Aalto scientific computing guide

Aalto Science-IT

Sep 13, 2018

Contents

This documentation is a public note of any topic regards computer science. May it be Machine Learning, Discrete Mathematics or Bayesian Data Analysis.

These docs are mainly maintained by Seyoung Park.

CHAPTER 1

Online courses

1.1 Current

- Mathematics for Machine Learning Linear Algebra by Imperial College London
- Introduction to Mathematical Thinking
- Linear Algebra by Khan academy
- Introduction to Probability and Statistics by MIT

CHAPTER 2

Contents

2.1 AI

2.1.1 AI – Fundamentals

2.1.2 Answer Set Programming(ASP)

ASP is a declarative programming method emphasizing on **what** to be computed rather than **how** to compute.

Fig. 1: < Formulas as rules and satisfying assignments as answer sets. Source: Aalto AI[1] >

Theory

Answer sets

1. The set is closed under the rules of the program.
2. If some particular atom belongs to the set, then there is at least one supporting rule instance having the atom in question as its head and the body of the rule is satisfied by the set. In other words, atoms cannot be true without a reason.
3. The set is minimal in this sense. In other words, atoms are false by default.

Positive programs

A positive rule is is an expression of the form

$$a \leftarrow b_1, \cdots, b_n$$

[1] https://www.youtube.com/watch?v=kdcd7Je2glc

where the head atom a can be inferred if the body atoms $b_1, \cdots, b_n$ have been inferred by other rules in the program. A rule with an empty body ($n = 0$) is called a fact.

Definition The (unique) answer set of a positive program is the least set S of ground atoms which is closed under the ground instances of its rules:

1. If there is a ground instance $h(t) \leftarrow b_1(t_1) \cdots b_n(t_n)$ of some rule in the program such that $b_1(t_1) \in S, \ldots, b_n(t_n) \in S$, then also $h(t) \in S$
2. If some other set $S' \subseteq S$ is closed in this way, then $S' = S$

ASP tutorial

Graph coloring[1]

n-Queen problem[2]

Basic

Advanced

The above code is much faster than the basic one but it will still take much time on solving due to `:- { queen(I, J) : D = I+J-1 } >= 2, D=1,..2*n-1.`. It is an exhaustive calculation. Let's be more efficient.

References

2.1.3 Logic

Logical Connectives

`NOR` and `NAND` gates are sufficient to express any Boolean function, i.e., they are functionally complete. If you want to verify that a set of operators is functionally complete, show that can be expressed with either `NOR` or `NAND`. Read Stackoverflow for more detail.

Fig. 2: < Logical connectives Venn diagram >

Logical Reasoning

Fig. 3: < The illustration of declarative problem solving. The logical approach provides the foundation for it.[1] >

---

[2] https://www.youtube.com/watch?v=d3arlJlGRTk&feature=youtu.be
[1] https://mycourses.aalto.fi/course/view.php?id=16956

Language of Propositional Logic

Syntax for Propositional Formulas

- negation $\neg$ ("not")
- conjunction $\wedge$ ("and")
- disjunction $\vee$ ("or")
- implication $\rightarrow$ ("if ... then")
- equivalence $\leftrightarrow$ ("if and only if" or "bi-implication")

Examples

- rainy $\rightarrow$ wet: If it rains, then it's wet.
- $\neg$ dry $\vee \neg$ swim: One can't be dry while he is swimming and vice versa.

Definition – formula

1. Every atomic formula is a **formula**.
2. If α and β are formulas, then also $(\neg\alpha), (\alpha \vee \beta), (\alpha \wedge \beta), (\alpha \rightarrow \beta), (\alpha \leftrightarrow \beta)$ are **formulas**.

Definition – subformula

The subformulas of a formula are defined recursively as follows.

1. The only subformula of an atomic proposition **a** is **a** it self.
2. The subformulas of $\not\alpha$ are $\not\alpha$ and all the subformulas of α.
3. If $*$ is any of the connectives $\vee, \wedge, \rightarrow, \leftrightarrow$, the subformulas of $(\alpha * \beta)$ are $(\alpha * \beta)$ and all the subformulas of α and β.

Examples

The subformulas of the formula $swim \rightarrow \neg dry$ are the formula itself, $swim$, $\neg dry$ and dry.

Example – Formulas from natural language

Consider the following statement in natural language:

If the file is too big, then it is compressed or removed.

For the sake formalization, we can identify the following atomic statements/formulas:

1. b: The file is too big.
2. c: The file is compressed.
3. r: The file is removed.

By substituting these statements into the original statement, we may derive a preliminary formalization that mixes natural language with the formal one:

If b, then c or r

Finally the formula is obtained by identifying the connectives and by incorporating them into the expression:

$$b \rightarrow c \vee r$$

Example for Terminologies

Fig. 4: < 2x2 sudoku example[1] >

We can study the above sudoku and reason that <2,1> can only contain 2. This can be expressed in a formal logical calculus as a disjunction

$$val(1,1,2) \vee val(1,2,2) \vee val(2,1,2) \vee val(2,2,2)$$

(i.e., 2 has to be in either in one of the 2x2 grid.)

The clue 2 at coordinates 1,3 and constraints for the first column (where x=1) contribute the following formulas

$$val(1,3,2), \quad \neg val(1,2,2) \vee \neg val(1,3,2), \quad \neg val(1,1,2) \vee \neg val(1,3,2).$$

(i.e., 2 cannot be placed simultaneously in the cells 1,2 and 1,3.)

Truth tables

α	β	$(\alpha \wedge \beta)$
F	F	F
F	T	F
T	F	F
T	T	T

α	β	$(\alpha \vee \beta)$
F	F	F
F	T	T
T	F	T
T	T	T

α	β	$(\alpha \rightarrow \beta)$
F	F	T
F	T	T
T	F	F
T	T	T

α	β	$(\alpha \leftrightarrow \beta)$
F	F	T
F	T	F
T	F	F
T	T	T

α	β	$(\alpha \oplus \beta)$
F	F	F
F	T	T
T	F	T
T	T	F

Models, Satisfiability and Unsatisfiability

Satisfaction is a relationship between specific sentences and specific truth assignments.[2]

- A sentence is *valid* if and only if it is satisfied by *every* truth assignment.
  - e.g. $(p \vee \neg p)$
  - *satisfiable*
- A sentence is unsatisfiable if and only if it is not satisfied by any truth assignment.
  - e.g. $(p \wedge \neg p)$
  - *falsifiable*
- A sentence is contingent if and only if there is some truth assignment that satisfies it and some truth assignment that falsifies it.
  - e.g. $(p \wedge q)$
  - *satisfiable* AND *falsifiable*.

Definition – model

Let α be a formula and Σ a set of formulas. A truth assignment v is called

1. a **model** of α, if $v \vDash \alpha$, and
2. a **model** of Σ, if $v \vDash \Sigma$, i.e., $v \vDash \beta$ for every formula $\beta \in \Sigma$

Logical entailment

We say that a sentence ϕ logically entails a sentence ψ (written $\phi \ \ \psi$) if and only if every truth assignment that satisfies ϕ also satisfies ψ. For example, the sentence p logically entails the sentence $(p \vee q)$. Since a disjunction is true whenever one of its disjuncts is true, then $(p \vee q)$ must be true whenever p is true.

Logical consistency

A sentence ϕ is consistent with a sentence ψ if and only if there is a truth assignment that satisfies both ϕ and ψ. For example, the sentence $(p \vee q)$ is consistent with the sentence $(p \wedge q)$. However it is NOT consistent with $(\neg p \wedge \neg q)$[3]

While consistency and entailment are very similar they don't entail each other.

[2] http://intrologic.stanford.edu/sections/section_03_01.html
[3] http://intrologic.stanford.edu/sections/section_03_05.html

Transformation Rules

1. $\alpha \leftrightarrow \beta \Longrightarrow (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha) \Longrightarrow (\neg\alpha \vee \beta) \wedge (\neg\beta \vee \alpha)$ (2.1)
2. $\alpha \rightarrow \beta \Longrightarrow \neg\alpha \vee \beta$ (2.2)
3. $\neg(\alpha \vee \beta) \Longrightarrow \neg\alpha \wedge \neg\beta$ (2.3)
4. $\neg(\alpha \wedge \beta) \Longrightarrow \neg\alpha \vee \neg\beta$ (2.4)
5. $\neg\neg\alpha \Longrightarrow \alpha$ (2.5)
6. $\alpha \vee (\beta \wedge \gamma) \Longrightarrow (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$ (2.6)
7. $(\alpha \wedge \beta) \vee \gamma \Longrightarrow (\alpha \vee \gamma) \wedge (\beta \vee \gamma)$ (2.7)
8. $\alpha \wedge (\beta \vee \gamma) \Longrightarrow (\alpha \wedge \beta) \vee (\alpha \vee \gamma)$ (2.8)
9. $(\alpha \vee \beta) \wedge \gamma \Longrightarrow (\alpha \wedge \gamma) \vee (\beta \vee \gamma)$ (2.9)

(2.10)

To transform a propositional formula into a CNF or DNF, follow the below steps:

1. Remove equivalences ($\leftrightarrow$)
2. Remove implications ($\rightarrow$)
3. Push negations inside until negations ($\neg$) occur as parts of negative literals only.
4. Organize conjunctions outside disjunctions (CNF) or disjunctions outside conjunctions (DNF).

Conjunctive Normal Form(CNF)

A statement is in conjunctive normal form if it is a conjunction (sequence of ANDs) consisting of one or more conjuncts, each of which is a disjunction (OR) of one or more literals. Examples of conjunctive normal forms include[4]

$$A$$
$$(A \vee B) \wedge (\neg A \vee C)$$
$$A \vee B$$
$$A \wedge (B \vee C)$$

Example

Transform the formula $\neg(p \wedge q) \leftrightarrow (r \wedge s)$ into CNF.

Disjunctive normal form (DNF)

A formula α is in disjunctive normal form (DNF) if and only if has the form $\beta_1 \vee \cdots \vee \beta_n$ where $n \geq 0$ and each disjunt β_i is a cube. Example:

$$(\neg fire \wedge \neg alarm) \vee (fire \wedge alarm)$$

[4] http://mathworld.wolfram.com/ConjunctiveNormalForm.html

Reference

2.1.4 Markov Decision Process

An MDP is defined by:

- $s \in S$ – a set of states
- $a \in A$ – a set of actions
- $T(s, a, s')$ – A transition_function/model/dynamics
  - prob that a from s leads to s', i.e., $P(s'|a, s)$
- $R(s, a, s')$ – a reward(cost) function
  - aka $R(s')$ or $R(s)$
  - We want to maximize the reward/cost.
- α – a start state
- γ – a discount factor
- Maybe a terminal state

- MDPs are non-deterministic/stochastic search problems

Markov Property

The future is independent of the past given the present. *[RL_course_mdp_DeepMind]*

A state S_t is *Markov* if and only if

$$\mathbb{P}\left[S_{t+1}|S_t\right] = \mathbb{P}\left[S_{t+1}|S_1, \ldots, S_t\right]$$

- The state S_t captures **all relevant information from the history** i.e., the state S_{t+1} is only dependent on S_t not on states before that.
- Once the state is known, the history may be thrown away i.e., the state is a sufficient statistic of the future

State transition matrix

For a Markov state s and successor state s', the state transition probability is defined by

$$\mathcal{P}_{SS'} = \mathbb{P}\left[S_{t+1} = s'|S_t = s\right]$$

State transition matrix $\mathcal{P}$ defines transition probabilities from all states s to all successor states s'.

$$\mathcal{P} = \text{ from } \begin{bmatrix} \mathcal{P}_{11} & \ldots & P_{1n} \\ \vdots & & \\ \mathcal{P}_{11} & \cdots & P_{nn} \end{bmatrix}$$

where each row of the matrix sum to 1.

Markov Process

A Markov process is a memoryless random process, i.e., a sequence of random states $S_1, S_2, ...$ with the Markov property.

- A Markov process/Chain is a tuple $< \mathcal{S}, \mathcal{P} >$
  - $\mathcal{S}$ is a finite set of states
  - $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}_{SS'} = \mathbb{P}\left[S_{t+1} = s' | S_t = 5\right]$

Markov Reward Process

A Markov reward process is a Markov chain with values, $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}\left[R_{t+1} | S_t = s\right]$
- γ is a discount factor, $\gamma \in [0, 1]$

Return

The return G_t is the total discounted reward from time-step t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma_{t+k+1}$$

Value Function

The value function $v(s)$ gives the long-term value of state s. It is the expected return starting from state s

$$v(s) = E\left[G_t | S_t = s\right]$$

Bellman Equation in Matrix Form

The Bellman equation can be expressed concisely using matrices.

$$v = \mathcal{R} + \gamma \mathcal{P} \mathbf{v}$$

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \ldots & P_{1n} \\ P_{11} & \ldots & P_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

You can solve the Bellman equation as a simple linear equation. The complexity is $O\left(n^3\right)$ for n states.

$$v = \mathcal{R} + \gamma P_\nu \tag{2.11}$$
$$(1 - \gamma P) v = \mathcal{R} \tag{2.12}$$
$$v = (1 - \gamma P)^{-1} \mathcal{R} \tag{2.13}$$

As we are using inversion, the direct solution is possible only for small MRPs. Other iterative solutions are

- Dynamic programming
- Monte-Carlo evalution
- Temporal-Difference Learning

Markov Decision Process

MDP is a MRP with decisions. It is an environment in which all states are Markov.

Policies

A policy π is a distribution over actions given states. It's a stochastic transition matrix.

$$\pi(a|s) = \mathbb{P}\left[A_t = a | S_t = s\right]$$

- A policy defines the behaviour of an agent.
- It depends on the current state.
- Policy is stationary(time-independnet)

Value Functions

The state-value function $V_\pi(s)$ of and MDP is the expected return starting from state s, and then following policy π.

$$v_\pi(s) = \mathbb{E}_\pi\left[G_t | S_t = s\right]$$

The action-value function $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π.

$$q_\pi(s, a) = \mathbb{E}_\pi\left[G_t | S_t = s, A_t = a\right]$$

Bellman expectation equation

The state-value function can again be decomposed into immediate reward plus discounted value of successor state.

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi\left(S_{t+1}\right) | S_t = s\right] \qquad (2.14) \\ &= \sum_{a \in A} \pi(a|s)\left(\mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^2 v_\pi(s)\right) \qquad (2.15) \end{aligned}$$

The action-value function can similarly be decomposed.

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi\left(S_{t+1}, A_{t+1}\right) | S_t = s, A_t = a\right] \qquad (2.16) \\ &= \gamma \sum_{s' \in S} \mathcal{P}_{SS'}^a \sum_{d' \in A} \pi\left(a'|s'\right) q_\pi\left(s', a'\right) \qquad (2.17) \end{aligned}$$

Optimal Value Function

The optimal state-value function $V_*(s)$ is the maximum state-value function over all policies

$$V_*(s) = \max_\pi v_\pi(s)$$

The optimal action-value function $q_*(s, a)$ is the maximum action-value function over all policies

$$q_*(s, a) = \max_\pi q_\pi(s, a)$$

An MDP is "solved" when we know the optimal value fn.

Define a partial ordering over policies

$$\pi \geq \pi' \text{ if } v_\pi(s) \geq v_{\pi'}(s), \forall_5$$

An optimal policy can be found by maximizing over $q_*(s, a)$,

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \arg\max q_x(5, a) \\ 0 & \text{otherwise} \end{cases}$$

- There is always a deterministic optimal policy for any MDP.
- If we know $q_*(s, a)$, we immediately have the optimal policy.

Solving the Bellman Optimality Equation

- Bellman optimality equation is non-linear
- No closed form solution in general
- Many iterative solution methods
  - Value iteration
  - Policy iteration
  - Q-learning
  - Sarsa

Q learning

"Q" is for "quality" of a certain action in a given state. We define a function $Q(s, a)$ representing the maximum discounted future reward when we perform action a in state s, and continue optimally from that point on.

$$Q(s_t, a_t) = maxR_{t+1}$$

Bellman equation represents the optimal Q-value of state *s* and action *a* in terms of the next state *s'* as following,

$$Q^*(s, a) = r + \gamma max_a Q^*(s', a')$$

In words, it means the optimal future reward for this state and action is the immediate reward plu maximum future reward for the next state.

Explore-exploit dilemma

Should an agent exploit the known working strategy or explore possibly better unknown strategies?

Reference

osascript -e 'set volume without output muted output volume 17 –100%'

2.1.5 Search

Note: This part of docs is mostly brought from Amit Patel's blog(https://www.redblobgames.com/pathfinding/a-star/).

Uninformed algorithms

DFS, BFS

Breadth First Search

The following code demonstrates how to BFS work.

```
# Author: Amit Patel
# https://www.redblobgames.com/pathfinding/a-star/introduction.html#breadth-first-
↪search
frontier = Queue()
frontier.put(start)
visited = {}
visited[start] = True

while not frontier.empty():
    current = frontier.get()

    # Following is for Early exit
    if current == goal:
      break    # End of the algorithm

    for next in graph.neighbors(current):
        if next not in visited:
            frontier.put(next)
            visited[next] = True
```

While search algorithms are mainly used to find shortest paths, it can be used to create procedural map generation.

```
# Author: Amit Patel
# https://www.redblobgames.com/pathfinding/a-star/introduction.html#breadth-first-
↪search

frontier = Queue()
frontier.put(start)
came_from = {}
came_from[start] = None

while not frontier.empty():
  current = frontier.get()
  for next in graph.neighbors(current):
    if next not in came_from:
      frontier.put(next)
      came_from[next] = current
```

It's easy to get a path from a specific point back to the start.

```
# Author: Amit Patel
# https://www.redblobgames.com/pathfinding/a-star/introduction.html#breadth-first-
↪search
current = goal
path = []
while current != start:
  path.append(current)
  current = came_from[current]
```

(continues on next page)

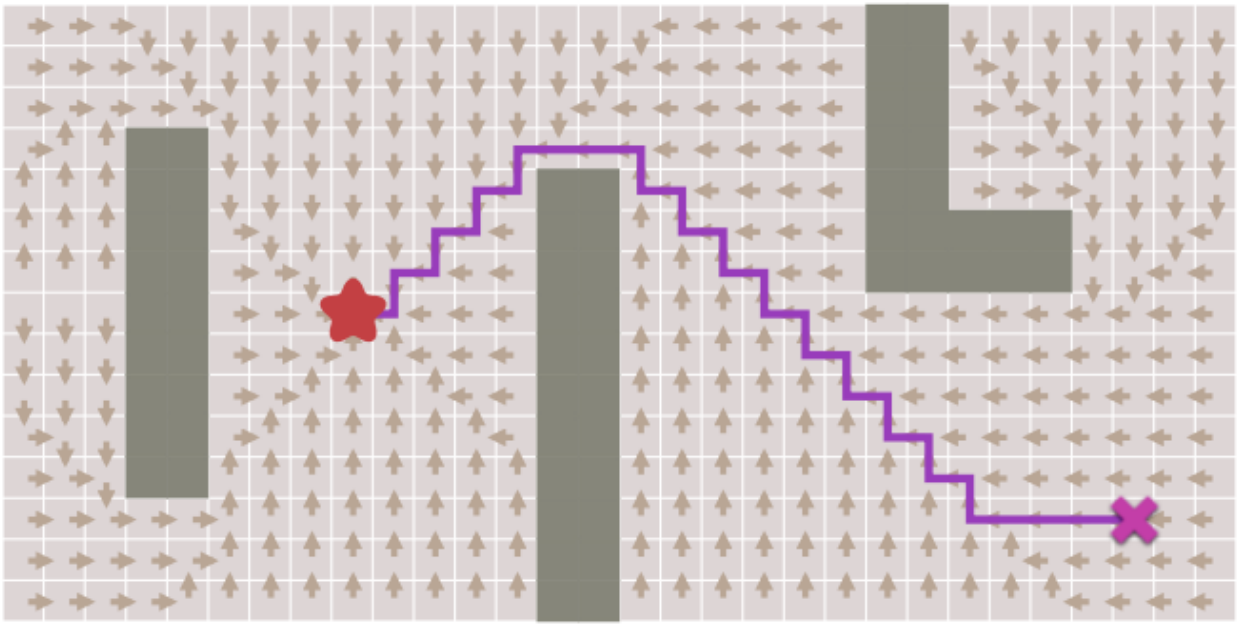


Fig. 5: < Procedural map generation by BFS - Amit Patel >

(continued from previous page)

```
path.append(start) # optional
path.reverse() # optional
```

Informed algorithms

Dijkstra's algorithm

AKA **Uniform Cost Search**. It prioritizes which path to explore. Instead of exploring all possible paths, it favors lower cost paths. We can assign lower costs to encourage moving on roads, higher costs to avoid forests, higher costs to discourage going near enemies, and more. When movement costs vary, we use this instead of Breadth First Search.

```
# Author: Amit Patel
# https://www.redblobgames.com/pathfinding/a-star/introduction.html#dijkstra
frontier = PriorityQueue()
frontier.put(start, 0)
came_from = {}
cost_so_far = {}
came_from[start] = None
cost_so_far[start] = 0

while not frontier.empty():
   current = frontier.get()

   if current == goal:
      break

   for next in graph.neighbors(current):
```

(continues on next page)

(continued from previous page)

```
      new_cost = cost_so_far[current] + graph.cost(current, next)
      # YES, you may add a same frontier multiple times. That's
      # why we have costs and priorities here.
      if next not in cost_so_far or new_cost < cost_so_far[next]:
         cost_so_far[next] = new_cost
         priority = new_cost
         frontier.put(next, priority)
         came_from[next] = current
```

Heuristic Search

BFS and Dijkstra's algorithm expand in all directions but many times we have a direction. **Heuristic function** tells us how close we are to the goal. Here's a simple example.

```
# Author: Amit Patel
# https://www.redblobgames.com/pathfinding/a-star/introduction.html#greedy-best-first
def heuristic(a, b):
  # Manhattan distance on a square grid
  return abs(a.x - b.x) + abs(a.y - b.y)
```

Greedy Best First Search

The queues of frontiers are now ordered by their priorities which tells you to which direction you should go.

```
# Author: Amit Patel
# https://www.redblobgames.com/pathfinding/a-star/introduction.html#greedy-best-first
frontier = PriorityQueue()
frontier.put(start, 0)
came_from = {}
came_from[start] = None

while not frontier.empty():
  current = frontier.get()

  if current == goal:
    break

  for next in graph.neighbors(current):
    if next not in came_from:
      priority = heuristic(goal, next)
      frontier.put(next, priority)
      came_from[next] = current
```

A^* algorithm

A^* is a modification of Dijkstra's Algorithm that is optimized for a single destination. So you may think it as a mixture of Dijkstra and Greedy Best First Search.

```
# Author: Amit Patel
# https://www.redblobgames.com/pathfinding/a-star/introduction.html#astar
frontier = PriorityQueue()
```

(continues on next page)

(continued from previous page)

```
frontier.put(start, 0)
came_from = {}
cost_so_far = {}
came_from[start] = None
cost_so_far[start] = 0

while not frontier.empty():
   current = frontier.get()

   if current == goal:
      break

   for next in graph.neighbors(current):
      new_cost = cost_so_far[current] + graph.cost(current, next)
      if next not in cost_so_far or new_cost < cost_so_far[next]:
         cost_so_far[next] = new_cost
         priority = new_cost + heuristic(goal, next)
         frontier.put(next, priority)
         came_from[next] = current
```

Choosing a Heuristic Function for A^*

From A*'s Use of the Heuristic

- At one extreme, if $h(n)$ is 0, then only $g(n)$ plays a role, and A^* turns into Dijkstra's Algorithm, which is guaranteed to find a shortest path.
- If $h(n)$ is always lower than (or equal to) the cost of moving from n to the goal, then A^* is guaranteed to find a shortest path. The lower $h(n)$ is, the more node A^* expands, making it slower.
- If $h(n)$ is exactly equal to the cost of moving from n to the goal, then A^* will only follow the best path and never expand anything else, making it very fast. Although you can't make this happen in all cases, you can make it exact in some special cases. It's nice to know that given perfect information, A^* will behave perfectly.
- If $h(n)$ is sometimes greater than the cost of moving from n to the goal, then A^* is not guaranteed to find a shortest path, but it can run faster.
- At the other extreme, if $h(n)$ is very high relative to $g(n)$, then only $h(n)$ plays a role, and A^* turns into Greedy Best-First-Search.

Note:

The following part is is made by studying

1. Aalto University CS-E4800 - AI
2. Reinforcement Learning: An Introduction by Sutton
3. Berkeley AI

Glossaries

Expansion

The expansion of a node n means that each successor node m of n is checked: if m has not been visited, it is visited now and recorded for further (recursive) expansion.

Admissible heuristic

> In computer science, specifically in algorithms related to pathfinding, a heuristic function is said to be admissible if it never overestimates the cost of reaching the goal, i.e. the cost it estimates to reach the goal is not higher than the lowest possible cost from the current point in the path. - Wikipedia: Admissible heuristic

Monotonic heuristic

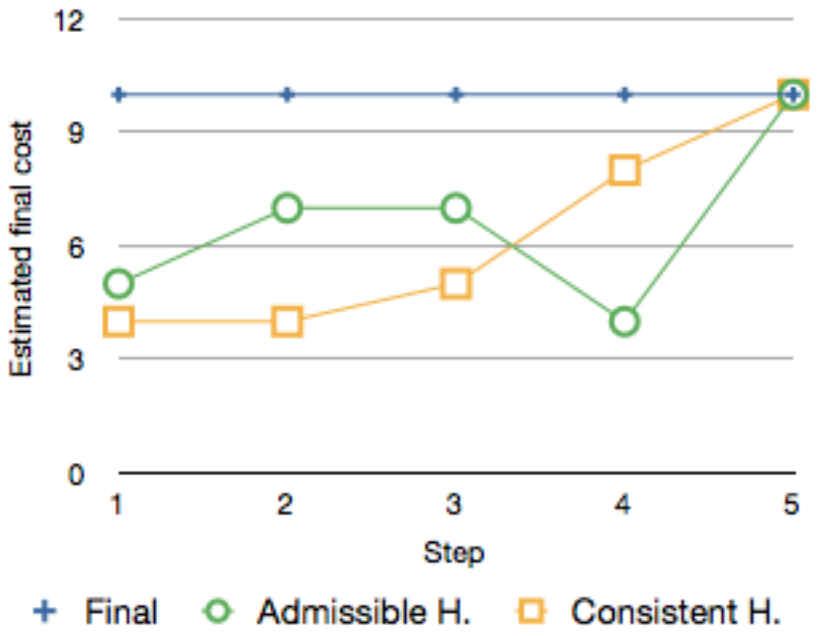


Fig. 6: < Comparison of an admissible but inconsistent and a consistent heuristic evaluation function. - Wikipedia: Consistent heuristic >

References

2.1.6 Reference

Textbooks

- https://www.people.vcu.edu/~rhammack/BookOfProof/

2.2 Bayesian

2.2.1 Bayesian – Fundamentals

Probabilistic modeling

The goal of probabilistic modeling:

- Classify the samples into groups
- Create prediction for future observations
- Select between competing hypothesis
- Estimate a parameter, such as the mean of a population

How do we run probabilistic modeling?

1. Models: Bayesian networks, Sparse Bayesian linear regression, Gaussian mixture models, latent linear models
2. Methods for inference: max likelihood, max a posteriori(MAP), Laplace approx., expectation maximization(EM), Variational Bayes(VB), Stochastic variational inference(SVI)
3. And choose a way to select different models based on your inference and update your model!

Bayes' Theorem

Bayes' theorem describes probabilities of an event, based on prior knowledge – called *hypothesis* in Bayesian inference – that might be related to the event – called *evidence* in Bayesian inference.

$$P(A|B) = \frac{P(B|A)}{P(B)} \times P(A) \tag{2.18}$$

$$\text{posterior} = \frac{\text{Likelihood}}{\text{Marginal likelihood/Evidence}} \times \text{prior} \tag{2.19}$$

$$\text{Evidence} = P(B) = \int P(B, A) \tag{2.20}$$

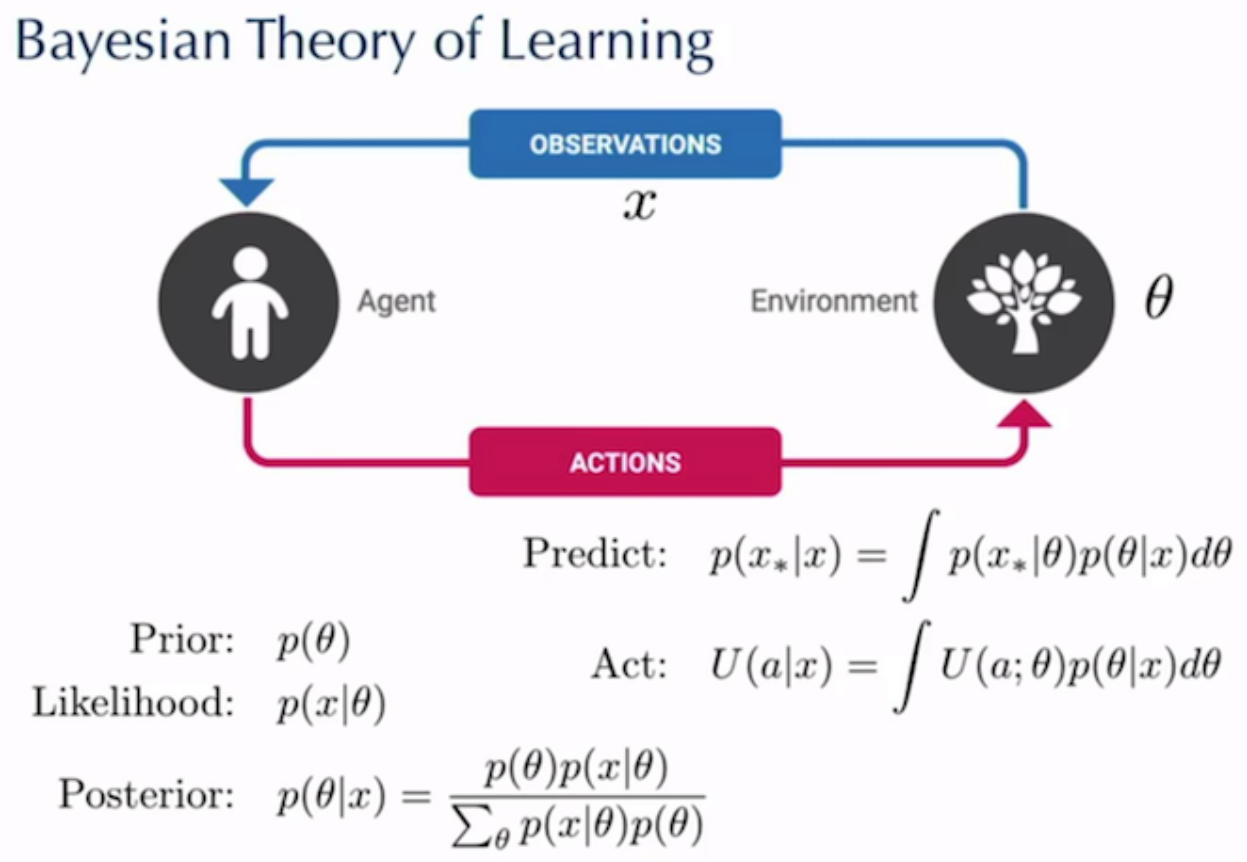


Fig. 7: < Yee Whye Teh: On Bayesian Deep Learning and Deep Bayesian Learning >

Bayesian vs. Frequentist

- In Probability domain They all use Bayes' formula when a prior $p(\theta)$ is known.
- In Statistics domain In statistics prior is unknown and it's where the two diverge.
  - Bayesians: they need a prior, so they develop one from the best information they have.
  - Frequentists: They draw inferences from likelihood func.

Fig. 8: < Using Bayes' theorem one can inverse conditional probabilities. The red items are new information. Source: Gnathan87 CC0[2]>

Conjugate priors

Suppose we have data with likelihood function $p(x|\theta)$ depending on a hypothesized parameter. Also suppose the prior distribution for θ is one of a family of parameterized distributions. If the posterior distribution for θ is in this family then we say the prior is a conjugate prior for the likelihood.

Examples

- *Beta* is conjugate to Bernoulli
- Gaussian is conjugate to Gaussian
- Any exponential family has a conjugate prior

Prior	Hypothesis	data	Likelihood	Posterior
$beta(a,b)$	$\theta \in [0,1]$	x	$Bernoulli(\theta)$	$beta(a+1,b)$ or $beta(a,b+1)$
$beta(a,b)$	$\theta \in [0,1]$	x	$Binomial(N,\theta)$	$beta(a+x,b+N-x)$
$beta(a,b)$	$\theta \in [0,1]$	x	$geometric(\theta)$	$beta(a+x,b+1)$
$\mathcal{N}(\mu_{prior},\sigma^2_{prior})$	$\theta \in (-\infty,\infty)$	x	$\mathcal{N}(\theta,\sigma^2)$	$\mathcal{N}(\theta_{post},\sigma^2_{post})$

Weakly informative prior

It a prior distribution which doesn't contain much information, but contains some enough information 'regularize' the posterior distribution, and to keep it roughly within reasonable bounds. It is often used in situations where we don't have much information but we want to ensure that the posterior distribution makes sense.

Improper prior

An improper prior is essentially a prior probability distribution that's infinitesimal over an infinite range, in order to add to one. For example, the uniform prior over all real numbers is an improper prior, as there would be an infinitesimal probability of getting a result in any finite range. It's common to use improper priors for when you have no prior information.[5]

---

[2] https://en.wikipedia.org/wiki/Bayes%27_theorem
[5] http://lesswrong.com/lw/6uk/against_improper_priors/

Bayesian Inference

Bayesian inference is just one application of Bayes' theorem[1]. We use it when we don't have as much data as you wish and want to maximize your predictive strenth. The bayes' theorem could be interpreted in the following way.

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

- H: Hypothesis.
  - A hypothesis is a possible answer to the question being asked. It can be affected by new data or Evidence, E. There can be multiple hypotheses and our job is to get the best one.
- E: Evidence.
  - It's new data observed.
- $P(H)$: Prior probability.
  - The probability of the hypothesis **before** the current evidence/data/E is observed. It is the initial degree of belief in H.
- $P(H|E)$: Posterior probability.
  - The probability of the hypothesis **after** the current evidence/data/E is observed. This is what we want to know ultimately. The higher the posterior is our hypothesis well-fits to new data. It is a degree of belief having accounted for E.
- $P(E|H)$: Likelihood.
  - The probability of current evidence/data/E given hypothesis. It's just another fancy name for conditional probability i.e., $P(A|B)$ could be read as *"How A is likely to occur given B?"*. Note, it's NOT a distribution[4]!
- $P(E)$: Marginal likelihood or model evidence.
  - This factor is the same for all hypotheses i.e., this does not enter into determining the relative probabilities of different hypotheses.

The Bayes' rule can be rewritten as

$$P(H|E) = \frac{P(E|H)}{P(E)} \cdot P(H)$$

where the factor $\frac{P(E|H)}{P(E)}$ represents the impact/support of E on H. This statement may be confusing but if one looks at the theorem it becomes obvious.

$$P(H|E) \propto \frac{P(E|H)}{P(E)}, \quad \text{AND} \tag{2.21}$$

$$P(H|E) \propto P(H) \tag{2.22}$$

$$\tag{2.23}$$

Both $P(H)$ and $\frac{P(E|H)}{P(E)}$ are factors of the posterior

[1] https://en.wikipedia.org/wiki/Bayesian_inference
[4] https://github.com/YoungxHelsinki/papers/blob/961603b8eccf5352580871dd43052164ae540962/tutorials/primer.pdf

Everyday-life example

I found a very intuitive example by Brandon[3]. In the article both frequentists and bayesians use Bayes' theorem. The difference is frequentists use uniform prior and bayesians use whatever they can. The prior could be empirical distribution such as age is between 0 and 130, temperature higher than -276 Celcius.

