

# Lecture 1b. Probability Review

COMP90051 Statistical Machine Learning

Sem2 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

COMP90051 Statistical Machine Learning

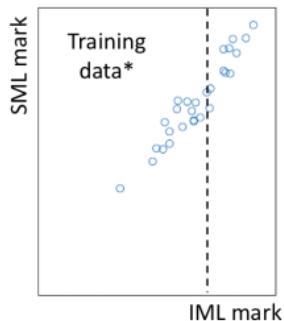
## This lecture

- About COMP90051
- **Review: Probability theory**
- Review: Linear algebra
- Review: Sequences and limits

2

COMP90051 Statistical Machine Learning

## Data is noisy (almost always)

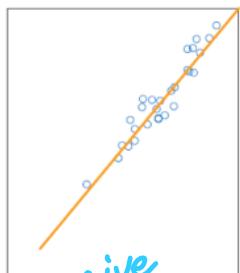


- Example:
  - \* given mark for Intro ML (IML)
  - \* predict mark for Stat Machine Learning (SML)

\* synthetic data :)

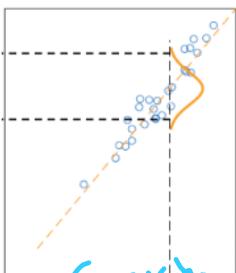
3

## Types of models



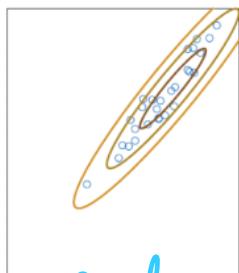
naive

$$\hat{y} = f(x)$$



Gaussian

$$P(y|x)$$



Random

$$P(x,y)$$

IntroML mark was 95,  
SML mark is predicted  
to be 95

fixed

IntroML mark was 95,  
SML mark is likely to  
be in (92, 97)

random

probability of having  
(IML =  $x$ , SML =  $y$ )

4

## Basics of probability theory



- A probability space:
  - \* Set  $\Omega$  of possible outcomes *universe*
  - \* Set  $F$  of events (subsets of outcomes)
  - \* Probability measure  $P: F \rightarrow \mathbb{R}$
- Example: a die roll
  - \*  $\Omega = \{1, 2, 3, 4, 5, 6\}$
  - \*  $F = \{\emptyset, \{1\}, \dots, \{6\}, \{1,2\}, \dots, \{5,6\}, \dots, \{1,2,3,4,5,6\}\}$
  - \*  $P(\emptyset) = 0, P(\{1\}) = 1/6, P(\{1,2\}) = 1/3, \dots$

5

## Axioms of probability\*

1.  $F$  contains all of:  $\Omega$ ; all complements  $\Omega \setminus f, f \in F$ ; the union of any countable set of events in  $F$ .
2.  $P(f) \geq 0$  for every event  $f \in F$ .
3.  $P(\bigcup_f f) = \sum_f P(f)$  for all countable sets of pairwise disjoint events.
4.  $P(\Omega) = 1$

\* We won't delve further into advanced probability theory, which starts with measure theory – a beautiful subject and the only way to "fully" formulate probability.

6

## Random variables (r.v.'s)



\$ \$ \$

- A random variable  $X$  is a numeric function of outcome  $X(\omega) \in \mathbb{R}$
  - $P(X \in A)$  denotes the probability of the outcome being such that  $X$  falls in the range  $A$
  - Example:  $X$  winnings on \$5 bet on even die roll
    - \*  $X$  maps 1,3,5 to -5
    - \*  $X$  maps 2,4,6 to 5
    - \*  $P(X=5) = P(X=-5) = \frac{1}{2}$
- uniform distribution*

7

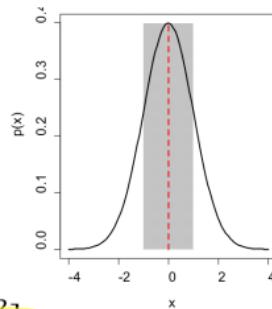
## Discrete vs. continuous distributions

- Discrete distributions
  - \* Govern r.v. taking discrete values
  - \* Described by probability mass function  $p(x)$  which is  $P(X=x)$
  - \*  $P(X \leq x) = \sum_{a=-\infty}^x p(a)$
  - \* Examples: Bernoulli, Binomial, Multinomial, Poisson
- Continuous distributions
  - \* Govern real-valued r.v.
  - \* Cannot talk about PMF but rather probability density function  $p(x)$
  - \*  $P(X \leq x) = \int_{-\infty}^x p(a)da$
  - \* Examples: Uniform, Normal, Laplace, Gamma, Beta, Dirichlet

8

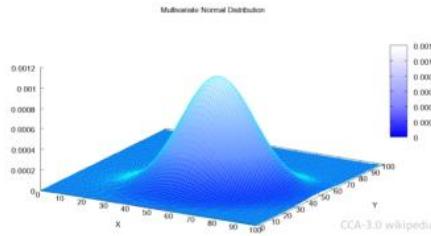
## Expectation

- Expectation  $E[X]$  is the r.v.  $X$ 's “average” value
  - \* Discrete:  $E[X] = \sum_x x P(X = x)$
  - \* Continuous:  $E[X] = \int_x x p(x) dx$
- Properties
  - \* Linear:  $E[aX + b] = aE[X] + b$   
 $E[X + Y] = E[X] + E[Y]$
  - \* Monotone:  $X \geq Y \Rightarrow E[X] \geq E[Y]$
- Variance:  $Var(X) = E[(X - E[X])^2]$



9

## Multivariate distributions



- Specify joint distribution over multiple variables
- Probabilities are computed as in univariate case, we now just have repeated summations or repeated integrals
- Discrete:  $P(X, Y \in A) = \sum_{(x,y) \in A} p(x, y)$
- Continuous:  $P(X, Y \in A) = \int_A p(x, y) dx dy$

10

## Independence and conditioning

- $X, Y$  are independent if
  - $P(X \in A, Y \in B) = P(X \in A)P(Y \in B)$
  - Similarly for densities:  $p_{X,Y}(x, y) = p_X(x)p_Y(y)$
  - Intuitively: knowing value of  $Y$  reveals nothing about  $X$
  - Algebraically: the joint on  $X, Y$  factorises!
- Conditional probability
  - $P(A|B) = \frac{P(A \cap B)}{P(B)}$
  - Similarly for densities  $p(y|x) = \frac{p(x,y)}{p(x)}$
  - Intuitively: probability event  $A$  will occur given we know event  $B$  has occurred
  - $X, Y$  independent equiv to  $P(Y = y|X = x) = P(Y = y)$

11

## Inverting conditioning: Bayes' Theorem



- In terms of events  $A, B$ 
    - $P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$
    - $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$
  - Simple rule that lets us swap conditioning order
  - Probabilistic and Bayesian inference make heavy use
    - Marginals: probabilities of individual variables
    - Marginalisation: summing away all but r.v.'s of interest
- $$P(A) = \sum_b P(A, B = b)$$

12

## Summary

- Probability spaces, axioms of probability
- Discrete vs continuous; Univariate vs multivariate
- Expectation, Variance
- Independence and conditioning
- Bayes rule and marginalisation

Next: Linear algebra primer/review

13

## Lecture 1c. Linear Algebra Review

COMP90051 Statistical Machine Learning

Sem2 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

## This lecture

- About COMP90051
- Review: Probability theory
- **Review: Linear algebra**
- Review: Sequences and limits

2

# Vectors

Link between geometric and algebraic interpretation of ML methods

3

## What are vectors?

Suppose  $\mathbf{u} = [u_1, u_2]'$ . What does  $\mathbf{u}$  really represent?



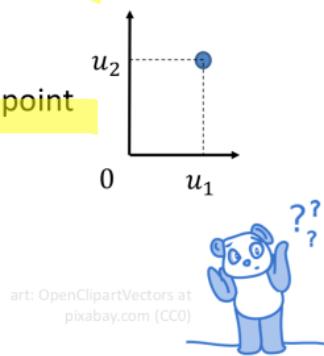
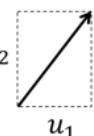
Ordered set of numbers  $\{u_1, u_2\}$



Cartesian coordinates of a point



A direction



4

## Dot product: Algebraic definition

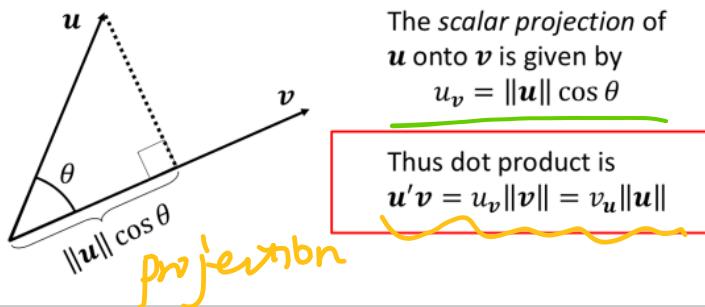
- Given two  $m$ -dimensional vectors  $\mathbf{u}$  and  $\mathbf{v}$ , their dot product is  $\mathbf{u} \cdot \mathbf{v} \equiv \mathbf{u}' \mathbf{v} \equiv \sum_{i=1}^m u_i v_i$  → transpose
  - E.g., weighted sum of terms is a dot product  $\mathbf{x}' \mathbf{w}$
- If  $k$  is a scalar,  $\mathbf{a}, \mathbf{b}, \mathbf{c}$  are vectors then
 
$$(k\mathbf{a})' \mathbf{b} = k(\mathbf{a}' \mathbf{b}) = \mathbf{a}' (k\mathbf{b})$$

$$\mathbf{a}' (\mathbf{b} + \mathbf{c}) = \mathbf{a}' \mathbf{b} + \mathbf{a}' \mathbf{c}$$

5

## Dot product: Geometric definition

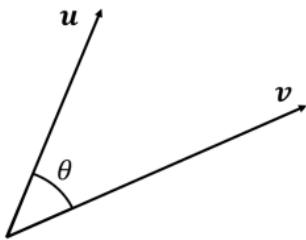
- Given two  $m$ -dimensional Euclidean vectors  $\mathbf{u}$  and  $\mathbf{v}$ , their dot product is  $\mathbf{u} \cdot \mathbf{v} \equiv \mathbf{u}'\mathbf{v} \equiv \|\mathbf{u}\| \|\mathbf{v}\| \cos \theta$ 
  - \*  $\|\mathbf{u}\|, \|\mathbf{v}\|$  are  $L_2$  norms for  $\mathbf{u}, \mathbf{v}$  also written as  $\|\mathbf{u}\|_2$
  - \*  $\theta$  is the angle between the vectors



6

## Geometric properties of the dot product

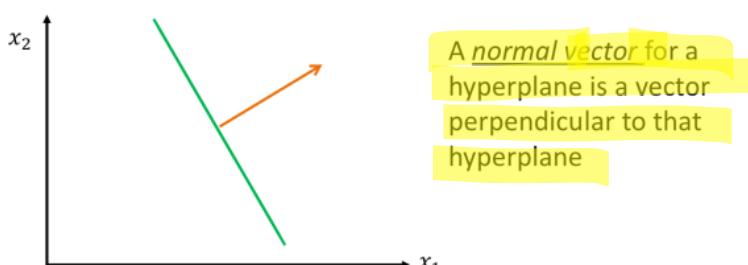
- If the two vectors are orthogonal then  $\mathbf{u}'\mathbf{v} = 0$
- If the two vectors are parallel then  $\mathbf{u}'\mathbf{v} = \|\mathbf{u}\| \|\mathbf{v}\|$ , if they are anti-parallel then  $\mathbf{u}'\mathbf{v} = -\|\mathbf{u}\| \|\mathbf{v}\|$
- $\mathbf{u}'\mathbf{u} = \|\mathbf{u}\|^2$ , so  $\|\mathbf{u}\| = \sqrt{u_1^2 + \dots + u_m^2}$  defines the Euclidean vector length



7

## Hyperplanes and normal vectors

- A hyperplane defined by parameters  $\mathbf{w}$  and  $b$  is a set of points  $\mathbf{x}$  that satisfy  $\mathbf{x}'\mathbf{w} + b = 0$
- In 2D, a hyperplane is a line: a line is a set of points that satisfy  $w_1x_1 + w_2x_2 + b = 0$



8

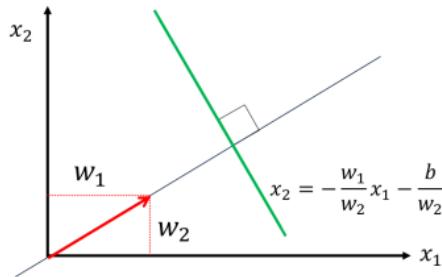
## Hyperplanes and normal vectors

- Consider a hyperplane defined by parameters  $w$  and  $b$ . Note that  $w$  is itself a vector
- Lemma: Vector  $w$  is normal to the hyperplane
- Proof sketch:
  - Choose any two points  $u$  and  $v$  on the hyperplane. Note that vector  $(u - v)$  lies on the hyperplane
  - Consider dot product  $(u - v)'w = u'w - v'w = (u'w + b) - (v'w + b) = 0$
  - Thus  $(u - v)$  lies on the hyperplane, but is perpendicular to  $w$ , and so  $w$  is a vector normal

9

## Example in 2D

- Consider a line defined by  $w_1, w_2$  and  $b$
- Vector  $w = [w_1, w_2]'$  is a normal vector



10

## $L_1$ and $L_2$ norms

- Throughout the subject we will often encounter norms that are functions  $\mathbb{R}^n \rightarrow \mathbb{R}$  of a particular form
  - Intuitively, norms measure lengths of vectors in some sense
  - Often component of objectives or stopping criteria in optimisation-for-ML
- More specifically, we will often use the  $L_2$  norm (aka Euclidean distance)

$$\|a\| = \|a\|_2 \equiv \sqrt{a_1^2 + \dots + a_n^2}$$
- And also the  $L_1$  norm (aka absolute norm or Manhattan distance)

$$\|a\|_1 \equiv |a_1| + \dots + |a_n|$$

11

# Vector Spaces and Bases

Useful in interpreting matrices and some algorithms like PCA

12

## Linear combinations, Independence

- For formal definition of **vector spaces**:  
[https://en.wikipedia.org/wiki/Vector\\_space#Definition](https://en.wikipedia.org/wiki/Vector_space#Definition)
- A **linear combination** of vectors  $v_1, \dots, v_k \in V$  some vector space, is a new vector  $\sum_{i=1}^k a_i v_i$  for some scalars  $a_1, \dots, a_k$
- A set of vectors  $\{v_1, \dots, v_k\} \subseteq V$  is called **linearly dependent** if one element  $v_j$  can be written as a linear combination of the other elements
- A set that isn't linearly dependent is **linearly independent**

13

## Spans, Bases

- The **span** of vectors  $v_1, \dots, v_k \in V$  is the set of all obtainable linear combinations (ranging over all scalar coefficients) of the vectors
- A set of vectors  $\{v_1, \dots, v_k\} \subseteq V$  is called a **basis** for a vector subspace  $V' \subseteq V$  if
  - The set is linearly independent; and
  - Every  $v \in V'$  is a linear combination of the set.
- An **orthonormal basis** is a basis in which each
  - Pair of basis vectors are orthogonal (zero dot prod); and
  - Basis vector has norm equal to 1.

14

# Matrices

Some useful facts for ML

15

## Basic matrices

- See more: [https://en.wikipedia.org/wiki/Matrix\\_\(mathematics\)](https://en.wikipedia.org/wiki/Matrix_(mathematics))
  - \* Including matrix-matrix and matrix-vector products
- A rectangular array, often denoted by upper-case, with two indices first for row, second for column
- Square matrix has equal dimensions (numbers of rows and columns)
- Matrix transpose  $A'$  or  $A^T$  of  $m$  by  $n$  matrix  $A$  is an  $n$  by  $m$  matrix with entries  $A'_{ij} = A_{ji}$
- A square matrix  $A$  with  $A = A'$  is called **symmetric**
- The (square) identity matrix  $I$  has 1 on the diagonal, 0 off-diagonal
- Matrix inverse  $A^{-1}$  of square matrix  $A$  (if it exists) satisfies  $A^{-1}A = I$

16

## Matrix eigenspectrum

- Scalar, vector pair  $(\lambda, v)$  are called an eigenvalue-eigenvector pair of a square matrix  $A$  if  $Av = \lambda v$ 
  - \* Intuition: matrix  $A$  doesn't rotate  $v$  it just stretches it
  - \* Intuition: the eigenvalue represents stretching factor
- In general eigenvalues may be zero, negative or even complex (imaginary) – we'll only encounter reals

17

## Spectra of common matrices

- Eigenvalues of symmetric matrices are always real (no imaginary component)
- A matrix with linear dependent columns has some zero eigenvalues (called rank deficient) → no matrix inverse exists

18

## Positive (semi)definite matrices

- A symmetric square matrix  $\mathbf{A}$  is called positive semidefinite if for all vectors  $\mathbf{v}$  we have  $\mathbf{v}'\mathbf{A}\mathbf{v} \geq 0$ .
  - Then  $\mathbf{A}$  has non-negative eigenvalues
  - For example, any  $\mathbf{A} = \mathbf{X}'\mathbf{X}$  since:  $\mathbf{v}'\mathbf{X}'\mathbf{X}\mathbf{v} = \|\mathbf{X}\mathbf{v}\|^2 \geq 0$
- Further if  $\mathbf{v}'\mathbf{A}\mathbf{v} > 0$  holds as a strict inequality then  $\mathbf{A}$  is called positive definite
  - Then  $\mathbf{A}$  has (strictly) positive eigenvalues

19

## Summary

- Vectors: Vector spaces, dot products, independence, hyperplanes
- Matrices: Eigenvalues, positive semidefinite matrices

Next time: Sequences and limits review/primer

20

# Lecture 1d. Limits Review

COMP90051 Statistical Machine Learning

Sem2 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

COMP90051 Statistical Machine Learning

## This lecture

- About COMP90051
- Review: Probability theory
- Review: Linear algebra
- **Review: Sequences and limits**

2

COMP90051 Statistical Machine Learning

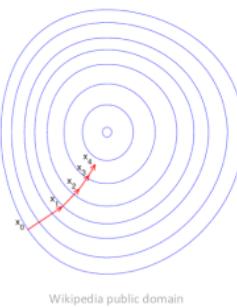
## Sequences and Limits

Sequences arise whenever we have iterations (e.g. training loops, growing data sample size). Limits tell us about where sequences tend towards.

3

## Infinite Sequences

- Written like  $x_1, x_2, \dots$  or  $\{x_i\}_{i \in \mathbb{N}}$
- Formally: a function from the positive (from 1) or non-negative (from 0) integers
- Index set: subscript set e.g.  $\mathbb{N}$
- Sequences allow us to reason about test error when training data grows indefinitely, or training error (or a stopping criterion) when training runs arbitrarily long



Wikipedia public domain

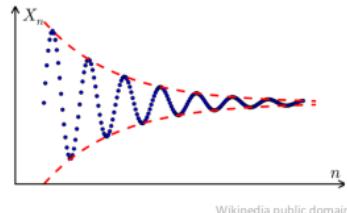
4

## Limits and Convergence

- A sequence  $\{x_i\}_{i \in \mathbb{N}}$  converges if its elements become and remain arbitrarily close to a fixed limit point  $L$ .
- Formally:  $x_i \rightarrow L$  if, for all  $\varepsilon > 0$ , there exists  $N \in \mathbb{N}$  such that for all  $n \geq N$  we have  $\|x_n - L\| < \varepsilon$

Notes:

- Epsilon  $\varepsilon$  represents distance of sequence to limit point
- Distance can be arbitrarily small
- Definition says we eventually get that close (at some finite  $N$ ) and we stay at least that close for ever more



Wikipedia public domain

5

## Supremum

Generalising the maximum: When a sequence never quite peaks.

6

## When does the Maximum Exist?

- Can you always take a **max of a set**?
- Finite sets: what's the max of  $\{1, 7, 3, 2, 9\}$ ? **exist**
- Closed, bounded intervals: what's the max of  $[0,1]$ ? **exist**
- Open, bounded intervals: what's the max of  $(0,1)$ ? **doesn't exist**
- Open, unbounded intervals: what's the max of  $[0,\infty)$ ? **doesn't exist**

7

## What about “Least Upper Bound”?

- Can you always take a **least-upper-bound of a set**? (much more often!)
- Finite sets: what's the max of  $\{1, 7, 3, 2, 9\}$ ?  $\text{max}=9 \quad \text{LUB}=9$
- Closed, bounded intervals: what's the max of  $[0,1]$ ?  $\text{max}=1 \quad \text{LUB}=1$
- Open, bounded intervals: what's the max of  $(0,1)$ ?  $\text{max}=\text{N/A} \quad \text{LUB}=1$
- Open, unbounded intervals: what's the max of  $[0,\infty)$ ?  $\text{max}=\text{N/A} \quad \text{LUB}=\infty$

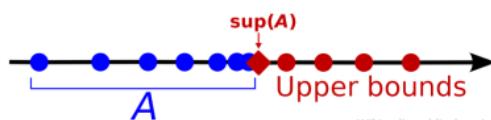
8

## The Supremum

- Consider any subset  $S$  of the reals
- **Upper bound**  $u \in \mathbb{R}^+$  of set  $S$  has:  $u \geq x$  for all  $x \in S$
- If  $u$  is no bigger than any other upper bound of  $S$ , then it's called a **least upper bound or supremum of  $S$ , written as  $\sup(S)$  and pronounced “soup”**:  
 $* z \geq u$  for all upper bounds  $z \in \mathbb{R}^+$  of  $S$
- When we don't know, or can't guarantee, that a set or sequence has a max, it is better to use its sup



FreeSVG public domain



Wikipedia public domain

9

## Infimum

- The greatest lower bound or infimum is generalisation of the minimum
- Written  $\inf(S)$  pronounced “inf”
- Useful if we’re minimising training error but don’t know if the minimum is ever attained.

10

## Stochastic Convergence

When random events or quantities can sometimes be expected to converge (e.g. test error likely drops to a minimal value)

11

## Why Simple Limits Aren’t Enough

- Consider running your favourite learner on varying numbers of  $n$  training examples giving classifier  $c_n$
- If your learner minimises training error, you’d wish its test error wasn’t much bigger than its training error
- If  $R_n = \text{err}_{\text{test}}(c_n) - \text{err}_{\text{train}}(c_n)$ , you’d wish for  $R_n \rightarrow 0$  as this would mean eventually tiny test error
- But both training data and test data are random!
- Even if  $R_n \rightarrow 0$  usually happens, it won’t always!!

tiny test error    SGD

12

## Stochastic Convergence

- A sequence  $\{X_n\}$  of random variables (CDFs  $F_n$ ) converges in distribution to random variable  $X$  (CDF  $F$ ) if  $F_n(x) \rightarrow F(x)$  for all constants  $x$
- A sequence  $\{X_n\}$  of random variables converges in probability to random variable  $X$  if for all  $\varepsilon > 0$ :  $\Pr(|X_n - X| > \varepsilon) \rightarrow 0$
- A sequence  $\{X_n\}$  of random variables converges almost surely to random variable  $X$  if:  $\Pr(X_n \rightarrow X) = 1$
- Chain of implications:

almost sure (strongest)  $\Rightarrow$  in probability  $\Rightarrow$  in distribution (weakest)

13

## But don't worry...

- We're not going to do *any* calculations with stochastic convergence
- Close understanding of it won't be necessary in this subject
- But it's good to be aware that its "out there" and we **may refer to it** (v briefly) within StatML theory



CC-BY 4.0 Vincent Le Moigne

14

## Summary

- Sequences
- Limits of sequences
- Supremum is the new maximum
- Stochastic convergence

Next time: L02 Statistical schools

Homework week #1: Watch all week 1 recordings.  
Jupyter notebooks setup and launch (at home)

15

# Lecture 2a. Statistical Schools of Thought: Frequentist

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

COMP90051 Statistical Machine Learning

## This lecture

### How do learning algorithms come about?

- **Frequentist statistics**
- Statistical decision theory
- Extremum estimators
- Bayesian statistics

Types of probabilistic models

- Parametric vs. Non-parametric
- Generative vs. Discriminative

2

COMP90051 Statistical Machine Learning

## Frequentist Statistics

Wherein unknown model parameters are treated  
as having fixed but unknown values.

3

## Frequentist statistics

- Abstract problem
  - Given:  $X_1, X_2, \dots, X_n$  drawn i.i.d. from some distribution
  - Want to: identify unknown distribution, or a property of it
- Parametric approach ("parameter estimation")
  - Class of models  $\{p_\theta(x) : \theta \in \Theta\}$  indexed by parameters  $\Theta$  (could be a real number, or vector, or ...)
  - Point estimate  $\hat{\theta}(X_1, \dots, X_n)$  a function (or statistic) of data
- Examples
  - Given  $n$  coin flips, determine probability of landing heads
  - Learning a classifier

Independent and identically distributed

Hat means estimate or estimator

best guess for unknown but fixed  $\theta$ 

4

## Estimator Bias

Frequentists seek good behaviour, in ideal conditions

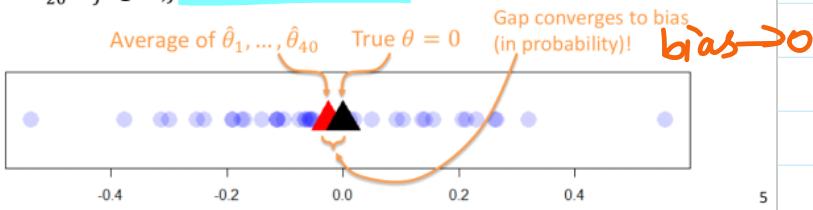
- Bias:  $B_\theta(\hat{\theta}) = E_\theta[\hat{\theta}(X_1, \dots, X_n)] - \theta$

Subscript  $\theta$  means data really comes from  $p_\theta$ Example: for  $i=1\dots 40$ 

- $X_{i,1}, \dots, X_{i,20} \sim p_\theta = \text{Normal}(\theta = 0, \sigma^2 = 1)$

- $\hat{\theta}_i = \frac{1}{20} \sum_{j=1}^{20} X_{i,j}$  the sample mean, plot as

1: 40 → □



5

## Estimator Variance

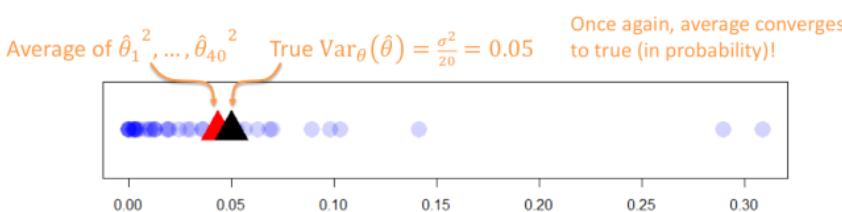
Frequentists seek good behaviour, in ideal conditions

- Variance:  $\text{Var}_\theta(\hat{\theta}) = E_\theta[(\hat{\theta} - E_\theta[\hat{\theta}])^2]$

 $\hat{\theta}$  still function of data

Example cont.