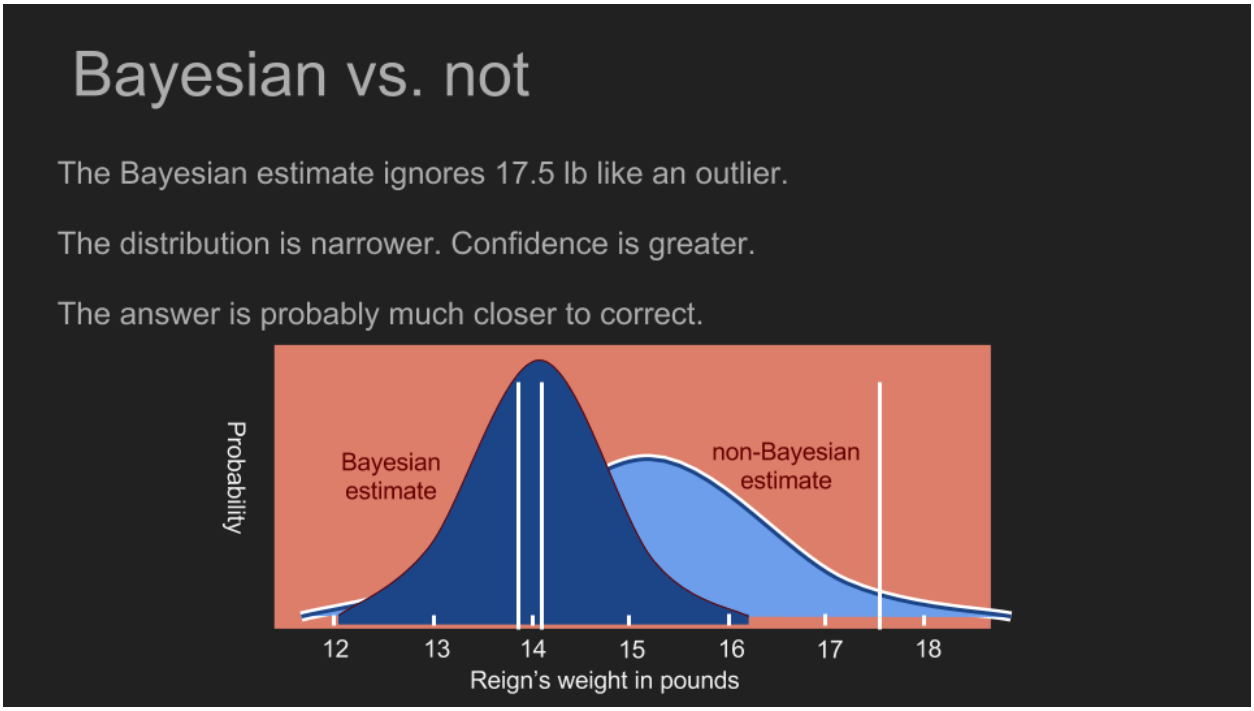


Fig. 9: < Bayesian vs. Frequentists. Source: Brandon Brohrer >

The peak/mean of frequentists' distribution is the Maximam Likelihood Estimate(MLE). The peak/mean of bayesian distribution is the Maximum A Posteriori estimate(MAP).

As Brandon mentions one should be aware of Bayesian traps. In Bayesian inference, we build prior ourselves and if the observed value doesn't exist in our prior then the posterior would be zero. Common trick is to use the normal distribution in order to keep the very low edges but which never becomes zero.

References

2.2.2 Bayesian Network

[3] https://brohrer.github.io/how_bayesian_inference_works.html

Note: Here I use $\perp\!\!\!\perp$ as an independence symbol.

A Bayesian network(BN) is a directed acyclic graph (DAG) in which nodes represent random variables, whose joint distribution is as follows,

$$p(x_1, ..., x_D) = \prod_{i=1}^{D} p\big(x_i | pa(x_i)\big)$$

where $pa(x_i)$ represents the parents of x_i.

$$p(a, b, c, d, e) = p(a)p(b)p(c|a, b)p(d|c)p(e|b, c)$$

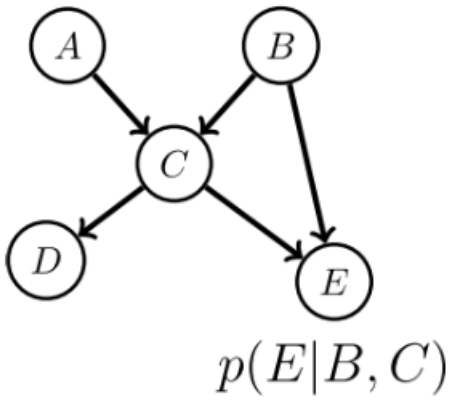
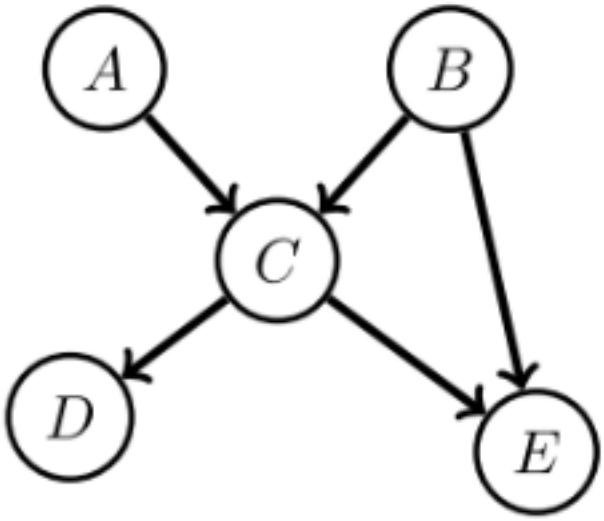


Fig. 10: < An example of a bayesian network. Source: Aalto course CS-E4820: Advanced probabilistic methods >

BNs are used in ML, because they are

- a concise way to represent and communicate the structure and assumptions of a model.
- a compact representation of the joint distribution. => efficient!



Independence in Bayesian networks

Example 1

Independent?	D connection
$A \perp\!\!\!\perp B$	–
$A \perp\!\!\!\perp B \mid C$	Separated
$A \perp\!\!\!\perp B \mid E$	Separated
$D \not\perp\!\!\!\perp E \mid C$	Connected

Example 2

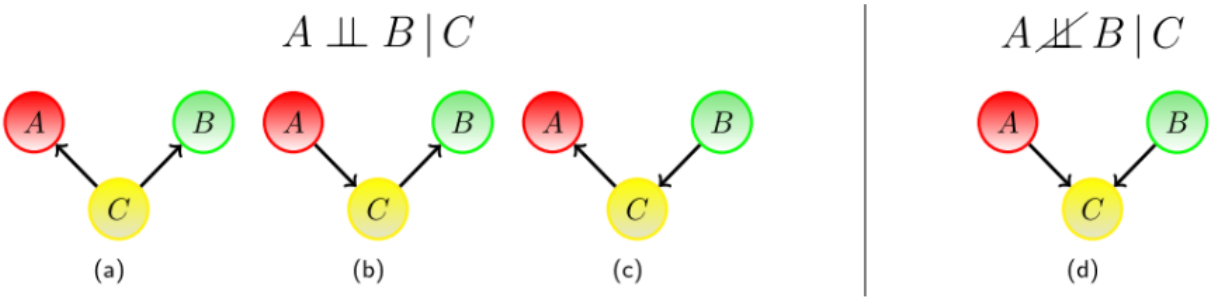


Fig. 11: < Conditional Independence >

Here's my real life example about marginal independence. Say you won a $1M lottery(C). You can either buy a $1M house(A) or buy a $1M Ferrari(B). If you've bought a Ferrari, you haven't bought a house for sure.

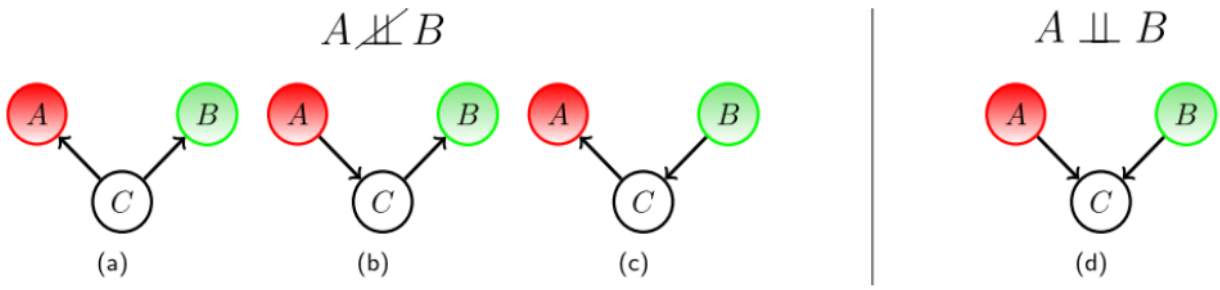
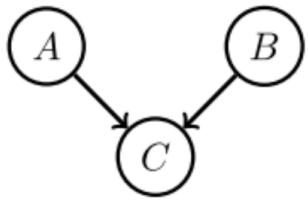


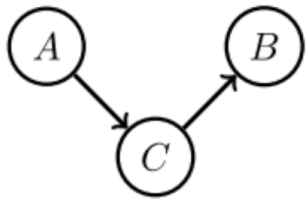
Fig. 12: < Marginal Independence >

Collider

A cllider (v-structure, head-to-head meeting) has two incoming arrows along a chosen path.



- C a collider
  - $A \perp\!\!\!\perp B$
  - $A \not\perp\!\!\!\perp B | C$
  - e.g. A ='Talented in sports', B ='Talented in maths', C ='Admitted to school'



- C a non-collider
  - $A \not\perp\!\!\!\perp B$
  - $A \perp\!\!\!\perp B | C$
  - e.g. A ='Cumulative sum of $n-1$ dice throws', C ='Cumsum of n throws', B ='Cumsum of $n+1$ throws'

Fig. 13: < Source: Aalto course CS-E4820: Advanced probabilistic methods >

D-Connection & D-Separation

- **A path** between variables A and B **is blocked** by a set of variables $\mathcal{C}$, if
  - there is a collider in the path such that neither the collider nor any of its descendasnts is in the conditioning **set** $\mathcal{C}$.
  - there is a non-collider in the path that is in the conditioning **set** $\mathcal{C}$.
- Sets of variables $\mathcal{A}$ and $\mathcal{B}$ are **d-separated** by $\mathcal{C}$ if all paths between $\mathcal{A}$ and $\mathcal{B}$ are blocked by $\mathcal{C}$.
  - d-separation implies $A \perp\!\!\!\perp B | C$

  - $\mathcal{X}$ and $\mathcal{Y}$ are d-separated by $\mathcal{Z}$ in G iff they are not d-connected by $\mathcal{Z}$ in G.

Bottom line

- Non-collider in the conditioning set ⇒ Blocked ⇒ d-separated ⇒ **conditionally independent** *BUT* **unconditionally dependent**
- Collider or its descendants in the conditioning set ⇒ Not blocked ⇒ d-connected ⇒ **conditionally dependent** *BUT* **unconditionally independent**

Examples

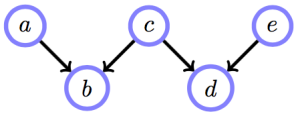


Fig. 14: < *b* d-separates *a* from *e*. *{b,d}* d-connect *a* from *e*. >

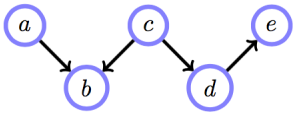


Fig. 15: < *c* and *e* are (unconditionally) d-connected. *b* d-connects *a* and *e* >

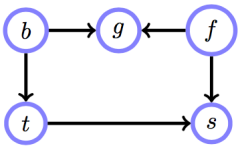


Fig. 16: < *t* and *f* are d-connected by *g* >

Markov equivalence

Two graphs are **Markov equivalent** if they

- entail(need) the same conditional independencies
- equivalently have the same d-separations

Graph

- Parent: pa(D) = {A,C}
- Children: ch(D) = E
- Family: A node itself and its parents.
  - fa(E) = {B,D,E,F}

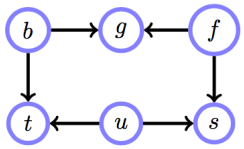


Fig. 17: < b and f are d-separated by u >

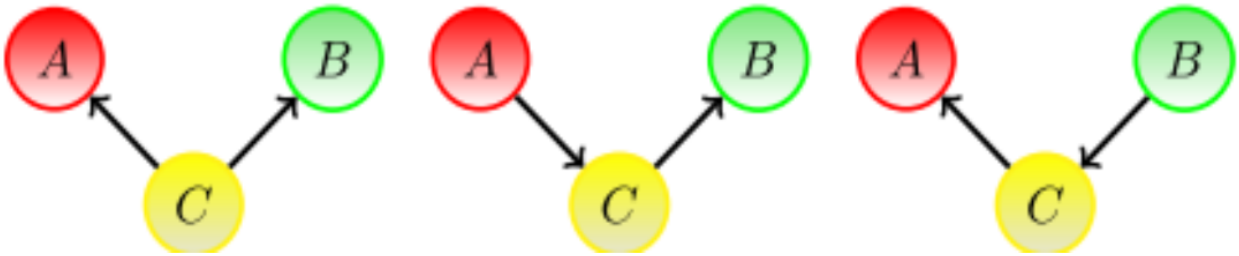
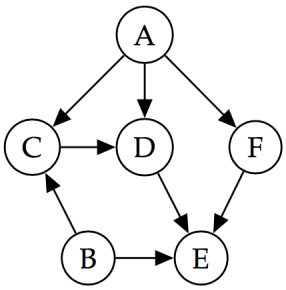


Fig. 18: < A markov equivalent example >



- Markov blanket: A node itself, its parents, children and the parents of its children.
  - MB(B) = {A,B,C,D,E,F}

2.2.3 Bayesian Models

Linear Parameter Models

Determining hyperparameters: ML-II

The hyperparameter posterior distribution is

$$p(\Gamma|\mathcal{D}) \propto p(\Gamma|\mathcal{D})p(\Gamma)$$

If $p(\Gamma) \approx const$ this is equivalent to

$$\Gamma^* = \arg\max_{\Gamma} p(\mathcal{D}|\Gamma),$$

where the **marginal likelihood**

$$p(\mathcal{D}|\Gamma) = \int p(\mathcal{D}|\Gamma, \boldsymbol{w})p(\boldsymbol{w}|\Gamma)d\boldsymbol{w}$$

ML vs. ML-II

In ML(Maximum likelihood), we select parameter values **w** that maximize the log-likelihood

$$\log p(y|\mathbf{w}.\mathbf{x}) = \sum_{i=1}^{N} \log N(y_i|\mathbf{w}^T\psi(\mathbf{x}_i), \beta^{-1}) \tag{2.24}$$

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \log p(y|\mathbf{w}, \mathbf{x}) \quad \text{(does not depend on } \beta\text{)} \tag{2.25}$$

In ML-II, we select hyperparameter values α and β that maximize the (log-)marginal likelihood (parameters **w** integrated out)

2.2.4 Markov Chain Monte Carlo

Introduction

> From 1990 a revolution took place in Bayesian computing. The main problem in the application of Bayesian methods was until then that computation was difficult. With conjugate priors analytical results could sometimes be obtained, but in complex models even that did not bring much relief. With the increasing speed of computers however, simulation methods became more and more feasible. Methods like "importance sampling ". But in 1990 the "Markov chain Monte Carlo " (MCMC) methods were found. **A Markov chain means a simulation where the next draw only depends on the previous one.** The technique has in itself nothing to do with Bayesian analysis. The question is whether it is possible to simulate from a complex distribution. As in Bayesian inference the posterior is known if prior and likelihood are known it is possible to write down the formula for it. And if there is a trick to simulate given the formula, that is sufficient.[1]

[1] https://github.com/YoungxHelsinki/papers/blob/961603b8eccf5352580871dd43052164ae540962/tutorials/primer.pdf

Schemes

Gibbs sampler

Algorithm

- Suppose there are parameters $\theta_1 \cdots \theta_n$ and that the distribution of each θ_i, conditional upon the other θ's is known.
- Draw in each step $\theta_i|\theta_1 \cdots \theta_{i-1}, \theta_{i+1} \cdots \theta_n$, and cycle through θ_i (so $\theta_1, \theta_2 \cdots \theta_n, \theta_1, \theta_2 \cdots$)

This is very efficient but requires the conditional distribution must be known. This preperty requires a lot of work and in some cases it is not analytically possible, the main problem being the normalizaing constant.

Metropolis-Hastings(aka MCMCMC)

Algorithm

- Start with some $\theta_1 (i = 1)$
- The proposal for $\theta_{i+1}, \theta^p_{i+1}$ is drawn from uniform$(\theta_i - a, \theta_i + a)$
- If $f(\theta^p_{i+1}) > f(\theta_i)$ then the proposal is accepted: $\theta_{i+1} := \theta^p_{i+1}$ else then proposal is accepted with probability $\alpha = f(\theta^p_{i+1}) \big/ f(\theta_i)$. If not accepted, $\theta_{i+1} := \theta_i$

We only need likelihood and the prior unlike Gibbs. The posterior normalizing constant, $P(E)$ denominator in Bayes' theorem, can be unknown.

References

2.2.5 Probabilistic modeling

Intro

What is a probabilistic programming language?

Probabilistic modeling is a powerful approach for analyzing empirical information and a probabilistic programming language is a language that enables you to implement the method.

Why is probabilistic programming needed?

We live in an era where data became abundant and the efficient computing hardware became accessible. Probabilistic modeling is essential to fields related to its methodology, such as statistics and machine learning, as well as fields related to its application, such as computation biology, NLP and etc.

We can take advantage of probabilistic modeling to tackle difficult challenges and therefore we need probabilistic modeling.

What is Edward?

It's a probabilisitic modeling library built on iterative process for probabilistic modeling. For software systems to enable fast experimentation, Edward provides rich abstractions that embeds both a broad class of probabilistic models and a broad class of algorithms for efficient inference. As well, in order to meet the needs of high performance computing Edward supports distributed training and integration of hardware such as (multiple) GPUs.

What are the three steps in the iterative process for probabilistic modeling that Edward is built around?

Given data from some unknown phenomena,

1. Formulate a model of the phenomena
2. Use an algorithm to infer the model's hidden structure, thus reasoning about the phenomena
3. Criticize how well the model captures the data's generative process. As we criticize our model's fit to the data, we revise components of the model and repeat to form an iterative loop.

Identify parts of the example probabilistic program (on p. 3–4) that correspond to the three steps mentioned in the previous question.

1. Modeling

```
import tensorflow as tf
from edward.models import Normal
W_0 = Normal(mu=tf.zeros([1, 2]), sigma=tf.ones([1, 2]))
W_1 = Normal(mu=tf.zeros([2, 1]), sigma=tf.ones([2, 1]))
b_0 = Normal(mu=tf.zeros(2), sigma=tf.ones(2))
b_1 = Normal(mu=tf.zeros(1), sigma=tf.ones(1))
x = x_train
# They defined a two-layer Bayesian NN below.
y = Normal(mu=tf.matmul(tf.tanh(tf.matmul(x, W_0) + b_0), W_1) + b_1, sigma=0.1)
```

2. Model hidden structure inference

```
# Use variational inferences to make inferences about the model from data.
qW_0 = Normal(mu=tf.Variable(tf.zeros([1, 2])),
              sigma=tf.nn.softplus(tf.Variable(tf.zeros([1, 2]))))
qW_1 = Normal(mu=tf.Variable(tf.zeros([2, 1])),
              sigma=tf.nn.softplus(tf.Variable(tf.zeros([2, 1]))))
qb_0 = Normal(mu=tf.Variable(tf.zeros(2)),
              sigma=tf.nn.softplus(tf.Variable(tf.zeros(2))))
qb_1 = Normal(mu=tf.Variable(tf.zeros(1)),
              sigma=tf.nn.softplus(tf.Variable(tf.zeros(1))))

import edward as ed
inference = ed.KLqp({W_0: qW_0, b_0: qb_0,
                     W_1: qW_1, b_1: qb_1}, data={y: y_train})
inference.run(n_iter=1000)
```

3. Criticize

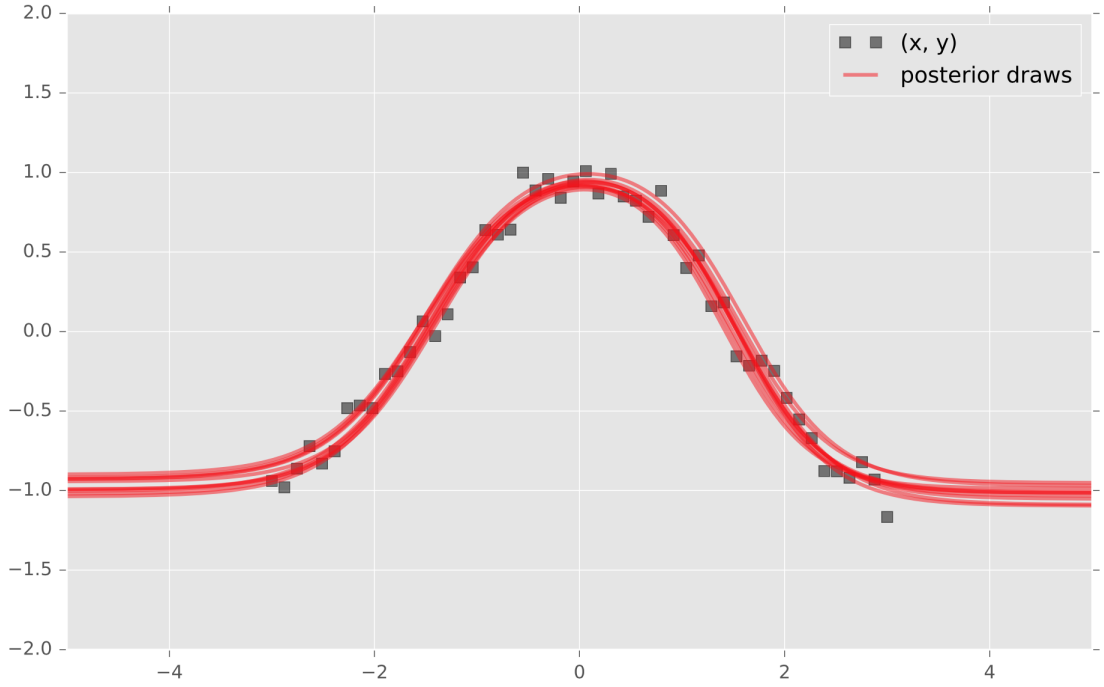


Figure 2: Posterior draws from the inferred Bayesian neural network.

Fig. 19: < The scatter plot is the observed data and the line the model. >

2.2.6 Variational Inference

Variational means you vary parameters in each step.

The idea is to approximate the posterior distribution of unknowns math:`p(Z|X)` with a tractable distribution math:`q(Z)`. For example math:`q(Z)` may be assumed to have a simple form, e.g., Gaussian, or to factorize in a certain way. For the GMM, it would be sufficient to assume

$$q(\mathbf{z}, \pi, \Lambda, \mu) = q(z)q(\pi, \Lambda, \mu)$$

Basis

When math:`q(Z)` is an approximation for math:`p(Z|X)`, it is always true that

$$\log p(x) = \mathcal{L}(q) + KL(q\|p), \text{ where}$$

$$\mathcal{L}(q) = \int q(\mathbf{z}) \log \left\{ \frac{p(x,z)}{q(z)} \right\} d\mathbf{z} \quad \text{(lower bound for } \log p(x)\text{)}$$

$$KL(q\|\rho) = -\int q(z) \log \left\{ \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})} \right\} d\mathbf{z} \quad \text{(KL-divergence btw q and p)}$$

- Goal: to maximize $\mathcal{L}(q)$ or, equivalently, to minimize the $KL(q||p)$.
- Note: $\mathcal{L}(q)$ is also called the 'ELBO'(evidence lower bound)

:math:`` :math:`` :math:`` :math:`` :math:``

Mixture models

- Probabilisitically-grounded way of doing soft clustering contrast to k-mean.
- Each cluster: a generative model(Gaussian or multinomial)
- Parameters(e.g. mean/covariance are unknown)

EM(Expectation Maximization) algorithm automatically discover all parameters for the K "sources".

Gaussian mixture models

Gaussian mixture models are a probabilistic model for representing normally distributed subpopulations within an overall population. Mixture models in general don't require knowing which subpopulation a data point belongs to, allowing the model to learn the subpopulations automatically. Since subpopulation assignment is not known, this constitutes a form of unsupervised learning.[1]

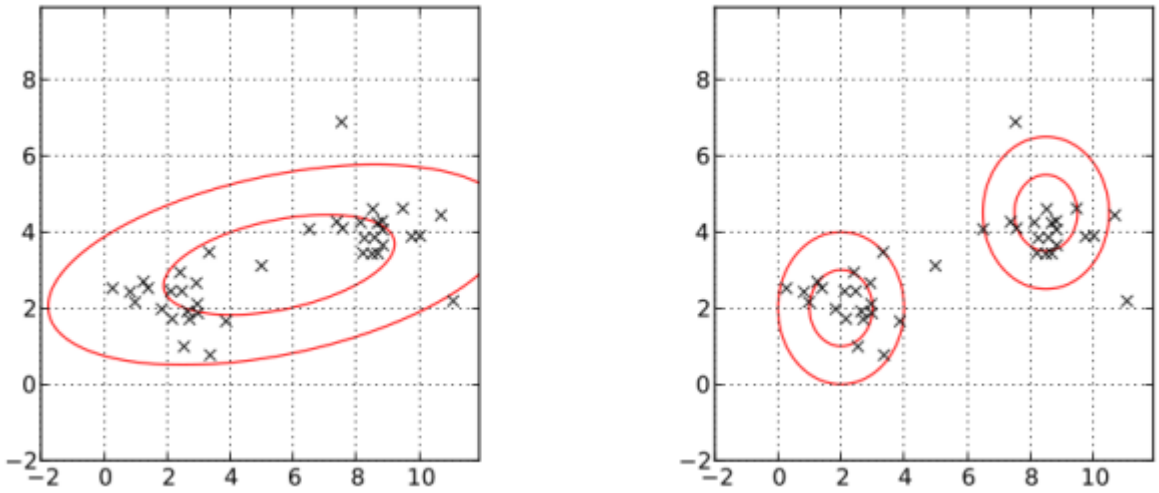


Fig. 20: < (Left) Fit with one Gaussian distribution (Right) Fit with Gaussian mixture model with two components[1] >

One-dimensional Model

$$p(x) = \sum_{i=1}^{K} \phi_i \mathcal{N}(x \mid \mu_i, \sigma_i) \tag{2.26}$$

$$\mathcal{N}(x \mid \mu_i, \sigma_i) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right) \tag{2.27}$$

$$\sum_{i=1}^{K} \phi_i = 1 \tag{2.28}$$

[1] https://brilliant.org/wiki/gaussian-mixture-model/

Multi-dimensional Model

$$p(\vec{x}) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i) \tag{2.29}$$

$$\mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^K |\Sigma_i|}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu}_i)^{\mathrm{T}} {\Sigma_i}^{-1} (\vec{x} - \vec{\mu}_i)\right) \tag{2.30}$$

$$\sum_{i=1}^{K} \phi_i = 1 \tag{2.31}$$

EM algorithm

EM solves a chicken and egg problem,[2]

- Need $\left(\mu_{a'} \sigma_a^2\right)$ and $\left(\mu_{b'} \sigma_b^2\right)$ to guess source of points
- Need to know the sources to estimate $\left(\mu_{a'} \sigma_a^2\right)$ and $\left(\mu_{b'} \sigma_b^2\right)$

Idea

1. Start with two randomly placed Gaussians $\left(\mu_{a'} \sigma_a^2\right)$ and $\left(\mu_{b'} \sigma_b^2\right)$
2. (E-step) for each point $P\left(b|x_i\right)$ does it look like it came from b?
3. (M-step) adjust $\left(\mu_{a'} \sigma_a^2\right)$ and $\left(\mu_{b'} \sigma_b^2\right)$ to fit points assigned to them
4. Iterate until convergence

Reference

2.2.7 Practices

- *Q. Conditional independence from Bayesian network*
- *Q. Burden of specification*
  - *a.*
  - *b.*
  - *c.*
- *Q. DAG representation*
- *Q. Derivation of the posterior distribution*
- *Q. Multivariate Gaussian*
  - *part a*
  - *Part-b*
- *Q3. Wishart distribution*

[2] https://www.youtube.com/watch?v=REypj2sy_5U

Q. Conditional independence from Bayesian network

Based on the Bayesian network in Figure 2, which of the following conditional independence statements follow? For each statement, give a "true/false" answer; for the false statements, also mention a path between the two nodes that is not blocked. (see Barber, 2012, ch. 3.3.4)

(a) $A \perp\!\!\!\perp B \mid C$ **(c)** $C \perp\!\!\!\perp E \mid B, D$ **(e)** $B \perp\!\!\!\perp F \mid A, C$

(b) $A \perp\!\!\!\perp B \mid \emptyset$ **(d)** $C \perp\!\!\!\perp D \mid A, B$ **(f)** $A \perp\!\!\!\perp E \mid D, F$

Furthermore, find a Bayesian network that is *Markov equivalent* to the network in Figure 2. (see Barber, 2012, ch. 3.3.6)

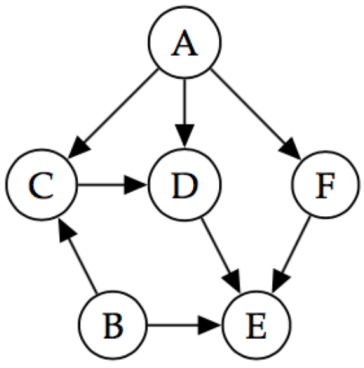


Fig. 21: < Source: Aalto course CS-E4820: Advanced probabilistic methods >

- $A \perp\!\!\!\perp B|C$
  - False. Collider C is in the path ACB and is in the conditioning set.
- $A \perp\!\!\!\perp B|\emptyset$

  - True
- $C \perp\!\!\!\perp E|B, D$
  - False. Collider D is in the path CDAFE and is in the conditioning set.
- $C \perp\!\!\!\perp D|A, B$
  - False. Path C – D is not blocked. Therfore it is not conditionally dependent.
- $B \perp\!\!\!\perp F|A, C$
  - True
- $A \perp\!\!\!\perp E|D, F$
  - False. In path ACBE, there is a collider C, and its descendent D is in the conditioning set.

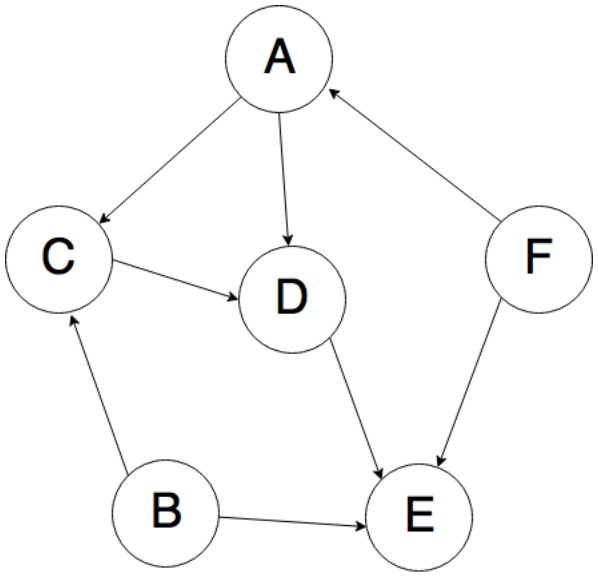


Fig. 22: < Markov Equivalent Graph >

Q. Burden of specification

a.

In total there are $2^5 = 32$ combinations of parameters i.e., states. The last probability could be defined by $1 - sum(31states)$. Thus the distribution could be defined by 31 combinations of parameters.

b.

Let's have alphabets instead of x_i.

$$p(a, b, c, d, e) = p(a)p(b|a)p(c|b)p(d|c)p(e|d)$$

The distribution needs 2+2+2+2+2 - 1 = 9 parameters. -1 is for the fact that you can deduce from the assumption the distribution sums to 1 and you can get the last probability by subtracting the rest from 1.

Consider a distribution of five binary variables x_i.

(a) What is the number of parameters needed to define the distribution $p(x_1, x_2, x_3, x_4, x_5)$ if no assumptions are made, i.e. p is an arbitrary distribution.

(b) How about if the Bayesian network in Figure 3 is assumed, i.e. p factorizes as implied by the graph.

(c) And how about if, additionally to (b), we assume that the conditional distributions are shared, i.e. $p(x_{i+1} \mid x_i) = p(x_i \mid x_{i-1}), i = 2, 3, 4$?

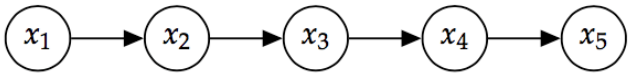


Fig. 23: < Source: Aalto course CS-E4820: Advanced probabilistic methods >

c.

$$p(a, b, c, d, e) = p(a)p(b|a)p(b|a)p(b|a)p(b|a)$$

The distribution needs 2+2 - 1 = 3 parameters.

Q. DAG representation

Let's have the following notation:

Notation	Explanation
$A = 1$	A person has brain cancer
$B = 1$	A person has a high blood calcium level
$C = 1$	A person has a brain tumor
$D = 1$	A person has seizures that cause unconsciousness
$E = 1$	A person has severe headaches

An expert have told us the following information about the relationships between variables:

Fig. 24: < Source: Aalto course CS-E4820: Advanced probabilistic methods >

Probability of severe headaches P(E = 1) depends only on the fact whether a person has a brain tumor (C) or not. On the other hand, if one knows the blood calcium level (B) and whether the person has a tumor or not (C), one can specify the probability of unconsciousness seizures P(D = 1). In this case, the probability of D doesn't depend on the presence of the headaches (E) or (directly) on the fact whether the person has brain cancer or not (A). The probability of a brain tumor (C) depends directly only on the fact, whether the person has brain cancer or not (A).