- Plot each  $(\hat{\theta}_i - E_\theta[\hat{\theta}_i])^2 = \hat{\theta}_i^2$  as



6

## Asymptotically Well Behaved

For our example estimator (sample mean), we could calculate its exact bias (zero) and variance ( $\sigma^2$ ). Usually can't guarantee low bias/variance exactly 😞

Asymptotic properties often hold! 😊

Bias closer and closer to zero

- Consistency:  $\hat{\theta}(X_1, \dots, X_n) \rightarrow \theta$  in probability
- Asymptotic efficiency:  $\text{Var}_\theta(\hat{\theta}(X_1, \dots, X_n))$  converges to the smallest possible variance of any estimator of  $\theta$

Variance closer & closer to optimal

eventually

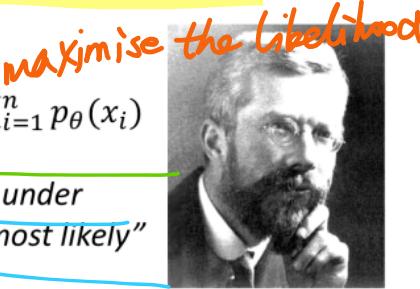
↓  
low bias  
low variance

Amazing Cramér-Rao lower bound (outside subject scope):

$$\text{Var}_\theta(\hat{\theta}) \geq \frac{1}{I(\theta)} \text{ with } I(\theta) \text{ the Fisher information of } p_\theta \text{ for any } \hat{\theta}$$

## Maximum-Likelihood Estimation

- A general principle for designing estimators
- Involves optimisation
- $\hat{\theta}(x_1, \dots, x_n) \in \underset{\theta \in \Theta}{\operatorname{argmax}} \prod_{i=1}^n p_\theta(x_i)$
- "The best estimate is one under which observed data is most likely"



Fisher

maximise the likelihood  
in training data

Later: MLE estimators usually well-behaved asymptotically

### Example I: Bernoulli

- Know data comes from Bernoulli distribution with unknown parameter (e.g., biased coin); find mean

$$L(\theta) = \prod_{i=1}^n p_\theta(x_i) = \prod_{i=1}^n \theta^{x_i} (1-\theta)^{1-x_i}$$

$$* p_\theta(x) = \begin{cases} \theta, & \text{if } x = 1 \\ 1 - \theta, & \text{if } x = 0 \end{cases} = \theta^x (1 - \theta)^{1-x}$$

(note:  $p_\theta(x) = 0$  for all other  $x$ )

$$* \text{Maximising likelihood yields } \hat{\theta} = \frac{1}{n} \sum_{i=1}^n x_i$$

$$\frac{dL}{d\theta} \hat{\theta}(\hat{\theta}) = \frac{\bar{x}}{\hat{\theta}} - \frac{n-\bar{x}}{1-\hat{\theta}} = 0 \Rightarrow \hat{\theta} = \frac{\bar{x}}{n}$$

choose  $\theta$  that optimize the function

$$\prod_{i=1}^n p_\theta(x_i)$$

$$\log \prod_{i=1}^n p_\theta(x_i)$$

$$= \log \prod_{i=1}^n \theta^{x_i} (1-\theta)^{1-x_i}$$

$$= \sum_{i=1}^n x_i \log \theta + (n-x_i) \log (1-\theta)$$

$$= (\sum_{i=1}^n x_i) \log \theta + (n - \sum_{i=1}^n x_i) \log (1-\theta)$$

$$\frac{dL}{d\theta} = \frac{\sum_{i=1}^n x_i}{\theta} - \frac{n - \sum_{i=1}^n x_i}{1-\theta} = 0$$

$$\downarrow \hat{\theta} = \frac{1}{n} \sum_{i=1}^n x_i$$

## Example II: Normal

- Know data comes from Normal distribution with variance 1 but unknown mean; find mean
- MLE for mean
  - \*  $p_\theta(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(x - \theta)^2\right)$
  - \* Maximising likelihood yields  $\hat{\theta} = \frac{1}{n} \sum_{i=1}^n X_i$
- Exercise: derive MLE for variance  $\sigma^2$  based on  
 $p_\theta(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right)$  with  $\theta = (\mu, \sigma^2)$

Derivation: <https://www.statlect.com/fundamentals-of-statistics/normal-distribution-maximum-likelihood>

10

## MLE ‘algorithm’

1. Given data  $X_1, \dots, X_n$  define probability distribution,  $p_\theta$ , assumed to have generated the data
2. Express likelihood of data,  $\prod_{i=1}^n p_\theta(X_i)$  (usually its *logarithm*... why?)
3. Optimise to find best (most likely) parameters  $\hat{\theta}$ 
  1. take partial derivatives of log likelihood wrt  $\theta$
  2. set to 0 and solve  
(failing that, use gradient descent)

11

## Summary

- Frequentist school of thought
- Point estimates
- Quality: bias, variance, consistency, asymptotic efficiency
- Maximum-likelihood estimation (MLE)

Next time: Statistical Decision Theory, Extremum estimators

Workshops week #2: learning Bayes a coin flip at a time!

12

# Lecture 2b. Statistical Schools of Thought: Statistical Decision Theory

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

COMP90051 Statistical Machine Learning

## This lecture

How do learning algorithms come about?

- Frequentist statistics
- **Statistical decision theory**
- **Extremum estimators**
- Bayesian statistics

Types of probabilistic models

- Parametric vs. Non-parametric
- Generative vs. Discriminative

2

COMP90051 Statistical Machine Learning

## Statistical Decision Theory

Branch within statistics, optimisation, economics,  
control, emphasising utility maximisation.

3

## Decision theory



Wald

- Act to maximise utility - connected to economics and operations research
- Decision rule  $\delta(x) \in A$  an action space
  - E.g. Point estimate  $\hat{\theta}(x_1, \dots, x_n)$
  - E.g. Out-of-sample prediction  $\hat{Y}_{n+1}|X_1, Y_1, \dots, X_n, Y_n, X_{n+1}$
- Loss function  $l(a, \theta)$ : economic cost, error metric
  - E.g. square loss of estimate  $(\hat{\theta} - \theta)^2$
  - E.g. 0-1 loss of classifier predictions  $1[y \neq \hat{y}]$

4

## Risk & Empirical Risk Minimisation (ERM)

- In decision theory, really care about expected loss
- Risk  $R_\theta[\delta] = E_{X \sim \theta}[l(\delta(X), \theta)]$ 
  - E.g. true test error
  - aka generalization error
- Want: Choose  $\delta$  to minimise  $R_\theta[\delta]$
- Can't directly! Why? do not have real  $\theta$
- ERM: Use training set  $X$  to approximate  $p_\theta$ 
  - Minimise empirical risk  $\hat{R}_\theta[\delta] = \frac{1}{n} \sum_{i=1}^n l(\delta(X_i), \theta)$

approximate true risk

do not have access to the whole population of error,  
only sample of it  
(training data)  
↓  
can't choose decision rule that minimize risk perfectly

## Decision theory vs. Bias-variance

We've already seen

- Bias:  $B_\theta(\hat{\theta}) = E_\theta[\hat{\theta}(X_1, \dots, X_n)] - \theta$
- Variance:  $\text{Var}_\theta(\hat{\theta}) = E_\theta[(\hat{\theta} - E_\theta[\hat{\theta}])^2]$

But are they equally important? How related?

- Bias-variance decomposition of square-loss risk

$$E_\theta [(\theta - \hat{\theta})^2] = [B(\hat{\theta})]^2 + \text{Var}_\theta(\hat{\theta})$$

Square loss

6

## Extremum estimators

Very general framework that covers elements of major statistical learning frameworks; enjoys good asymptotic behaviour in general!!

7

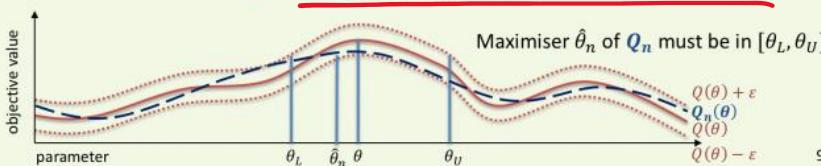
## Extremum estimators

- $\hat{\theta}_n(X) \in \operatorname{argmin}_{\theta \in \Theta} Q_n(X, \theta)$  for any objective  $Q_n()$
- point estimate objective function (a lot of things)
- Generalises bits of all statistical frameworks. *Woot!*
  - \* MLE and ERM seen earlier this lecture; and
  - \* MAP seen later in this lecture.
  - \* These are all  $M$ -estimators, with  $Q$  as a sum over data (i.e. of log-likelihood, loss, or log-likelihood plus log prior)
- And it generalises other frameworks too!

8

## Consistency of Extremum Estimators

- Recall consistency: stochastic convergence to 0 bias
- Theorem for extremum estimators:  $\hat{\theta}_n \rightarrow \theta$  in prob, if there's a ("limiting") function  $Q()$  such that:
  - $Q()$  is uniquely maximised by  $\theta$ .  
That is, no other parameters make  $Q()$  as large as  $Q(\theta)$ .
  - The parameter family  $\Theta$  is "compact"  
(a generalisation of the familiar "closed" & "bounded" set, like  $[0,1]$ )
  - $Q()$  is a continuous function
  - Uniform convergence:  $\sup_{\theta \in \Theta} |Q_n(\theta) - Q(\theta)| \rightarrow 0$  in probability.



9

## A game changer

- Frequentists: estimators that aren't even correct with infinite data (inconsistent), aren't adequate in practice
- Proving consistency for every new estimator? Ouch! *not easy*
- So many estimators are extremum estimators – general guarantees make it much easy (but not easy!) to prove
- Asymptotic normality
  - Extremum estimators converge to Gaussian in distribution
  - Asymptotic efficiency: the variance of that limiting Gaussian
- Practical: **Confidence intervals** - think error bars!!

→ Frequentists like to have this asymptotic theory for their algorithms



average  
↓ converge  
Gaussian

10

## Summary

- Decision theory: Utility-based, Minimise risk
- Many familiar learners minimise loss over data (ERM)
- Extremum estimators generalise ERM, MLE, (later: MAP)
  - Amazingly, consistent: Gives us confidence that they work (eventually)
  - Amazingly, asymptotically normal: Helps make confidence intervals

Next time: Last but not least, the Bayesian paradigm

Workshops week #2: learning Bayes one coin flip at a time!

11

## Lecture 2c. Statistical Schools of Thought: The Bayesian Paradigm

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

## This lecture

How do learning algorithms come about?

- Frequentist statistics
- Statistical decision theory
- Extremum estimators
- **Bayesian statistics**

### Types of probabilistic models

- **Parametric vs. Non-parametric**
- **Generative vs. Discriminative**

2

## Bayesian Statistics

Wherein unknown model parameters have associated distributions reflecting prior belief.

3

## Bayesian statistics



Laplace

- Probabilities correspond to beliefs
- Parameters
  - \* Modeled as r.v.'s having distributions
  - \* Prior belief in  $\theta$  encoded by prior distribution  $P(\theta)$ 
    - Parameters are modeled like r.v.'s (even if not really random)
    - Thus: data likelihood  $P_\theta(X)$  written as conditional  $P(X|\theta)$
  - \* Rather than point estimate  $\hat{\theta}$ , Bayesians update belief  $P(\theta)$  with observed data to  $P(\theta|X)$  the posterior distribution

4

# Tools of probabilistic inference



Bayes

- Bayesian probabilistic inference
  - \* Start with prior  $P(\theta)$  and likelihood  $P(X|\theta)$
  - \* Observe data  $X = x$
  - \* Update prior to posterior  $P(\theta|X = x)$

- Primary tools to obtain the posterior

- \* Bayes Rule: reverses order of conditioning

$$P(\theta|X = x) = \frac{P(X = x|\theta)P(\theta)}{P(X = x)}$$

- \* Marginalisation: eliminates unwanted variables

$$P(X = x) = \sum_t P(X = x, \theta = t)$$

This quantity is called the evidence

These are general tools of probability and not specific to Bayesian stats/ML

evidence

5

## Example

- We model  $X|\theta$  as  $N(\theta, 1)$  with prior  $N(0, 1)$
- Suppose we observe  $X=1$ , then update posterior

$$\begin{aligned} P(\theta|X = 1) &= \frac{P(X = 1|\theta)P(\theta)}{P(X=1)} \\ &\propto P(X = 1|\theta)P(\theta) \\ &= \left[ \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(1-\theta)^2}{2}\right) \right] \left[ \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\theta^2}{2}\right) \right] \\ &\propto N(0.5, 0.5) \end{aligned}$$

NB: allowed to push constants out front and "ignore" as these get taken care of by normalisation

6

$$\begin{aligned} P(\theta|X = 1) &= \frac{P(X = 1|\theta)P(\theta)}{P(X=1)} \\ &\propto P(X = 1|\theta)P(\theta) \end{aligned}$$

Name of the game is to get posterior into a recognisable form. exp of quadratic must be a Normal

*Discard constants w.r.t  $\theta$*

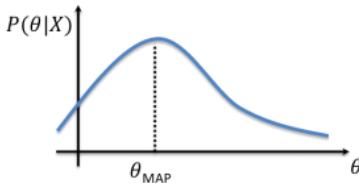
$$\begin{aligned} &= \left[ \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(1-\theta)^2}{2}\right) \right] \left[ \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\theta^2}{2}\right) \right] \\ &\propto \exp\left(-\frac{(1-\theta)^2 + \theta^2}{2}\right) \\ &\simeq \exp\left(-\frac{2\theta^2 - 2\theta + 1}{2}\right) \\ &\simeq \exp\left(-\frac{\theta^2 - \theta + \frac{1}{2}}{2}\right) \\ &\simeq \exp\left(-\frac{\theta^2 - \theta + \frac{1}{4}}{2} + \frac{\frac{1}{4}}{2}\right) \cdot \exp\left(-\frac{\frac{1}{4}}{2}\right) \\ &\propto \exp\left(-\frac{\theta^2 - \theta + \frac{1}{4}}{2}\right) \\ &\text{Factorise} = \exp\left(-\frac{(\theta - \frac{1}{2})^2}{2}\right) \\ &\text{Recognise as (unnormalized) Normal!} \propto \mathcal{N}(0.5, 0.5) \end{aligned}$$

Constant underlined  
Variance/std deviation circled

7

## How Bayesians make point estimates

- They don't, unless forced at gunpoint!
  - \* The posterior carries full information, why discard it?
- But, there are common approaches
  - \* Posterior mean  $E_{\theta|X}[\theta] = \int \theta P(\theta|X)d\theta$
  - \* Posterior mode  $\operatorname{argmax}_{\theta} P(\theta|X)$  (max a posteriori or MAP)
  - \* There're Bayesian decision-theoretic interpretations of these



8

## MLE in Bayesian context

- MLE formulation: find parameters that best fit data  
 $\hat{\theta} \in \operatorname{argmax}_{\theta} P(X = x|\theta)$
- Consider the MAP under a Bayesian formulation
 
$$\begin{aligned}\hat{\theta} &\in \operatorname{argmax}_{\theta} P(\theta|X = x) \\ &= \operatorname{argmax}_{\theta} \frac{P(X = x|\theta)P(\theta)}{P(X = x)} \\ &= \operatorname{argmax}_{\theta} P(X = x|\theta)P(\theta)\end{aligned}$$

weighted maximisation
- Prior  $P(\theta)$  weights; MLE like uniform  $P(\theta) \propto 1$
- Extremum estimator: Max  $\log P(X = x|\theta) + \log P(\theta)$

9

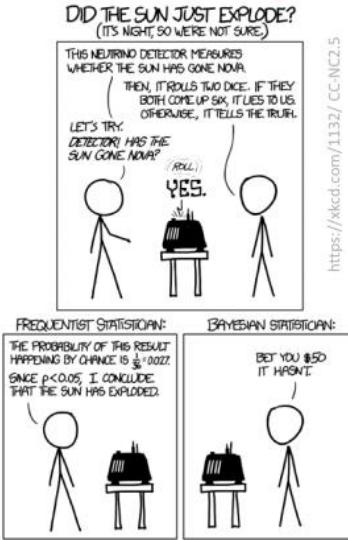
## Frequentists vs Bayesians – Oh My!

- Two key schools of statistical thinking
  - \* Decision theory complements both
- Past: controversy; animosity; almost a 'religious' choice
- Nowadays: deeply connected

I declare the Bayesian vs. Frequentist debate over for data scientists

Are You a Bayesian or a Frequentist?

Michael I. Jordan  
Department of EECS  
Department of Statistics  
University of California, Berkeley  
<http://www.cs.berkeley.edu/~jordan>



10

# (Some) Categories of Probabilistic Models

11

## Parametric vs non-parametric models

Parametric	Non-Parametric
Determined by fixed, finite number of parameters	Number of parameters grows with data, potentially infinite
Limited flexibility	More flexible
Efficient statistically and computationally	Less efficient

*Examples to come!* There are non/parametric models in both the frequentist and Bayesian schools.

12

## Generative vs. discriminative models

- X's are instances, Y's are labels (supervised setting!)
  - \* Given: i.i.d. data  $(X_1, Y_1), \dots, (X_n, Y_n)$
  - \* Find model that can predict  $Y$  of new  $X$
- Generative approach
  - \* Model full joint  $P(X, Y)$
- Discriminative approach
  - \* Model conditional  $P(Y|X)$  only
- Both have pro's and con's

*Examples to come!* There are generative/discriminative models in both the frequentist and Bayesian schools.

13

## Summary

- Bayesian paradigm: Its all in the prior!
- Bayesian point estimate: MAP (an extremum estimator)
- Parametric vs Non-parametric models
- Discriminative vs. Generative models

Next: Logistic regression (unlike you've ever seen before)

Workshops week #2: learning Bayes one coin flip at a time!



week 2

# Lecture 3a. Linear Regression – Decision Theory

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

COMP90051 Statistical Machine Learning

## This lecture

- **Linear regression**
  - \* Simple model (convenient maths at expense of flexibility)
  - \* Often needs less data, “interpretable”, lifts to non-linear
  - \* Derivable under all Statistical Schools: Lect 2 case study
    - This week: Frequentist + **Decision theory derivations**
  - Later in semester: Bayesian approach
  - \* Convenient optimisation: Training by “analytic” (exact) solution
- Basis expansion: Data transform for more expressive models

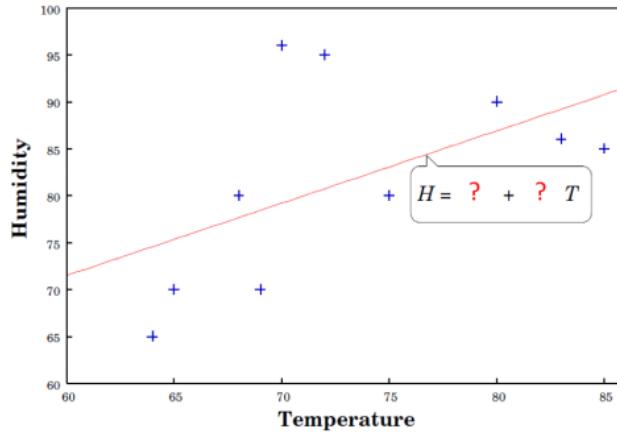
# Linear Regression via Decision Theory

A warm-up example

3

Example: Predict humidity from temperature

Temperature	Humidity
<b>TRAINING DATA</b>	
85	85
80	90
83	86
70	96
68	80
65	70
64	65
72	95
69	70
75	80
<b>TEST DATA</b>	
75	70

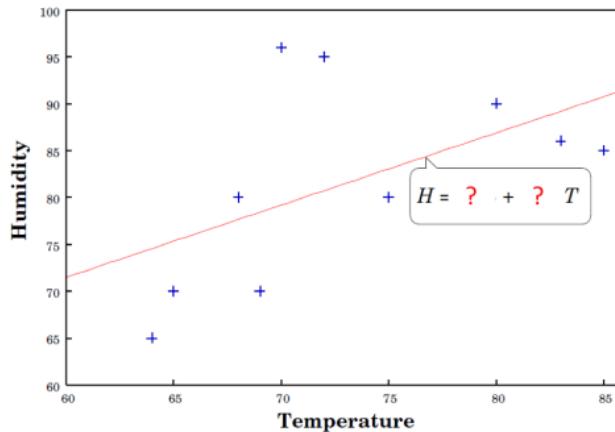


In regression, the task is to predict numeric response (aka dependent variable) from features (aka predictors or independent variables)  
 Assume a linear relation:  $H = a + bT$   
 ( $H$  – humidity;  $T$  – temperature;  $a, b$  – parameters)

4

## Example: Problem statement

- The model is  $H = a + bT$
- Fitting the model = finding "best"  $a, b$  values for data at hand
- Important criterion: minimise the sum of squared errors (aka residual sum of squares)



5

## Example: Minimise Sum Squared Errors

To find  $a, b$  that minimise  $L = \sum_{i=1}^{10} (H_i - (a + b T_i))^2$

set derivatives to zero:

$$\frac{\partial L}{\partial a} = -2 \sum_{i=1}^{10} (H_i - a - b T_i) = 0$$

if we know  $b$ , then  $\hat{a} = \frac{1}{10} \sum_{i=1}^{10} (H_i - b T_i)$

$$\frac{\partial L}{\partial b} = -2 \sum_{i=1}^{10} T_i (H_i - a - b T_i) = 0$$

if we know  $a$ , then  $\hat{b} = \frac{1}{\sum_{i=1}^{10} T_i^2} \sum_{i=1}^{10} T_i (H_i - a)$

High-school optimisation:

- Write derivative
- Set to zero
- Solve for model
- (Check 2<sup>nd</sup> derivatives)

empirical risk minimization  
→ analytic solution

Can we be more systematic?

6

## Example: Analytic solution

- We have two equations and two unknowns  $a, b$
- Rewrite as a system of linear equations

$$\begin{pmatrix} 10 & \sum_{i=1}^{10} T_i \\ \sum_{i=1}^{10} T_i & \sum_{i=1}^{10} T_i^2 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{10} H_i \\ \sum_{i=1}^{10} T_i H_i \end{pmatrix}$$

- **Analytic solution:**  $a = 25.3, b = 0.77$
- (Solve using `numpy.linalg.solve` or sim.)

7

## More general decision rule

- Adopt a linear relationship between response  $y \in \mathbb{R}$  and an instance with features  $x_1, \dots, x_m \in \mathbb{R}$

$$\hat{y} = w_0 + \sum_{i=1}^m x_i w_i$$

Here  $w_0, \dots, w_m \in \mathbb{R}$  denote weights (model parameters)

- Trick: add a dummy feature  $x_0 = 1$  and use vector notation

$$\hat{y} = \sum_{i=0}^m x_i w_i = \mathbf{x}' \mathbf{w}$$

A lowercase symbol in **bold face** indicates a vector;  $\mathbf{x}'$  denotes transpose

8

## Summary

- Linear regression
  - \* Simple, effective, “interpretable”, basis for many approaches
  - \* Decision-theoretic frequentist derivation

Next time:

Frequentist derivation; Solution/training approach

9

## Lecture 3b. Linear Regression - Frequentist.

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

# This lecture

- **Linear regression**
  - \* Simple model (convenient maths at expense of flexibility)
  - \* Often needs less data, “interpretable”, lifts to non-linear
  - \* Derivable under all Statistical Schools: Lect 2 case study
    - This week: Frequentist + Decision theory derivations
    - Later in semester: Bayesian approach
  - \* Convenient optimisation: Training by “analytic” (exact) solution
- Basis expansion: Data transform for more expressive models

2

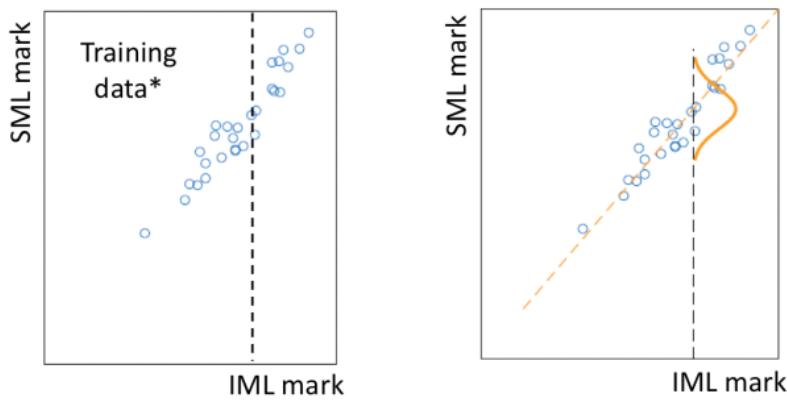
## Linear Regression via Frequentist Probabilistic Model

Max-Likelihood Estimation

3

# Data is noisy!

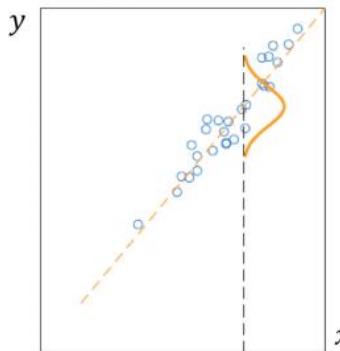
Example: predict mark for Statistical Machine Learning (SML)  
from mark for Intro ML (IML aka KT)



\* synthetic data :)

4

# Regression as a probabilistic model



- Assume a probabilistic model:  $Y = \mathbf{X}'\mathbf{w} + \varepsilon$ 
  - Here  $\mathbf{X}$ ,  $Y$  and  $\varepsilon$  are r.v.'s
  - Variabile  $\varepsilon$  encodes noise
- Next, assume Gaussian noise (indep. of  $\mathbf{X}$ ):  
 $\varepsilon \sim \mathcal{N}(0, \sigma^2)$

$$Y \sim \mathcal{N}(\mathbf{X}'\mathbf{w}, \sigma^2)$$

- Recall that  $\mathcal{N}(x; \mu, \sigma^2) \equiv \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

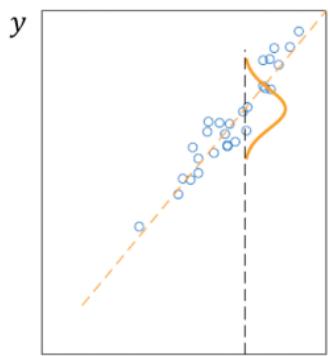
this is a squared error!

- Therefore

$$p_{\mathbf{w}, \sigma^2}(y|\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \mathbf{x}'\mathbf{w})^2}{2\sigma^2}\right)$$

5

## Parametric probabilistic model



- Using simplified notation, **discriminative model** is:

$$p(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - x'w)^2}{2\sigma^2}\right)$$

- Unknown parameters:  $w, \sigma^2$

$\sigma^2$  known

- Given observed data  $\{(X_1, Y_1), \dots, (X_n, Y_n)\}$ , we want to find parameter values that "best" explain the data
- Maximum-likelihood estimation:** choose parameter values that maximise the probability of observed data

6

## Maximum likelihood estimation

- Assuming independence of data points, the probability of data is

$$p(y_1, \dots, y_n | x_1, \dots, x_n) = \prod_{i=1}^n p(y_i | x_i)$$

- For  $p(y_i | x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - x_i'w)^2}{2\sigma^2}\right)$
- "Log trick": Instead of maximising this quantity, we can maximise its logarithm (Why? Explained soon)

$$\sum_{i=1}^n \log p(y_i | x_i) = -\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - x_i'w)^2 + C$$

here  $C$  doesn't depend on  $w$  (it's a constant)

the sum of squared errors!

- Under this model, maximising log-likelihood as a function of  $w$  is equivalent to minimising the sum of squared errors

7

# Method of least squares

- Training data:  $\{(x_1, y_1), \dots, (x_n, y_n)\}$ . Note bold face in  $x_i$
- For convenience, place instances in rows (so attributes go in columns), representing training data as an  $n \times (m + 1)$  matrix  $X$ , and  $n$  vector  $y$

Analytic solution:

- Write derivative
- Set to zero
- Solve for model

- Probabilistic model/decision rule assumes  $y \approx Xw$
- To find  $w$ , minimise the sum of squared errors

$$L = \sum_{i=1}^n \left( y_i - \sum_{j=0}^m X_{ij} w_j \right)^2$$

$$\frac{\partial L}{\partial w} = 2X'(y - Xw) = 0$$

$$X'y = X'Xw$$

- Setting derivative to zero and solving for  $w$  yields

$$\hat{w} = (X'X)^{-1}X'y$$

- \* This system of equations called the **normal equations**
- \* System is well defined only if the inverse exists

In this slide: UPPERCASE symbol in bold face means matrix;  $X'$  denotes transpose;  $X^{-1}$  denotes matrix inverse

8



# Wherefore art thou: Bayesian derivation?

- Later in the semester: return of linear regression
- Fully Bayesian, with a posterior:
  - \* **Bayesian linear regression**
- Bayesian (MAP) point estimate of weight vector:
  - \* Adds a penalty term to sum of squared losses
  - \* Equivalent to  $L_2$  “regularisation” to be covered next week
  - \* Called: **ridge regression**

9

## Summary

- Linear regression
    - \* Simple, effective, “interpretable”, basis for many approaches
    - \* Probabilistic frequentist derivation
    - \* Solution by normal equations
- Later in semester: Bayesian approaches

Next time: Basis expansion for non-linear regression

10

## Lecture 3c. Basis Expansion.

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

# This lecture

- Linear regression
  - \* Simple model (convenient maths at expense of flexibility)
  - \* Often needs less data, “interpretable”, lifts to non-linear
  - \* Derivable under all Statistical Schools: Lect 2 case study
    - This week: Frequentist + Decision theory derivations
  - Later in semester: Bayesian approach
  - \* Convenient optimisation: Training by “analytic” (exact) solution
- **Basis expansion: Data transform for more expressive models**

2

# Basis Expansion

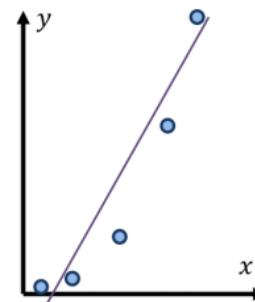
Extending the utility of models via  
data transformation

3

## Basis expansion for linear regression

- Real data is likely to be non-linear
- What if we still wanted to use a linear regression?
  - Simple, easy to understand, computationally efficient, etc.
- How to marry non-linear data to a linear method?