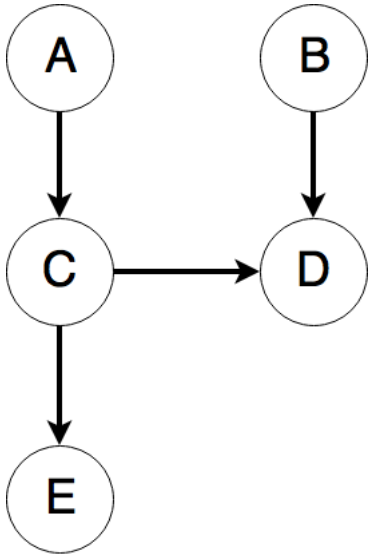


Fig. 25: < DAG representation >

Q. Derivation of the posterior distribution

- http://halweb.uc3m.es/esp/Personal/personas/mwiper/docencia/English/PhD_Bayesian_Statistics/ch3_2009.pdf
- https://www.youtube.com/watch?v=0XD6C_MQXXE

Q. Multivariate Gaussian

Reference: https://learnbayes.org/index.php?option=com_content&view=article&id=77%3Acompletesquare&catid=83&Itemid=479&showall=1&limitstart=

part a

We have a model,

$$to$$

$$\boldsymbol{x}_i \overset{iid}{\sim} \text{Multivariate Normal}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\boldsymbol{\mu} \sim \text{Multivariate Normal}_p(\boldsymbol{m}_0, \boldsymbol{S}_0) \quad (2.32)$$

Multivariate Normal$_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})\boldsymbol{\mu}$
Multivariate Normal$_p(\boldsymbol{m}_0, \boldsymbol{S}_0)$

Here's the general Multivariate Gaussian Distrubution(MVN):

$$p(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{p}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}$$

Our posterior is proportional to the product of the likelihood and prior.

$$p(\mu \mid \boldsymbol{X}) \propto \text{likelihood} \times \text{prior} \tag{2.32}$$

$$\propto -\frac{1}{2}\left(\boldsymbol{\mu}^T\left(N\boldsymbol{\Sigma}^{-1} + \boldsymbol{S}_0^{-1}\right)\boldsymbol{\mu} - \boldsymbol{\mu}^T\left(N\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{x}} + \boldsymbol{S}_0^{-1}\boldsymbol{m}_0\right) - \left(N\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{x}} + \boldsymbol{S}_0^{-1}\boldsymbol{m}_0\right)^T\boldsymbol{\mu}\right) \tag{2.33}$$

where, $\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}$. In order to interpret the above as the general form we need to substitute some terms. Let $\boldsymbol{A} = N\boldsymbol{\Sigma}^{-1} + \boldsymbol{S}_0^{-1}$ and let $\boldsymbol{b} = N\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{x}} + \boldsymbol{S}_0^{-1}\boldsymbol{m}_0$.

$$p(\mu \mid \boldsymbol{X}) \propto -\frac{1}{2}\left(\boldsymbol{\mu}^T\boldsymbol{A}\boldsymbol{\mu} - \boldsymbol{\mu}^T\boldsymbol{b} - \boldsymbol{b}^T\boldsymbol{\mu}\right).$$

In order to complete the square we can add some helper term that is not dependent on μ:

$$p(\mu \mid \boldsymbol{X}) \propto -\frac{1}{2}\left(\boldsymbol{\mu}^T\boldsymbol{A}\boldsymbol{\mu} - \boldsymbol{\mu}^T\boldsymbol{b} - \boldsymbol{b}^T\boldsymbol{\mu} + \boldsymbol{b}^T\boldsymbol{A}^{-1}\boldsymbol{b}\right)$$

Remember that A is symmetric – it's a weighted sum of symmetric matrices – and invertible – it's a sum of full-rank covariance matrices. Hence, the above is rewritten as

$$p(\mu \mid \boldsymbol{X}) \propto -\frac{1}{2}\left(\boldsymbol{\mu}^T\boldsymbol{A}\boldsymbol{\mu} - \boldsymbol{\mu}^T\boldsymbol{A}\boldsymbol{A}^{-1}\boldsymbol{b} - \boldsymbol{b}^T\boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b}^T\boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{A}^{-1}\boldsymbol{b}\right)$$

We introduce new helper terms in order to complete the square. Let $\boldsymbol{\Sigma}_n = \boldsymbol{A}^{-1}$ and $\boldsymbol{\mu}_n = \boldsymbol{A}^{-1}\boldsymbol{b}$. The above is rewritten as

$$p(\mu \mid \boldsymbol{X}) \propto -\frac{1}{2}\left(\boldsymbol{\mu}^T\boldsymbol{\Sigma}_n^{-1}\boldsymbol{\mu} - \boldsymbol{\mu}^T\boldsymbol{\Sigma}_n^{-1}\boldsymbol{\mu}_n - \boldsymbol{\mu}_n^T\boldsymbol{\Sigma}_n^{-1}\boldsymbol{\mu} + \boldsymbol{\mu}_n^T\boldsymbol{\Sigma}_n^{-1}\boldsymbol{\mu}_n\right)$$

Organize the terms

$$p(\mu \mid \boldsymbol{X}) \propto -\frac{1}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}_n)^T\boldsymbol{\Sigma}_n^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}_n)$$

So here is the posterior distribution

$$\boldsymbol{\mu} \mid \boldsymbol{X} \sim \text{Multivariate Normal}_p(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n),$$

where

$$to$$

$$\boldsymbol{\Sigma}_n = \boldsymbol{A}^{-1} = \left(N\boldsymbol{\Sigma}^{-1} + \boldsymbol{S}_0^{-1}\right)^{-1},$$

$$\boldsymbol{\mu}_n = \boldsymbol{A}^{-1}\boldsymbol{b} = \boldsymbol{\Sigma}_n\left(N\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{x}} + \boldsymbol{S}_0^{-1}\boldsymbol{m}_0\right) \tag{2.34}$$

=
$=\left(N\boldsymbol{\Sigma}^{-1}+\boldsymbol{S}_0^{-1}\right)^{-1}, \boldsymbol{\mu}_n=\boldsymbol{A}^{-1}\boldsymbol{b}$
$\boldsymbol{\Sigma}_n\left(N\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{x}}+\boldsymbol{S}_0^{-1}\boldsymbol{m}_0\right)$

Part-b

Here's code:

```
m_0 = np.array([0,0]).T
s_0 = np.array([0.1,0,0,0.1]).reshape(2,2)

mu = np.array([0.0,0.0])
sigma = np.array([1.0,0,0,1]).reshape(2,2)

def mle():
  t = np.array([0.0,0])
  for _ in range(10):
      t += np.random.multivariate_normal(u, var)
  return t / 10

def inv(m):
  return np.linalg.inv(m)

N = 10
x_mean = mle()
sigma_n = inv( N* inv(sigma) + inv(s_0) )
mu_n = sigma_n.dot( N* inv(sigma).dot(x_mean) + inv(s_0).dot(m_0) )
print(x_mean)    # [-0.20467891  0.24346118]
print(mu_n)      # [-0.10233945  0.12173059]
```

The Bayesian estimate is half of MLE hence, it is closer to the tru value.

Q3. Wishart distribution

part a

- Mean: νW
- Variance: $Var(\Lambda_{ij}) = \nu(V_{ij}^2 + V_{ii}V_{jj})$

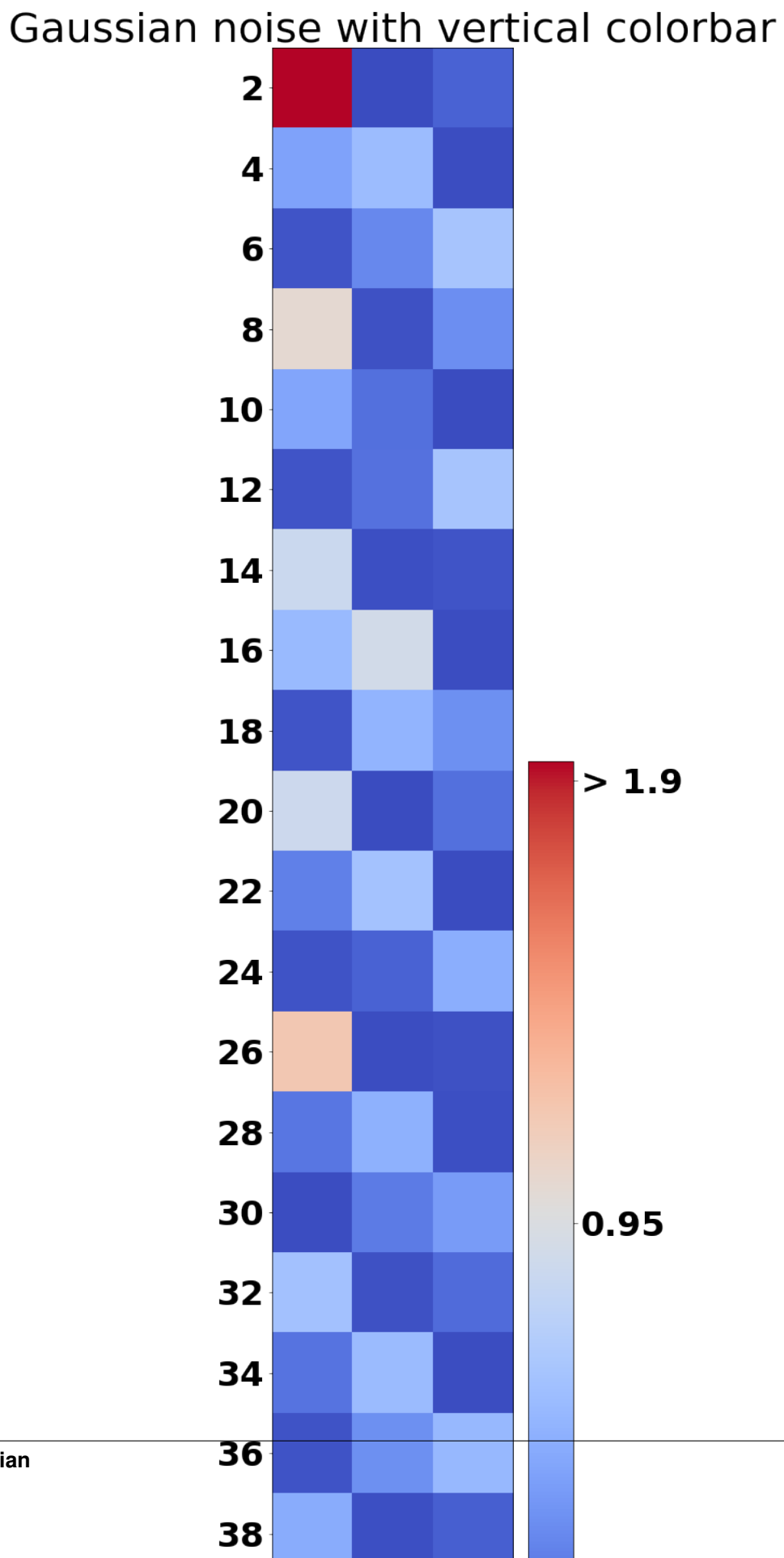
part b&c

As we are given a mean A and we know that mean is equal to νW, we only need to set on parameter, the degree-of-freedom. We run grid search for degree-of-freedom and sample_size.

Below is a grid search plot I ran for parameter tuning for sample_sizes = [1, 10, 1000] and degree-of-freedom = range(2,60,2). For this example I found the degree-of-freedom 52 and sample size 1000 is the most optimal. However, it's not straightforward which parameters are the best. For instance, degree-of-freedom 2 seems to be really bad for the sample size 1 but it is okay for the sample size 10. So the decision for parameters are left to a stakeholder.

CODE

```
A = np.array([2,0.3,0.3,0.5]).reshape(2,2)

sample_sizes = [1,10,1000]
distances = []
def run_wishart(df):
    for ss in sample_sizes:
        V = A/df
        samples = wishart.rvs(df, V , ss)
        if ss == 1:
            distances.append( np.linalg.norm(A - samples))
        else:
            distances.append( np.linalg.norm(A - samples.mean(axis=0)))

    return samples.mean(axis=0)

for df in range(2,60,2):
    l = run_wishart(df)

data = np.array(distances).reshape(3, 29).T; data

from matplotlib import cm
from numpy.random import randn

font = {'family' : 'normal',
        'weight' : 'bold',
        'size'   : 32}

import matplotlib
matplotlib.rc('font', **font)

fig, ax = plt.subplots(figsize = (5, 55))


# data = np.clip(randn(10, 10), -1, 1)

cax = ax.imshow(data, interpolation='nearest', cmap=cm.coolwarm)
ax.set_title('Gaussian noise with vertical colorbar')
ax.set_xticks(range(3))
ax.set_xticklabels(sample_sizes)
ax.set_yticks(range(29))
ax.set_yticklabels(list(range(2,60,2)))



# Add colorbar, make sure to specify tick locations to match desired ticklabels
cbar = fig.colorbar(cax, ticks=[0.02, 0.95, 1.9])
cbar.ax.set_yticklabels(['< 0.02', '0.95', '> 1.9'])  # vertically oriented colorbar

plt.show()
```


Gaussian noise with vertical colorbar
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
> 1.9
0.95

Q. Posterior of regression weights

We have

- prior $p(w|\alpha)$
- likelihood $p(y|\mathbf{X}, w, \beta)$
- posterior $p(w|y, \mathbf{X}, \alpha, \beta)$

By Bayesian we can get posterior by multiplying the prior and liklihood. Here we want to just derive the posterior thus we ignore constants. Also for multivariate gaussian distribution, it is easier to work with logarithms. Thus,

$$\log p(w|y, \mathbf{X}, \alpha, \beta) = \log p(w|\alpha) + \log p(y|\mathbf{X}, w, \beta)$$

$$\propto \frac{-1}{2} w^T (\alpha^{-1} I)^{-1} w - \frac{1}{2}(y - \mathbf{X}w)^T (\beta^{-1} I)^{-1} (y - \mathbf{X}w)$$

$$\propto \frac{-\alpha}{2} w^T w - \frac{\beta}{2}\left[y^T y - 2w^T \mathbf{X}^T y + w^T \mathbf{X}^T \mathbf{X} w\right]$$

$$\propto \frac{-1}{2} w^T \left[\alpha + \beta \mathbf{X}^T \mathbf{X}\right] w + \beta w^T \mathbf{X}^T y$$

In *WEEK3 problem 2* we derived the logarithm of the multivariate normal distribution $x|\mu \sim \mathcal{N}(\mu, \Sigma)$

$$\propto \frac{-1}{2} x^T \Sigma^{-1} x + x^T \Sigma^{-1} \mu$$

If we compare this to what we have derived above we see the same pattern i.e., we have derived the posterior.

$$\mathbf{S} = (\alpha + \beta \mathbf{X}^T \mathbf{X})^{-1} \tag{2.34}$$

$$\mathbf{m} = \mathbf{S}(\mathbf{S}^{-1}\mathbf{m}) \tag{2.35}$$

$$= \beta \mathbf{S} \mathbf{X}^T y \tag{2.36}$$

Q. Poisson regression with Laplace approximation

(a)

Poisson maximum likelihood is as follows

$$p(y_i|\theta) = \prod_{i=1}^{N} \frac{exp(y_i \theta^T x_i) exp(-e^{\theta^T x_i})}{y_i!}$$

Then,

$$\log p(\theta|y) = \log p(y|\theta) + \log p(\theta) \tag{2.37}$$

$$= \sum_{i=1}^{N} \left[y_i \theta^T x_i - e^{\theta^T x_i})\right] - \frac{\alpha}{2}\theta^T \theta + \text{constant} \tag{2.38}$$

Now get the gradient

$$\nabla \log p(\theta|y) = \sum_{i=1}^{N} \left[y_i x_i - e^{\theta^T x_i} x_i\right] - \alpha\theta$$

Now get the Hessian

$$\nabla\left(\nabla \log p(\theta|y)\right)^T = \sum_{i=1}^{N} \left[- e^{\theta^T x_i} x_i x_i^T\right] - \alpha I$$

(b)

In general,

$$p(w|\alpha, \mathcal{D}) \propto exp(-E(w)), \quad E(w) = -\log p(w|\alpha, \mathcal{D})$$

Let $E(\theta) = -\log p(\theta|y)$.

Apply Laplace approximation

$$\widetilde{E}(\theta) \approx E(\bar{\theta}) + \frac{1}{2}(\theta - \bar{\theta})^T H_{\bar{\theta}}(\theta - \bar{\theta})$$

$\bar{\theta}$ is the minimum of $E(\theta)$.

The mean is $\bar{\theta}$ and the covariance $H_{\bar{\theta}}$.

(c)

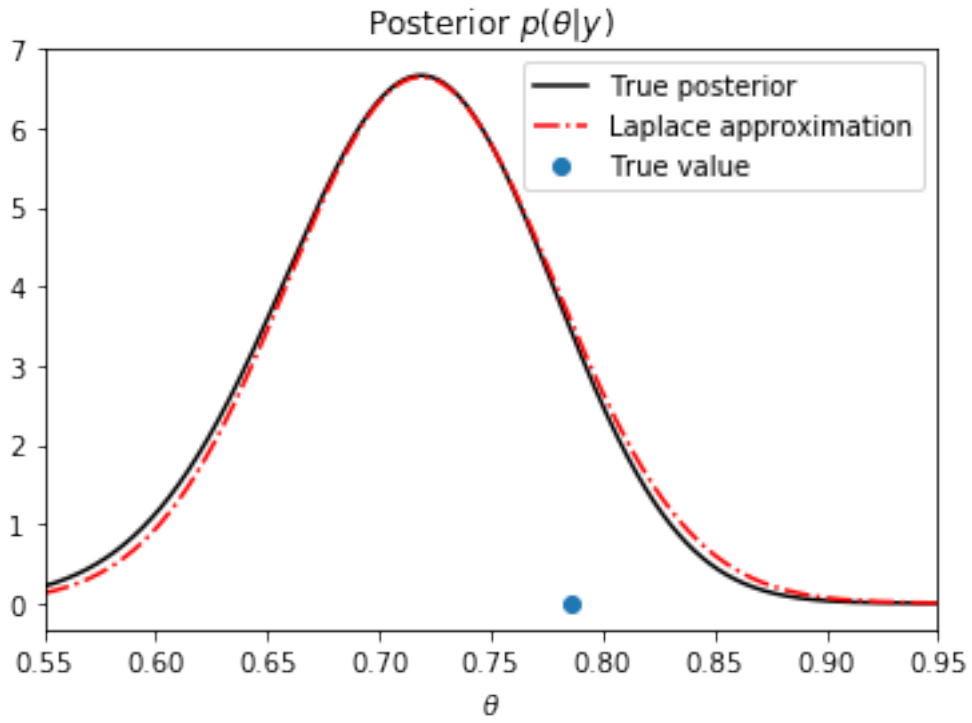


Fig. 27: < Laplace approximation vs. true posterior >

```
# ML: Advanced Probabilistic Methods
# Round 4, problem 4.

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# get some data
```

(continues on next page)

(continued from previous page)

```
data = np.loadtxt('ex4_4_data.txt')
x = data[:,0]
y = data[:,1]

theta_true = np.pi / 4 # true parameter used to generate the data
alpha = 1e-2 # prior's parameter

# compute Laplace approximation
theta_lapl = 0.5 # initial

# compute Laplace approximation
theta_lapl = 0.5 # initial

# iterate to optimum with newton's method to find the MAP estimate for theta
for iter in range(100):
    # compute gradient
    grad = -np.dot(np.exp(theta_lapl * x), x) + np.dot(x, y) - alpha * theta_lapl
    # compute Hessian
    H = - alpha - np.dot(np.exp(theta_lapl * x).T, x**2)
    theta_lapl = theta_lapl - grad / H # do newton step

# compute Hessian at optimum
H = -np.dot(np.exp(theta_lapl * x), x**2) - alpha

difference = theta_lapl - theta_true

# plot posterior densities
theta = np.linspace(0.55, 0.95, 1000)
post_true = np.zeros(1000)
for i in range(len(theta)):
    # log posterior:
    from scipy.misc import factorial
    post_true[i] = (np.dot(y, x * theta[i]) - np.sum(np.exp(x * theta[i]) -
                    np.log(factorial(y))) - 0.5*alpha*np.dot(theta[i], theta[i]))

M = np.amax(post_true)
post_true = np.exp(post_true-M) / np.sum(np.exp(post_true-M)) / (theta[1]-theta[0]) #␣
↪normalize

post_laplace = norm.pdf(theta, theta_lapl, np.sqrt(-1/H))
               # compute approximative density at the points 'theta'
               # Hint: you can use norm.pdf from scipy.stats

plt.figure(1)
plt.plot(theta, post_true, '-k', label="True posterior")
plt.plot(theta, post_laplace, '-.r', label="Laplace approximation")
plt.plot(theta_true, 0, 'o', label="True value")
plt.xlim(0.55, 0.95)
plt.xlabel('$\\theta$')
plt.title('Posterior $p(\\theta|y)$')
plt.legend()

plt.figure(2)
plt.plot(x, y, 'o', x, np.exp(theta_lapl*x), '-r')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Data')
```

(continues on next page)

(continued from previous page)

```
plt.show()
```

Q. Variational approximation for a simple distribution

Problem 4. *"Variational approximation for a simple distribution."*

Consider a model with two binary random variables x_1 and x_2, defined by the distributions:

$p(x_1)$	
x_1=0	0.4
x_1=1	0.6

$p(x_2 \mid x_1)$	x_1=0	x_1=1
x_2=0	0.5	0.9
x_2=1	0.5	0.1

Find a fully factorized distribution $q(x_1, x_2) = q_1(x_1)q_2(x_2)$ that best approximates the joint $p(x_1, x_2)$, in the sense of minimizing $\mathrm{KL}\,(p \parallel q)$.

1. Get the joint distribution $p\,(x_1, x_2)$

Use the rule: $p\,(x_1, x_2) = p\,(x_2|x_1)\,p\,(x_1)$

$p(x_1, x_2)$	$x_1 = 0$	$x_1 = 1$
$x_2 = 0$	0.2	0.54
$x_2 = 1$	0.2	0.06

2. Make the similar probability distribution table for the approximation $q\,(x_1, x_2)$

Suppose $q\,(x_1 = 1) = a$ and $q\,(x_2 = 1) = b$. Then we get

$q(x_1, x_2)$	$x_1 = 0$	$x_1 = 1$
$x_2 = 0$	$(1-a)(1-b)$	$a(1-b)$
$x_2 = 1$	$(1-a)b$	ab

3. Get KL-divergence $KL(p\|q)$.

$$\begin{aligned}
\mathrm{KL}(p\|\|q) &= \sum_x p(x) \log \frac{p(x)}{q(x)} && (2.39)\\
&= -\sum_x p(x) \log \frac{q(x)}{p(x)} && (2.40)\\
&= -0.2 \log \frac{(1-a)(1-b)}{0.2} - 0.2 \log \frac{(1-a)b}{0.2} - 0.54 \log \frac{a(1-b)}{0.54} - 0.06 \log \frac{ab}{0.06} && (2.41)\\
&= -0.4 \log(1-a) - 0.74 \log(1-b) - 0.26 \log b - 0.6 \log a + \text{constant} && (2.42)\\
& && (2.43)
\end{aligned}$$

4. Minimize $KL(p\|q)$ by partial derivatives being equal to 0.

> The Hessian matrix of a convex function is positive semi-definite. Refining this property allows us to test if a critical point x is a local maximum, local minimum, or a saddle point.

Here is the expression of the determinant,

$$D(x,y) = \det(H(x,y)) = f_{xx}(x,y) f_{yy}(x,y) - (f_{xy}(x,y))^2$$

And we know that if $D(a,b) > 0$ and $f_{xx}(a,b) > 0$ then (a,b) is a local minimum of f.

Here is a plot of the derivatives

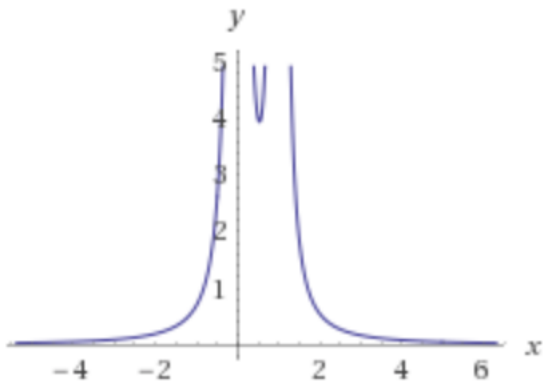


Fig. 28: < plots for $f_{xx}(x,y)$ and $f_{yy}(x,y)$ look pretty much identical and they are positve values. >

And we can see that $f_{xy}(x,y) = 0$. There for the determinant is $D(x,y)$ is positive and $f_{xx}(a,b) > 0$. Hence, the critical point is a local minimum.

$$\begin{aligned}
\frac{\partial KL(p||q)}{\partial a} = 0 \Rightarrow -\frac{0.6}{a} + \frac{0.4}{1-a} = 0 \Rightarrow a = 0.6 && (2.44)\\
\frac{\partial KL(p||q)}{\partial b} = 0 \Rightarrow -\frac{0.26}{b} + \frac{0.74}{1-b} = 0 \Rightarrow b = 0.26 && (2.45)
\end{aligned}$$

Done.

References

2.2.8 References

1. http://statistics.zone

2.3 Calculus

2.3.1 Fundamentals

Why are we using e?[1]

e is a special number. The derivative of exponential e^t is always e^t. Let's look at non-e exponentail 2^t and its derivative.

$$\frac{\delta}{\delta t}2^t = \lim_{\delta t \to 0} \frac{2^{t+\delta t} - 2^t}{\delta t} \tag{2.46}$$

$$= 2^t \Big(\lim_{\delta t \to 0} \frac{2^{\delta t} - 1}{\delta t} \tag{2.47}$$

```
t= .1
for i in range(1, 10):
    dt = t ** i
    d = (2**(dt) - 1 )/(dt)
    print(dt, d)

        # 0.1 0.7177346253629313
        # 0.010000000000000002 0.6955550056718883
        # 0.0010000000000000002 0.6933874625807411
        # 0.00010000000000000002 0.6931712037649972
        # 1.0000000000000003e-05 0.6931495828199628
        # 1.0000000000000004e-06 0.6931474207938491
        # 1.0000000000000004e-07 0.6931472040783146
        # 1.0000000000000005e-08 0.6931471840943001
        # 1.0000000000000005e-09 0.6931470952764581
```

So we see that as δt becomes finitely small, $\frac{\delta}{\delta t}2^t$ converges to 0.693. It's good to know that it converges but it would be handy if we can find a pattern and that's where e comes in!

Let's rewrite 2^t

$$2^t = e^{t \log 2} \quad \text{(as } e \text{ and 2 can be exchanged)} \tag{2.48}$$

$$\frac{\delta}{\delta t}2^t = \frac{\delta}{\delta t}(e^{t \log 2}) \tag{2.49}$$

$$= \log 2 e^{t \log 2} \quad (\text{as } \frac{\delta}{\delta t}e^{at} = \tag{2.50}$$

$$(\log 2 = 0.693147180559 \tag{2.51}$$

So we got the same results by using e and it is more universal as we can easily plug any constants and get the derivative using logarithm!

[1] https://youtu.be/m2MIpDrF7Es

Matrix Differentiation

Let's demonstrate matrix differentiation with the following example.

$$f(w) = (1/|X|)\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

I will divide the two terms into separate expressions f_1 and f_2.

$$\begin{aligned} f_1(w) &= (1/|X|)\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \\ f_2(w) &= \lambda\|\mathbf{w}\|_2^2 \end{aligned} \tag{2.52}$$

To get $\nabla f_1(w)$ I will use the chain rule.

$$\begin{aligned} h(w) &= \mathbf{y} - \mathbf{X}\mathbf{w} \\ f_1(w) &= (1/|X|)\|h\|_2^2 \end{aligned} \tag{2.54}$$

(2.56)

Now let's get the derivative! Remember that the norm sign with double 2 on the right means the square of euclidean distance.

$$\begin{aligned} \delta f_1(w) &= (1/|X|)2h\delta h \\ \delta h &= -\mathbf{X}\delta\mathbf{w} \\ \delta f_1(w) &= (-2/|X|)\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) \end{aligned} \tag{2.57}$$

(2.60)

The transpose for $\mathbf{X}^T$ is for matrix calculation. Solve $\nabla f_2(w)$

$$\delta f_2(w) = 2\lambda\mathbf{w}\delta\mathbf{w} \tag{2.61}$$

Finally add them together

$$\begin{aligned} \frac{\delta f(w)}{\delta w} &= \frac{\delta f_1(w)}{\delta w} + \frac{\delta f_2(w)}{\delta w} \\ &= \frac{\delta f_1}{\delta h}\frac{\delta h}{\delta w} + \frac{\delta f_2(w)}{\delta w} \\ &= (\frac{-2}{|X|})\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + 2\lambda\mathbf{w} \end{aligned} \tag{2.62}$$

(2.63)

(2.64)

Matrix expression example 1

We have a function

$$f(w) = w^T\mathbf{A}w + \mathbf{b}^T w + c$$

Given $\mathbf{A} = (1/N)\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$, $\mathbf{b} = -(2/N)\mathbf{X}^T\mathbf{y}$ and $c = (1/N)y^T y$, complete the function.

$$\begin{aligned} f(w) &= w^T\big((1/N)\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\big)w + \big(-(2/N)\mathbf{X}^T\mathbf{y}\big)^T w + (1/N)y^T y \\ &= \frac{1}{N}w^T\mathbf{X}^T\mathbf{X}w + w^T\lambda w + \frac{-2}{N}\mathbf{y}^T\mathbf{X}w + \frac{1}{N}y^T y \end{aligned} \tag{2.65}$$

(2.66)

Then we realize it is the expanded form of

$$f(w) = (1/|X|)\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

KL-divergence

A measure of how one probability distribution diverges from a second, expected probability distribution.[2]

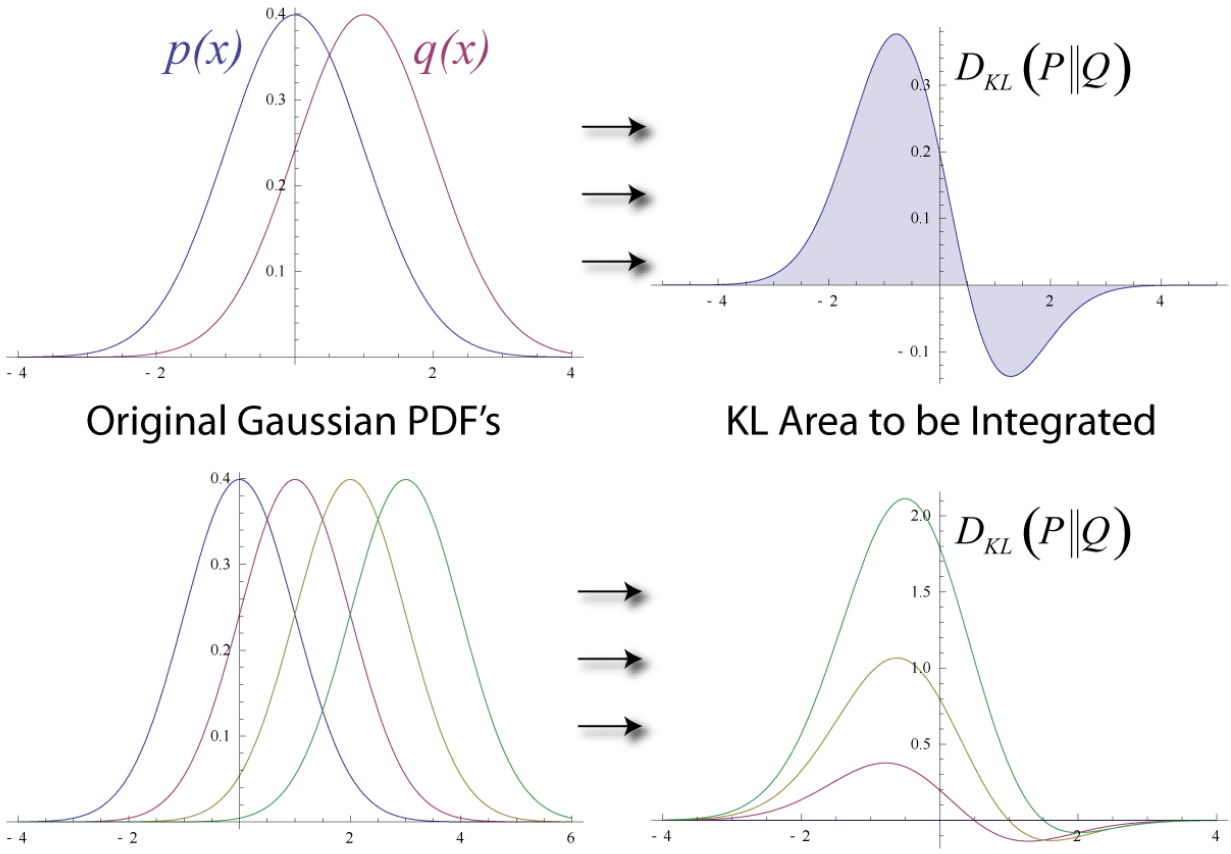


Fig. 29: < The Kullback-Leibler divergence for a normal Gaussian probability distribution. On the top left is an example of two Gaussian PDF's and to the right of that is the area which when integrated gives the KL metric. >

Here's another example,

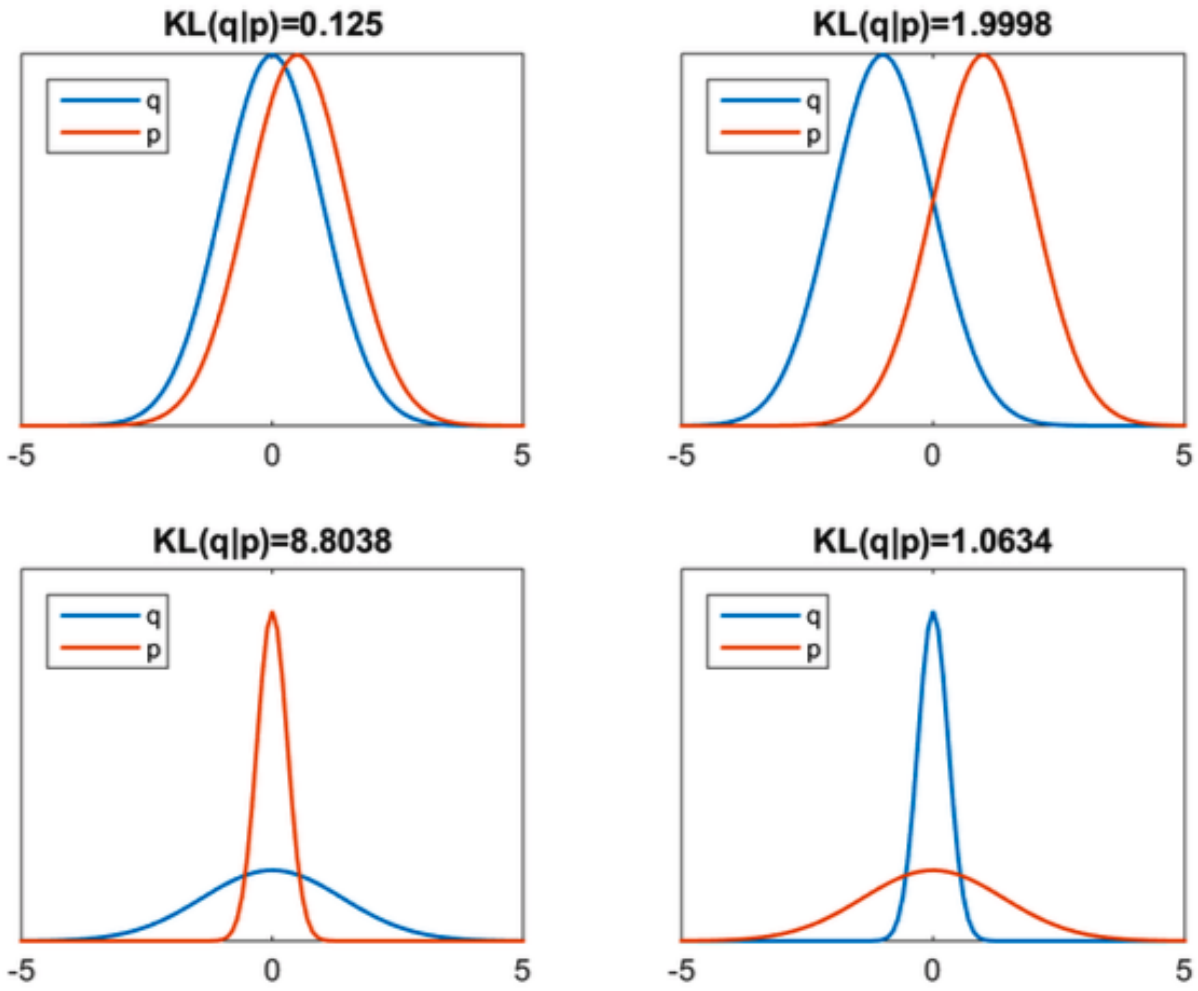
Reference

2.3.2 Partial derivatives

Second partial derivative test

It is used to determine if a critical point of a function is a local minimum, maximum or saddle point.[1]

[2] https://en.wikipedia.org/wiki/Kullback–Leibler_divergence#/media/File:KL-Gauss-Example.png
[1] https://en.wikipedia.org/wiki/Second_partial_derivative_test


KL(q|p)=0.125
q
p
-5
0
5
KL(q|p)=1.9998
q
p
-5
0
5
KL(q|p)=8.8038
q
p
-5
0
5
KL(q|p)=1.0634
q
p
-5
0
5

References

2.3.3 Series

Taylor series

- Coursera video: How Do Taylor Series Provide Intuition For Limits?

References

2.4 Computer Graphics

2.4.1 Linear Algebra

2.5 Deep Learning

2.5.1 Fundamentals

Intro

$\Theta^{(j)}$: matrix of weights controlling function mapping from layer j to layer $j+1$.

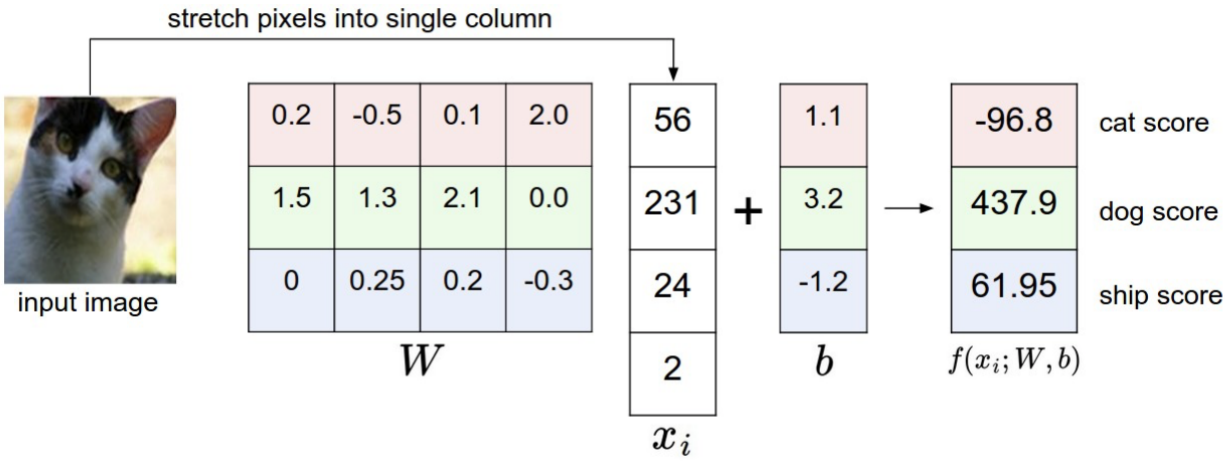


Fig. 30: < Representation of a linear classifier neural network. Source: cs231n >

The values for each of the *activation* nodes is obtained as follows:

$$\text{3x4 matrix} \begin{cases} a_1^{(2)} & = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3) \\ a_2^{(2)} & = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_2 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3) \\ a_3^{(2)} & = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_3 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3) \end{cases}$$

$$h_\Theta(x) = a_1^{(3)}$$
$$a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

As you can see each layer gets its own matrix of weights, $\Theta^{(j)}$.

If a network has S_j units in layer *j* and S_{j+1} units in layer *j+1*, then $\Theta^{(j)}$ will be of dimension $S_{j+1} \times (S_j + 1)$. (1 is a bias node, x_0)

Score function

A function that takes an input and returns class scores. In linear classifier it would be $Wx + b$.

Cost/Loss Function

Loss function measures how good a given set of weights/parameters are compared to the ground truth labels in the training set. Popular loss functions are

- relu
- svm
- softmax
- sigmoid
- tanh

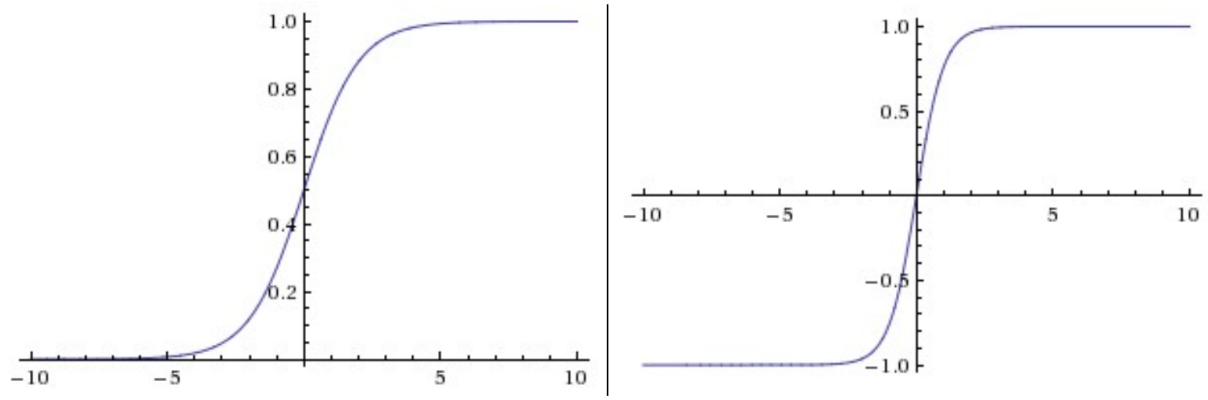


Fig. 31: < Left: Sigmoid non-linearity squashes real numbers to range between [0,1] Right: The tanh non-linearity squashes real numbers to range between [-1,1]. Source: Stanford cs231n >

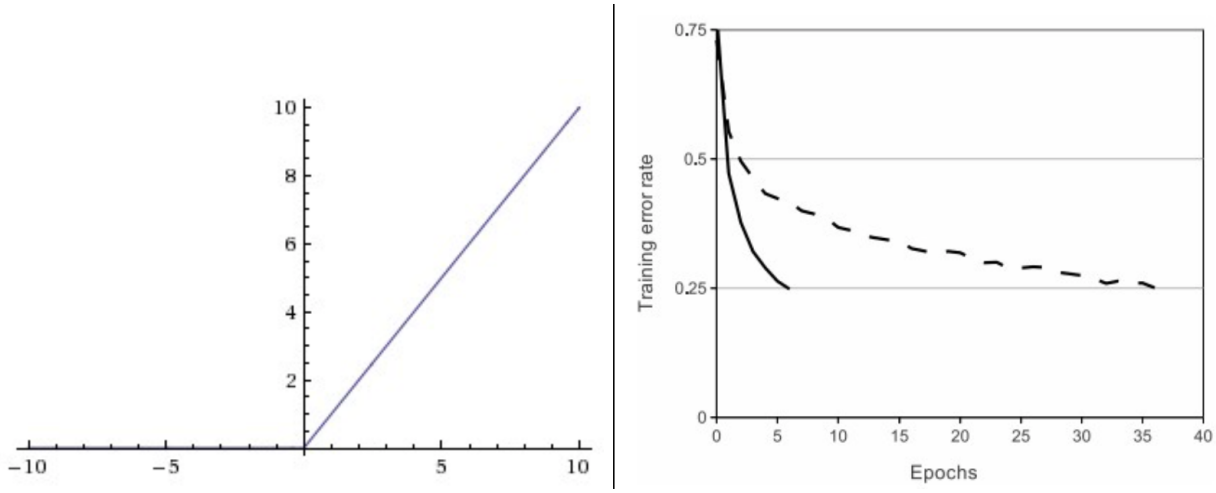


Fig. 32: < Left: Rectified Linear Unit (ReLU) activation function, which is zero when x < 0 and then linear with slope 1 when x > 0. Right: A plot from Krizhevsky et al. (pdf) paper indicating the 6x improvement in convergence with the ReLU unit compared to the tanh unit. >

Sigmoid: The sigmoid non-linearity has the mathematical form $\sigma(x) = 1/(1+e^{-x})$. It's not zero-centered so in gradient descent it may cause zig-zag effects.

tanh: tanh neuron is simply a scaled sigmoid neuron, in particular the following holds: $\tanh(x) = 2\sigma(2x) - 1$.

Maxout: The Maxout neuron computes the function $\max\left(w_1^T x + b_1, w_2^T x + b_2\right)$. It's a generalized form of ReLU and Leaky ReLU.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}[y_k^i \log((h_\theta(x^i))_k) + (1-y_k^i)\log(1-(h_\theta(x^i))_k)] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_{l+1}}(\theta_{j,i}^{(l)})^2$$

Here λ is regularized term.

Weight optimization

Practically it often works better to compute the numeric gradient using the centered difference formula: $[f(x+h) - f(x-h)]/2h$ (Wiki).

Gradient Descent

Here are pseudo-code for vanila GD and SGD.

```
# Source: http://cs231n.github.io/optimization-1/
# Vanilla Gradient Descent
while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```

```
# Vanilla Minibatch Gradient Descent
while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```

There two ways to find gradients

- numeric
- analytic

Numerical solution is simple but doens't give the exact solution but rather an approximation. Analytical solution is fast and give the exact solution but is error-prone as one could make a mistake during mathematical derivation. Therefore, in practice you'd use a gradient check(compare numerical and analytical solution).

Epochs, batches and iterations

- Epoch: A single through of an entire dataset
- Batch: A single dataset can be divided into batches.
- Iteration: A number of batches to complete an epoch.

$$\text{A number of dataset} = \text{Batch} \times \text{Iteration}$$

Back-propagation

It's a recursive application of a chain rule along a computational graph to compute the gradients of all parameters. It's contrary to stockastic gradient descent which is used to perform learning using the gradient. It's an algorithm that computes the chain rule of calculus , with a specific order of operations that is highly efficient *[Goodfellow-et-al]*. It modifies the connection weight parameters layer-by-layer starting from the output layer and progressing toward the input layer.

In a stanford lecture about backpropagation the TA shows that analytical gradient search could be represented as a computational graph.

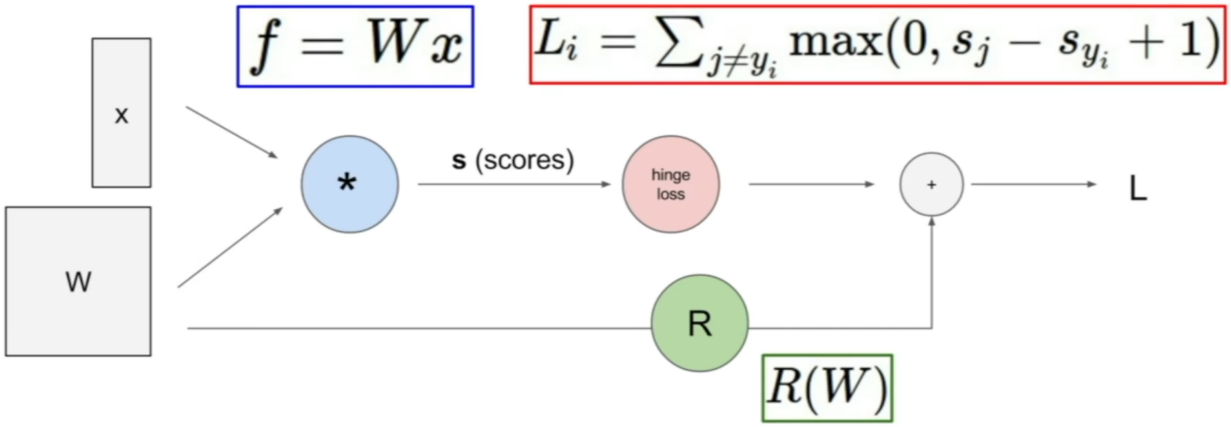


Fig. 33: < Source: Stanford cs231n >

Example

$$\frac{\delta f}{\delta x} = \frac{\delta q}{\delta x}\frac{\delta f}{\delta q} = 1 \times -4 = -4 \tag{2.67}$$

$$\frac{\delta f}{\delta y} = \frac{\delta q}{\delta y}\frac{\delta f}{\delta q} = 1 \times -4 = -4 \tag{2.68}$$

$$\frac{\delta f}{\delta z} = -2 + 0 = -2 \tag{2.69}$$

So what do we do with the **local gradients** in the computational graph? We send the upstream gradient going down and multiply it by the local gradients in order to get the gradient respect to the input.

Here's a bit more complicated example.

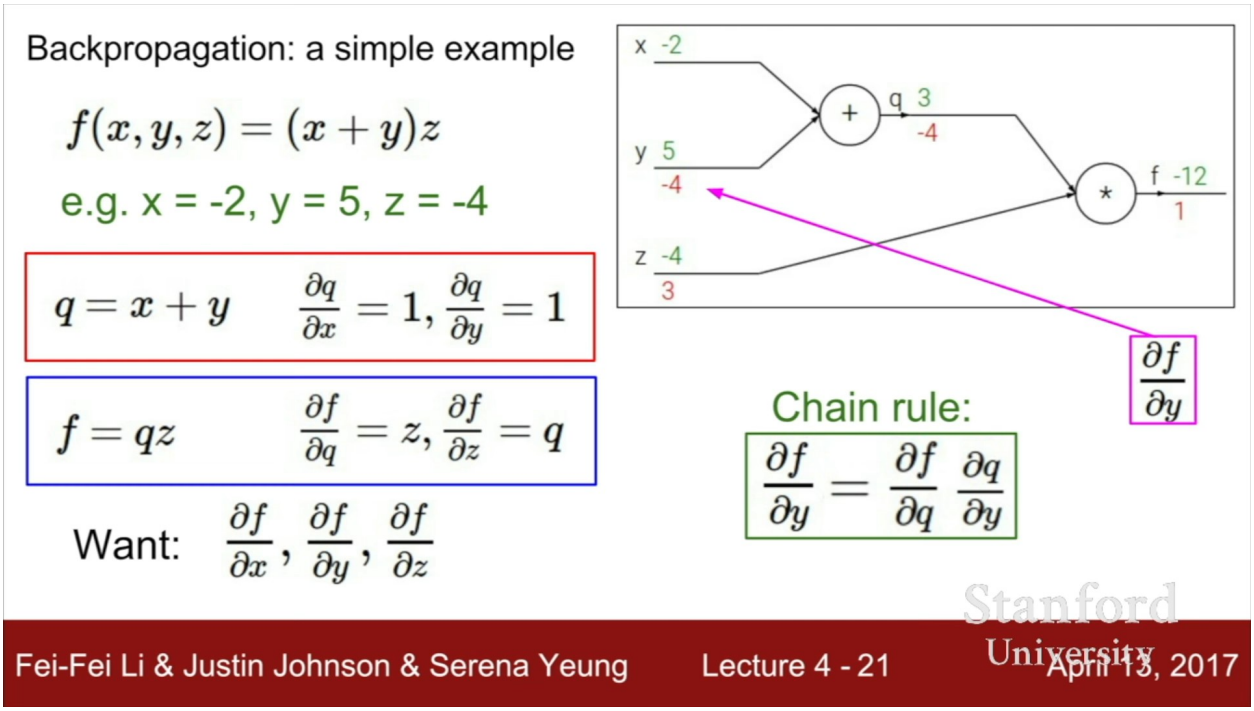
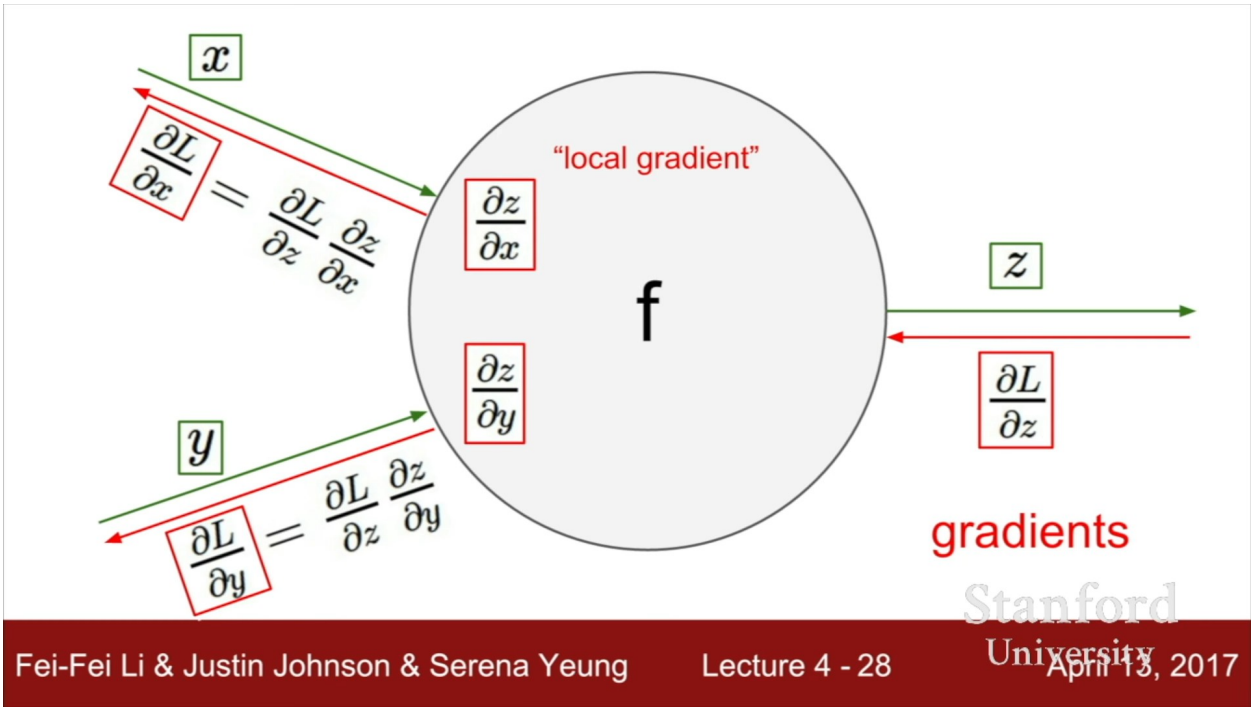


Fig. 34: < In the figure, the upper digits are the values of the nodes and the lower its gradient/derivative. The node values are filled by forward pass and the gradients by back propagation. Source: Stanford cs231n >



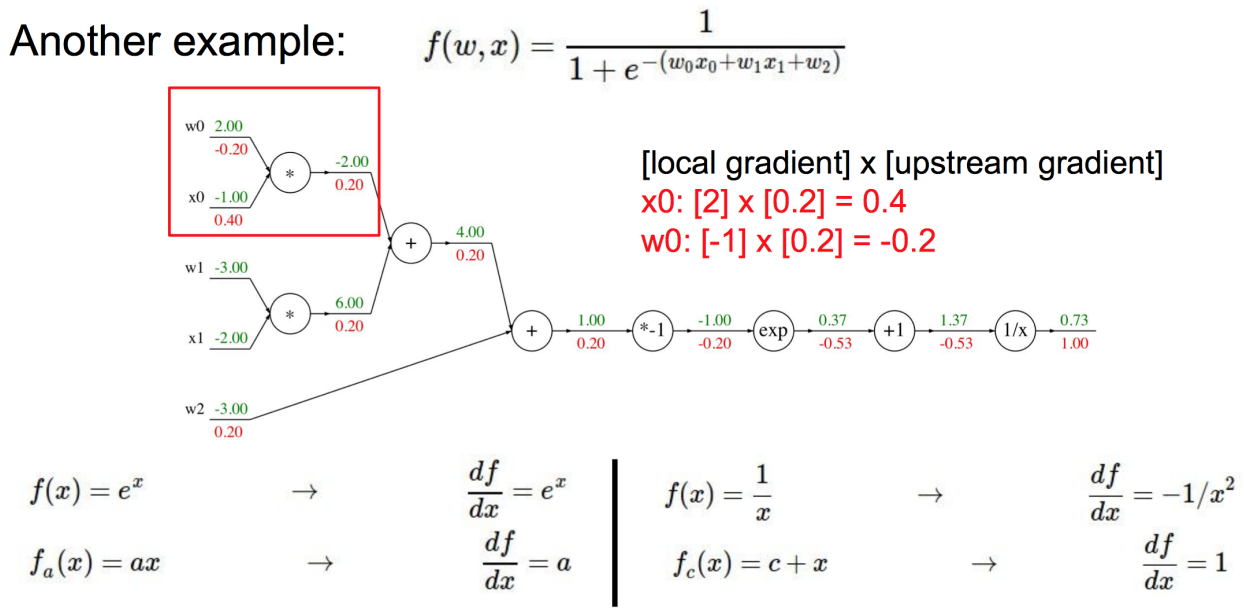


Fig. 35: < Source: Stanford cs231n >

I will show the back propagation step-by-step.

$$\frac{\delta f}{\delta f} = 1$$
$$\frac{\delta q}{\delta x} = \frac{\delta 1/x}{\delta x} = \frac{-1}{x^2} = \frac{-1}{1.37^2} = -0.53$$
$$\frac{\delta w}{\delta x} = \frac{\delta c + x}{\delta x} = 1,1 \times -0.53 = -0.53$$
$$\frac{\delta e}{\delta x} = \frac{\delta e^x}{\delta x} = e^x = e^{-1} = 0.37, 0.37 \times -0.53 = -0.2$$
$$\frac{\delta r}{\delta x} = \frac{\delta - x}{\delta x} = -1, -1 \times -0.2 = 0.2$$
$$\frac{\delta t}{\delta x} = \frac{\delta c + x}{\delta x} = 1,1 \times 0.2 = 0.2$$
$$\frac{\delta y}{\delta x} = \frac{\delta c + x}{\delta x} = 1,1 \times 0.2 = 0.2$$
$$\frac{\delta u}{\delta x} = \frac{\delta x_0 x}{\delta x} = x_0 = -1, -1 \times 0.2 = -0.2$$
$$\frac{\delta p}{\delta x} = \frac{\delta w_0 x}{\delta x} = w_0 = 2, 2 \times 0.2 = 0.4$$
$$\frac{\delta s}{\delta x} = \frac{\delta x_1 x}{\delta x} = x_1 = -2, -2 \times 0.2 = -0.4$$
$$\dots$$

However, there isn't only one way to draw a computational graph. One can decide the level of complexity like in the bottom, in which it substitutes a sigmoid gate with 4 nodes on the right.:

Patterns in back propagation

In the example you could observe a **pattern** in the back propagation. The **add** gate distributes gradients. The **mul** gate switches scaler and multiply it to the upstream gradient. So in the above example for w0 local gradient it is 0.2×-1.

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}} \qquad \boxed{\sigma(x) = \frac{1}{1+e^{-x}}} \quad \text{sigmoid function}$$

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\,\sigma(x)$$

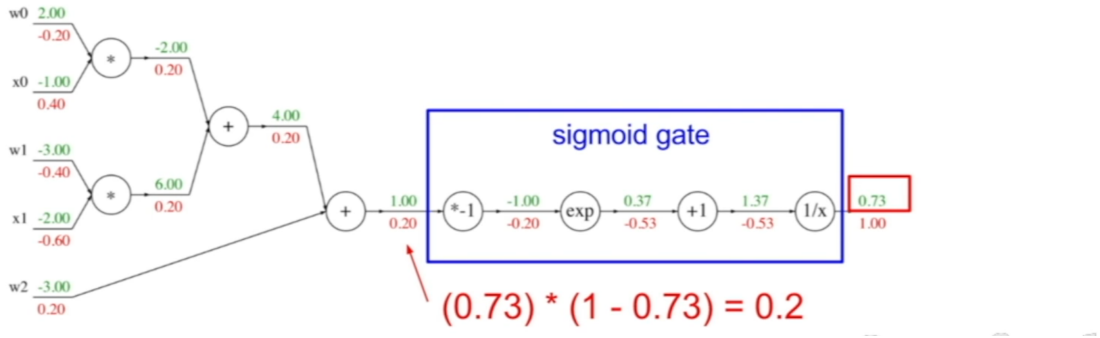


Fig. 36: < Source: Stanford cs231n >

max gate is interesting. It routes the gradient only to the max node.

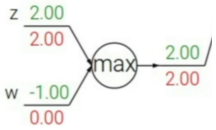


Fig. 37: < Source: Stanford cs231n >

So to summarize:

Gates	Rules
add gate	Gradient distributor
max gate	Gradient router
mul gate	Scaler switcher

Vectorized example

The idea is the same with scalar example. For instance, in order to get the gradient of W, you follow the **scaler switcher** rule.

```
np.array([0.2,0.4]).reshape(2,1).dot(np.array([0.44,.52]).reshape(1,2))
```

Activation functions

In NN, we use non-linear activation functions. This excellent Stackoverflow answer explains why we use non-linear activation functions.

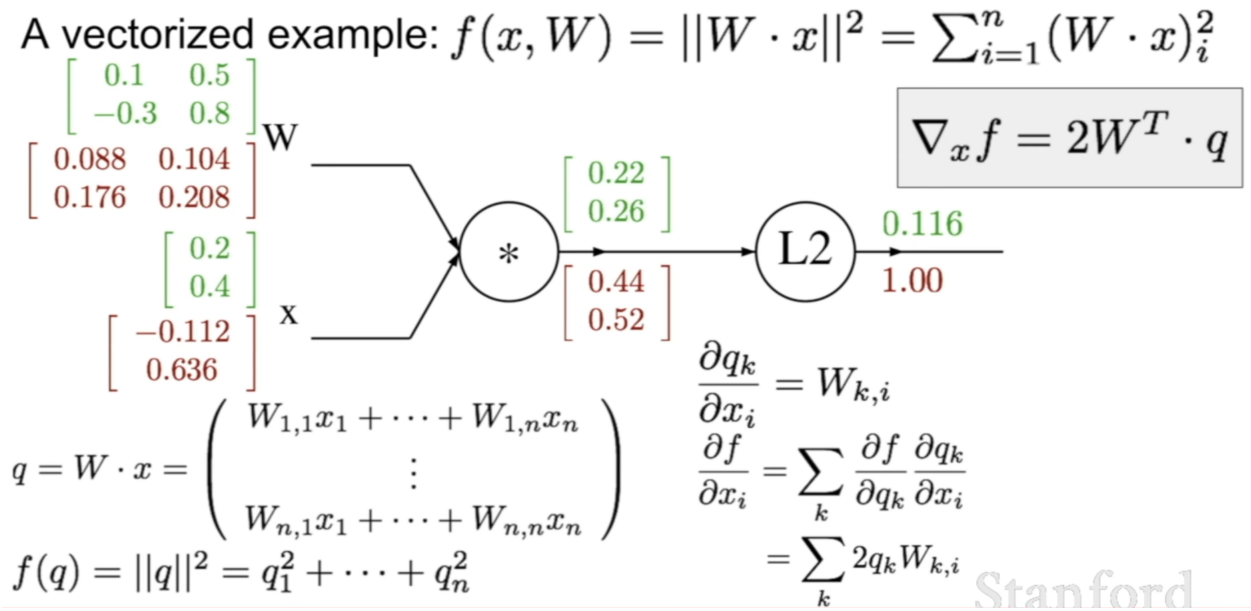


Fig. 38: < Source: Stanford cs231n >

The purpose of the activation function is to introduce **non-linearity into the network**.

In turn, this allows you to model a response variable (aka target variable, class label, or score) that varies non-linearly with its explanatory variables

non-linear means that the output cannot be reproduced from a linear combination of the inputs (which is not the same as output that renders to a straight line–the word for this is affine).

another way to think of it: without a non-linear activation function in the network, a NN, **no matter how many layers it had, would behave just like a single-layer perceptron**, because summing these layers would give you just another linear function (see definition just above).

Popular activation functions

Learning rate vs. Momentum

When performing gradient descent, **learning rate** measures how much the current situation affects the next step, while **momentum** measures how much past steps affect the next step. *[Quara-What-is-the-difference-between-momentum-and-learning-rate]*

Batch normalization

Adaptive reparameterization method

1. Normalize output from activation function.

$$z = \frac{x - m}{s}$$

2. Multiply normlized output by arbitrary parameter, g.

$$z * g$$

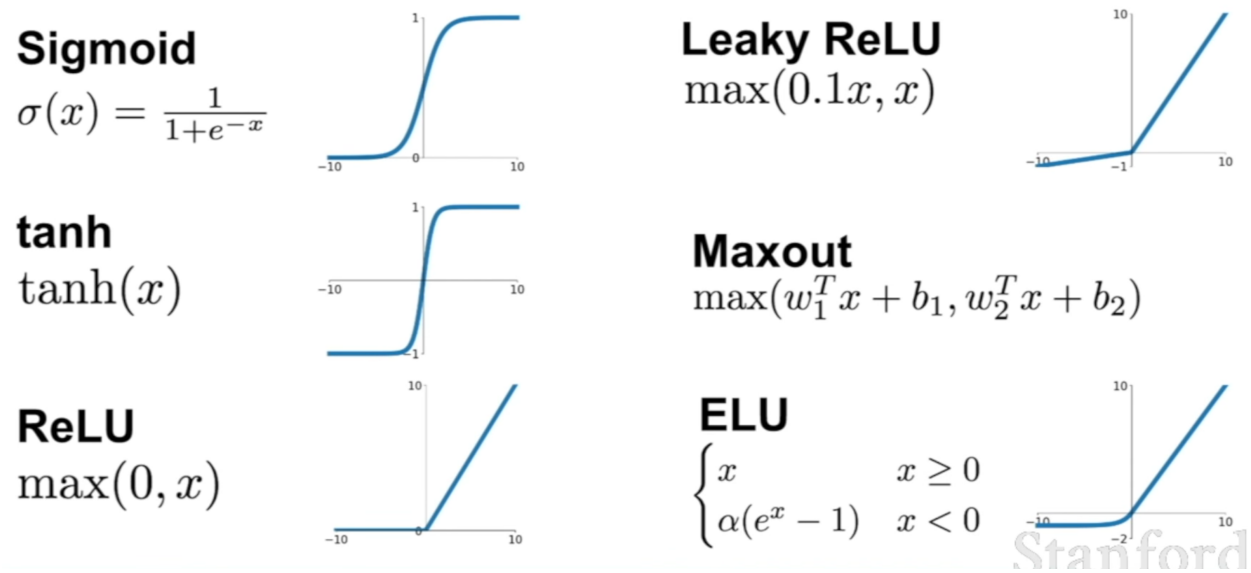


Fig. 39: < Source: Stanford cs231n >

Fig. 40: < Momentum and other gradient descent techiques visualized. Source >

3. Add arbitrary parameter, b, to resulting product

$$(z * g) + b$$

At test time

https://www.youtube.com/watch?v=5qefnAek8OA

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$
$$\sigma^2 = \frac{1}{m}\sum_i \left(z^{(i)} - \mu\right)^2$$
$$Z^{(i)}_{\text{norm}} = \frac{Z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$
$$\tilde{Z}^{(i)} = \gamma Z^{(i)}_{\text{norm}} + \beta$$

2.5.2 Gradient descent

http://ruder.io/optimizing-gradient-descent/index.html#gradientdescentoptimizationalgorithms

2.5.3 Regularization

- *Bagging*

Bagging

Bootstrap AGGregatING. Reduces generalization error by combining several models. Train several different models separately, then have all the models vote on the output for test examples. It's an example of **model averaging** or **ensemble methods**. Averaging works because different models will usually not make all the same errors on the test set. Boosting constructs an ensemble with higher capacity than the individual models.

Dropout

Technique for resolving overfitting in a large network. It randomly drop units(along with their connections) from the NN during training. *[Dropout_A_Simple_Way_to_Prevent_Neural_Networks_from_Overfitting]*

Dropout rate

Dropout rate p = 1, implies no dropout and low values of p mean more dropout. Typical values of p for hidden units are in the range 0.5 to 0.8. For input layers, the choice depends on the kind of input. For real-valued inputs (image patches or speech frames), a typical value is 0.8. For hidden layers, the choice of p is coupled with the choice of number of hidden units n. Smaller p requires big n which slows down the training and leads to underfitting. Large p may not produce enough dropout to prevent overfitting.

Learning rate and Momentum

Dropout introduces a lot of noise in the gradients compared to standard stochastic gradient descent. Therefore, a lot of gradients tend to cancel each other. You can use 10-100 times the learning rate to fix this. While momentum values of 0.9 are common for standard nets, with dropout 0.95-0.99 works well.

Max-norm Regularization

Large momentum and learning rate can cause the network weights to grow very large. Max-norm regularization constrains the norm of the vector of incoming weights at each hidden unit to be bound by a constant c in the interval of 3-4.

2.5.4 Convolutional Neural Network

Intro

CNN is robust for images compared to Regular Neural Nets because images are huge! A single image have millions of features and an image dataset can have millions of images as well.

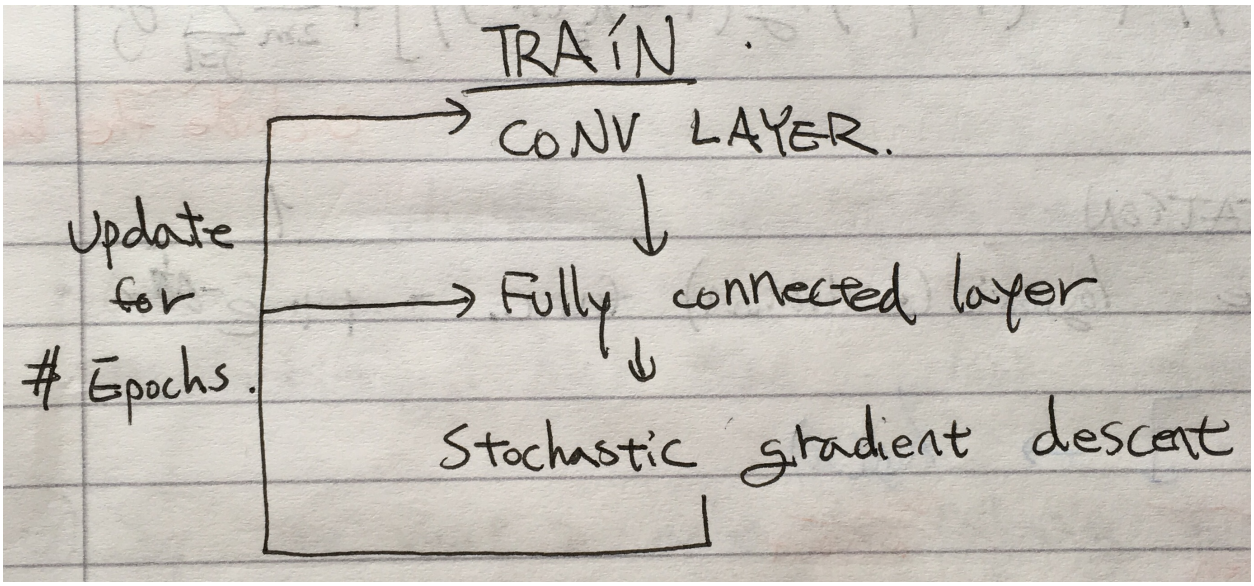


Fig. 41: < Mechamisom of CNN. >

Convolution operation

Convolution

Pooling

Data augmentation

It is a technique used in computer vision for increasing training data set based on the given data, in order to build a better model. Keras(ImageDataGenerator), Tensorflow(tflearn.data_augmentation.DataAugmentation) and mxnet(Augmenter) have ready-made implementations

Common techniques[1]

- Scaling
- Translation
- Rotation (at 90 degrees)
- Rotation (at finer angles)
- Flipping
- Adding Salt and Pepper noise
- Lighting condition
- Perspective transform

[1] https://medium.com/ymedialabs-innovation/data-augmentation-techniques-in-cnn-using-tensorflow-371ae43d5be9

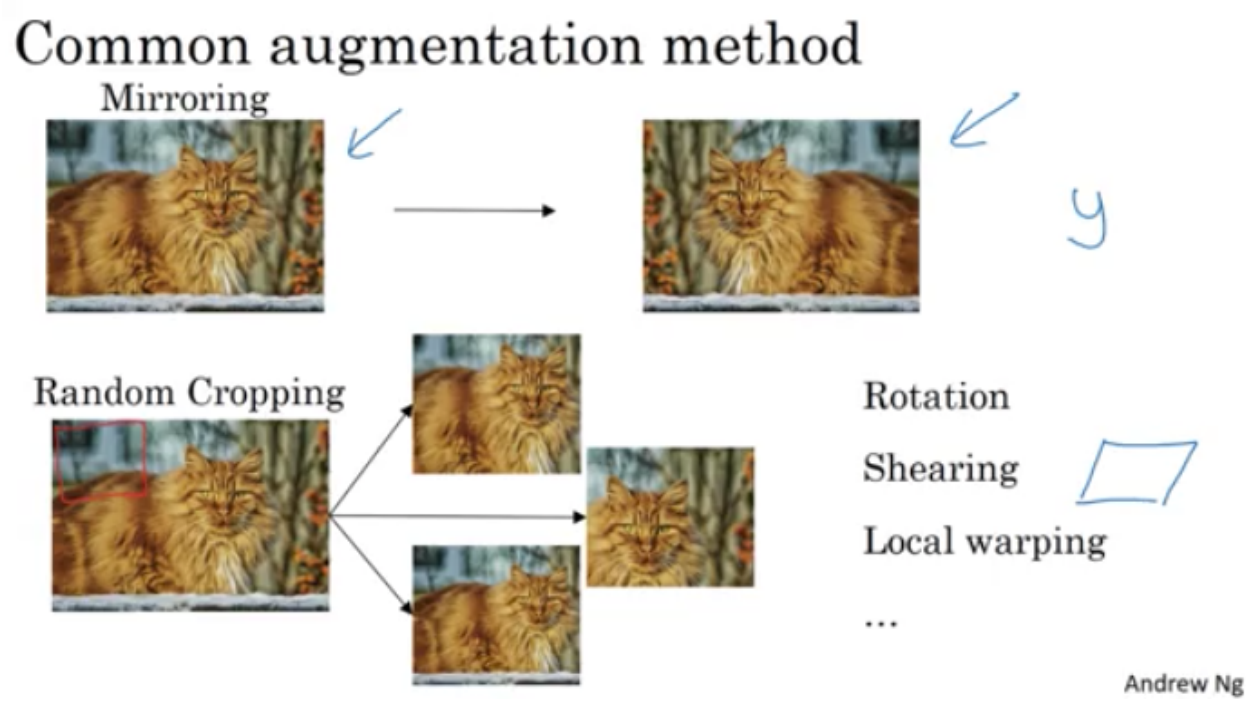


Fig. 42: < Source: *[Deep_Learning_Andrew_Ng]* >

Deconvolution

"A deconvnet can be thought of as a convnet model that uses the same components (filtering, pooling) but in reverse, so instead of mapping pixels to features does the opposite. In (Zeiler et al., 2011), deconvnets were proposed as a way of performing unsupervised learning. Here, they are not used in any learning capacity, just as a probe of an already trained convnet."[2]

In short, deconvolution layer is just a transposed convolutional layer.