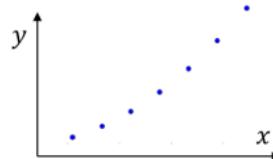
*If you can't beat'em, join'em*



4

## Transform the data

- The trick is to transform the data: Map data into another features space, s.t. data is linear in that space
- Denote this transformation  $\varphi: \mathbb{R}^m \rightarrow \mathbb{R}^k$ . If  $x$  is the original set of features,  $\varphi(x)$  denotes new feature set
- Example: suppose there is just one feature  $x$ , and the data is scattered around a parabola rather than a straight line

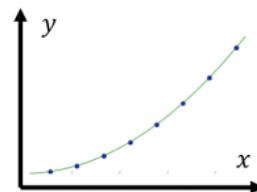


5

## Example: Polynomial regression

- No worries, mate: define

$$\begin{aligned}\varphi_1(x) &= x \\ \varphi_2(x) &= x^2\end{aligned}$$



- Next, apply linear regression to  $\varphi_1, \varphi_2$

$$y = w_0 + w_1 \varphi_1(x) + w_2 \varphi_2(x) = w_0 + w_1 x + w_2 x^2$$

and here you have **quadratic regression**

- More generally, obtain **polynomial regression** if the new set of attributes are powers of  $x$
- Similar idea basis of autoregression for time series

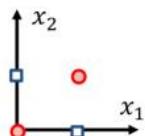
6

## Example: linear classification

- Example binary classification problem: Dataset not linearly separable
- Define transformation as

$$\varphi_i(\mathbf{x}) = \|\mathbf{x} - \mathbf{z}_i\|, \text{ where } \mathbf{z}_i \text{ some pre-defined constants}$$

- Choose  $\mathbf{z}_1 = [0,0]', \mathbf{z}_2 = [0,1]', \mathbf{z}_3 = [1,0]', \mathbf{z}_4 = [1,1]'$



there exist weights that make new data separable, e.g.:

$w_1$	$w_2$	$w_3$	$w_4$
1	0	0	1

The transformed data is linearly separable!

$x_1$	$x_2$	$y$
0	0	Class A
0	1	Class B
1	0	Class B
1	1	Class A

$\varphi_1$	$\varphi_2$	$\varphi_3$	$\varphi_4$
0	1	1	$\sqrt{2}$
1	0	$\sqrt{2}$	1
1	$\sqrt{2}$	0	1
$\sqrt{2}$	1	1	0

$\varphi'w$	$y$
$\sqrt{2}$	Class A
2	Class B
2	Class B
$\sqrt{2}$	Class A

7

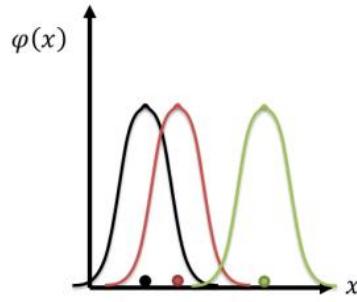
## Radial basis functions

- Previous example: motivated by approximation theory where sums of RBFs approx. functions
- A **radial basis function** is a function of the form  $\varphi(x) = \psi(\|x - z\|)$ , where  $z$  is a constant

- Examples:

- $\varphi(x) = \|x - z\|$

- $\varphi(x) = \exp\left(-\frac{1}{\sigma} \|x - z\|^2\right)$



8

## Challenges of basis expansion

- Basis expansion can significantly increase the utility of methods, especially, linear methods
- In the above examples, one limitation is that the transformation needs to be defined beforehand
  - \* Need to choose the size of the new feature set
  - \* If using RBFs, need to choose  $z_i$
- Regarding  $z_i$ , one can choose uniformly spaced points, or cluster training data and use cluster centroids
- Another popular idea is to use training data  $z_i \equiv x_i$ 
  - \* E.g.,  $\varphi_i(x) = \psi(\|x - x_i\|)$
  - \* However, for large datasets, this results in a large number of features → computational hurdle

9



## Further directions

- There are several avenues for taking the idea of basis expansion to the next level
  - \* Will be covered later in this subject
- One idea is to *learn* the transformation  $\varphi$  from data
  - \* E.g., Artificial Neural Networks
- Another powerful extension is the use of the *kernel trick*
  - \* “Kernelised” methods, e.g., kernelised perceptron
- Finally, in *sparse kernel machines*, training depends only on a few data points
  - \* E.g., SVM

10

## Summary

- Basis expansion
  - \* Extending model expressiveness via data transformation
  - \* Examples for linear and logistic regression
  - \* Theoretical notes

### Next time:

First/second-order iteration optimisation;  
 Logistic regression - linear probabilistic model for classification.

11

# Lecture 4a. Iterative Optimisation.

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

[COMP90051 Statistical Machine Learning](#)

## This lecture

- **Iterative optimisation** for extremum estimators
  - \* First-order method: **Gradient descent**
  - \* Second-order: **Newton-Raphson method**
  - Later: Lagrangian duality
- Logistic regression: workhorse linear classifier
  - \* Possibly familiar derivation: frequentist
  - \* Decision-theoretic derivation
  - \* Training with Newton-Raphson looks like repeated, weighted linear regression

# Gradient Descent

Brief review of most basic optimisation approach in ML

3

## Optimisation formulations in ML

- Training = Fitting = Parameter estimation
- Typical formulation

$$\hat{\theta} \in \operatorname{argmin}_{\theta \in \Theta} L(\text{data}, \theta)$$

- \* argmin because we want a minimiser not the minimum
  - Note: argmin can return a set (minimiser not always unique!)
- \*  $\Theta$  denotes a model family (including constraints)
- \*  $L$  denotes some objective function to be optimised
  - E.g. MLE: (conditional) likelihood
  - E.g. Decision theory: (regularised) empirical risk

4

## One we've seen: Log trick

product → sub

- Instead of optimising  $L(\theta)$ , try convenient  $\log L(\theta)$
- Why are we allowed to do this?
- Strictly monotonic function:  $a > b \Rightarrow f(a) > f(b)$ 
  - \* Example: log function!
- Lemma:** Consider any objective function  $L(\theta)$  and any strictly monotonic  $f$ .  $\theta^*$  is an optimiser of  $L(\theta)$  if and only if it is an optimiser of  $f(L(\theta))$ .
  - \* Proof: Try it at home for fun!

5

## Two solution approaches

- Analytic (aka closed form) solution**
  - \* Known only in limited number of cases
  - \* Use 1<sup>st</sup>-order necessary condition for optimality\*:
 
$$\frac{\partial L}{\partial \theta_1} = \dots = \frac{\partial L}{\partial \theta_p} = 0$$
- Approximate iterative solution**
  - Initialisation: choose starting guess  $\theta^{(1)}$ , set  $i = 1$
  - Update:  $\theta^{(i+1)} \leftarrow \text{SomeRule}[\theta^{(i)}]$ , set  $i \leftarrow i + 1$
  - Termination: decide whether to Stop
  - Go to Step 2
  - Stop: return  $\hat{\theta} \approx \theta^{(i)}$

Assuming unconstrained, differentiable  $L$

\* Note: to check for local minimum, need positive 2<sup>nd</sup> derivative (or Hessian positive definite); this assumes unconstrained – in general need to also check boundaries. See also Lagrangian techniques later in subject.

6

## Reminder: The gradient

- Gradient at  $\theta$  defined as  $\left[ \frac{\partial L}{\partial \theta_1}, \dots, \frac{\partial L}{\partial \theta_p} \right]'$  evaluated at  $\theta$
- The gradient points to the direction of maximal change of  $L(\theta)$  when departing from point  $\theta$
- Shorthand notation
  - \*  $\nabla L \stackrel{\text{def}}{=} \left[ \frac{\partial L}{\partial \theta_1}, \dots, \frac{\partial L}{\partial \theta_p} \right]'$  computed at point  $\theta$
  - \* Here  $\nabla$  is the “nabla” symbol
- Hessian matrix at  $\theta$ :  $\nabla^2 L_{ij} = \frac{\partial^2 L}{\partial \theta_i \partial \theta_j}$



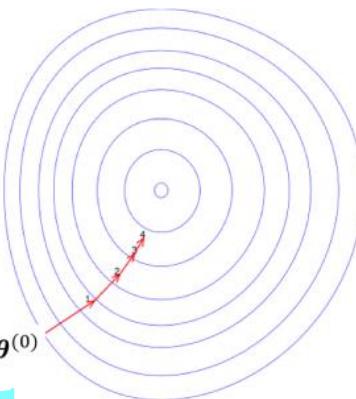
7

## Gradient descent and SGD

- Choose  $\theta^{(1)}$  and some  $T$
  - For  $i$  from 1 to  $T^*$ 
    - $\theta^{(i+1)} = \theta^{(i)} - \eta \nabla L(\theta^{(i)})$
  - Return  $\hat{\theta} \approx \theta^{(i)}$
- Note:  $\eta$  dynamically updated per step
  - Variants: Momentum, AdaGrad, ...
  - Stochastic gradient descent: two loops
    - Outer for loop: each loop (called epoch) sweeps through all training data
    - Within each epoch, randomly shuffle training data; then for loop: do gradient steps only on batches of data. Batch size might be 1 or few

\*Other stopping criteria can be used

faster to compute

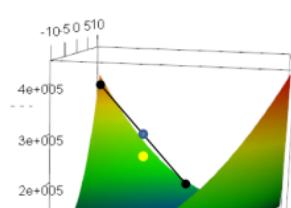
 Assuming  $L$  is differentiable


Wikimedia Commons. Authors: Olegalexandrov, Zerodamage

8

## Convex objective functions

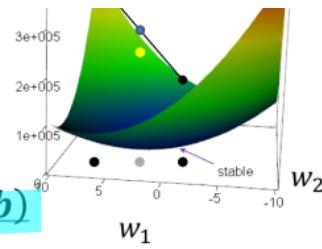
- ‘Bowl shaped’ functions → global minimum
- Informally: if line segment between any two points on graph of function lies above or on graph



any two points on graph of function lies above or on graph

- Formally\*  $f: D \rightarrow \mathbf{R}$  is convex if  $\forall \mathbf{a}, \mathbf{b} \in D, t \in [0,1]$ :  
$$f(t\mathbf{a} + (1-t)\mathbf{b}) \leq tf(\mathbf{a}) + (1-t)f(\mathbf{b})$$
 Strictly convex if inequality is strict ( $<$ )

- Gradient descent on (strictly) convex function guaranteed to find a (unique) global minimum!



\* Aside: Equivalently we can look to the second derivative. For  $f$  defined on scalars, it should be non-negative; for multivariate  $f$ , the Hessian matrix should be positive semi-definite (see linear algebra supplemental deck).

9

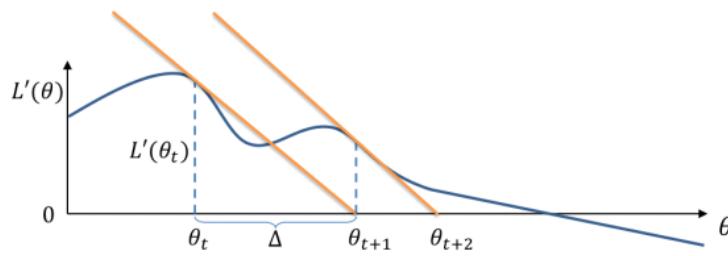
COMP90051 Statistical Machine Learning

## Newton-Raphson

A second-order method:  
Successive root finding in the  
objective's derivative.