Reference

2.5.5 Recurrent Neural Network

Note: This part of the documentation is largely referring to Aalto CS-E4890.

RNN is a specialized NN for processing sequential data $x^{(1)}, \cdots, x^{(\mathcal{T})}$. RNN employs parameter sharing just as CNN does. In CNN it is a kernel applied to a grid within images, and in RNN an n-gram sequence on sentences. RNNs are able to preserve information about the sequence order.

[2] https://github.com/YoungxHelsinki/papers/blob/4bc6eee3a68cb7da5277ff66ccefd8815a7f778d/papers/Visualizing%20and%20Understanding%20Convolutional%20Networks.pdf

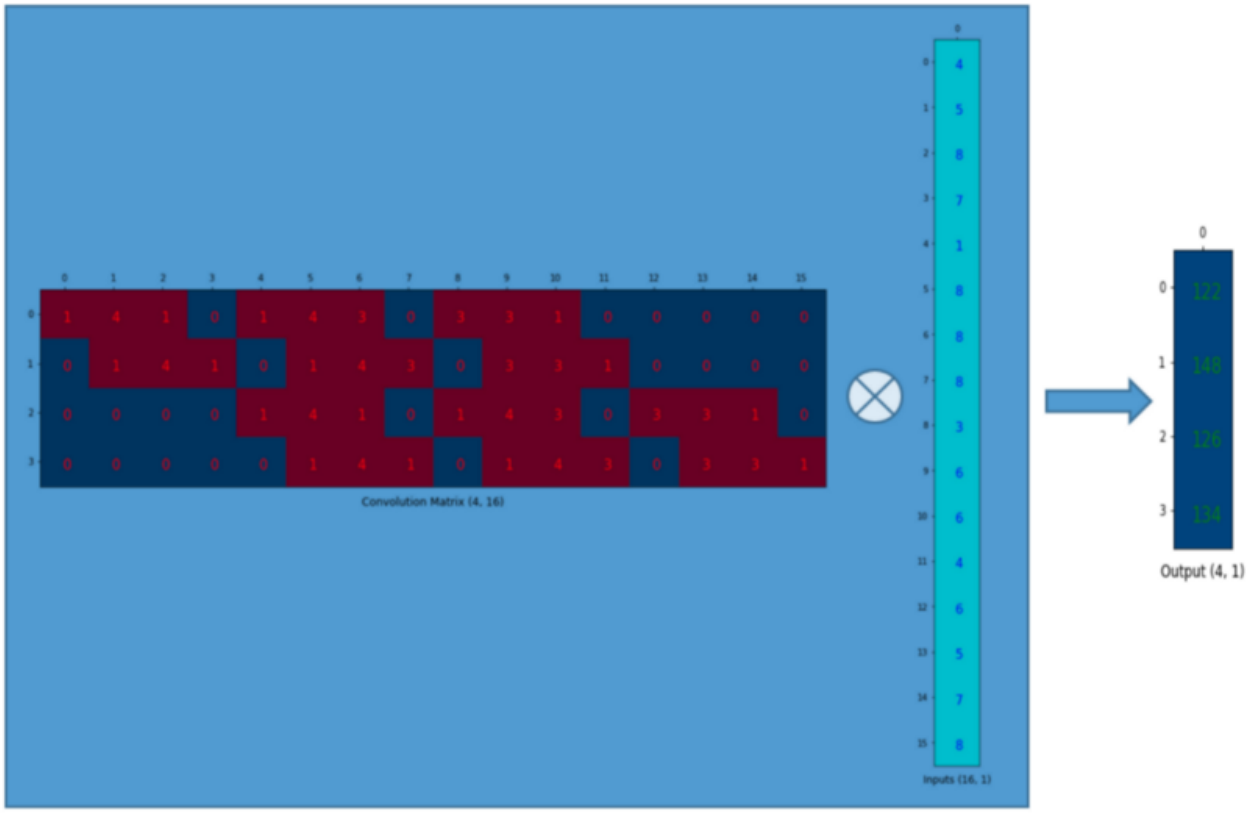


Fig. 43: < Convolution operation is a many-to-one relationship. Source: https://towardsdatascience.com/up-sampling-with-transposed-convolution-9ae4f2df52d0 >

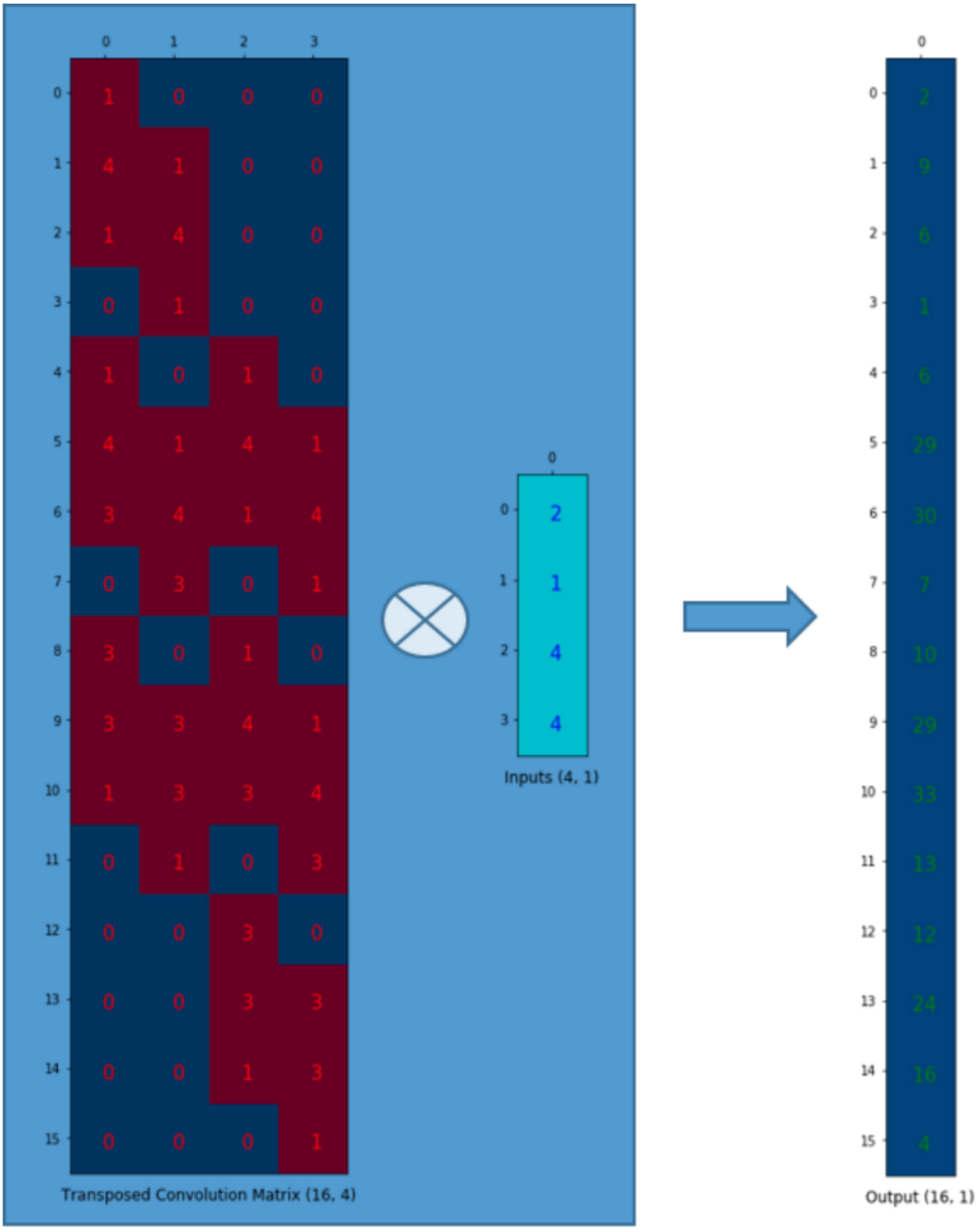


Fig. 44: < Deconvolution operation is a one-to-many relationship. Source: https://towardsdatascience.com/up-sampling-with-transposed-convolution-9ae4f2df52d0 >

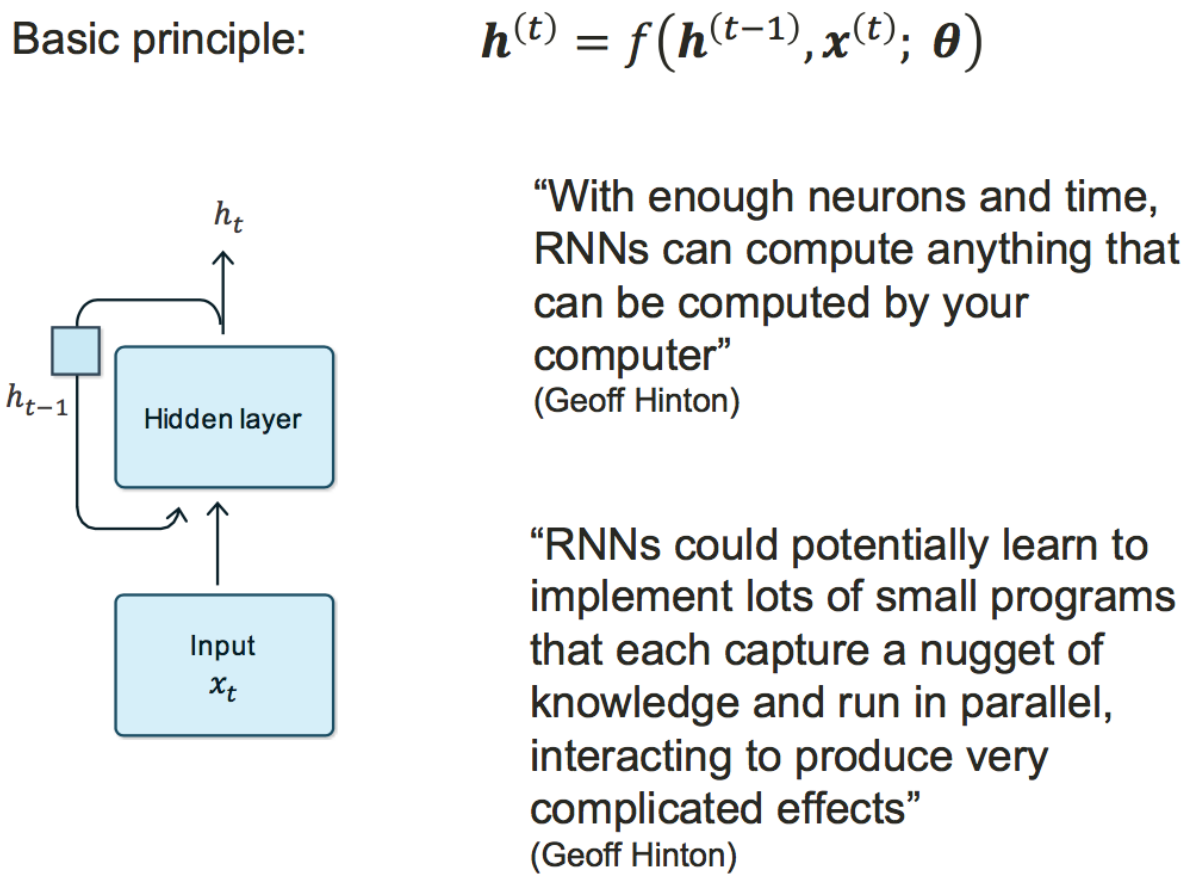


Fig. 45: < Source: Aalto CS-E4890 >

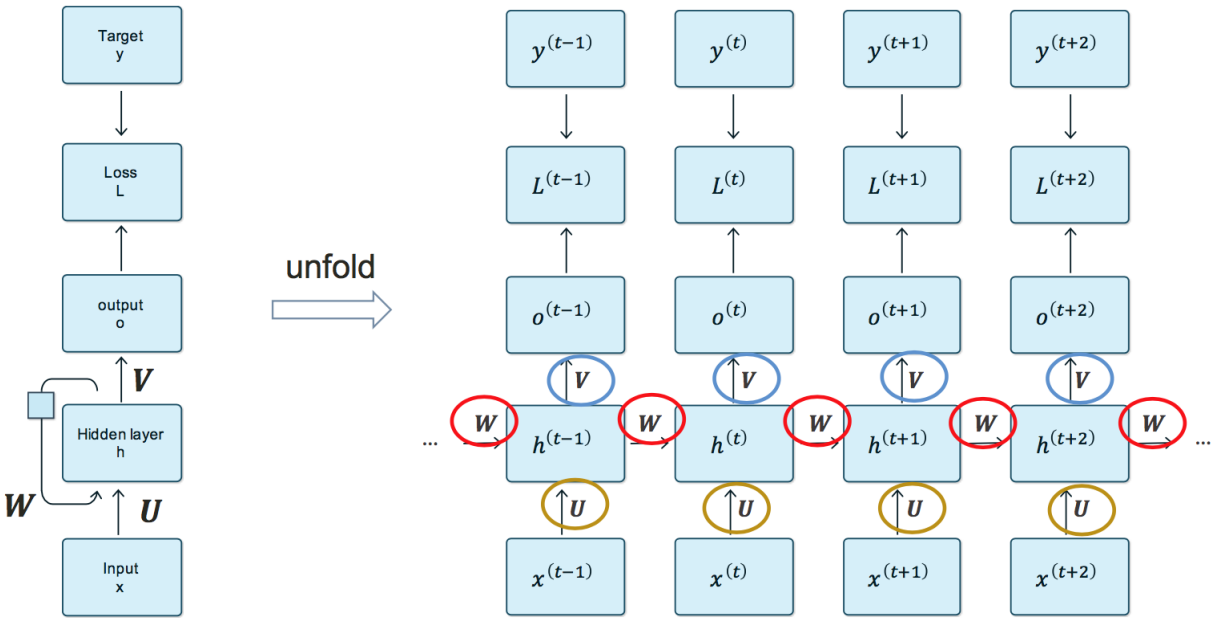


Fig. 46: < The unfolding the computational graph makes the RNN to resemble feedforward network structure. The back propagation is done normally. The parameters are shared over the training. Source: Aalto CS-E4890 >

Vanishing & exploding gradient problem

If a computatinal graph is deep and a shared parameters are repeatedly multiplied, as in RNN, there may be either a vanishing or exploding gradient problem. Here's a simple recurrence without input or activation function:

$$h^{(t)} = W^T h^{(t-1)}$$

This can be also presented as several multiplications by the same weight matrix

$$h^{(t)} = (W^T)^t h^{(0)}$$

W can be factorized as

$$\mathbf{W} = \mathcal{Q} * \mathcal{Q}^{\mathcal{T}}, \tag{2.70}$$

($\mathcal{Q}$: orthogonal matrix composed of eigenvectors of W,

Λ: diagonal matrix of eigenvalues)

We can thus conclude that:

$$h^{(t)} = \mathcal{Q} *^{\sqcup} \mathcal{Q}^{\mathcal{T}} h^{(0)}$$

Since the eigenvalues are raised to the power of t, the gradient will explode if the largest eigenvalue is >1, and vanish if the largest eigenvalue is < 1

You can solve this issue by clipping gradients; just clip the gradient if it is larger than the threshold. Clipping can be done element-wise or in vectorized way.

$$if \|g\| > v, \quad g \leftarrow \frac{gv}{\|g\|}$$

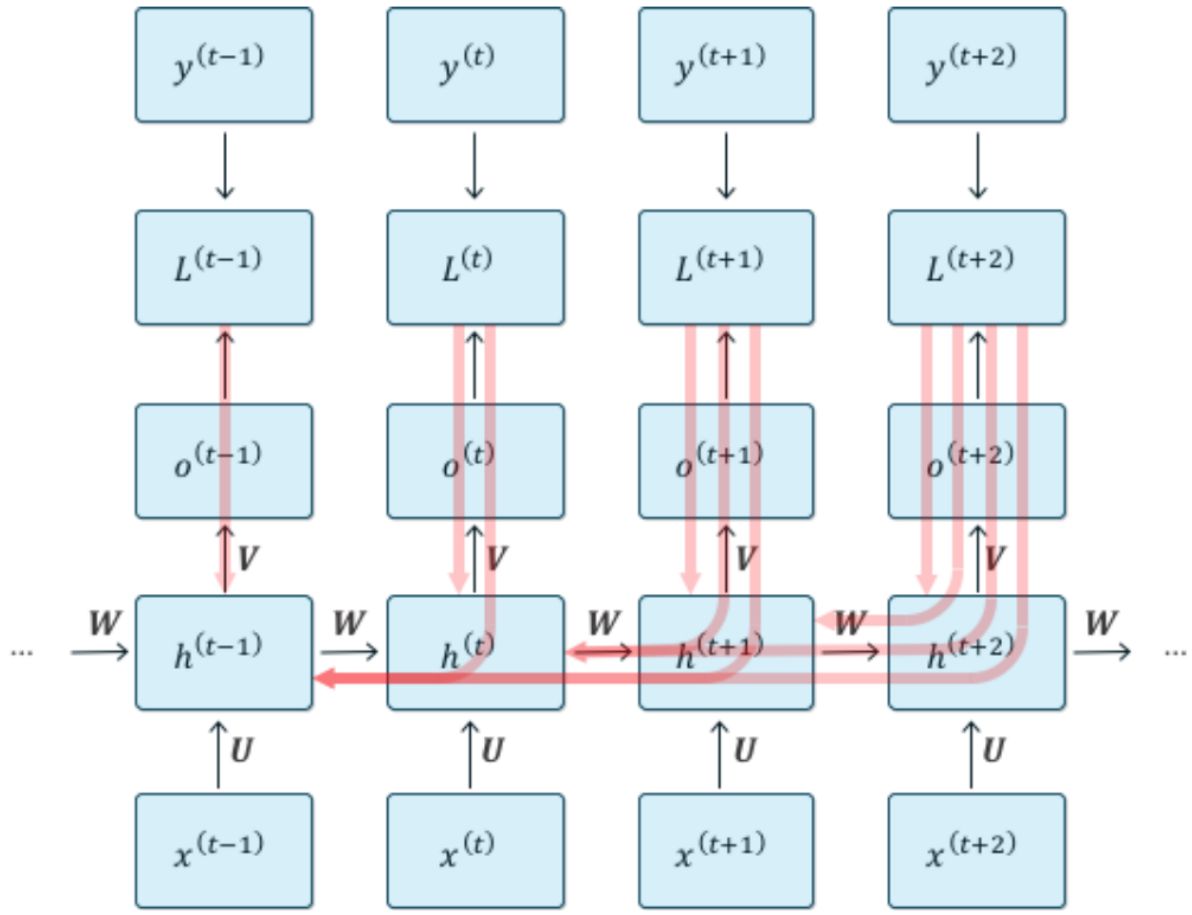


Fig. 47: < The back propagation through time is expensive as one has to calculate loss L from each time step to every hidden state i.e., $L = \sum_{t=1}^{T} L^{(t)}$ Source: Aalto CS-E4890 >

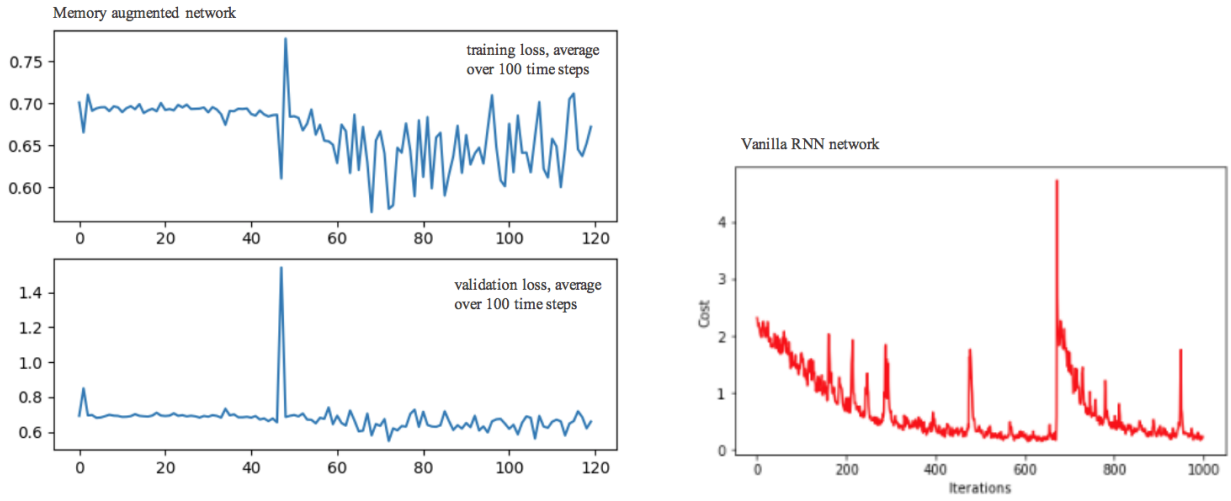


Fig. 48: < Training RNNs is unstable. Source: Aalto CS-E4890 >

2.5.6 Generative Adversarial Learning

GANs consist of two networks: generator and discriminator.

Autoencoder GANs <http://elarosca.net/slides/iccv_autoencoder_gans.pdf>_

It combines the reconstruction power of autoencoders with the sampling power of GANs.

Reference

2.5.7 Network Comparisons

Here, we try to compare different kinds of networks.

2.5.8 Exam

TERMs

Capacity

The capacity of a model is its ability to fit a wide variety of functions, often related to the number of free parameters. Too low capacity may lead to underfitting, too high to overfitting.

CNN

Neural networks with certain constraints enforced on the network topology, and parametrization; convolution may be used in the computations of at least one layer. A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers.

Jacobian matrix

A matrix of all first-order partial derivatives of a vector-valued function. When the matrix is a square matrix, both the matrix and its determinant are referred to as the Jacobian in literature.

$$J = \begin{bmatrix} \frac{\delta f}{\delta x_1} & \cdots & \frac{\delta f}{\delta x_n} \end{bmatrix} = \begin{bmatrix} \frac{\delta f_1}{\delta x_1} & \cdots & \frac{\delta f_1}{\delta x_n} \\ \vdots & \ddots & \vdots \\ \frac{\delta f_n}{\delta x_1} & \cdots & \frac{\delta f_n}{\delta x_n} \end{bmatrix} \tag{2.73}$$

The Jacobian matrix is important because if the function f is differentiable at a point x, then the Jacobian matrix defines a linear map n $\rightarrow$ m, which is the best (pointwise) linear approximation of the function f near the point x. This linear map is thus the generalization of the usual notion of derivative, and is called the derivative or the differential of f at x.

Minibatch

It is a training method which lies in-between Batch algorithm and SGD. Batch algorithm trains a model with the entire dataset. SGD trains with a single data point. Each batch is small enough to be fit in memory, and the model updates are frequent enough not to converge prematurely. It is not too computationally expensive compared to SGD and the variance is not too high over training epochs.

Momentum

It is a gradient descent optimization method inspired by physical concept of momentum which enhances global minima search. Momentum accelerates gradient descent in the early training phase. Later, when you are close to convergence, the update delta become very small so the momentum becomes small so you won't jump over the global minima. Momentum value is within [0,1]

Vanishing and exploding gradient problem

Name THREE regularization methods

L1 regularization

It is a sume of the weights. The solution is not unique. It has a built-in feature selection and it is a result of having sparse coefficients.

L2 regularization

It is a sume of the squared of the weights. It is computationally efficient due to having analytical solutions. Non-sparse outputs

Bagging

Bootstrap AGGregatING. Reduces generalization error by combining several models. Train several different models separately, then have all the models vote on the output for test examples. It's an example of **model averaging** or **ensemble methods**. Averaging works because different models will usually not make all the same errors on the test set.

Dropout

Technique for resolving overfitting in a large network. It randomly drop units(along with their connections) from the NN during training. *[Dropout_A_Simple_Way_to_Prevent_Neural_Networks_from_Overfitting]*

Dropout rate

Dropout rate p = 1, implies no dropout and low values of p mean more dropout. Typical values of p for hidden units are in the range 0.5 to 0.8. For input layers, the choice depends on the kind of input. For real-valued inputs (image patches or speech frames), a typical value is 0.8. For hidden layers, the choice of p is coupled with the choice of number of hidden units n. Smaller p requires big n which slows down the training and leads to underfitting. Large p may not produce enough dropout to prevent overfitting.

Learning rate and Momentum

Dropout introduces a lot of noise in the gradients compared to standard stochastic gradient descent. Therefore, a lot of gradients tend to cancel each other. You can use 10-100 times the learning rate to fix this. While momentum values of 0.9 are common for standard nets, with dropout 0.95-0.99 works well.

Max-norm Regularization

Large momentum and learning rate can cause the network weights to grow very large. Max-norm regularization constrains the norm of the vector of incoming weights at each hidden unit to be bound by a constant c in the interval of 3-4.

2.6 Information visualization

2.6.1 Fundamentals

Terms

Lie factor

The "Lie Factor" is a value to describe the relation between the size of effect shown in a graphic and the size of effect shown in the data. To ensure the Integrity of a graphic, its Lie Factor should have a value between 0.95 and 1.05.

Space-time narrative

It is a information visualization method where the time series data is visualized on spatial domain such as a map. One common example is Carte Figurative by Minard.

Design variation & data variation

Data variation is when your data varies which are the most cases. The problem occurs when one mixes design variation with data variation as it generates ambiguity and deception. A typical design variation is having different y-axes without clear notes.

Data ink maximization

Data-ink ratio is how much ink is spent on actual data. To maximize it, simply erase redundant & non data-ink. One example would be deleting grids on graphs or ticks.

Chartjunk

A visualization is chartjunk if it contains redundant visualization elements which troubles viewers understanding the visualization. Types of chartjunk is ducks, vibrations and grids.

Graphical excellence

- More ideas in shorter time.
- Don't waste space, ink
- Eliminate non-essentials

Tufte's principles

- Show the data
- Induce the viewer to think about the substance
- Avoid distortion
- Data ink ratio maximization
- Make large data sets coherent
- Encourage the eye to compare different piece of data
- Reveal the data at several levels of detail
- serve a reasonably clear purpose: description, exploration,
- Be integrated with the statistcal and verbal descriptions of a data set

Human perception

Gibson's affordance theory

We perceive possibilities for action in the environment, known as affordance. For instance, an open terrain affords walking; a stone on the ground affords tripping while walking.

Visual acuities

Visual acuities are measurements of our ability to see detail. Simple acuities are restricted by the spacing of the receptor cells at the centre of the fovea. Superacuity is the ability to achieve better resolution by integrating information over space (or time).

Receptor cells

rods: detect black/white/grey colours but not much detail • function best in dim light • located around the edges of the retina • 120 million in each eye

cones: detect fine detail and colours • function best in bright light • densely packed in fovea (centreof retina) • 5 million in each eye

Acuity is at maximum at the centre of the fovea.

Attentiveness

- Non-preattentive feature: Cound numbers of 3s.
- Preattentive feature: Size(Form)

Gestalt laws

- Proximity: Things that are near to each other appear to be grouped together. Place the data elements into proximity to emphasize connections between them.
- Similarity: Similar objects appear to be grouped together

- Good continuation: Visual complete objects are more likely to be constructed from visual elements that are smooth and continuous, rather than ones that contain abrupt changes in direction. It is one of the most powerful grouping principles. It's easier to perceive connections when contours run smoothly.

Sensory vs. arbitrary symbols

- **sensory symbols**
  - understandable without learning
  - processing is hard-wired and fast
  - resistant to instructional bias
  - cross-cultural
- **arbitrary symbols**
  - hard to learn and easy to forget (except when overlearned)
  - formally powerful
  - capable of rapid change
  - culture-specific

How light is perceived by eyes

Light falling on retina activates (1) receptor cells (i.e., rods and cones) which in turn activate (2) bipolar cells and then (3) ganglion cells through cascading photochemical reactions that transform the light into neural impulses, which carry visual information via the optic nerve to the visual processing areas in the visual cortex at the back of the brain where meaningful images are composed

Integral and separable dimensions

Separable features are perceived independent of each other such as size and color.

Integral features are perceived holistically such as width and height.

Use separable dimensions to encode different variables in glyphs.

Dimensionality Reduction

What is PCA?

Mathematically PCA is about selecting most significant eigenvectors. The eigenvector with the largest eigenvalue is the principal component. The number of dimensions equal to the number of eigenvectors and the number of significant eigenvectors(or dimensions) one selects is determined by the user. It is useful as oftentimes the data may be more comprehendable when the PCA is conducted.

The steps are as following,

1. Standardize the data
2. Obtain the eigen vectors and eigenvalues.
3. Sort the eigen values in descending order and choose k largest eigen values. k is the number of dimensions

4. Construct the projection matrix W from the selected k eigenvectors
5. Transform the original dataset X via W to obtain a k-dimensional feature subspace Y.

When not to use PCA?

When the dataset is non-linear.

What is MDS?

Metric Multi-Dimensional Scaling is similar to PCA. PCA creates plots based on **correlations** among samples while MDS creates plots based on **distances** among samples. That is the only difference.

Graphical visualization and navigation

What is a graph?

A graph is a visual representation of data which contrasts to tables. A graph is visual/pictorial so it is intuitive and easy to comprehend compared to tables. On the other hand, as graphs are visual it inherently cannot convey sharp accuracies as tables or raw data and it is easy to misguide viewers. Some of the popular graphs are bar, violin, heatmap and lines.

High-dimensional data

- small multiples with simple plots
- heatmaps
- parallel coordinates
- glyphs
- dimension reduction

Design principles for eyes

- physical luminance and perceived brightness can be quite different
- gray scale is bad at encoding absolute values, good at encoding relative values and shapes
- **if outline of the shapes of objects is important:**
  - background should have maximal contrast with foreground objects
- **if it is important to see variations in grayscale:**
  - background should have minimal contrast with foreground objects

2.7 Linear Algebra

2.7.1 Fundamentals

Basis

It's a set of vectors in a vector space which are linearly independent. All vectors in the vector space are linear combinations of the basis. Read wiki.

Eigenvector & Eigenvalue

Intuitively, eigenvectors and eigenvalues are related to transformation. When a transformation is applied to a vector space, veectors span. While most vectors drift away from their spans some remain on its own span – eigenvectors. Eigenvectors remain on its own span after a transformation but they may scale – by a factor of their eigenvalues. An interesting fact is that any vector that lies on the same span as eigenvectors is itself an eigenvector. Therefore **there can be infinitely many eigenvectors**. However, **there could be only one eigenvector as well**. Consider a 3D transformation matrix A:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.74}$$

A will squash everything into *null* and there will be only one eigenvector $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

Eigenvector in 3D rotations

An eigenvectors in a 3D rotation means the **axis of a rotation**. And because a rotation doesn't change the scale, the **eigenvalue should be 1**.

Eigenvector in 2D rotations

There is no eigenvector in 2D rotations. No eigenvector implies no real-valued eigenvalue but imaginary-valued.

Single eigenvalue with multiple eigenvectors

There could be multiple eigenvectors but only one single eigenvalue. Consider a transformation A:

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \tag{2.75}$$

A scales every eigenvector by 2 and only 2.

Calculation

$$\overbrace{A\vec{v}}^{\text{Matrix-vector multiplication}} = \underbrace{\lambda\vec{v}}_{\text{Scalar multiplication}}$$

The matrix A changes only the scale of the vector $\vec{v}$ by a factor of λ. We could rewrite the right hand as

$$\begin{aligned}\lambda\vec{v} &= \begin{bmatrix}\lambda & 0 & 0\\ 0 & \lambda & 0\\ 0 & 0 & \lambda\end{bmatrix}\vec{v} \\ &= \lambda\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\vec{v} \\ &= \lambda I\vec{v}\end{aligned} \tag{2.76}$$

To get the value of the eigenvalue, take the righthand to the other side:

$$(A - \lambda I)\vec{v} = \vec{0}$$

Remember the squashification? $(A - \lambda I)$ is squashing $\vec{v}$. This implies the following:

$$\det(A - \lambda I) = 0$$

Eigenbasis

Consider a 2D vectorspace. If both basis vectors are eigenvectors then its transformation matrix would be diagonal. Here's step-by-step:

A typical set of eigen vectors in 2D:

$$\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}$$

A transformation matrix:

$$\begin{bmatrix}-1 & 0\\ 0 & 2\end{bmatrix}$$

The columns of the transformation matrix happnes to be the eigenvectors and, **the basis vectors as well** with the diagonal values being their eigenvalues. Pay attention to the matrix that the matrix is diagonal. Diagonal matrices have a handy property – their power is just a power of the elements:

$$\begin{bmatrix}-1 & 0\\ 0 & 2\end{bmatrix}^n = \begin{bmatrix}(-1)^n & 0\\ 0 & 2^n\end{bmatrix}$$

Eigenbasis for easier power

Consider a transformation A,

$$\begin{bmatrix}3 & 1\\ 0 & 2\end{bmatrix}$$

A is not diagonal so its power will be expensive to calculate. Let's change its basis as the eigenvectors **in order to make** A **diagonal**. The eigenvectors of A,

$$\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix}$$

We can build **"Change of basis matrix"** from the eigenvectors,

$$\begin{bmatrix}1 & -1\\ 0 & 1\end{bmatrix}$$

Now let's change its basis,

$$\begin{aligned}\left[\text{Change of basis matrix}^{-1}\right] A \left[\text{Change of basis matrix}\right] \\ \Rightarrow \begin{bmatrix}1 & -1\\ 0 & 1\end{bmatrix}^{-1} \begin{bmatrix}3 & 1\\ 0 & 2\end{bmatrix} \begin{bmatrix}1 & -1\\ 0 & 1\end{bmatrix} \\ = \begin{bmatrix}3 & 0\\ 0 & 2\end{bmatrix}\end{aligned}$$

Hessian matrix

The Hessian Matrix is a square matrix of second ordered partial derivatives of a scalar function. It is of immense use in linear algebra as well as for determining points of local maxima or minima.[1]

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

Conditions for Minima,Maxima,Saddle point

The Hessian of a function is denoted by $\Delta^2 f(x, y)$ where f is a twice differentiable function & if (x_0, y_0) is one of it's stationary points then :

- If $\Delta^2 f(x_0, y_0) > 0$ i.e positive definite , (x_0, y_0) is a point of local minimum.
- If $\Delta^2 f(x_0, y_0) < 0$, i.e. negative definite , (x_0, y_0) is a point of local maximum.
- If $\Delta^2 f(x_0, y_0)$ is neither positive nor negative i.e. Indefinite , (x_0, y_0) is a saddle point

Reference

2.7.2 Quiz

1. Differentiate $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$

Consider $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{n\times n}$. Show that for a symmetric matrix $\mathbf{A}^T = \mathbf{A}$, the gradient of $f(\mathbf{x})$ is given as $2\mathbf{A}\mathbf{x}$.