10

## Newton-Raphson: Derivation (1D)



- Critical points of  $L(\theta) = \text{Zero-crossings of } L'(\theta)$
- Consider case of scalar  $\theta$ . Starting at given/random  $\theta_0$ , iteratively:
  1. Fit tangent line to  $L'(\theta)$  at  $\theta_t$
  2. Need to find  $\theta_{t+1} = \theta_t + \Delta$  using linear approximation's zero crossing
  3. Tangent line given by derivative: rise/run =  $-L''(\theta_t) = L'(\theta_t)/\Delta$
  4. Therefore iterate is  $\theta_{t+1} = \theta_t - L'(\theta_t)/L''(\theta_t)$

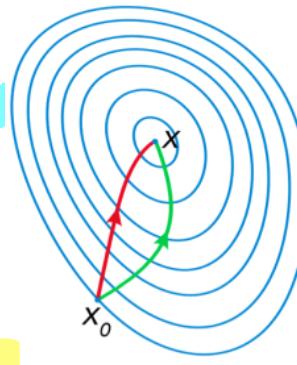
11

## Newton-Raphson: General case

- Newton-Raphson summary
  - \* Finds  $L'(\theta)$  zero-crossings
  - \* By successive linear approximations to  $L'(\theta)$
  - \* Linear approximations involve derivative of  $L'(\theta)$ , ie.  $L''(\theta)$
- Vector-valued  $\theta$ :
 

How to fix scalar  $\theta_{t+1} = \theta_t - L'(\theta_t)/L''(\theta_t)???$

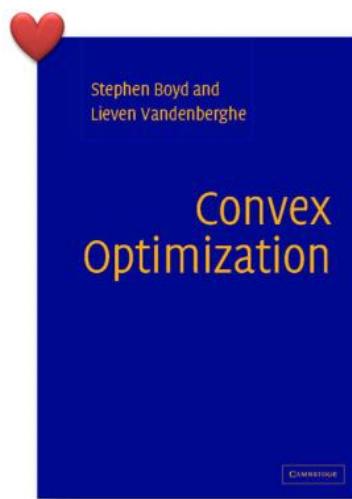
  - \*  $L'(\theta)$  is  $\nabla L(\theta)$
  - \*  $L''(\theta)$  is  $\nabla_2 L(\theta)$
  - \* Matrix division is matrix inversion
- General case:  $\theta_{t+1} = \theta_t - (\nabla_2 L(\theta_t))^{-1} \nabla L(\theta_t)$ 
  - \* Pro: May converge faster; fitting a quadratic with curvature information
  - \* Con: Sometimes computationally expensive, unless approximating Hessian



12

## ...And much much more

- What if you have constraints?
  - \* See Lagrangian multipliers (let's you bring constraints into objective)
  - \* Or, projected gradient descent (you iterate between GD on objective, and GD on each constraints)
- What about speed of convergence?
- Do you really need differentiable objectives? (no, subgradients)
- Are there more tricks? (Hell yeah! But outside scope here)



Free at <http://web.stanford.edu/~boyd/cvxbook/>

13

## Summary

- Iterative optimisation for ML
  - \* First-order: Gradient Descent and Stochastic GD
  - \* Convex objectives: Convergence to global optima
  - \* Second-order: Newton-Raphson can be faster, can be expensive to build/invert full Hessian

Next time: Logistic regression for binary classification

14

# Lecture 4b. Logistic Regression.

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

COMP90051 Statistical Machine Learning

## This lecture

- Iterative optimisation for extremum estimators
  - \* First-order method: Gradient descent
  - \* Second-order: Newton-Raphson method
  - Later: Lagrangian duality
- **Logistic regression:** workhorse linear classifier
  - \* Possibly familiar derivation: **frequentist**
  - \* **Decision-theoretic** derivation
  - \* Training with Newton-Raphson looks like repeated, weighted linear regression

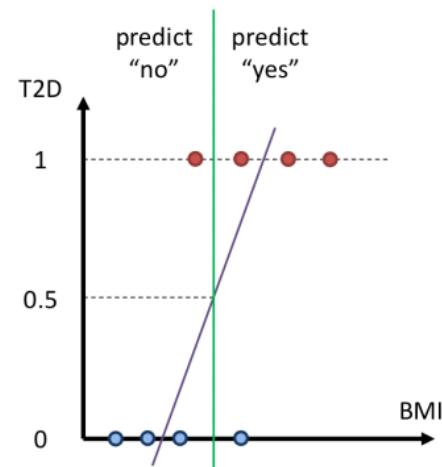
# Logistic Regression Model

A workhorse linear, binary classifier;  
 (A review for some of you; new to some.)

3

## Binary classification: Example

- **Example:** given body mass index (BMI) does a patient have type 2 diabetes (T2D)?
- This is (supervised) **binary classification**
- One *could* use linear regression
  - \* Fit a line/hyperplane to data (find weights  $w$ )
  - \* Denote  $s \equiv x'w$
  - \* Predict "Yes" if  $s \geq 0.5$
  - \* Predict "No" if  $s < 0.5$

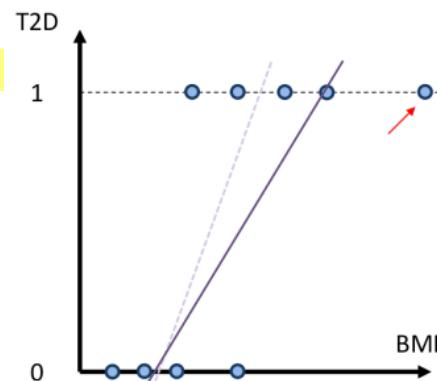


4

## Why not linear regression

- Due to the square loss, points far from boundary have loss squared – even if they're confidently correct!

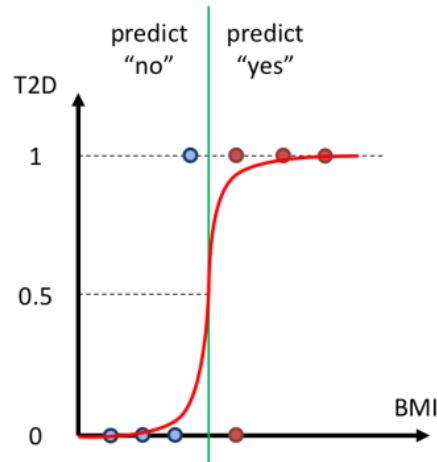
- Such “outliers” will “pull at” the linear regression
- Overall, the least-squares criterion looks unnatural in this setting



5

## Logistic regression model

- Probabilistic approach to classification
  - \*  $P(Y = 1|x) = f(x) = ?$
  - \* Use a linear function? E.g.,  $s(x) = \mathbf{x}'\mathbf{w}$
- Problem: the probability needs to be between 0 and 1.
- Logistic function  $f(s) = \frac{1}{1 + \exp(-s)}$
- Logistic regression model
 
$$P(Y = 1|x) = \frac{1}{1 + \exp(-x'\mathbf{w})}$$



6

# How is logistic regression *linear*?

- Logistic regression model:

$$P(Y=1|x) = \frac{1}{1 + \exp(-x'w)}$$

- Classification rule:

if  $(P(Y=1|x) > \frac{1}{2})$  then class "1", else class "0"

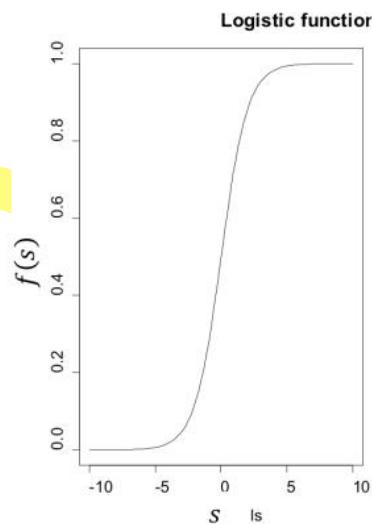
- Decision boundary is the set of  $x$ 's such that:

$$\frac{1}{1 + \exp(-x'w)} = \frac{1}{2}$$

$$\exp(-x'w) = 1$$

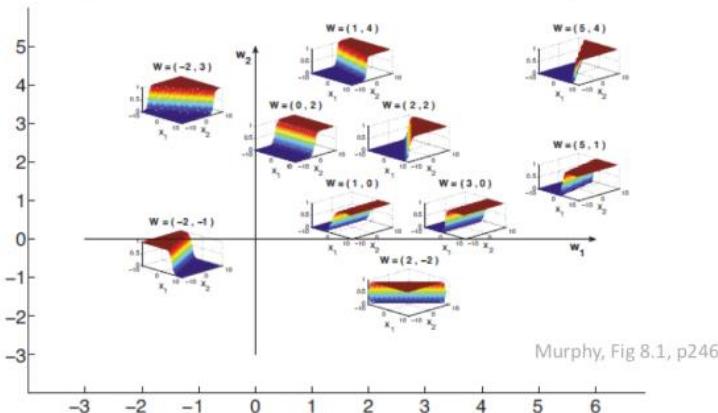
$$x'w = 0$$

hyperplane



7

## Effect of parameter vector (2D problem)



- Decision boundary is the line where  $P(Y=1|x) = 0.5$ 
  - In higher dimensional problems, the decision boundary is a plane or hyperplane
- Vector  $w$  is perpendicular to the decision boundary (see linear algebra review topic)
  - That is,  $w$  is a normal to the decision boundary
  - Note: in this illustration we assume  $w_0 = 0$  for simplicity

Probability  $\rightarrow$  non-linear

8

## Linear vs. logistic probabilistic models

- Linear regression assumes a Normal distribution with a fixed variance and mean given by linear model

$$p(y|x) = \text{Normal}(x'w, \sigma^2)$$

$$\text{Var} = \sigma^2$$

- Logistic regression assumes a Bernoulli distribution with parameter given by logistic transform of linear model

- Logistic regression assumes a Bernoulli distribution with parameter given by logistic transform of linear model  

$$p(y|x) = \text{Bernoulli}(\text{logistic}(x'w))$$
- Recall that Bernoulli distribution is defined as  

$$p(1) = \theta \text{ and } p(0) = 1 - \theta \text{ for } \theta \in [0,1]$$
- Equivalently  $p(y) = \theta^y(1-\theta)^{1-y}$  for  $y \in \{0,1\}$

9

COMP90051 Statistical Machine Learning

## Training as Max-Likelihood Estimation

- Assuming independence, probability of data  

$$p(y_1, \dots, y_n | x_1, \dots, x_n) = \prod_{i=1}^n p(y_i | x_i)$$
- Assuming Bernoulli distribution we have  

$$p(y_i | x_i) = (\theta(x_i))^{y_i} (1 - \theta(x_i))^{1-y_i}$$
  
where  $\theta(x_i) = \frac{1}{1 + \exp(-x_i'w)}$
- Training: maximise this expression wrt weights  $w$

10

## Apply log trick, simplify

- Instead of maximising likelihood, maximise its logarithm

$$\begin{aligned}
 \log\left(\prod_{i=1}^n p(y_i|\mathbf{x}_i)\right) &= \sum_{i=1}^n \log p(y_i|\mathbf{x}_i) \\
 &= \sum_{i=1}^n \log\left((\theta(\mathbf{x}_i))^{y_i}(1-\theta(\mathbf{x}_i))^{1-y_i}\right) \\
 &= \sum_{i=1}^n (y_i \log(\theta(\mathbf{x}_i)) + (1-y_i) \log(1-\theta(\mathbf{x}_i))) \\
 &= \sum_{i=1}^n ((y_i - 1)\mathbf{x}'_i \mathbf{w} - \log(1 + \exp(-\mathbf{x}'_i \mathbf{w})))
 \end{aligned}$$

Can't do this analytically

11

## Logistic Regression: Decision-Theoretic View

Via cross-entropy loss

12

## Background: Cross entropy

- Cross entropy is an information-theoretic method for comparing two distributions
- Cross entropy is a measure of a divergence between reference distribution  $g_{ref}(a)$  and estimated distribution  $g_{est}(a)$ . For discrete distributions:

$$H(g_{ref}, g_{est}) = - \sum_{a \in A} g_{ref}(a) \log g_{est}(a)$$

$A$  is support of the distributions, e.g.,  $A = \{0,1\}$

13

## Training as cross-entropy minimisation

- Consider log-likelihood for a single data point  
 $\log p(y_i | \mathbf{x}_i) = y_i \log(\theta(\mathbf{x}_i)) + (1 - y_i) \log(1 - \theta(\mathbf{x}_i))$
- Cross entropy  $H(g_{ref}, g_{est}) = - \sum_a g_{ref}(a) \log g_{est}(a)$ 
  - \* If reference (true) distribution is  
 $g_{ref}(1) = y_i$  and  $g_{ref}(0) = 1 - y_i$
  - \* With logistic regression estimating this distribution as  
 $g_{est}(1) = \theta(\mathbf{x}_i)$  and  $g_{est}(0) = 1 - \theta(\mathbf{x}_i)$

It finds  $w$  that minimises sum of cross entropies per training point

minimise loss  $\rightarrow$  equivalent MLE

14

## Summary

- Logistic regression formulation
  - \* A workhorse linear binary classifier
  - \* Frequentist: Bernoulli label with coin bias logistic-linear in  $\mathbf{x}$
  - \* Decision theory: Maximising cross entropy with labels

## Summary

- Logistic regression *formulation*
  - \* A workhorse linear binary classifier
  - \* Frequentist: Bernoulli label with coin bias logistic-linear in  $\mathbf{x}$
  - \* Decision theory: Maximising cross entropy with labels

Next time: Training quickly with Newton-Raphson, and how that is repeated (weighted) linear regression under the hood!

15

## Lecture 4c. Training logistic regression with the IRLS algorithm

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

## This lecture

- Iterative optimisation for extremum estimators
  - \* First-order method: Gradient descent
  - \* Second-order: Newton-Raphson method
- Later: Lagrangian duality
- Logistic regression: workhorse linear classifier
  - \* Possibly familiar derivation: frequentist
  - \* Decision-theoretic derivation
  - \* **Training with Newton-Raphson** looks like repeated, weighted linear regression

2

## Training Logistic Regression: the IRLS Algorithm

Analytical? Newton-Raphson!

3

## Iterative optimisation

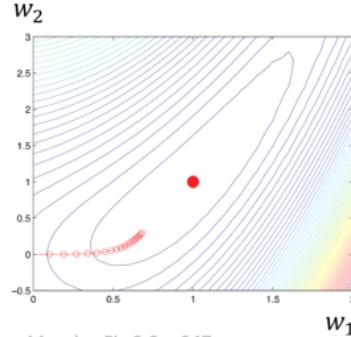
- Training logistic regression:  $\mathbf{w}$  maximising log-likelihood  $L(\mathbf{w})$  or cross-entropy loss
- **Bad news:** No closed form solution
- **Good news:** Problem is strictly convex, if no irrelevant features  $\rightarrow$  convergence!



Look ahead: regularisation for irrelevant features

How does gradient descent work?