[1] https://brilliant.org/wiki/hessian-matrix/

Solution

Let $\mathbf{x}^{n\times 1} = (x_1, \ldots, x_n)'$ be a vector, the derivative of $\mathbf{y} = f(\mathbf{x})$ with respect to the vector $\mathbf{x}$ is defined by

$$\frac{\partial f}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

Let

$$\begin{aligned}
\mathbf{y} &= f(\mathbf{x}) && (2.79)\\
&= \mathbf{x}'\mathbf{A}\mathbf{x} && (2.80)\\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j && (2.81)\\
&= \sum_{i=1}^{n} a_{i1}x_i x_1 + \sum_{j=1}^{n} a_{1j}x_1 x_j + \sum_{i=2}^{n}\sum_{j=2}^{n} a_{ij}x_i x_j && (2.82)\\
\frac{\partial f}{\partial x_1} &= \sum_{i=1}^{n} a_{i1}x_i + \sum_{j=1}^{n} a_{1j}x_j && (2.83)\\
&= \sum_{i=1}^{n} a_{1i}x_i + \sum_{i=1}^{n} a_{1i}x_i \text{ [since } a_{1i} = a_{i1}\text{]} && (2.84)\\
&= 2\sum_{i=1}^{n} a_{1i}x_i && (2.85)\\
\frac{\partial f}{\partial \mathbf{x}} &= \begin{pmatrix} 2\sum_{i=1}^{n} a_{1i}x_i \\ \vdots \\ 2\sum_{i=1}^{n} a_{ni}x_i \end{pmatrix} && (2.86)\\
&= 2\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{11} & a_{12} & \ldots & a_{1n} \end{pmatrix}\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} && (2.87)\\
&= 2\mathbf{A}\mathbf{x} && (2.88)
\end{aligned}$$

2.7.3 Reference

Textbooks

2.8 Machine Learning

2.8.1 Linear Regression

Consider a hypothesis function as a linear function of x.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \qquad (\theta_i\text{'s are parameters/weights}) \qquad (2.89)$$

To simplify our notation, we introduce the convention of let the intercept be 1: $x_0 = 1$.

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

How do we learn the parameters? Make $h(x)$ close to y. In other words, make the cost as small as possible. Cost function $J(\theta)$,

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2$$

Gradient Descent

This is the tool to minimize your cost. Following is how it works.

Repeat until convergence {

$$\theta_j := \theta_j - \alpha\frac{\delta}{\delta\theta_j}J(\theta_0, \theta_1), \qquad \text{(for j = 0 and j = 1)}$$

}

Be careful not to update θ separately. Update them all together at the end of each loop. i.e.,

$$temp_0 := \theta_0 - \alpha\frac{\delta}{\delta\theta_j}J(\theta_0, \theta_1)$$
$$temp_1 := \theta_1 - \alpha\frac{\delta}{\delta\theta_j}J(\theta_0, \theta_1)$$
$$\theta_0 := temp_0$$
$$\theta_1 := temp_1$$

Bivariate Gradient Descent

$$\begin{aligned}\frac{\delta}{\delta\theta_j}J(\theta_0, \theta_1) &= \frac{\delta}{\delta\theta_j}J(\theta_0, \theta_1)\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2 \\ &= \frac{\delta}{\delta\theta_j}J(\theta_0, \theta_1)\frac{1}{2m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^i - y^i)^2\end{aligned}$$

$$\theta_0 \Rightarrow j = 0 : \tfrac{\delta}{\delta\theta_j}J(\theta_0, \theta_1) = \tfrac{1}{m}\textstyle\sum_{i=1}^{m}(h_\theta(x^i) - y^i)$$
$$\theta_1 \Rightarrow j = 1 : \tfrac{\delta}{\delta\theta_j}J(\theta_0, \theta_1) = \tfrac{1}{m}\textstyle\sum_{i=1}^{m}(h_\theta(x^i) - y^i)$$

Repeat until convergence {

$$\begin{aligned}\theta_0 &:= \theta_0 - \alpha\tfrac{1}{m}\textstyle\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_0^i \\ \theta_1 &:= \theta_1 - \alpha\tfrac{1}{m}\textstyle\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_1^i\end{aligned}$$

}

Here x_0^i is 0.

Multivariate gradient descent

Repeat until convergence {

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i$$

}

Normal Equation

In linear gression, instead of a loop as above, gradient descent can be expressed in a non-loop equation:

$$\theta = (X^T X)^{-1} X^T y$$

Regularization

- Overfitting: low bias, high variance
- Underfitting: high bias, low variance

You could think bias in respect of the training dataset, and variance the test set. If a model overfits the bias would be almost 0 because it perfectly fits the training set. However, it wouldn't be a good model for new dataset so results in high variance.

Watch Andrew Ng's lecture.

Example

Say we want to make the following function more quadratic:

$$H = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Without actually removing $\theta_3 x^3 + \theta_4 x^4$ or changing the form of H, we can modify our cost function:

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2 + 1000\theta_3^2 + 1000\theta_4^2$$

We could also regularize all of our theta parameters in a single summation

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2 + \lambda \sum_{j=1}^{n} \theta_j^2$$

The square in the second sum comes from the first sum.

Regularization - Gradient Descent

Repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x_0^i \qquad \text{(Don't penalize the intercept } \theta_0\text{)}$$

$$\theta_j := \theta_j - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i \right) x_j^i \right) + \frac{\lambda}{m} \theta_j \right] \qquad j \in 1, 2, \ldots, n \quad (2.90)$$

}

$\frac{\lambda}{m}$ is a *regularization performer*.

The above can be represented as:

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x_j^i$$

$1 - \alpha \frac{\lambda}{m}$ is always less than 1. Thus, regularized.

Regularization - Normal Equation

$$X = \begin{bmatrix} (x^1)^T \\ \vdots \\ (x^m)^T \end{bmatrix}, \qquad \text{size is } (m) \times (n+1) \tag{2.91}$$

$$\vec{y} = \begin{bmatrix} y^1 \\ \vdots \\ y^m \end{bmatrix}, \qquad \text{size is } (m) \times (1) \tag{2.92}$$

$$\theta = (X^T X + \lambda L)^{-1} X^T y$$

where L is a pseudo-diagonal matrix of

$$L = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}, \qquad \text{size is } (n+1) \times (n+1) \tag{2.93}$$

If $m \leq n$, then $X^T X$ is non-invertable and so is $(X^T X + \lambda L)$.

2.8.2 Logistic Regression

Used for classification. We want $\theta \leq h_\theta(x) \leq 1$.

$$h_\theta(x) = g(\theta^T x) = g(z) = \frac{1}{1+e^z} \Rightarrow h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

$\frac{1}{1+e^{-\theta^T x}}$ is a sigmoid/logistic function

Fig. 49: < Logistic function plot >

The idea of logistic regression is

- Suppose predict $y = 1$ if $h_\theta(x) \geq 0.5$
- Suppose predict $y = 0$ if $h_\theta(x) < 0.5$

Linearity of Logistic Regression

From Stackoverflow.

The logistic regression model is of the form,

$$\text{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \cdots + \beta_p x_{p,i}.$$

It is called a generalized linear model not because the estimated probability is linear, but because the logit of the estimated probability response is a linear function of the parameters.

Cost Function

$$\text{cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

We use a separate cost function for logistic regression which differs from linear regression because otherwise it will be too wavy; cause too many local optima.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{cost}(h_\theta(x^i), y^i)$$

Simplified Logistic Regression Cost Function

$$\text{cost}(h_\theta(x^i), y^i) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x))$$
$$J(\theta) = -\frac{1}{m} [\sum_{i=1}^{m} y^i \log(h_\theta(x^i)) + (1 - y^i) \log(1 - h_\theta(x^i))]$$

Logistic Regression Cost Function Gradient Descent

To minimize $J(\theta)$, repeat until convergence:

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x_j^i$$

This algorithm is identical to linear regression.

Normal Equation of Gradient Descent

$$\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y})$$

Regularization

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^i \log(h_\theta(x^i)) + (1 - y^i) \log(1 - h_\theta(x^i))] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

2.8.3 Loss Function

Cross-entropy

It describes the loss between two probability distributions.

$$H(p, q) = -\sum_x p(x) \log q(x)$$

Cross-entropy sums over every class.

Read: Stackoverflow

2.8.4 Optimization

Empirical Risk Minimization

This part is mostly from Deep Learning by Goodfellow et al.

In ML, we want to optimize performance measure P by reducing a different cost function $J(\theta)$.

$$J(\theta) = \mathbb{E}_{(x,y)\sim\hat{p}_{data}} L(f(x;\theta), y)$$

- L: per-example loss function.
- $f(x;\theta)$: predicted output when the input is x.
- $\hat{p}_{data}$: empirical distribution
- y: target output in the supervised learning

The above defines an objective function with respect to the training set. We would prefer to minimize the corresponding objective function where the expectation is taken across *the data-generating distribution* p_{data} rather than just over the finite training set:

$$J^*(\theta) = \mathbb{E}_{(x,y)\sim p_{data}} L(f(x;\theta), y)$$

The above quantity is called the **risk**. **NOTE**: here we have a true distribution p_{data}. In ML we only have empirical distribution so we have to replace it with the empirical distribution $\hat{p}_{data}$ defined by the training set. Minimize the **empirical risk**

$$\mathbb{E}_{(x,y)\sim p_{data}}[L(f(x;\theta), y)] = \frac{1}{m}\sum_{i=1}^{m} L(f(x^{(i)};\theta), y^{(i)})$$

where *m* is the number of training examples.

Remember, empirical risk minimization is prone to overfitting. In DL, we rarely use empirical risk minimization but algorithms based on gradient descent.

Batch and Minibatch Algorithms

Optimization algorithms that use the entire training set are called **batch** or **deterministic** gradient methods, because they process all the training examples simultaneously in a large batch. Algorithms that use only a **single** example at a time are called **stochastic**. **However**, most algorithms used for DL fall somewhere in between, using more than one but fewer than all the training examples. They are called **minibatch stochastic** or simply **stochastic** methods.

Minibatch size selection factors

> **Warning:** Ask a TA for detailed explanation.

- Larger batches provide a more accurate estimate of the gradient, but with less than linear returns.
- Multicore architectures are usually underutilized by extremely small batches. For a fix, use an absolute minimum batch size.

- If all examples in the batch are to be processed in parallel, then the amount of memoery scales with the batch size. Usually this is the limiting factor in batch size.
- Some kinds of HW achieve better runtime with specific sizes of arrays. Especially when using GPUs, it is common for power of 2 batch sizes to offer better runtime(typically from 32 to 256. You may try 16 for large models).
- Small batches can offer a regularizing effect (Wilson and Martinez, 2003), perhaps due to the noise they add to the learning process. Generalization error is often best for a batch size of 1. Training with such a small batch size might require a small learning rate to maintain stability because of the high variance in the estimate of the gradient. The total runtime can be very high as a result of the need to make more steps.

Adam

Adam learns scale of loss.

2.8.5 Data Manipulation

Feature Scaling

Make sure features are on a similar scale such as $-1 \leq x_i \leq 1$. For instance,

$$\begin{aligned} x_1 &= \text{size}, && \text{(0 to 2000 meters)} \\ x_2 &= \text{number of bedrooms}, && \text{(1 to 5 rooms)} \end{aligned}$$

Rescale:

$$\begin{aligned} x_1 &= \frac{\text{size}}{2000} \\ x_2 &= \frac{\text{number of bedrooms}}{5} \end{aligned}$$

2.8.6 Cross-validation

A methodology for tuning hyperparameters without overfitting.

Divide the dataset into 3 partitions: training, validation and test. Use the training data for learning parameters and evaluate the hyperparameters using the validation set. Use the test set at very last only once as it would represent general data.

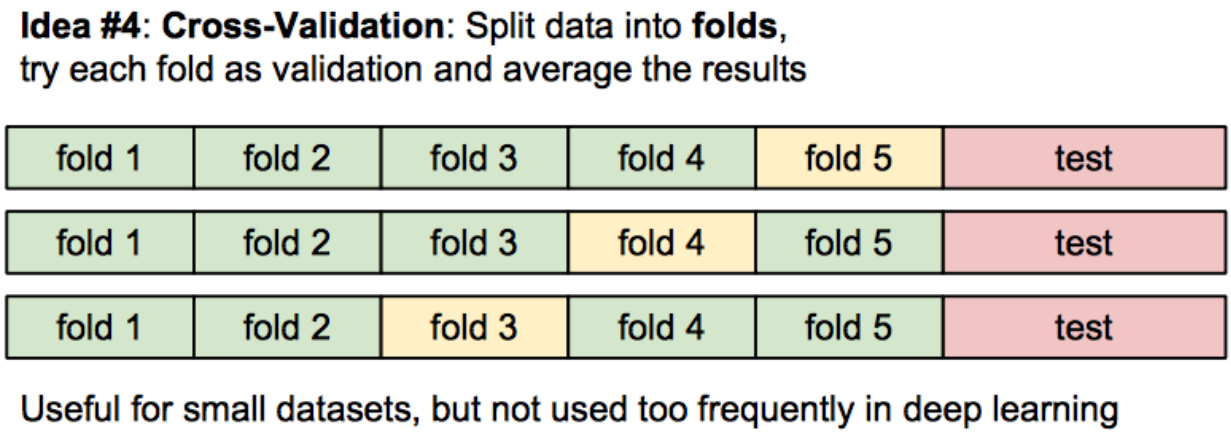


Fig. 50: < An image from a Stanford course, Neural Networks for Visual Recognition >

Exhuastive cross-validation

Leave-p-out cross-validation

Leave-one-out cross-validation

Non-exhuastive cross-validation

k-fold cross-validation

2.8.7 Algorithms

K-Nearest Neighbors: Distance Metric

L1(manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

L1 value may change if the coordinate system changes.

L2(Euclidean) distance

$$d_1(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

More generic. Values stay the same even when the coordinate system changes.

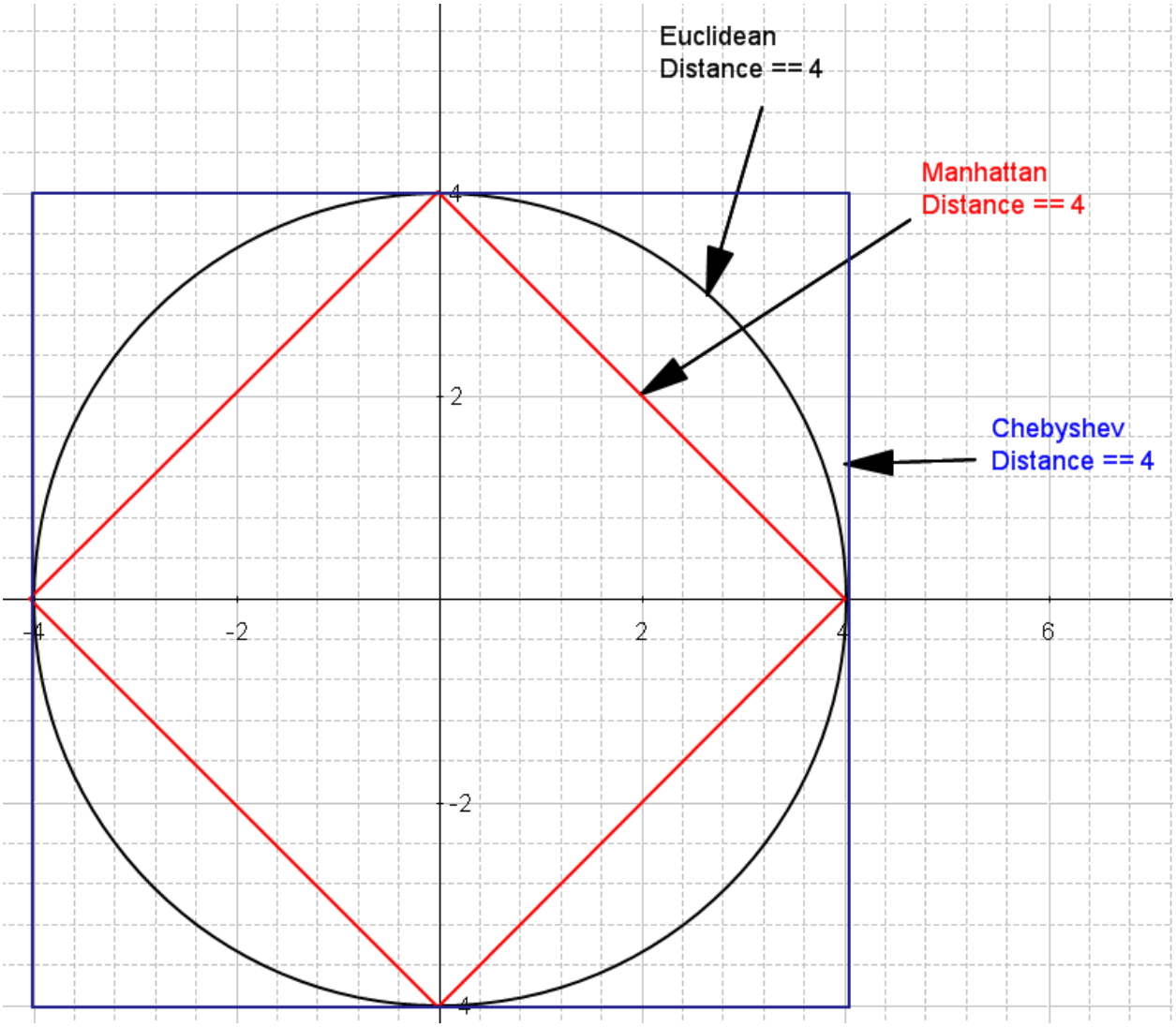


Fig. 51: < L1, L2, Chebyshev distances >

Disadvantage

- The classifier must remember all of the training data and store it for future comparisons
- Not suitable for images. Classifying a test image is expensive since it requires a comparison to all training images and images contain millions of pixels. Don't even talk about videos.

2.8.8 Glossaries

Hyperparameter

A parameter we "just give" to algorithms, not learn via training. Hyperparameters are evaluated by cross-validation sets and fine-tuned hyperparameters are your ultimate goals. For instance, distance function selection is a hyperparameter setting.

Stochastic System

It does not always produce the same output for a given input. A few components of systems that can be stochastic in nature include stochastic inputs, random time-delays, noisy(modeled as random) disturbances.

2.8.9 Project Management

This post is a machine learning project management guideline. This assumes you have chosen your dataset.

Pull existing Kernels

For just a sake of a little kick to move your project going, pull existing kernels from dataset owner or Kaggle and see what people have done. Play around with it a little.

Analyze the data

Super important. It's very easy to just dive into algorithms and treat data as just digits but it is cruicial that you comprehend your data; distribution, symmetry, correlation, pixel aggregation(for visual data) and etc. Visualize with histograms, violin, swarm or heatmaps.

Write related work

Write **Related work** section like you would in your thesis. During this phase you will read existing works and get some idea from academia. Definitely checkout dataset owner as they usually have many related papers as references. For instance, Zalando's Fashion-MNIST has good number of references.

Tune your hyperparameters

In NN, you have to make several choices:

- optimization methods
- cost function
- hidden activation function (e.g. ReLu)

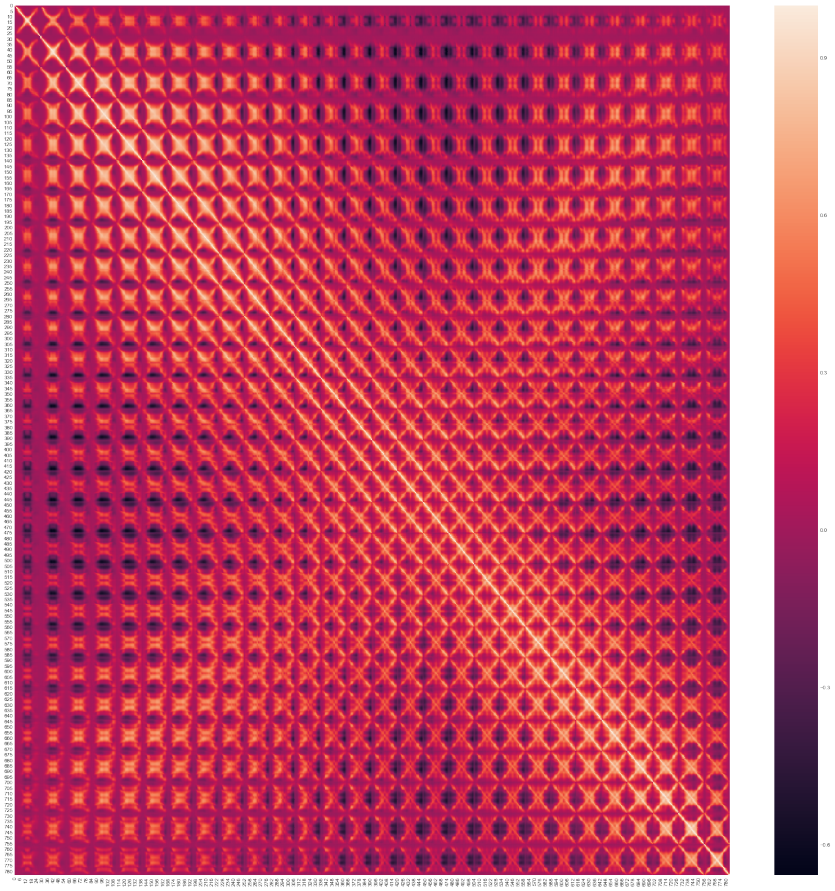


Fig. 52: < Correlation heatmap of Fashion-MNIST >

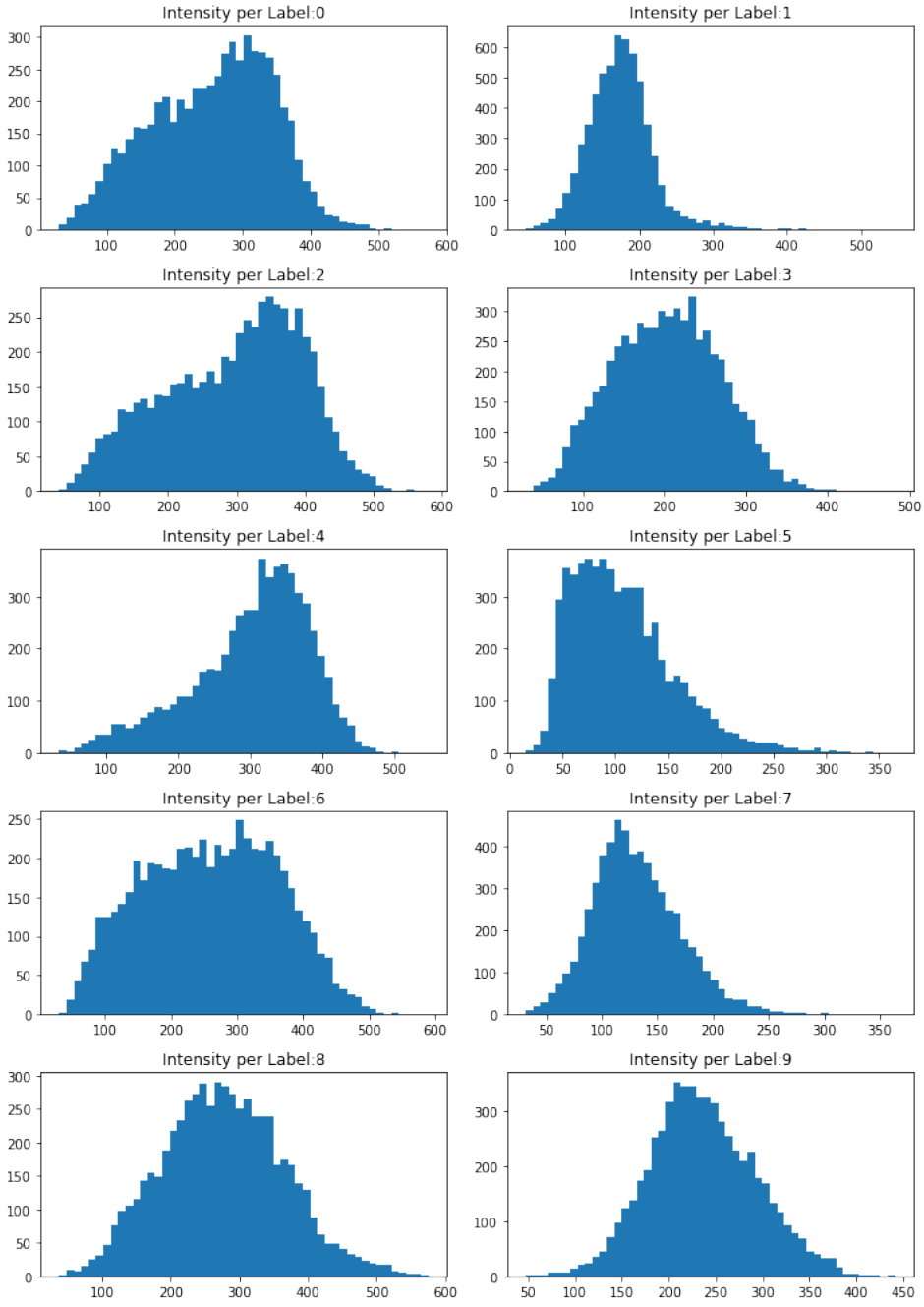


Fig. 53: < Pixel intensity distribution of Fashion-MNIST >

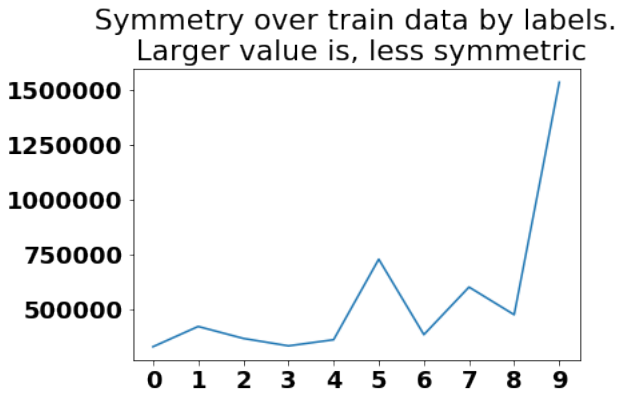


Fig. 54: < Symmetry per category of Fashion-MNIST >

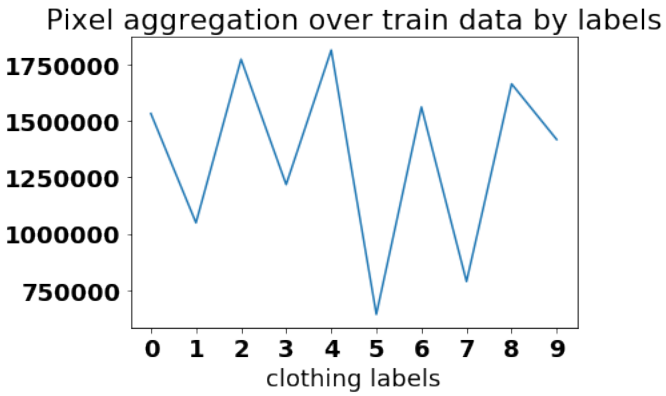


Fig. 55: < Pixel aggregation per category of Fashion-MNIST >

- output activation function (e.g. softmax)
- regularization parameters
- depth of network
- width of each hidden layer

This is a very time consuming phase. You could run grid search but they can be exhuasting. Instead, you could do a step-by-step tuning. For a CNN model for Fashion-MNIST my team tuned in the following order:

1. regularization parameters
2. activation functions
3. dropout rates
4. optimization methods

Check which class of data is hard to train

Check and fix

Make a Telegram notification bot training models

Send a message of a output file name with plots of the data. It saves you time and also it's a good log.

Benchmark your model on different dataset

If you are doing image classification run it on MNIST or CIFAR10.

2.9 Papers

This is a collection of the papaers I've read.

2.9.1 PROGRESSIVE GROWING OF GANS FOR IMPROVED QUALITY, STABILITY, AND VARIATION

Authors: Tero Karras, Timo Aila, Samuli Laine, Jaakko Lehtinen, 2018

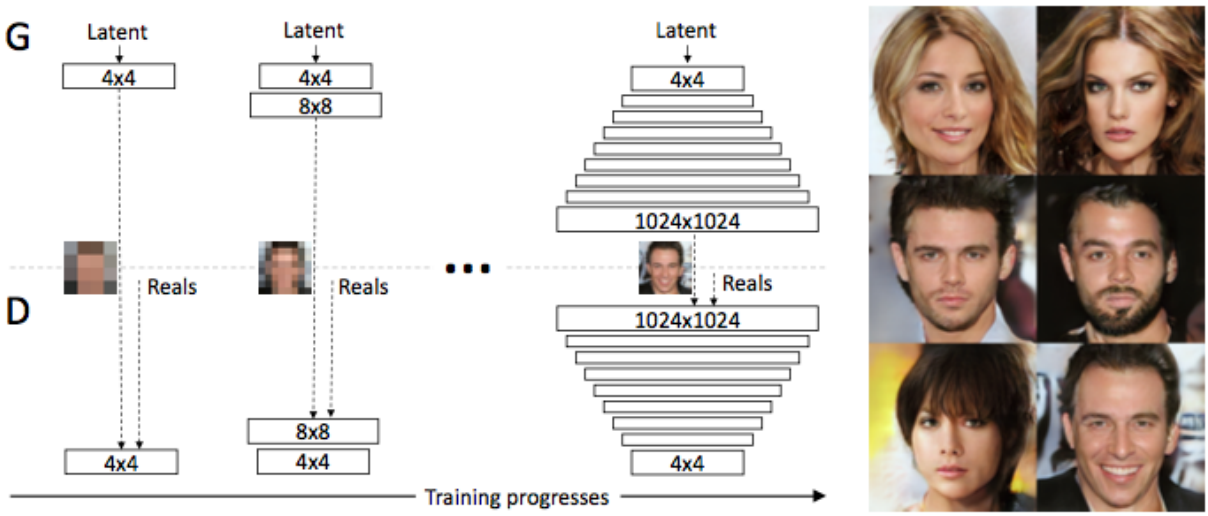


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $N \times N$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. One the right we show six example images generated using progressive growing at 1024×1024.

- Github, Youtube, Trained network
- Image generation from CELEBA with GAN(Generative Adversarial Networks) using generator and discriminator progressively.
- 8.80 in unsupervised CIFAR10
- GAN is differentiable and this allows us to guide generators and discriminators to the right direction.
- High-resolution image generation is difficult as discriminator can more easily distinguish fakes from real images, thus amplifies gradient problem.
- The paper used low-resolution training sets in the beginning, and add new layers that introduce higher-resolution details as the training progresses.
- They used minibatch discrimination in order to compute feature statistics across the minibatch which is added towards the end of the discriminator.
- Used $\mathcal{N}(0, 1)$ for weight initialization and then scale at runtime (He et al., 2015).
- In order to prevent the escalation of signal magnitudes, used a variant of "local response normalization"(Krizhevsky et al., 2012), $b_{x,y} = a_{x,y} \Big/ \sqrt{\frac{1}{N} \sum_{j=0}^{N-1} (a^j_{x,y})^2 + \epsilon}$ where
  - $\epsilon = 10^{-s}$
  - N, the number of feature maps
  - $a_{x,y}, b_{x,y}$ original and normalized feature vector in pixel (x, y)

- Used sliced Wasserstein distance(SWD) and multi-scale structural similarity(MS- SSIM) (Odena_et_al_2017) to evaluate the importance our individual contributions, and also percep- tually validate the metrics themselves
- Progressive variant offers two main benefits: it converges to a considerably better optimum and also reduces the total training time by about a factor of two.

2.9.2 UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

Authors: Alec Radford & Luke Metz, Soumith Chintala, 2016

Related work on unsupervised representation learning

- Hierarchical clustring of image patches(Coates & Ng 2012)
- auto-encoder training (Vincent et al. 2010), (Zhao et al., 2015), (Rasmus et al., 2015)
- Deep belief networks (Lee et al., 2009)

Generating natural images

- parametric
  - samples often suffer from being blurry
  - iterative forward diffusion process (Sohl-Dickstein et al., 2015)
  - GAN (Goodfellow et al., 2014) suffers from being noisy and incomprehensible.
    - A laplacian pyramid extension to this approach (Denton et al., 2015) showed higher quality images, but they still suffered from the objects looking wobbly because of noise introduced in chaining multiple models.
    - A recurrent network approach (Gregor et al., 2015) and a deconvolution network approach (Dosovitskiy et al., 2014) have also recently had some success with generating natural images. However, they have not leveraged the generators for supervised tasks.
- non-parametric
  - do matching from a database of existing images

Approach & Architecture

Until LAPGAN (Denton et al., 2015) appeared GANs using CNNs to model images was not scalable. LAPGAN is an alternative approach to iteratively upscale low resolution generated images which can be modeled more reliably.

- Used convolutional net (Springenberg et al., 2014) which replaces deterministic spatial pooling functions (such as maxpooling) with strided convolutions, allowing the network to learn its own spatial downsampling/upsampling. Used in generators and discriminators.
- Eliminated fully connected layers on top of convolutional features. (Mordvintsev et al.) used this approach in their art image classifiers with global average pooling.

- Batch Normalization (Ioffe & Szegedy, 2015) which stabilizes learning by normalizing the input to each unit to have zero mean and unit variance. This helps deal with training problems that arise due to poor initialization and helps gradient flow in deeper models. Applying batchnorm to all layers however, resulted in sample oscillation and model instability. This was avoided by not applying batchnorm to the generator output layer and the discriminator input layer.
- Architecture guidelines for stable Deep Convolutional GANs
  - Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
  - Use batchnorm in both the generator and the discriminator.
  - Remove fully connected hidden layers for deeper architectures.
  - Use ReLU activation in generator for all layers except for the output, which uses Tanh.
  - Use LeakyReLU activation in the discriminator for all layers.

Vector arithmetic for visual concepts

2.9.3 Visualizing and Understanding Convolutional Networks

Why do they run rectifier in deconvnet??? -> The author says it's because the rectifier is used in forward passing so should be used in the backward passing as well. I don't think they have a really good reason. Read this Quora answer.

In computer vision it is important to have image patterns(*filters*) that cause high activations. The purpose of deconvolution is to visualize those. . . .

References

- Visualizing and Understanding Convolutional Networks
- Youtube_Visualizing and Understanding Deep Neural Networks by Matt Zeiler
- How does a deconvolutional neural network work
- Deconvolutional Networks Slides
- Deconvolutional Networks Paper
- Visualizing what ConvNets learn

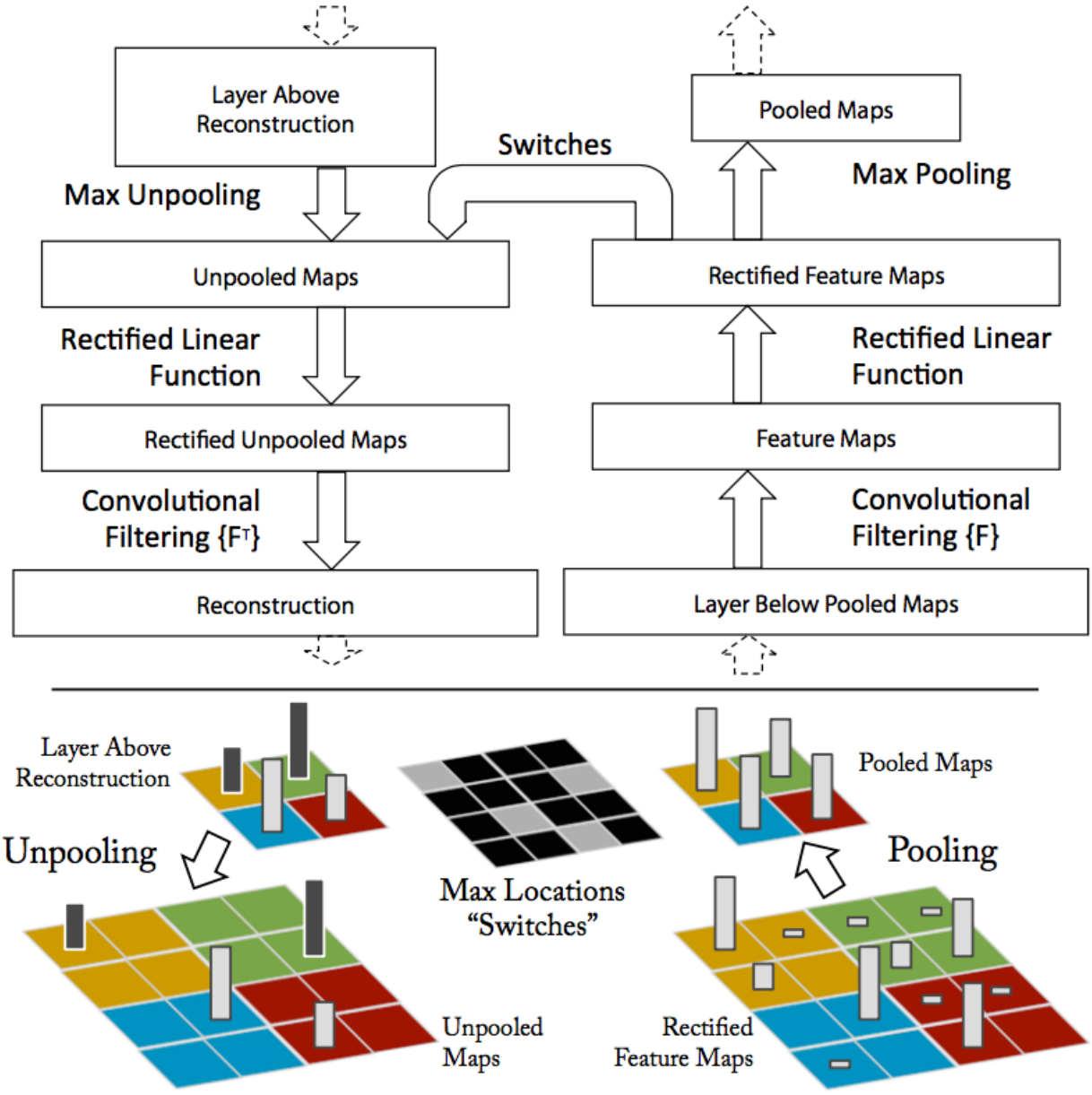


Fig. 56: < Top: A deconvnet layer (left) attached to a con- vnet layer (right). The deconvnet will reconstruct an approximate version of the convnet features from the layer beneath. Bottom: An illustration of the unpooling operation in the deconvnet, using switches which record the location of the local max in each pooling region (colored zones) during pooling in the convnet. >

Fig. 57: < Visualization of features in a fully trained model. For layers 2-5 we show the top 9 activations in a random subset of feature maps across the validation data, projected down to pixel space using our deconvolutional network approach. Our reconstructions are not samples from the model: they are reconstructed patterns from the validation set that cause high activations in a given feature map. For each feature map we also show the corresponding image patches. Note: (i) the the strong grouping within each feature map, (ii) greater invariance at higher layers and (iii)

2.10 Probabilities and Statistics

2.10.1 Fundamentals

Odds

$$O(E) = \frac{P(E)}{P(\bar{E})}$$

Variance and Standard Deviation

Spread of probability mass about the mean.

Variance

$$Var(X) = E[(X - E[X])^2] = E[X^2] - E[X]^2$$

Computation as sum:

$$Var(X) = \sum_{i=1}^{n} p(x_i)(x_i 0 \mu)^2$$

Standard Deviation

$$\sigma = \sqrt{Var(X)}$$

Standard Error

Standard error(SE) of a parameter is the standard deviation(SV) of its sampling distribution or an estimate of the SV. If the parameter or the statistic is the mean, it is called the **standard error of the mean(SEM)**. SEM can be expressed as

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$$

where *n* is the sample size.

Standard Deviation vs. SEM

SD quantifies scatter, and SEM how precisely you know the true mean of the population. SEM takes into account both the values of the SD and the sample size. They both use the same units of the data. SEM, by definition, is always smaller than the SD. SEM gets smaller as your sample size, n, gets larger. This makes sense, because the mean of a large sample is likely to be closer to the true population mean than is mean of a small sample.

Likelihood

A function of the parameters of a statistical model given data.

Joint distribution

A probability distribution for two or more variables. OR A joint distribution is a distribution which is joint.

Conditional Probabilities

$$P(A|B) = \frac{A \cap B}{P(B)}$$

Multiplication Rule

Reference: Introduction to Probability by Bertsekas pg.24. Assuming that all of the conditioning events have positive probability,

$$P(\cap_{i=1}^{n} A_i) = P(A_1)P(A_2|A_1)P(A_3|A_1 \cap A_2) \cdots P(A_n| \cap_{i=1}^{n-1} A_i)$$

Total Probability Theorem

Let $A_1, \cdots, A_n$ be disjoint events and assume that $P(A_i) > 0$ for all *i*. Then, for any event B, we have

$$\begin{aligned} P(B) &= P(A_1 \cap B) + \cdots + P(A_n \cap B) \\ &= P(A_1)P(B|A_1) + \cdots + P(A_n)P(B|A_n) \end{aligned}$$

Conditional distribution

For random variables A and B, $P(A|B)$ is the probability distribution which describes the change in A when B is changed. Therefore the distribution sums to 1 over all the events in A.

$$A = a_1, a_2, ...a_n$$
$$\sum_{i=1}^{n} P(a_i|B) = 1$$

Conditional independence

$X \perp\!\!\!\perp Y|Z$ denotes that variable X and Y are conditionally independent of each other, given the state of variable Z.

$$P_{X,Y|Z}(x, y|z) = P_{X|Z}(x|z)P_{Y|Z}(y|z)$$

Intuitively, this means that if we know the value of Z, knowing in addition the value of Y does not provide any information about the value of X. Indeed, provided $P(y, z) > 0$, we have

$$X \perp\!\!\!\perp Y|Z \implies P_{X,|Y,Z}(x|y, z) = P_{X|Z}(x|z)$$

Proof:

$$p(x|y,z) = \frac{p(x,y,z)}{p(y,z)} = \frac{p(x,y|z)p(z)}{p(z)p(y|z)} \tag{2.94}$$
$$= \frac{p(x|z)p(y|z)p(z)}{p(z)p(y|z)} = p(x|z) \tag{2.95}$$
$$p(x,y,z) = p(x|y,z)p(y|z)p(z) \quad \text{(General chain rule of probability)} \tag{2.96}$$

Intuitive examples

1. Let $X_1, X_2, ..., X_n$ denote the cumulative sum of *n* dice throws, such that $dom(X_1) = 1, ..., 6, dom(X_2) = 2, ..., 12$, etc.

- Is X_{n+1} independent of X_{n-1}? **NO**.
- Is X_{n+1} conditionally independent of X_{n-1} given X_n? **YES**.

2. X ='Location of an airplane now', Y ='Location of the plane 15s ago', Z='Location 15s from now'

- Is Y independent of Z? **NO**
- Is Y conditionally independent of Z given X? **NO**

Practices

Example 1.

Consider the Bayesian network which represents Mr Holmes' burglary worries as given in the figure: (B)urglar, (A)larm, (W)atson, Mrs (G)ibbon. All vari- ables are binary with states True and False.

Part a). $p(B|W)$

$$p(B|W) = \frac{p(B,W)}{p(W)}$$

Let's separate the numerator and denominator. Numerator:

$$p(B,W) = p(W|B)p(B) \tag{2.97}$$
$$p(W|B) = p(W|A)p(A|B) + p(W|\bar{A})p(\bar{A}|B) = 0.896 \tag{2.98}$$
$$p(B,W) = 0.896 * 0.01 = 0.00896 \tag{2.99}$$

Now denominator:

$$p(W) = p(W|B)p(B) + p(W|\bar{B})p(\bar{B}) \tag{2.100}$$
$$= 0.52376$$

$$p(B|W) = \frac{p(B,W)}{p(W)} = \frac{0.00896}{0.52376} \approx 0.0171$$

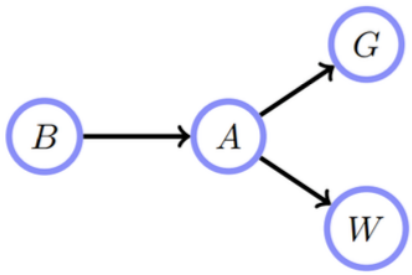


Figure 1: Bayesian network for Mr Holmes' burglary worries in Problem 1.

The probabilities are

$$
\begin{aligned}
p(B = tr) &= 0.01 \\
p(A = tr|B = tr) &= 0.99 \qquad p(A = tr|B = fa) = 0.05 \\
p(W = tr|A = tr) &= 0.90 \qquad p(W = tr|A = fa) = 0.5 \\
p(G = tr|A = tr) &= 0.7 \qquad p(G = tr|A = fa) = 0.2.
\end{aligned}
$$

Compute the conditional probabilities

(a) $p(B = tr|W = tr)$

(b) $p(B = tr|W = tr, G = fa)$

Fig. 58: < Source: Aalto course CS-E4820: Advanced probabilistic methods >

Part b). $p(B|W,\bar{G})$

$$p(B|W,\bar{G}) = \frac{p(B,W,\bar{G})}{p(W,\bar{G})}$$

Again, Let's separate the numerator and denominator. Numerator:

$$\begin{aligned} p(B,W,\bar{G}) &= p(W,\bar{G}|B)p(B) && (2.102)\\ p(W,\bar{G}|B) &= p(W|A)p(\bar{G}|A)p(A|B) + p(W|\bar{A})p(\bar{G}|\bar{A})p(\bar{A}|B) && (2.103)\\ &= (0.95)(0.03)(0.95) + (0.05)(0.90)(0.05) && (2.104)\\ &= 0.027075 + 0.002250 = 0.029325 && (2.105)\\ p(B,W,\bar{G}) &= 0.029325 \times 0.095 = 0.002713 && (2.106) \end{aligned}$$

Now denominator:

$$\begin{aligned} p(W,\bar{G}) &= p(W,\bar{G}|B)p(B) + p(W,\bar{G}|\bar{B})p(\bar{B}) && (2.107)\\ p(W,\bar{G}|\bar{B}) &= P(W|A)p(\bar{G}|A)p(A|\bar{B}) && (2.108)\\ &\quad + P(W|\bar{A})p(\bar{G}|\bar{A})p(\bar{A}|\bar{B}) && (2.109)\\ &= (0.95)(0.03)(0.05) + (0.05)(0.90)(0.95) && (2.110)\\ p(W,\bar{G}) &= 0.002713 + 0.389565 = 0.392278 && (2.111) \end{aligned}$$

Therefore,

$$\begin{aligned} p(B|W,\bar{G}) &= \frac{p(B,W,\bar{G})}{p(W,\bar{G})} && (2.112)\\ &= \frac{0.002713}{0.392278} \approx 6.916 \times 10^{-3} && (2.113) \end{aligned}$$

Marginal distribution

Consider a joint probability distribution $P(\theta 1, \theta 2)$. A marginal distribution is obtained by integrating over one parameter,

$$P(\theta_1) = \int P(\theta_1, \theta_2) d\theta_2$$

It gives the probabilities of the variables without reference to the other variables. The contrary of conditional distribution.

For discrete random variables, the marginal probability mass function(PMF) can be written as $P(X = x)$.

$$P(X = x) = \sum_y P(X = x, Y = y) = \sum_y P(X = x|Y = y)P(Y = y)$$

where $P(X = x, Y = y)$ is the joint distribution of X and Y.

Marginal independence

Random variable X is marginally independent of random variable Y if, for all $x_i \in dom(X), y_j \in dom(Y), y_k \in dom(Y)$,,

$$P(X = x_i|Y = y_j) = P(X = x_i|Y = y_k) = P(X = x_i)$$

(NOTE: the differences in *j* and *k*)

That is, knowledge of Y's value doesn't affect your belief in the value of X.

Sample space

- Set of all possible outcomes of an experiment
- Size of the set is **NOT** the sample space
- Outcomes can be sequence of numbers

Discrete sample space

Discrete = listable

e.g.

$$\begin{aligned} a, b, c &\quad \text{(finite)} \\ 0, 1, 2, ... &\quad \text{(infinite)} \end{aligned} \tag{2.113}$$

Independence

- Events A & B are independent if $P(A \cap B) = P(A) \times P(B)$
- Random variables X and Y are independent if $F(x, y) = F_X(x)F_Y(y)$
- Discrete random variables X and Y are independent if $P(x_i, y_j) = P_X(x_i)P_Y(y_j)$
- Continuous random variables X and Y are independent if $f(x, y) = f_X(x)f_Y(y)$
- $cov(X, Y) = 0 \iff E[XY] = E[X]E[Y]$

Covariance and Correlation

> The two are very similar. Both describe the degree to which two random variables or sets of random variables tend to deviate from their expected values in similar ways. - Wikipedia

Covariance

Measures the degree to which two random variables vary together, e.g. height and weight of people.

Random variables X, Y with means μ_x, μ_y.

$$\sigma_{X,Y} cov(X, Y) = E((X - \mu_x)(Y - \mu_y))$$

Properties

- $cov(aX + b, cY + d) = accov(X, Y)$ for constants a, b, c, d
- $cov(X_1 + X_2, Y) = cov(X_1, Y) + cov(X_2, Y)$
- $cov(X, X) = Var(X)$
- $cov(X, Y) = E(XY) - \mu_x \mu_y$
- If X, Y are independent then $Cov(X, Y) = 0$. **Warning**: The converse is not true, when covariance is 0 the variables might not be independent.

Correlation

It's like covariance, but it removes the scale. The population correlation coefficient $\rho_{X,Y}$ between X and Y is defined by

$$\rho_{X,Y} = corr(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

WARNING: It's not causation.

Standardization

$$Y = \frac{X - \mu}{\sigma}$$

- Y has mean 0 and $\sigma_Y = 1$
- Standardizing any normal random variable produces the standard normal.
- If $X \approx normal$, then standardized $X \approx$ standardized normal
- Z: standardized normal random variable.

Random Variables

Random Variable(RV)

$$X : \Omega \longrightarrow \mathbb{R}$$

Probability Mass Function(PMF)

$$f_X(x) = P[X = x] = P[\omega \in \Omega : X(\omega) = x]$$

Probability Density Function(PDF)

$$P[a \leq X \leq b] = \int_a^b f(x)dx$$

Cumulative Distribution Function(CDF)

$$F_X : \mathbb{R} \longrightarrow [0,1] \quad F_X(x) = P[X \leq x]$$

Exchangability & i.i.d

i.i.d(independently and identically distributed) $\Rightarrow$ exchangable

, but

i.i.d(independently and identically distributed) $\nLeftarrow$ exchangable

Coin tossing is a good example; $[P(H,H,T) = P(T,H,H)] \Longrightarrow$ events are independent and their order can be exchanged.

References

2.11 Python

2.11.1 Python cheatsheet

Tuples

Tuple unpacking

```
In [2]: def divmod(x,y):
            return x // y, x % y

        args = (20,8)
        divmod(20, 8) == divmod(*args)
```

Using $\star$ to grab excess items

```
In [8]: a, b, *rest = range(5)
        a, b, rest
```

```
Out[8]: (0, 1, [2, 3, 4])
```

```
In [9]: a, b, *rest = range(3)
        a, b, rest
```

```
Out[9]: (0, 1, [2])
```

```
In [10]: a, b, *rest = range(2)
         a, b, rest
```

```
Out[10]: (0, 1, [])
```

`*` prefix can appear in any position

```
In [12]: *head, a, b = range(5)
         a, b, head
```

```
Out[12]: (3, 4, [0, 1, 2])
```

Nested Tuple unpacking

```
In [13]: metro_areas = [
             ('Tokyo', 'JP', 36.933, (35.689722, 139.691667)),
             (' Delhi NCR', 'IN', 21.935, (28.613889, 77.208889)),
             ('Mexico City', 'MX', 20.142, (19.433333, -99.133333)),
             ('New York-Newark', 'US', 20.104, (40.808611, -74.020386)),
             ('Sao Paulo', 'BR', 19.649, (-23.547778, -46.635833)),
         ]
```

```
In [16]: print('{:15} | {:^9} | {:^9}'.format('', 'lat.', 'long.'))
         fmt = '{:15} | {:9.4f} | {:9.4f}'
         for name, cc, pop, (lat, long) in metro_areas:
             if long <= 0:
                 print(fmt.format(name, lat, long))
```

```
                |   lat.    |   long.
Mexico City     |   19.4333 |  -99.1333
New York-Newark |   40.8086 |  -74.0204
Sao Paulo       |  -23.5478 |  -46.6358
```

Named Tuples

```
In [18]: from collections import namedtuple
```

```
In [19]: City = namedtuple('City', 'name country population coordinates')
```

```
In [20]: tokyo = City('Tokyo', 'JP', 36.933, (35.689722, 133.5846))
```

```
In [21]: tokyo
```

```
Out[21]: City(name='Tokyo', country='JP', population=36.933, coordinates=(35.689722, 133.5846))
```

```
In [23]: tokyo.country
```

```
Out[23]: 'JP'
```

```
In [24]: tokyo[1]
```

```
Out[24]: 'JP'
```

```
In [25]: City._fields
```

```
Out[25]: ('name', 'country', 'population', 'coordinates')
```

```
In [26]: LatLong = namedtuple('LatLong', 'lat long')
```

```
In [28]: delhi_data = ('Delhi NCR', 'IN', 21.935, LatLong(28.354, 77.2))
```

```
In [29]: delhi = City._make(delhi_data);delhi
```

```
Out[29]: City(name='Delhi NCR', country='IN', population=21.935, coordinates=LatLong(lat=28.354, long
```

```
In [30]: delhi._asdict()

Out[30]: OrderedDict([('name', 'Delhi NCR'),
                      ('country', 'IN'),
                      ('population', 21.935),
                      ('coordinates', LatLong(lat=28.354, long=77.2))])

In [32]: for k, v in delhi._asdict().items():
             print(k + ":", v)

name: Delhi NCR
country: IN
population: 21.935
coordinates: LatLong(lat=28.354, long=77.2)
```

Slicing

```
In [33]: s = 'bicycle'
         s[::3]

Out[33]: 'bye'

In [34]: s[::-1]

Out[34]: 'elcycib'

In [35]: s[::-3]

Out[35]: 'eyb'

In [38]: # You can name a slice
         first_two = slice(0,2)
         s[first_two]

Out[38]: 'bi'

In [42]: # You can assign to slices
         numbers = list(range(10))
         numbers[2:5] = [20, 30]; numbers

Out[42]: [0, 1, 20, 30, 5, 6, 7, 8, 9]

In [43]: del numbers[5:7]; numbers

Out[43]: [0, 1, 20, 30, 5, 8, 9]

In [44]: numbers[3::2] = [11, 22]; numbers

Out[44]: [0, 1, 20, 11, 5, 22, 9]

In [46]: numbers[2:5] = [100]; numbers

Out[46]: [0, 1, 100]
```

Python bytecode disassembly tool – `dis`

```
In [50]: import dis

         dis.dis('s[a] += b')

  1           0 LOAD_NAME                0 (s)
              2 LOAD_NAME                1 (a)
              4 DUP_TOP_TWO
              6 BINARY_SUBSCR
              8 LOAD_NAME                2 (b)
```

```
             10 INPLACE_ADD
             12 ROT_THREE
             14 STORE_SUBSCR
             16 LOAD_CONST               0 (None)
             18 RETURN_VALUE
```

Key is brilliant

```
In [51]: l = [28,14,'28',5,1,'1','23',19]
         sorted(l, key=int)
```

```
Out[51]: [1, '1', 5, 14, 19, '23', 28, '28']
```

```
In [52]: sorted(l, key=str)
```

```
Out[52]: [1, '1', 14, 19, '23', 28, '28', 5]
```

Functions as first-class objects

```
In [54]: def factorial(n):
             '''returns n!'''
             return n if n < 2 else factorial(n-1) * n

         factorial(10)
```

```
Out[54]: 3628800
```

```
In [55]: help(factorial)
```

```
Help on function factorial in module __main__:

factorial(n)
    returns n!
```

```
In [56]: [factorial(i) for i in range(10)]
```

```
Out[56]: [0, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880]
```

```
In [61]: list(map(factorial, range(10)))
```

```
Out[61]: [0, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880]
```

Higher-order functions

```
In [57]: def reverse(word):
             return word[::-1]

         reverse('seyoung')
```

```
Out[57]: 'gnuoyes'
```

```
In [59]: fruits = 'banaani mustikka appelsiini omena'.split()
         sorted(fruits, key=reverse)
```

```
Out[59]: ['mustikka', 'omena', 'banaani', 'appelsiini']
```

```
In [62]: sorted(fruits, key=lambda word: word[::-1])
```

```
Out[62]: ['mustikka', 'omena', 'banaani', 'appelsiini']
```

User-defined callable types

```
In [65]: import random

         class BingoCage:

             def __init__(self, items):
                 self._items = list(items)
                 random.shuffle(self._items)

             def pick(self):
                 try:
                     return self._items.pop()
                 except IndexError:
                     raise LookupError('Pick from empty BingoCage')

             def __call__(self):
                 return self.pick()
```

```
In [69]: bingo = BingoCage(range(30))
         bingo(), bingo.pick()
```

```
Out[69]: (17, 7)
```

Decorators

- https://github.com/GrahamDumpleton/wrapt/blob/develop/blog/01-how-you-implemented-your-python-decorator-is-wrong.md

```
In [ ]: '''
        The following two are equivalent
        '''
        class A:
            @classmethod
            def method(cls):
                pass


        class B:
            def method(cls):
                pass
            method classmethod(method)
```

```
In [71]: def deco(func):
             def inner():
                 print('running inner()')
                 func()
             return inner

         @deco
         def target():
             print('running target()')

         target()
```

```
running inner()
running target()
```

Decorator is just a syntatic sugar as passing another function to a function:

```
In [74]: def inner(func):
             print('running inner()')
             func()

         def target():
             print('running target()')


         inner(target)
```

```
running inner()
running target()
```

Decorator – practical usecase

```
In [76]: promos = []

         def promotion(promo_func):
             promos.append(promo_func)
             return promo_func

         '''
         Decorators are run on import time. Thus `promos` will be filled up
         as soon as the module is imported.
         '''

         @promotion
         def fidelity(order):
             '''
             5% discount for customers with 1000+ fidelity points
             '''
             return order.total * .05 if order.customer.fidelity >= 1000 else 0

         @promotion
         def bulk_item(order):
             '''10% discount for each LineItem with 20 or mor units'''
             discount = 0
             for item in order.cart:
                 if item.quantity >= 20:
                     discount += item.total() * .1
             return discount


         def best_promo(order):
             '''Select best discount available'''
             return max(promo(order) for promo in promos)
```

Closure

A closure is a function within a function which accesses nonglobal variable that is declared outside its body

```
In [77]: def make_averager():
             series = []

             def avg(new_value):
                 series.append(new_value)
                 return sum(series) / len(series)
```

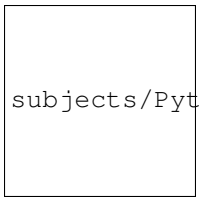


Fig. 59: closure_variable_scope

```
            return avg

        avg = make_averager()
        avg(10), avg(11), avg(12)
```

```
Out[77]: (10.0, 10.5, 11.0)
```

```
In [78]: avg.__code__.co_varnames
```

```
Out[78]: ('new_value',)
```

```
In [79]: avg.__code__.co_freevars
```

```
Out[79]: ('series',)
```

```
In [80]: avg.__closure__
```

```
Out[80]: (<cell at 0x1118e90a8: list object at 0x112697d88>,)
```

```
In [82]: avg.__closure__[0].cell_contents
```

```
Out[82]: [10, 11, 12]
```

Nonlocal declaration

```
In [83]: def make_averager():
             count = 0
             total = 0

             def avg(new_value):
                 nonlocal count, total
                 count += 1
                 total += new_value
                 return total / count

             return avg
```

Runtime measure decorator

```
In [35]: import time
         import functools

         def clock(func):
             @functools.wraps(func)
             def clocked(*args, **kwargs):
                 t0 = time.perf_counter()
                 result = func(*args, **kwargs)
                 elapsed = time.perf_counter() - t0
                 name = func.__name__
```

```
                arg_list = []
                if args:
                    arg_list.append(', '.join(repr(arg) for arg in args))
                if kwargs:
                    pairs = ['%s=%r' % (k, w) for k, w in sorted(kwargs.items())]
                    arg_list.append(', '.join(pairs))
                arg_str = ', '.join(arg_list)
                print('[{:.8f}s] {:s}({:s}) -> {}'.format(
                    elapsed,
                    name,
                    arg_str,
                    result
                ))
                return result
            return clocked
```

```
In [53]: @clock
         def snooze(seconds):
             time.sleep(seconds)

         @clock
         def factorial(n):
             '''returns n!'''
             return n if n < 2 else factorial(n-1) * n

         snooze(1)
```

```
[1.00002574s] snooze(1) -> None
```

```
In [44]: factorial(10)
```

```
Out[44]: 3628800
```

Unwrapping a decorator

```
In [54]: snooze.__wrapped__(2)
```

Momoization decorator

```
In [103]: '''
          @functools.lru_cache(maxsize=128, typed=False)
          maxsize should be a power of 2 for optimal performance.
          '''
          @functools.lru_cache(maxsize=128, typed=False)
          @clock
          def fibonacci(n):
              if n < 2:
                  return n
              return fibonacci(n-2) + fibonacci(n-1)

          fibonacci(100)
```

```
[0.00000047s] fibonacci(0) -> 0
[0.00000062s] fibonacci(1) -> 1
[0.00018753s] fibonacci(2) -> 1
[0.00000126s] fibonacci(3) -> 2
[0.00023754s] fibonacci(4) -> 3
```

```
[0.00000094s] fibonacci(5) -> 5
[0.00028544s] fibonacci(6) -> 8
[0.00000091s] fibonacci(7) -> 13
[0.00033311s] fibonacci(8) -> 21
[0.00000088s] fibonacci(9) -> 34
[0.00038029s] fibonacci(10) -> 55
[0.00000106s] fibonacci(11) -> 89
[0.00042941s] fibonacci(12) -> 144
[0.00000100s] fibonacci(13) -> 233
[0.00047744s] fibonacci(14) -> 377
[0.00000095s] fibonacci(15) -> 610
[0.00052555s] fibonacci(16) -> 987
[0.00000098s] fibonacci(17) -> 1597
[0.00057387s] fibonacci(18) -> 2584
[0.00000093s] fibonacci(19) -> 4181
[0.00062187s] fibonacci(20) -> 6765
[0.00000091s] fibonacci(21) -> 10946
[0.00067036s] fibonacci(22) -> 17711
[0.00000106s] fibonacci(23) -> 28657
[0.00071860s] fibonacci(24) -> 46368
[0.00000088s] fibonacci(25) -> 75025
[0.00076595s] fibonacci(26) -> 121393
[0.00000088s] fibonacci(27) -> 196418
[0.00081273s] fibonacci(28) -> 317811
[0.00000088s] fibonacci(29) -> 514229
[0.00085992s] fibonacci(30) -> 832040
[0.00000092s] fibonacci(31) -> 1346269
[0.00090667s] fibonacci(32) -> 2178309
[0.00000092s] fibonacci(33) -> 3524578
[0.00095364s] fibonacci(34) -> 5702887
[0.00000090s] fibonacci(35) -> 9227465
[0.00100009s] fibonacci(36) -> 14930352
[0.00000097s] fibonacci(37) -> 24157817
[0.00104787s] fibonacci(38) -> 39088169
[0.00000089s] fibonacci(39) -> 63245986
[0.00109493s] fibonacci(40) -> 102334155
[0.00000104s] fibonacci(41) -> 165580141
[0.00114280s] fibonacci(42) -> 267914296
[0.00000092s] fibonacci(43) -> 433494437
[0.00119226s] fibonacci(44) -> 701408733
[0.00000107s] fibonacci(45) -> 1134903170
[0.00124033s] fibonacci(46) -> 1836311903
[0.00000107s] fibonacci(47) -> 2971215073
[0.00128768s] fibonacci(48) -> 4807526976
[0.00000105s] fibonacci(49) -> 7778742049
[0.00133551s] fibonacci(50) -> 12586269025
[0.00000108s] fibonacci(51) -> 20365011074
[0.00149767s] fibonacci(52) -> 32951280099
[0.00000117s] fibonacci(53) -> 53316291173
[0.00154963s] fibonacci(54) -> 86267571272
[0.00000100s] fibonacci(55) -> 139583862445
[0.00159663s] fibonacci(56) -> 225851433717
[0.00000101s] fibonacci(57) -> 365435296162
[0.00164442s] fibonacci(58) -> 591286729879
[0.00000104s] fibonacci(59) -> 956722026041
[0.00169188s] fibonacci(60) -> 1548008755920
[0.00000101s] fibonacci(61) -> 2504730781961
[0.00173856s] fibonacci(62) -> 4052739537881
[0.00000097s] fibonacci(63) -> 6557470319842
```

```
[0.00178615s] fibonacci(64) -> 10610209857723
[0.00000098s] fibonacci(65) -> 17167680177565
[0.00183278s] fibonacci(66) -> 27777890035288
[0.00000101s] fibonacci(67) -> 44945570212853
[0.00187957s] fibonacci(68) -> 72723460248141
[0.00000108s] fibonacci(69) -> 117669030460994
[0.00192640s] fibonacci(70) -> 190392490709135
[0.00000095s] fibonacci(71) -> 308061521170129
[0.00197353s] fibonacci(72) -> 498454011879264
[0.00000098s] fibonacci(73) -> 806515533049393
[0.00202084s] fibonacci(74) -> 1304969544928657
[0.00000100s] fibonacci(75) -> 2111485077978050
[0.00303772s] fibonacci(76) -> 3416454622906707
[0.00000149s] fibonacci(77) -> 5527939700884757
[0.00316910s] fibonacci(78) -> 8944394323791464
[0.00000113s] fibonacci(79) -> 14472334024676221
[0.00321860s] fibonacci(80) -> 23416728348467685
[0.00000100s] fibonacci(81) -> 37889062373143906
[0.00326648s] fibonacci(82) -> 61305790721611591
[0.00000097s] fibonacci(83) -> 99194853094755497
[0.00331421s] fibonacci(84) -> 160500643816367088
[0.00000098s] fibonacci(85) -> 259695496911122585
[0.00336403s] fibonacci(86) -> 420196140727489673
[0.00000106s] fibonacci(87) -> 679891637638612258
[0.00341073s] fibonacci(88) -> 1100087778366101931
[0.00000110s] fibonacci(89) -> 1779979416004714189
[0.00345798s] fibonacci(90) -> 2880067194370816120
[0.00000103s] fibonacci(91) -> 4660046610375530309
[0.00350508s] fibonacci(92) -> 7540113804746346429
[0.00000098s] fibonacci(93) -> 12200160415121876738
[0.00355214s] fibonacci(94) -> 19740274219868223167
[0.00000095s] fibonacci(95) -> 31940434634990099905
[0.00359954s] fibonacci(96) -> 51680708854858323072
[0.00000101s] fibonacci(97) -> 83621143489848422977
[0.00364663s] fibonacci(98) -> 135301852344706746049
[0.00000094s] fibonacci(99) -> 218922995834555169026
[0.00369384s] fibonacci(100) -> 354224848179261915075
```

```
Out[103]: 354224848179261915075
```

`singledispatch` for function overloading

```
In [110]: from functools import singledispatch
          from collections import abc
          import numbers
          import html

          @singledispatch
          def example_func(object):
              print("any: ", object)

          @example_func.register(str)
          def _(text):
              print("str: ", text)

          @example_func.register(int)
          def _(number):
              print("int: ", number)
```

```
@example_func.register(tuple)
@example_func.register(list)
def _(number):
    print("sequence ", number)


example_func(123)
example_func('123')
example_func([123,543])
example_func((123,543))
example_func(23.4)
```

```
int:  123
str:  123
sequence  [123, 543]
sequence  (123, 543)
any:  23.4
```

Parameterized clock decorator

Notice the parenthesis after `@clock` below.

```
In [2]: import time

        DEFAULT_FMT = '[{elapsed:0.8f}s] {name}({args}) -> {result}'

        def clock(fmt=DEFAULT_FMT):
            def decorate(func):
                def clocked(*_args):
                    t0 = time.time()
                    _result = func(*_args)
                    elapsed = time.time() - t0
                    name = func.__name__
                    args = ', '.join(repr(arg) for arg in _args)
                    result = repr(_result)
                    print(fmt.format(**locals()))
                    return _result
                return clocked
            return decorate

        @clock()
        def snooze(seconds):
            time.sleep(seconds)

        for _ in range(3):
            snooze(.123)

        print("\nWith a custom argument\n")

        @clock('{name}: {elapsed}s')
        def snooze(seconds):
            time.sleep(seconds)

        for _ in range(3):
            snooze(.123)
```

```
[0.12447476s] snooze(0.123) -> None
[0.12471581s] snooze(0.123) -> None
[0.12503409s] snooze(0.123) -> None

With a custom argument

snooze: 0.12709403038024902s
snooze: 0.12314391136169434s
snooze: 0.12816691398620605s
```

Parameterized clock decorator with `partial` for removing `()`

Notice the parenthesis after `@clock` is GONE below.

```
In [3]: import time
        from functools import wraps, partial
        DEFAULT_FMT = '[{elapsed:0.8f}s] {name}({args}) -> {result}'

        def clock(func=None, fmt=DEFAULT_FMT):
            if not func:
                return partial(clock, fmt=fmt)

            @wraps(func)
            def clocked(*_args):
                t0 = time.time()
                _result = func(*_args)
                elapsed = time.time() - t0
                name = func.__name__
                args = ', '.join(repr(arg) for arg in _args)
                result = repr(_result)
                print(fmt.format(**locals()))
                return _result
            return clocked

        @clock
        def snooze(seconds):
            time.sleep(seconds)

        for _ in range(3):
            snooze(0.123)

        print("\nWith a custom argument\n")

        @clock(fmt='{name}: {elapsed}s')
        def snooze(seconds):
            time.sleep(seconds)

        for _ in range(3):
            snooze(.123)
```

```
[0.12613082s] snooze(0.123) -> None
[0.12330794s] snooze(0.123) -> None
[0.12451982s] snooze(0.123) -> None

With a custom argument

snooze: 0.12448596954345703s
snooze: 0.12357401847839355s
```

```
snooze: 0.12677001953125s
```

Logger decorator

```
In [1]: from functools import wraps
        import logging

        logging.basicConfig(
            filename='code/example.log',
            format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
            level=logging.DEBUG
        )

        def logged(level, name=None, message=None):
            def decorate(func):
                logname = name if name else func.__module__
                logger = logging.getLogger(logname)
                logmsg = message if message else func.__name__

                @wraps(func)
                def wrapper(*args, **kwargs):
                    logger.log(level, logmsg)
                    return func(*args, **kwargs)
                return wrapper
            return decorate

        @logged(logging.DEBUG)
        def add(x, y):
            return x + y

        @logged(logging.CRITICAL, 'example')
        def mul(x, y):
            return x * y

        add(2, 34), mul(5,3)
```

```
Out[1]: (36, 15)
```