- $\mu(z) = \frac{1}{1+\exp(-z)}$  then  $\frac{d\mu}{dz} = \mu(z)(1-\mu(z))$
- Then  $\nabla L(\mathbf{w}) = \sum_{i=1}^n (y_i - \mu(\mathbf{x}_i)) \mathbf{x}_i = \mathbf{X}'(\mathbf{y} - \boldsymbol{\mu})$ , stacking instances in  $\mathbf{X}$ , labels in  $\mathbf{y}$ ,  $\mu(\mathbf{x}_i)$  in  $\boldsymbol{\mu}$



Murphy, Fig 8.3, p247

Note I'm abusing notation:  
 $\mu(\mathbf{x}_i) = \mu(z)$  where  $z = \mathbf{w}'\mathbf{x}_i$   
 Meaning by input type

4

## Iteratively-Reweighted Least Squares

- Instead of GD, let's apply Newton-Raphson  $\rightarrow$  IRLS algorithm
- Recall:  $\nabla L(\mathbf{w}) = \mathbf{X}'(\mathbf{y} - \boldsymbol{\mu})$ . Differentiate again for Hessian:
 
$$\nabla_2 L(\mathbf{w}) = -\sum_i \frac{d\mu}{dz_i} \mathbf{x}_i \mathbf{x}_i' = -\sum_i \mu(\mathbf{x}_i)(1-\mu(\mathbf{x}_i)) \mathbf{x}_i \mathbf{x}_i'$$

$$= -\mathbf{X}' \mathbf{M} \mathbf{X}, \text{ where } M_{ii} = \mu_i(1-\mu_i) \text{ otherwise 0}$$
- Newton-Raphson then says (now with  $\mathbf{M}_t, \boldsymbol{\mu}_t$  dependence on  $\mathbf{w}_t$ )
 
$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t - (\nabla_2 L)^{-1} \nabla L = \mathbf{w}_t + (\mathbf{X}' \mathbf{M}_t \mathbf{X})^{-1} \mathbf{X}'(\mathbf{y} - \boldsymbol{\mu}_t) \\ &= (\mathbf{X}' \mathbf{M}_t \mathbf{X})^{-1} [\mathbf{X}' \mathbf{M}_t \mathbf{X} \mathbf{w}_t + \mathbf{X}'(\mathbf{y} - \boldsymbol{\mu}_t)] \\ &= (\mathbf{X}' \mathbf{M}_t \mathbf{X})^{-1} \mathbf{X}' \mathbf{M}_t \mathbf{b}_t, \text{ where } \mathbf{b}_t = \mathbf{X} \mathbf{w}_t + \mathbf{M}_t^{-1} (\mathbf{y} - \boldsymbol{\mu}_t) \end{aligned}$$
- Each IRLS iteration solves a least squares problem weighted by  $\mathbf{M}_t$ , which are reweighted iteratively!

Compare to  
normal  
equations

5

## IRLS intuition: Putting labels on linear scale

IRLS:  $\mathbf{w}_{t+1} = (\mathbf{X}' \mathbf{M}_t \mathbf{X})^{-1} \mathbf{X}' \mathbf{M}_t \mathbf{b}_t$   
 where  $\mathbf{b}_t = \mathbf{X} \mathbf{w}_t + \mathbf{M}_t^{-1} (\mathbf{y} - \boldsymbol{\mu}_t)$   
 and  $M_{ii} = \mu_t(\mathbf{x}_i)[1 - \mu_t(\mathbf{x}_i)]$  otherwise 0  
 and  $\mu_t(\mathbf{x}) = [1 + \exp(-\mathbf{w}_t' \mathbf{x})]^{-1}$

- The  $\mathbf{y}$  are not on linear scale. Invert logistic function?
- The  $\mathbf{b}_t$  are a “linearised” approximation to these: the  $\mathbf{b}_t$  equation matches a linear approx. to  $\mu_t^{-1}(\mathbf{y})$ .
- Linear regression on new labels!

6

## IRLS intuition: Equalising label variance

IRLS:  $\mathbf{w}_{t+1} = (\mathbf{X}' \mathbf{M}_t \mathbf{X})^{-1} \mathbf{X}' \mathbf{M}_t \mathbf{b}_t$   
 where  $\mathbf{b}_t = \mathbf{X} \mathbf{w}_t + \mathbf{M}_t^{-1} (\mathbf{y} - \boldsymbol{\mu}_t)$   
 and  $M_{ii} = \mu_t(\mathbf{x}_i)[1 - \mu_t(\mathbf{x}_i)]$  otherwise 0  
 and  $\mu_t(\mathbf{x}) = [1 + \exp(-\mathbf{w}_t' \mathbf{x})]^{-1}$

- In linear regression, each  $y_i$  has equal variance  $\sigma^2$
- Our  $y_i$  are Bernoulli, variance:  $\mu_t(\mathbf{x}_i)[1 - \mu_t(\mathbf{x}_i)]$
- Our reweighting standardises, dividing by variances!!

$i$  changes  
 $\sigma^2$  not change  
 change

Fun exercise: Show that Newton-Raphson for linear regression gives you the normal equations!

7

## Summary

- Training logistic regression
  - \* No analytical solution
  - \* Gradient descent possible, but convergence rate not ideal
  - \* Newton-Raphson: iteratively reweighted least squares

Next time: Regularised linear regression for avoiding overfitting and ill-posed optimisation

## Lecture 05

2020年8月18日 星期二 13:34



05

# Lecture 5. Regularisation

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

# This lecture

- How irrelevant features make optimisation ill-posed
- Regularising linear regression
  - \* Ridge regression
  - \* The lasso
  - \* Connections to Bayesian MAP
- Regularising non-linear regression
- Bias-variance (again)

30/07/2013 Week 1, Lecture 2

2

# Regularisation

Process of introducing additional information in order to solve an ill-posed problem or to prevent overfitting

- Major technique & theme, throughout ML
- Addresses one or more of the following related problems
  - \* Avoids ill-conditioning (a computational problem)
  - \* Avoids overfitting (a statistical problem)
  - \* Introduce prior knowledge into modelling
- This is achieved by augmenting the objective function
- In this lecture: we cover the first two aspects. We will cover more of regularisation throughout the subject

3

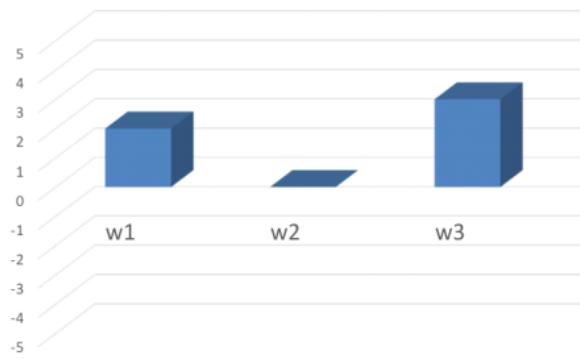
# The Problem with Irrelevant Features

*Linear regression on rank-deficient data.*

4

## Example 1: Feature importance

- Linear model on three features
  - \*  $\mathbf{X}$  is matrix on  $n = 4$  instances (rows)
  - \* Model:  $y = w_1x_1 + w_2x_2 + w_3x_3 + w_0$

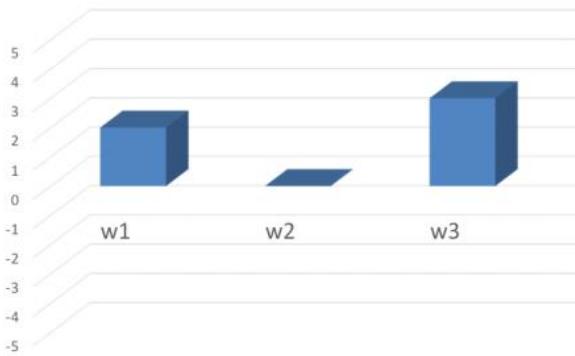


Question: Which feature is more important?

5

## Example 1: Feature importance

- Linear model on three features
  - \*  $\mathbf{X}$  is matrix on  $n = 4$  instances (rows)
  - \* Model:  $y = w_1x_1 + w_2x_2 + w_3x_3 + w_0$



6

## Example 1: Irrelevant features

- Linear model on three features, first two same
  - \*  $\mathbf{X}$  is matrix on  $n = 4$  instances (rows)
  - \* Model:  $y = w_1x_1 + w_2x_2 + w_3x_3 + w_0$
  - \* First two columns of  $\mathbf{X}$  identical
  - \* Feature 2 (or 1) is irrelevant

3	3	7
6	6	9
21	21	79
34	34	2



- Effect of perturbations on model predictions?
  - \* Add  $\Delta$  to  $w_1$
  - \* Subtract  $\Delta$  from  $w_2$

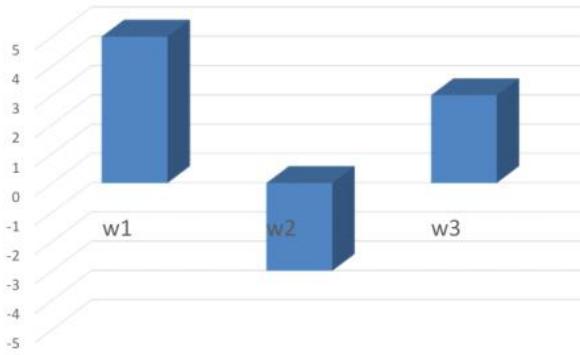
7

## Example 1: Irrelevant features

- Linear model on three features, first two same

- \*  $\mathbf{X}$  is matrix on  $n = 4$  instances (rows)
- \* Model:  $y = w_1x_1 + w_2x_2 + w_3x_3 + w_0$
- \* First two columns of  $\mathbf{X}$  identical
- \* Feature 2 (or 1) is **irrelevant**

3	3	7
6	6	9
21	21	79
34	34	2



- Effect of perturbations on model predictions?
- \* Add  $\Delta$  to  $w_1$
- \* Subtract  $\Delta$  from  $w_2$

8

## Problems with irrelevant features

- In example, suppose  $[\hat{w}_0, \hat{w}_1, \hat{w}_2, \hat{w}_3]'$  is “optimal”
- For any  $\delta$  new  $[\hat{w}_0, \hat{w}_1 + \delta, \hat{w}_2 - \delta, \hat{w}_3]'$  get
  - \* Same predictions!
  - \* Same sum of squared errors!
- Problems this highlights
  - \* The solution is not unique
  - \* Lack of interpretability
  - \* Optimising to learn parameters is **ill-posed problem**

9

## Irrelevant (co-linear) features in general

- Extreme case: features complete clones
- For linear models, more generally
  - \* Feature  $\mathbf{X}_{\cdot j}$  is irrelevant if
  - \*  $\mathbf{X}_{\cdot j}$  is a linear combination of other columns
$$\mathbf{X}_{\cdot j} = \sum_{l \neq j} \alpha_l \mathbf{X}_{\cdot l}$$

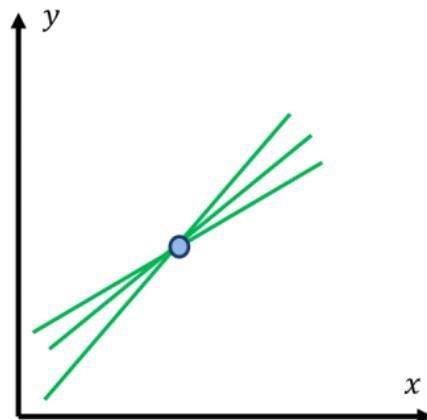
... for some scalars  $\alpha_l$ . Also called multicollinearity
- Equivalently: Some eigenvalue of  $\mathbf{X}'\mathbf{X}$  is zero
- Even near-irrelevance/colinearity can be problematic
  - \* Very small eigenvalues of  $\mathbf{X}'\mathbf{X}$
- Not just a pathological extreme; *easy to happen!*

$\mathbf{X}_{\cdot j}$  denotes the  $j$ -th column of  $X$

10

## Example 2: Lack of data

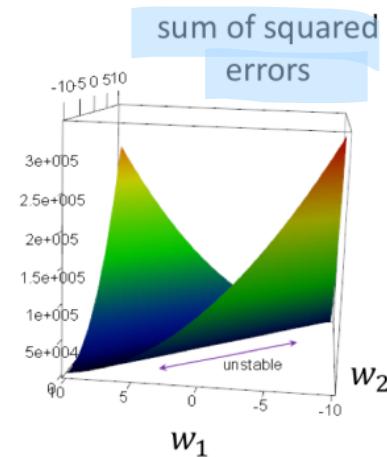
- Extreme example:
  - \* Model has two parameters (slope and intercept)
  - \* Only one data point
- Underdetermined system



11

## III-posed problems

- In both examples, finding the best parameters becomes an **ill-posed problem**
- This means that **the problem solution is not defined**
  - \* In our case  $w_1$  and  $w_2$  cannot be uniquely identified
- Remember normal equations solution of linear regression:  
 $\hat{\mathbf{w}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$
- With irrelevant/multicollinear features, **matrix  $\mathbf{X}'\mathbf{X}$  has no inverse**



convex, but not strictly convex

12

## Mini Summary

- Irrelevant features as collinearity
- Leads to
  - \* Ill-posed optimisation for linear regression
  - \* Broken interpretability
- Multiple intuitions: algebraic, geometric
- Next: Regularisation to the rescue!

13

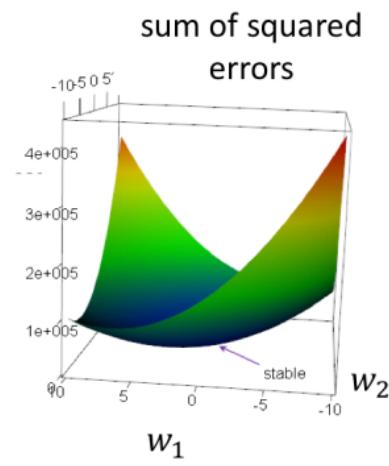
# Regularisation in Linear Models

*Ridge regression and the Lasso*

14

## Re-conditioning the problem

- Regularisation: introduce an **additional condition** into the system
- The original problem is to **minimise**  $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$
- The regularised problem is to minimise  $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$  for  $\lambda > 0$
- The solution is now  $\hat{\mathbf{w}} = (\mathbf{X}'\mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}'\mathbf{y}$
- This formulation is called **ridge regression**
  - \* Turns the ridge into a deep, singular valley
  - \* Adds  $\lambda$  to eigenvalues of  $\mathbf{X}'\mathbf{X}$ ; makes invertible



strictly convex

15

## Regulariser as a prior

- Without regularisation, parameters found based entirely on the information contained in the training set  $\mathbf{X}$ 
  - \* Regularisation introduces additional information
- Recall our probabilistic model  $Y = \mathbf{x}'\mathbf{w} + \varepsilon$ 
  - \* Here  $Y$  and  $\varepsilon$  are random variables, where  $\varepsilon$  denotes noise
- Now suppose that  $\mathbf{w}$  is also a random variable (denoted as  $\mathbf{W}$ ) with a Normal prior distribution Bayesian

$$\mathbf{W} \sim \mathcal{N}(0, 1/\lambda)$$
  - \* I.e. we expect small weights and that no one feature dominates
  - \* Is this always appropriate? E.g. data centring and scaling
  - \* We could encode much more elaborate problem knowledge

16

## Computing posterior using Bayes rule

- The prior is then used to compute the posterior

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$

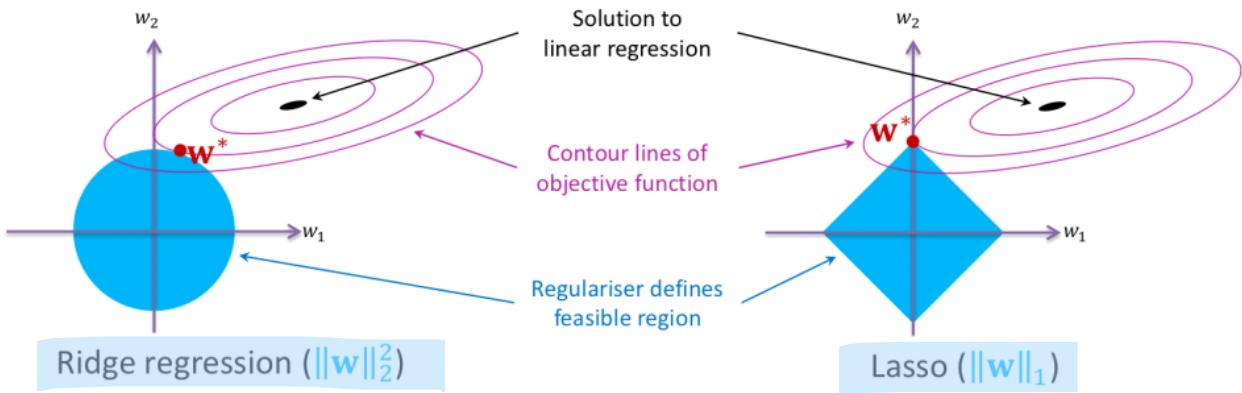
- Instead of maximum likelihood (MLE), take maximum a posteriori estimate (MAP)
  - Apply log trick, so that  $\log(\text{posterior}) = \log(\text{likelihood}) + \log(\text{prior}) - \log(\text{marg})$
  - Arrive at the problem of minimising  $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$
- doesn't depend on  $w$
- this term doesn't affect optimisation

17

## Regulariser as a constraint

- For illustrative purposes, consider a *modified problem*:

$$\text{minimise } \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \text{ subject to } \|\mathbf{w}\|_2^2 \leq \lambda \text{ for } \lambda > 0$$

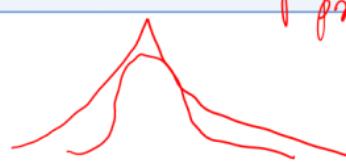


- Lasso ( $L_1$  regularisation) encourages solutions to sit on the axes  
→ Some of the weights are set to zero → Solution is sparse

18

## Regularised linear regression

Algorithm	Minimises	Regulariser	Solution
Linear regression	$\ \mathbf{y} - \mathbf{X}\mathbf{w}\ _2^2$	None	$(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ (if inverse exists)
Ridge regression	$\ \mathbf{y} - \mathbf{X}\mathbf{w}\ _2^2 + \lambda \ \mathbf{w}\ _2^2$	$L_2$ norm 	$(\mathbf{X}'\mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}'\mathbf{y}$
Lasso	$\ \mathbf{y} - \mathbf{X}\mathbf{w}\ _2^2 + \lambda \ \mathbf{w}\ _1$	$L_1$ norm 	No closed-form, but solutions are sparse and suitable for high-dim data



19

## Mini Summary

- $L_2$  regularisation: Ridge regression
  - \* Re-conditions the optimisation
  - \* Equivalent to MAP with Gaussian prior on weights
- $L_1$  regularisation: The Lasso
  - \* Particularly favoured in high-dim, low-example regimes
- Next: Regularisation and non-linear regression

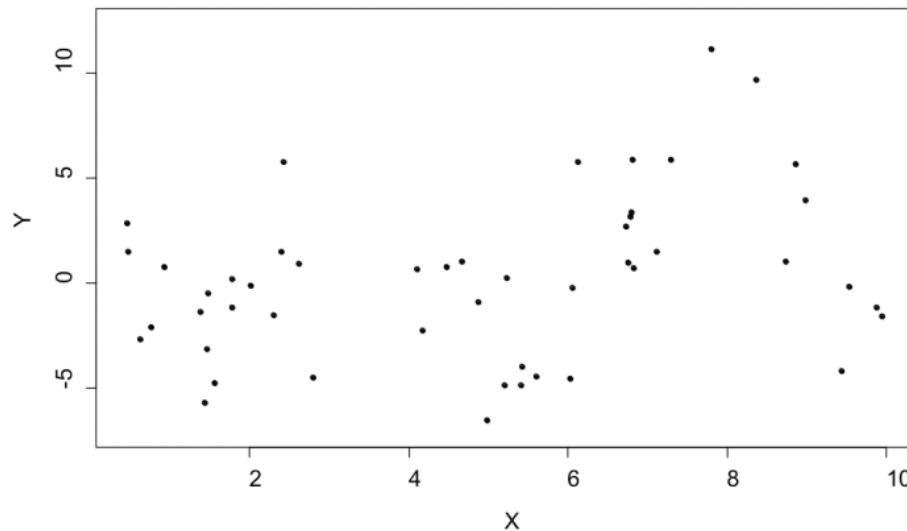
20

## Regularisation in Non-Linear Models

*Model selection in ML*

21

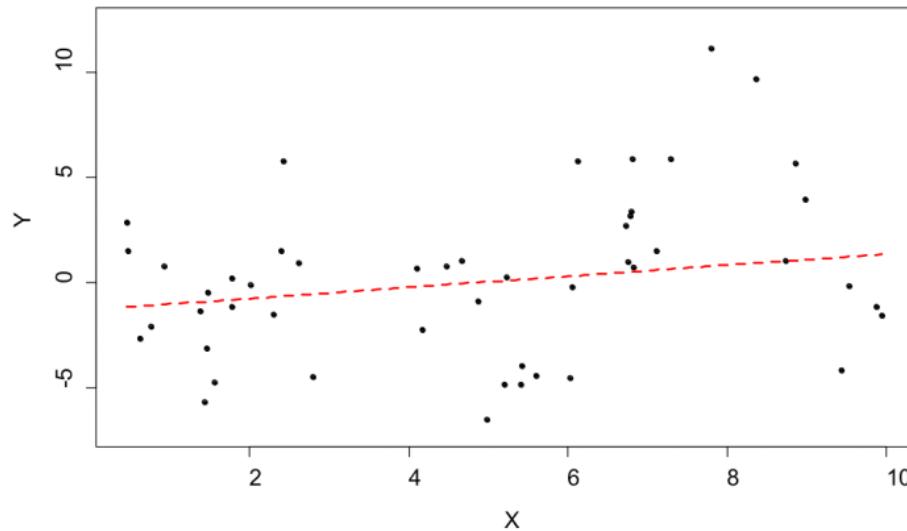
## Example regression problem



How complex a model should we use?

22

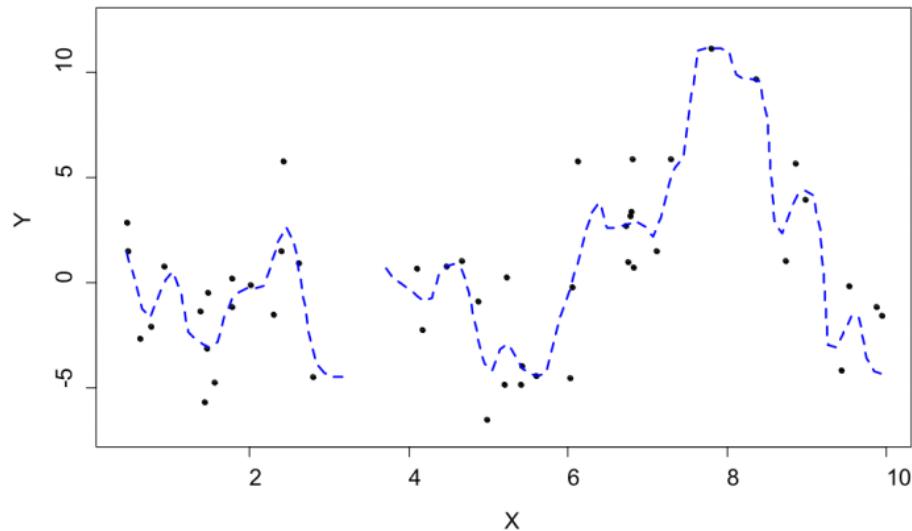
## Underfitting (linear regression)



Model class  $\Theta$  can be too simple to possibly fit true model.

23

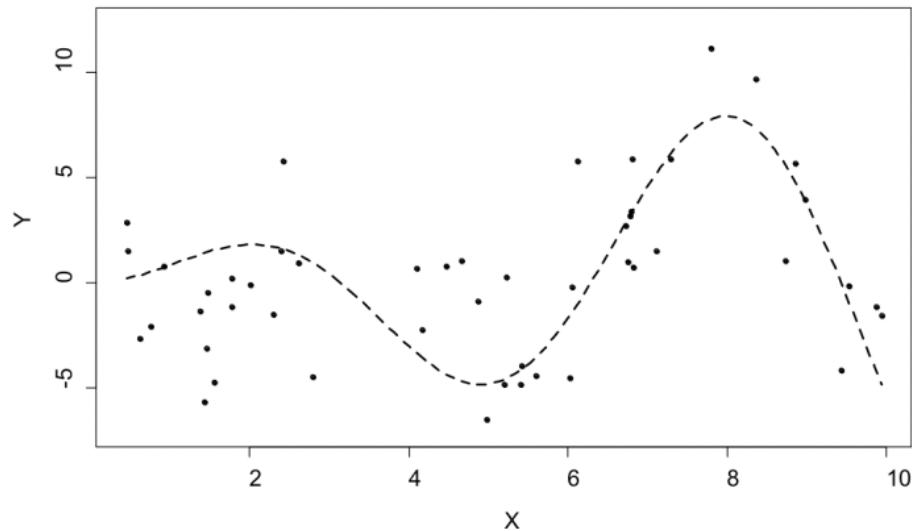
## Overfitting (non-parametric smoothing)



Model class  $\Theta$  can be **so complex** it can fit true model + noise

24

## Actual model ( $x \sin x$ )



The **right** model class  $\Theta$  will sacrifice some training error, for test error.

25

## Approach: Explicit model selection

- Try different classes of models. Example, try polynomial models of various degree  $d$  (linear, quadratic, cubic, ...)
- Use held out validation (cross validation) to select the model
  1. Split training data into  $D_{train}$  and  $D_{validate}$  sets
  2. For each degree  $d$  we have model  $f_d$ 
    1. Train  $f_d$  on  $D_{train}$
    2. Test  $f_d$  on  $D_{validate}$
  3. Pick degree  $\hat{d}$  that gives the best test score
  4. Re-train model  $f_{\hat{d}}$  using all data

26

## Approach: Regularisation

- Augment the problem:  

$$\hat{\theta} \in \operatorname{argmin}_{\theta \in \Theta} (L(data, \theta) + \lambda R(\theta))$$
- E.g., ridge regression  

$$\hat{w} \in \operatorname{argmin}_{w \in W} \|y - Xw\|_2^2 + \lambda \|w\|_2^2$$
- Note that regulariser  $R(\theta)$  does not depend on data
- Use held out validation/cross validation to choose  $\lambda$

27

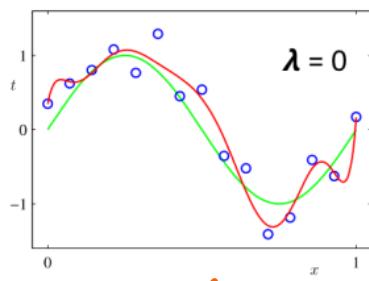
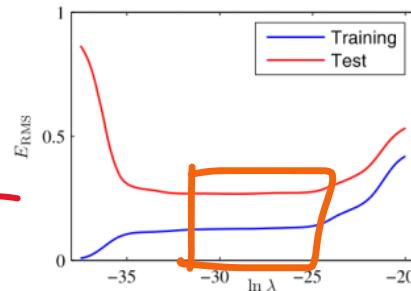
## Example: Polynomial regression

- 9<sup>th</sup>-order polynomial regression

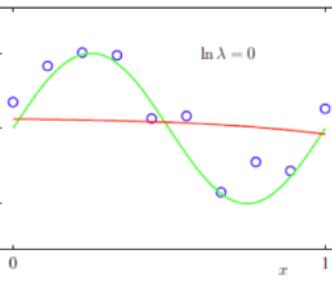
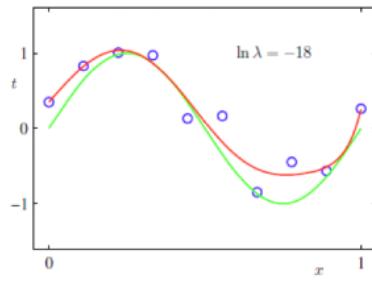
- \* model of form

$$\hat{f} = w_0 + w_1 x + \dots + w_9 x^9$$

- \* regularised with  $\lambda \|\mathbf{w}\|_2^2$  term



overfitting



underfitting

Figure 3.8: Overfitting vs underfitting

## Mini Summary

- Overfitting vs underfitting
- Effect of regularisation on nonlinear regression
  - \* Controls balance of over- vs underfitting
  - \* Controlled in this case by the penalty hyperparameter
- Next: Bias-variance view