Enforcing type checking on a function using a decorator

```
In [9]: from inspect import signature

        def typeassert(*ty_args, **ty_kwargs):
            def decorate(func):
                # If in optimized mode, disable type checking
                if not __debug__:
                    return func

                # Map function argument names to supplied types
                sig = signature(func)
                bound_types = sig.bind_partial(*ty_args, **ty_kwargs).arguments

                @wraps(func)
                def wrapper(*args, **kwargs):
                    bound_values = sig.bind(*args, **kwargs)
                    # Enforce type assertion across supplied arguments
                    for name, value in bound_values.arguments.items():
```

```
                        if name in bound_types:
                            if not isinstance(value, bound_types[name]):
                                raise TypeError(
                                    'Argument {} must be {}'.format(
                                        name,
                                        bound_types[name]
                                    )
                                )
                        return func(*args, **kwargs)
                    return wrapper
                return decorate


        @typeassert(int, z=int)
        def spam(x,y,z=42):
            print(x,y,z)

        spam(1,2,3)
        spam('1',2,3)
        spam(1,'23',3)
        spam(1,'23','23')
```

```
1 2 3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-c3a04188cd07> in <module>()
     37
     38 spam(1,2,3)
---> 39 spam('1',2,3)
     40 spam(1,'23',3)
     41 spam(1,'23','23')

<ipython-input-9-c3a04188cd07> in wrapper(*args, **kwargs)
     21                                 'Argument {} must be {}'.format(
     22                                     name,
---> 23                                     bound_types[name]
     24                                 )
     25                             )

TypeError: Argument x must be <class 'int'>
```

Anatomy of a function wrapper with a class v.1

```
class function_wrapper(object):
    def __init__(self, wrapped):
        self.wrapped = wrapped
    def __call__(self, *args, **kwargs):
        return self.wrapped(*args, **kwargs)

@function_wrapper
def function():
    pass
```

```
In [6]: import functools

        class function_wrapper(object):
            def __init__(self, wrapped):
```

```
                self.wrapped = wrapped
                functools.update_wrapper(self, wrapped)

            def __call__(self, *args, **kwargs):
                print("wrapper")
                return self.wrapped(*args, **kwargs)

        @function_wrapper
        def function():
            print("actual func")

        function()
```

```
wrapper
actual func
```

Anatomy of a function wrapper with a class v.2

```
In [18]: import types
         from functools import wraps

         class function_wrapper:
             ncalls = 0
             def __init__(self, wrapped):
                 wraps(wrapped)(self)
         #         self.ncalls = 0

             def __call__(self, *args, **kwargs):
                 function_wrapper.ncalls += 1
                 print("wrapper")
         #         self.ncalls += 1
         #         ncalls += 1
                 return self.__wrapped__(*args, **kwargs)

             def __get__(self, instance, cls):
                 '''
                 This method is necessary when using a class decorator
                 to an instance method.

                 Maybe it is just better to use a closure decorator.
                 '''
                 if not instance:
                     return self
                 else:
                     return types.MethodType(self, instance)

         @function_wrapper
         def function():
             print("actual func")

         function()
         function.ncalls
```

```
wrapper
actual func
```

```
Out[18]: 1
```

Applying decorators to Class and

Proper decorators

```
In [32]: import functools

         '''
         Factory for creating decorators
         '''
         class object_proxy(object):

             def __init__(self, wrapped):
                 self.wrapped = wrapped
                 try:
                     self.__name__ = wrapped.__name__
                 except AttributeError:
                     pass

             @property
             def __class__(self):
                 return self.wrapped.__class__

             def __getattr__(self, name):
                 return getattr(self.wrapped, name)


         class bound_function_wrapper(object_proxy):

             def __init__(self, wrapped, instance, wrapper):
                 super(bound_function_wrapper, self).__init__(wrapped)
                 self.instance = instance
                 self.wrapper = wrapper

             def __call__(self, *args, **kwargs):
                 if self.instance is None:
                     instance, args = args[0], args[1:]
                     wrapped = functools.partial(self.wrapped, instance)
                     return self.wrapper(wrapped, instance, args, kwargs)
                 return self.wrapper(self.wrapped, self.instance, args, kwargs)


         class function_wrapper(object_proxy):

             def __init__(self, wrapped, wrapper):
                 super(function_wrapper, self).__init__(wrapped)
                 self.wrapper = wrapper

             def __get__(self, instance, owner):
                 '''
                  If the wrapper is applied to a method of a class,
                  the __get__() method is called when the attribute is
                  accessed, which returns a new bound wrapper and the
                  __call__() method of that is invoked instead when a
                  call is made.
                 '''
                 wrapped = self.wrapped.__get__(instance, owner)
                 return bound_function_wrapper(wrapped, instance, self.wrapper)
```

```
            def __call__(self, *args, **kwargs):
                return self.wrapper(self.wrapped, None, args, kwargs)


        def decorator(wrapper):
            '''
            A decorator for creating decorators
            '''
            @functools.wraps(wrapper)
            def _decorator(wrapped):
                print("_decorator: wrapped: ", wrapped.__name__)
                print("_decorator: wrapper: ", wrapper.__name__)
                return function_wrapper(wrapped, wrapper)
            return _decorator
```

```
In [33]: @decorator
         def my_function_wrapper(wrapped, instance, args, kwargs):
             print('INSTANCE', instance)
             print('ARGS', args)
             return wrapped(*args, **kwargs)

         @my_function_wrapper
         def function(a, b):
             pass

         function(3,24)

         # class Class(object):
         #     @my_function_wrapper
         #     def function_im(self, a, b):
         #         pass

         # c = Class()  # __get__ is called

         # c.function_im(1,2)
```

```
_decorator: wrapped:  function
_decorator: wrapper:  my_function_wrapper
INSTANCE None
ARGS (3, 24)
```

```
In [34]: class Class(object):
             @my_function_wrapper
             def function_im(self, a, b):
                 pass

         c = Class()  # __get__ is called
         Class.function_im(c, 1, 2)

         # c.function_im(1,2)
```

```
_decorator: wrapped:  function_im
_decorator: wrapper:  my_function_wrapper
INSTANCE <__main__.Class object at 0x106c92e10>
ARGS (1, 2)
```

```
In [35]: def decorator(wrapper):
             @functools.wraps(wrapper)
             def _decorator(wrapped):
                 return function_wrapper(wrapped, wrapper)
             return _decorator
```

```
        def optional_arguments(wrapped=None, arg=1):
            if wrapped is None:
                return functools.partial(optional_arguments, arg=arg)

            @decorator
            def _wrapper(wrapped, instance, args, kwargs):
                return wrapped(*args, **kwargs)

            return _wrapper(wrapped)

        @optional_arguments(arg=2)
        def function1():
            pass

        @optional_arguments
        def function2():
            pass

        function1()
        function2()
```

```
In [36]: import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

```
In [39]: x = 4
         y = 2
         if x == 4: print(x, y); x, y = y, x
```

```
4 2
```

```
In [42]: def powOfTwo(n: int = 2) -> int:
             return n * n

         powOfTwo(5), powOfTwo()
```

```
Out[42]: (25, 4)
```

Getter and Setter with `@property`

```
In [45]: class Person:
             def __init__(self, age):
                 self.age = age

             @property
             def value(self):
                 return 80 - self.age

             @value.setter
             def value(self, age):
                 self.age = age

         jack = Person(10)
         print(jack.value)
         jack.value = 12
         print(jack.value)
```

```
70
68
```

```
In [21]: 'asdf'.suffix(1)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-21-6e590c9db7d1> in <module>()
----> 1 'asdf'.suffix(1)

AttributeError: 'str' object has no attribute 'suffix'
```

```
In [16]: class TimemachineServerMonitor:

             def __init__(self):
                 pass

             @property
             def timestamp(self):
                 return format(datetime.now(), '%Y/%B/%d %A %H:%M:%S')

         t = TimemachineServerMonitor()
         t.timestamp
```

```
Out[16]: '2018/July/06 Friday 12:38:47'
```

Getter & Setter using Descriptor

```
In [30]: '''
         Getter & Setter using Descriptor. With WeakKeyDictionary, it's free from memory leak
         '''

         from weakref import WeakKeyDictionary


         class Grade(object):
             def __init__(self):
                 self._value = WeakKeyDictionary()

             def __get__(self, instance, instance_type):
```

```
            if not instance: return self
            return self._value.get(instance, 0)

        def __set__(self,instance, value):
            if not (0 <= value <= 100):
                raise ValueError
            self._value[instance] = value

    class Exam:
        math_grade = Grade()
        writing_grade = Grade()
        science_grade = Grade()

    exam = Exam()

    exam.writing_grade = 40;
    print(exam.writing_grade)

    exam2 = Exam()
    print(exam2.writing_grade)
```

```
40
0
```

```
In [31]: seyoung = Exam()
```

```
In [32]: seyoung.math_grade = 100; seyoung.math_grade
```

```
Out[32]: 100
```

```
In [33]: fabio = Exam(); fabio.math_grade
```

```
Out[33]: 0
```

```
In [34]: my_grade = Grade()
```

Datetime

```
In [11]: from datetime import datetime
         format(datetime.now(), '%Y/%B/%d %A %H:%M')
```

```
Out[11]: '2018/July/06 Friday 12:34'
```

```
In [ ]:
```

2.11.2 Dictionaries & Sets

```
In [1]: # How to initialize a dict
        a = dict(one=1, two=2, three=3)
        b = {'one': 1, 'two': 2, 'three': 3}
        c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
        d = dict([('two', 2), ('one', 1), ('three', 3)])
        e = dict({'three': 3, 'one': 1, 'two': 2})
        a == b == c == d == e
```

```
Out[1]: True
```

Dict comprehension

```
In [2]: DIAL_CODES = [
            (86, 'China'),
            (91, 'India'),
            (1, 'United States'),
            (62, 'Indonesia'),
            (55, 'Brazil'),
            (92, 'Pakistan'),
            (880, 'Bangladesh'),
            (234, 'Nigeria'),
            (7, 'Russia'),
            (81, 'Japan'),
        ]
        country_code = {country: code for code, country in DIAL_CODES}
        country_code
```

```
Out[2]: {'China': 86,
         'India': 91,
         'United States': 1,
         'Indonesia': 62,
         'Brazil': 55,
         'Pakistan': 92,
         'Bangladesh': 880,
         'Nigeria': 234,
         'Russia': 7,
         'Japan': 81}
```

```
In [3]: {code: country.upper() for country, code in country_code.items() if code < 66}
```

```
Out[3]: {1: 'UNITED STATES', 62: 'INDONESIA', 55: 'BRAZIL', 7: 'RUSSIA'}
```

collections

```
In [6]: import collections

        ct = collections.Counter('abracadabra')
        print(ct)
        ct.update('aaaaazzz');
        print(ct)
        ct.most_common(2)
```

```
Counter({'a': 5, 'b': 2, 'r': 2, 'c': 1, 'd': 1})
Counter({'a': 10, 'z': 3, 'b': 2, 'r': 2, 'c': 1, 'd': 1})
```

```
Out[6]: [('a', 10), ('z', 3)]
```

Sets

Find occurrences

```
In [8]: found = 0
        for n in needles:
            if n in haystack:
                found += 1
```

The above does the same thing as the below. The above is faster than below but the below works on any iterables.

```
In [ ]: found = len(set(needles) & set(haystack))
        # another way:
        found = len(set(needles).intersection(haystack))
```

Immutable sets

```
In [9]: frozenset(range(10))
Out[9]: frozenset({0, 1, 2, 3, 4, 5, 6, 7, 8, 9})
```

Set Comprehension

```
In [10]: {i for i in range(10)}
Out[10]: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
In [11]:
Out[11]: True
In [ ]:
```

2.11.3 Python – Web Stuffs

Start a filesystem based web server

```
python -m http.server

# To override default parameters:
python -m http.server 8000 --bind 127.0.0.1
```

```
In [ ]:
```

2.11.4 Python Analysis tools

Bytecode inspection

```
In [1]: from dis import dis
        dis('{1}')
  1           0 LOAD_CONST               0 (1)
              2 BUILD_SET                1
              4 RETURN_VALUE

In [2]: dis('set([1])')
  1           0 LOAD_NAME                0 (set)
              2 LOAD_CONST               0 (1)
              4 BUILD_LIST               1
              6 CALL_FUNCTION            1
              8 RETURN_VALUE
```

Breakpoint debuggin

```
In [4]: import pdb

        # Put the following where you want to break
        # pdb.set_trace()
```

```
In [ ]:
```

2.11.5 Data Visualization in Python

```
In [2]: import plotly as py
        import plotly.graph_objs as go
        import ipywidgets as widgets
        import numpy as np
        from scipy import special

        py.offline.init_notebook_mode(connected=True)
```

```
In [21]: x = np.linspace(0, np.pi, 1000)

         layout = go.Layout(
             title="Simple Example",
             yaxis=dict(
                 title='volts',
             ),
             xaxis=dict(
                 title='nanoseconds'
             )
         )
         data=[

         ]
         trace1 = go.Scatter(
             x=x,
             y=np.cos(x),
             mode='lines',
             name='bessel {}'.format(x),
             line=dict(
                 shape='spline'
             )
         )

         data.append(trace1)

         fig = go.Figure(data=data, layout=layout)
         py.offline.iplot(fig)
         #     py.plotly.iplot(fig)

         # signals = widgets.SelectMultiple(options=list(range(6)), value=(0,), description='Bessel
         # freq = widgets.FloatSlider(min=1, max=20,value=1, description='Freq')
         # widgets.interactive(update_plot, signals=signals, freq=freq)
```

Data type cannot be displayed: text/html, application/vnd.plotly.v1+json, text/vnd.plotly.v1+html

```
In [17]: x = np.linspace(0, np.pi, 1000)
```

```
        layout = go.Layout(
            title="Simple Example",
            yaxis=dict(
                title='volts',
            ),
            xaxis=dict(
                title='nanoseconds'
            )
        )

        def update_plot(signals, freq):

            data = []
            for s in signals:
                trace1 = go.Scatter(
                    x=x,
                    y=special.jv(s, freq * x),
                    mode='lines',
                    name='bessel {}'.format(s),
                    line=dict(
                        shape='spline'
                    )
                )

                data.append(trace1)

            fig = go.Figure(data=data, layout=layout)
            py.offline.iplot(fig)
        #     py.plotly.iplot(fig)

        signals = widgets.SelectMultiple(options=list(range(6)), value=(0,), description='Bessel Orc
        freq = widgets.FloatSlider(min=1, max=20,value=1, description='Freq')
        widgets.interactive(update_plot, signals=signals, freq=freq)
```

```
interactive(children=(SelectMultiple(description='Bessel Order', index=(0,), options=(0, 1, 2, 3, 4,
```

```
In [ ]:
```

2.12 UNIX

2.12.1 Backup Server

This docs is about building a desktop backup server. Under progress atm.

macOS

Host(Ubuntu zfs) with Netatalk 2

Dependencies

- netatalk
- zfs-utils

Steps

We should remember that some zfs file system options cannot be changed once created.

```
# First figure out your target disks
sudo lsblk -o NAME,FSTYPE,SIZE,MOUNTPOINT,LABEL

# Create a pool
zpool create -f timemachine_backup_zpool /dev/sda ...
zpool status
zfs list

# Create a fs
zfs create timemachine_backup_zpool/fs
zfs list

# https://linux.die.net/man/8/zfs
zfs create -o normalization=formD -o nbmand=off -o utf8only=on -o␣
↪aclinherit=passthrough

# Properties of the fs
zfs get all timemachine_backup_zpool/fs

# Set properties of the fs
zfs set quota=500G timemachine_backup_zpool/fs
zfs set compression=on timemachine_backup_zpool/fs

chown danny timemachine_backup_zpool/fs

# Install netatalk
sudo apt-get install netatalk

sudo vim /etc/netatalk/AppleVolumes.default
# Edit the ending.
# :DEFAULT: options:upriv,usedots
# /home/danny/tm "Danny's Time Machine" options:tm allow:danny

sudo service netatalk restart
```

ZFS setup

From ZFS Linux mail list:

> On ext4 (or ZFS with normalization=none), you get two files that appear to have the same name, because one is NFC (uses LATIN SMALL LETTER E WITH ACUTE) and the other is NFD (LATIN SMALL LETTER E, COMBINING ACUTE ACCENT). If you use any of the other normalization options on ZFS, they will be treated as the same file. This seems like a desirable behavior to me, and should help avoid interoperability problems across systems which generally use different normal forms (e.g. Linux vs. OS X).

```
zfs create -o normalization=formD
```

In macOS, options `-O casesensitivity=insensitive -O normalization=formD` are essential.

```
sudo zpool create timemachine_pool -f \
-o ashift=12 \
```

(continues on next page)

(continued from previous page)

```
 /dev/sda /dev/sdb /dev/sdc

sudo zfs create timemachine_pool/user \
-o nbmand=off \
-o utf8only=on \
-o aclinherit=passthrough \
-o compression=lz4 \
-o casesensitivity=insensitive \
-o atime=off \
-o normalization=formD \
-o quota=300G


zfs create -o normalization=formD -o nbmand=off -o utf8only=on -o␣
↪aclinherit=passthrough

-o ashift=12 # For advanced. http://louwrentius.com/zfs-performance-and-capacity-
↪impact-of-ashift9-on-4k-sector-drives.html
```

```
#!/bin/bash

USER=$1

# Test user exists
id $USER

zfs create timemachine_pool/$USER \
  -o nbmand=off \
  -o utf8only=on \
  -o aclinherit=passthrough \
  -o compression=lz4 \
  -o casesensitivity=insensitive \
  -o atime=off \
  -o normalization=formD \
  -o quota=300G


chown -R $USER /timemachine_pool/$USER

chmod 700 /timemachine_pool/$USER
```

- compression=lz4, which not only saves space, but is faster as well. Loading a file from even an SSD is slow, decompressing it the CPU faster. So, the reduced file size helps loading it faster, while the time needed for decompression is still smaller, resulting in overall lesser time used. Follow this link for experimental results.
- atime=off switches of the access time file attribute. Otherwise every time a file is read the access time would be set to the current date, issuing an unnecessary write (wearing down the hard drive and endangering the file).
- `ashift=12` This specifies that your disk is Advanced Format, which is the same as saying it has 4096 byte sectors instead of the old 512 byte sectors. Most disks made after 2011 are advanced format so you'll need this option most of the time. If you forget, ZFS assumes the sector size is 512. If that's the wrong answer, you'll take a big performance hit.

Here's a script that automates the filesystem creation and acl update. Use with `./setup.sh username`.

```
#!/bin/bash
# setup.sh
```

(continues on next page)

(continued from previous page)

```
username=$1

zfs create timemachine_pool/$username \
  -o nbmand=off \
  -o utf8only=on \
  -o aclinherit=passthrough \
  -o compression=lz4 \
  -o casesensitivity=insensitive \
  -o atime=off \
  -o normalization=formD \
  -o quota=500G

chown -R $username /timemachine_pool/$username
```

Netatalk setup

Set logging

```
# Put this at the end of /etc/netatalk/afpd.conf
-setuplog "default log_info /var/log/afpd.log"

service netatalk restart

tail -f /var/log/afpd.log
```

To set number of clients, edit `/etc/default/netatalk` and restart Netatalk.

```
# /etc/default/netatalk
# Netatalk 2.x configuration

################################################################################
# Global configuration
################################################################################

#### machine's AFPserver/AppleTalk name.
#ATALK_NAME=machinename

#### server (unix) and legacy client (<= Mac OS 9) charsets
ATALK_UNIX_CHARSET='LOCALE'
ATALK_MAC_CHARSET='MAC_ROMAN'

#### Don't Edit. export the charsets, read form ENV by apps
export ATALK_UNIX_CHARSET
export ATALK_MAC_CHARSET

################################################################################
# AFP specific configuration
################################################################################

#### Set which daemons to run.
#### If you use AFP file server, run both cnid_metad and afpd.
CNID_METAD_RUN=yes
AFPD_RUN=yes
```

(continues on next page)

(continued from previous page)

```
#### maximum number of clients that can connect:
AFPD_MAX_CLIENTS=50

#### UAMs (User Authentication Modules)
#### available options: uams_dhx.so, uams_dhx2.so, uams_guest.so,
####                    uams_clrtxt.so(legacy), uams_randnum.so(legacy)
#AFPD_UAMLIST="-U uams_dhx2.so,uams_clrtxt.so"

#### Set the id of the guest user when using uams_guest.so
#AFPD_GUEST=nobody

#### config for cnid_metad. Default log config:
#CNID_CONFIG="-l log_note"

########################################################################
# AppleTalk specific configuration (legacy)
########################################################################

#### Set which legacy daemons to run.
#### If you need AppleTalk, run atalkd.
#### papd, timelord and a2boot are dependent upon atalkd.
#ATALKD_RUN=no
#PAPD_RUN=no
#TIMELORD_RUN=no
#A2BOOT_RUN=no

#### Control whether the daemons are started in the background.
#### If it is dissatisfied that legacy atalkd starts slowly, set "yes".
#### In case using systemd/systemctl, this is not so significant.
#ATALK_BGROUND=no

#### Set the AppleTalk Zone name.
#### NOTE: if your zone has spaces in it, you're better off specifying
####       it in atalkd.conf
#ATALK_ZONE=@zone
```

After the edit, you run `service netatalk status` and you see `-c 50` which sets the max num. of clients as 50.

```
 netatalk.service
   Loaded: loaded (/etc/init.d/netatalk; bad; vendor preset: enabled)
   Active: active (running) since Thu 2018-03-08 12:45:10 EET; 46s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 953 ExecStart=/etc/init.d/netatalk start (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/netatalk.service
           ├─974 /usr/sbin/cnid_metad -l log_note
           └─986 /usr/sbin/afpd -U uams_dhx2.so,uams_clrtxt.so -g nobody -c 50 -n␣
↪TimeMachine
```

Client(Macs)

Backup

- Set the destination via GUI in order to set encryption on

  - Mount via Finder. (e.g. afp://gorilla.org.gakkou.fi)
  - Then set it as the destination
- You can set via command line but you won't be able to encrypt the backup.
  - `sudo tmutil setdestination -p "afp://danny@gorilla.org.gakkou.fi/Danny's Time Machine"`
  - The name specified in `/etc/netatalk/AppleVolumes.default` should be given.

Restore

Enter the Backup from Time Machine in Recovery Mode.

Set source as `afp://username@gorilla.org.gakkou.fi/tm`. The final directory which you defined to share in `/etc/netatalk/AppleVolumes.default` should be given after the ip address. If you cannot check the server you could try when mounted,

```
# Show all mounted disks
mount
```

Miscellaneous

sudo apt-get install netatalk avahi-daemon sudo adduser danny

mkdir -R /home/danny/tm/ sudo chown -R danny /home/danny/tm/

sudo vim /etc/nsswitch.conf hosts: files mdns4_minimal [NOTFOUND=return] dns mdns4 mdns

sudo vim /etc/avahi/services/afpd.service

<?xml version="1.0" standalone="no"?> <!DOCTYPE service-group SYSTEM "avahi-service.dtd">

<service-group> <name replace-wildcards="yes">%h</name>

<service> <type>_device-info._tcp</type> <port>0</port> <txt-record>model=Aalto Time Machine Beta</txt-record>

</service>

</service-group>

sudo vim /etc/avahi/services/smb.service

<?xml version="1.0" standalone='no'?><!–-*nxml*–-> <!DOCTYPE service-group SYSTEM "avahi-service.dtd">

<service-group> <name replace-wildcards="yes">%h</name> <service>

<type>_smb._tcp</type> <port>445</port>

</service> <service>

<type>_device-info._tcp</type> <port>0</port> <txt-record>model=AaltoTMTest</txt-record>

</service>

</service-group>

sudo service avahi-daemon restart

---

docker run -dt -v /l/backup_server_tools/smb.conf:/etc/samba/smb.conf -v /timemachine_backup_zpool/parks1/dozer:/dozer -v /l/backup_server_tools/share:/share -p 445:445 –name samba –restart=always stanback/alpine-samba

docker run -dt -v /home/leon/smb.conf:/etc/samba/smb.conf -v /timemachine_backup_zpool/parks1/dozer:/dozer -v /timemachine_backup_zpool/parks1/share:/share -p 445:445 –name samba –restart=always stanback/alpine-samba

docker run -d -v /l/backup_server_tools/services:/etc/avahi/services –net=host –name=avahi –restart=always stanback/alpine-avahi

[global] workgroup = WORKGROUP server string = %h server (Samba, Alpine) security = user map to guest = Bad User encrypt passwords = yes load printers = no printing = bsd printcap name = /dev/null disable spoolss = yes disable netbios = yes server role = standalone server services = -dns, -nbt smb ports = 445 name resolve order = hosts ;log level = 3 create mask = 0664 directory mask = 0775 veto files = /.DS_Store/ nt acl support = no inherit acls = yes ea support = yes vfs objects = catia fruit streams_xattr recycle acl_xattr:ignore system acls = yes recycle:repository = .recycle recycle:keeptree = yes recycle:versions = yes

[Dozer] path = /timemachine_backup_zpool/parks1/dozer comment = ZFS browseable = yes writable = yes valid users = leon

[Shared] path = /timemachine_backup_zpool/parks1/share comment = Shared Folder browseable = yes read only = yes write list = leon guest ok = yes

Linux

References

2.12.2 Docker

Containerization means, that your application runs in an isolated container, that is an explicitly defined, reproducable and portable environment. The analogy is taken from freight transport where you ship your goods in containers. [1]_

Docker solves the problem of having identical environments across various stages of development and having isolated environments for your individual applications. [1]_ They persist one main executable/app per container.

Basic management

```
## List Docker CLI commands
docker
docker container --help

## Display Docker version and info
docker --version
docker version
docker info

## List Docker images
docker image ls

## List Docker containers (running, all, all in quiet mode)
docker container ls
docker container ls --all
docker container ls -aq
```

(continues on next page)

(continued from previous page)

```
## Delete an image
docker image rm <image id>
# Remove all images from this machine
docker image rm $(docker image ls -a -q)
```

```
docker pull

# port forwarding
# -p host-port:container-port
docker run --name my-nginx -p 80:80 nginx:1.10.1-alpine
```

```
docker info

docker ps
docker ps -a    # List all, even the stopped ones
docker stop
docker start
docker restart
docker logs --tail 50 --follow --timestamps instance-name
docker inspect
docker rm
```

```
# Launch shell
docker exec -ti my-nginx /bin/sh
```

Dockerfile

```
# An example from https://docs.docker.com/get-started/part2/#dockerfile
# Use an official Python runtime as a parent image
FROM python:2.7-slim

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
ADD . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

Build

```
# Create image using current directory's Dockerfile
docker build -t image-name .
```

Run

```
# Map port host:container
docker run -p 4000:80 image

# Run in background
docker run -p 4000:80 image

# Run Ubuntu Bash
docker run -it ubuntu bash

docker run -p 5900:5900 ubuntu-firefox
```

Stop

```
# Gracefully stop the specified container
docker container stop <hash>

# Force shutdown of the specified container
docker container kill <hash>
```

Share

```
# 1.
docker tag image username/repository:tag

docker push username/repository:tag
```

Docker swarm

```
# docker-compose.yml
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: username/repo:tag
    deploy:
      replicas: 5
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
      restart_policy:
        condition: on-failure
    ports:
```

(continues on next page)

(continued from previous page)

```
      - "80:80"
    networks:
      - webnet
networks:
  webnet:
```

```
docker swarm init

docker stack deploy -c docker-compose.yml <appname>

docker service ls

# A single container running in a service is called a task.
# Tasks are given unique IDs that numerically increment,
# up to the number of replicas you defined.
# List the tasks for your service:
docker service ps <service>

docker inspect <task or container>

# Take down an app
docker stack rm <appname>

# Take down the swarm
docker swarm leave --force
```

Cleanup containers & images for disk space

```
# https://www.digitalocean.com/community/tutorials/how-to-remove-docker-images-
↪containers-and-volumes
# https://github.com/moby/moby/issues/21925
docker rm $(docker ps -a -q)
```

```

```

```

```

```

```

```

```

```

**RUN apt-get install -y** libportaudio0 libportaudio2 libportaudiocpp0 portaudio19-dev python-dev –no-install-recommends gcc
```

2.12.3 Netatalk3

Installation

Dependencies

```
sudo apt-get install autoconf \
        libtool-bin \
        libtool \
        automake \
        build-essential \
        libssl-dev \
        libgcrypt11-dev \
        libkrb5-dev \
        libpam0g-dev \
        libwrap0-dev \
        libdb-dev \
        libmysqlclient-dev \
        libavahi-client-dev \
        libacl1-dev \
        libldap2-dev \
        libcrack2-dev \
        systemtap-sdt-dev \
        libdbus-1-dev \
        libdbus-glib-1-dev \
        libglib2.0-dev \
        tracker \
        libtracker-miner-1.0-dev \
        libtracker-sparql-1.0-dev \
        libevent-dev \
        libtdb-dev  # You need it for `./configure --without-tdb `
```

Configuration

```
      ./configure \
--with-init-style=systemd \
--without-libevent \
--without-tdb \
--with-cracklib \
--enable-krbV-uam \
--with-pam-confdir=/etc/pam.d \
--with-dbus-sysconf-dir=/etc/dbus-1/system.d \
--with-tracker-pkgconfig-version=1.0
```

2.12.4 Raspberry pi

Set up

Locale issue

Set locale with `sudo raspi-config`.

2.12.5 References

2.12.6 Singularity

Intro

```
# Test
singularity selftest

####### Output #######
# + sh -c test -f /etc/singularity/singularity.conf                          (retval=0) OK
# + test -u /usr/lib/x86_64-linux-gnu/singularity/bin/action-suid            (retval=0) OK
# + test -u /usr/lib/x86_64-linux-gnu/singularity/bin/mount-suid             (retval=0) OK
# + test -u /usr/lib/x86_64-linux-gnu/singularity/bin/start-suid             (retval=0) OK
```

Build

```
# Build from Docker hub
singularity build englishspeechupsampler-latest.simg docker://shinyeyes/
↪englishspeechupsampler
```

Shell

```
# Enter the shell of an image with specified shell and GPU
singularity shell -s /bin/bash --nv englishspeechupsampler-latest.simg

# Bind a dir
singularity shell -s /bin/bash --nv -B /l:/l englishspeechupsampler-latest.simg

singularity shell -s /bin/bash --nv -B /l/EnglishSpeechUpsampler/settings:/settings -
↪B /storage_zpool/data:/storage_zpool/data english-speech-upsampler-v0.1.simg


source /environment
```

Exec

```
# Run Python from the image
singularity exec python englishspeechupsampler-latest.simg
```

Run

Execute the "Run" scripts from recipe.

```
singularity run --nv -B /l:/l englishspeechupsampler-latest.img

singularity run  -B _build/html:/usr/share/nginx/html ${SINGULARITY_PULLFOLDER}/nginx.
↪simg

singularity run  -B /usr/share/nginx/html:_build/html ${SINGULARITY_PULLFOLDER}/nginx.
↪simg
```

References

singularity shell -s /bin/bash –nv -B /l/gadichs1/gitrepos/temp/fsecure-exercise/:/workspace $SINGULARITY_PULLFOLDER/audio-u-net-v0.1.simg

CHAPTER 3

Metadocs

These docs are open source: all content is licensed under CC-BY 4.0 and all examples under CC0 (public domain). Additionally, this is an *open project* and we *strongly* encourage anyone to *contribute*. For information, see the *About these docs* and the Github links at the top of every page.

3.1 About these docs

These docs are literally my digitized master studies and Aalto University. I make notes about courses I take. Much content is brought from somewhere else such as blogs, Stackoverflow, Wikipedia and etc. I try to be strict with citation as much as possible. You may fork it or do whatever you may want to do.

The structure of this docs is forked from Aalto University CS-IT documentation page, http://scicomp.aalto.fi. I work there.

3.1.1 Contributing

This documentation is Open Source (CC-BY 4.0), and we welcome contributions from the Aalto community. The project is run on Github (https://github.com/AaltoScienceIT/triton-docs).

To contribute, you can always use the normal Github contribution mechanims: make a pull request or comments. If you are at Aalto, you can also get direct write access. Make a github issue, then contact us in person/by email for us to confirm.

The worst contribution is one that isn't made. Don't worry about making things perfect: since this is in version control, we track all changes and will just fix anything that's not perfect. This is also true for formatting errors - if you can't do ReStructudedText perfectly, just do your best (and pretend it's markdown because all the basics are similar).

Contributing gives consent to use content under the licenses (CC-BY 4.0 or CC0 for examples).

3.1.2 Requirements and building

The only software needed is Sphinx: Debian package `python-sphinx`, : PyPI: `python-sphinx`. It is already installed on Aalto workstations.

To build the docs, run `make html`.

HTML output is in `_build/html/index.html`, and other output formats are available as well.

3.1.3 Editing

Look at examples and copy. To add sections, add a new page in a subfolder. Link it from the main Table of Contents (`toctree`) in `index.rst` to have the document appear and be cross-referenced.

You can see a complete example from UiT: source and compiled HTML.

3.1.4 ReStructured text

ReStructured Text is similar to markdown for basics, but has a more strictly defined syntax and more higher level structure. This allows more semantic markup, more power to compile into different formats (since there isn't embedded HTML), and advanced things like indexing, permanent references, etc.

Restructured text home and quick reference.

Note: Literal inline text uses `` instead of a single ` (second works but gives warning).

A very quick guide is below.

Header style:

```
\=====
Title
\=====

Heading1
\=======

Heading2
\#######

Heading3
\^^^^^^^

Heading4
\*******

Heading5
\+++++++
```

`Inline code/monospace`, *emphasis*, **strong emphasis**

```
``Inline code/monospace``, *emphasis*, **strong emphasis**
```

```
Block quote
Block quote
```

```
::

  Block quote
  Block quote
```

Block quotes can also start with paragraph ending in double colon, like this:

```
Block quote
```

```
Block quotes can also start with paragraph ending in double colon,
like this::

    Block quote
```

Inline link, or anonymous, or separate, or different text links. Trailing underscores indicate links.

```
Inline `link <http://python.org>`_, or
anonymous__, or
separate_, or
`different text <separate_>`_ links.
Trailing underscores indicate links.

__ http://python.org

.. _separate: http://python.org
```

Linking to the web. If possible use a permanent reference (next section), but you can also refer to specific files by name. Note, that for internal links there are no trailing underscores:

```
:doc:`../tut/interactive.rst`   (recommended)
`../tut/interactive.rst`        (short, no warning if link breaks)

With different text:
:doc:`Text <../tut/interactive.rst>`   (recommended)
`Text <../tut/interactive.rst>`        (short, no warning if link breaks)
```

Internal links. Permanent references across files

Label things this way (note only one colon):

```
.. _label-name:
```

Reference them this way:

```
:ref:`label-name`      (recommended)
`label-name`           (short, no warning if link breaks)
`Text <label-name>`    (short, no warning if link breaks)
```

Admonitions: attention, caution, danger, error, hint, important, note, tip, warning, etc.

Note: This is a note

Warning: This is a warning

See also:

This is a simple **seealso** note.

```
.. note::

  This is a note

.. warning::

  This is a warning

.. seealso::

  This is a simple **seealso** note.
```

—

Topic Title

Subsequent indented lines comprise the body of the topic, and are interpreted as body elements.

Sidebar Title

Optional Sidebar Subtitle

Subsequent indented lines comprise the body of the sidebar, and are interpreted as body elements.

- genindex
- search

Bibliography

[RL_course_mdp_DeepMind] https://www.youtube.com/watch?v=lfHX2hHRMVQ

[Goodfellow-et-al] Deep Learning

[Quara-What-is-the-difference-between-momentum-and-learning-rate] https://www.quora.com/What-is-the-difference-between-momentum-and-learning-rate

[Dropout_A_Simple_Way_to_Prevent_Neural_Networks_from_Overfitting] https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

[Deep_Learning_Andrew_Ng] https://www.coursera.org/learn/convolutional-neural-networks/lecture/AYzbX/data-augmentation?authMode=login

[Dropout_A_Simple_Way_to_Prevent_Neural_Networks_from_Overfitting] https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf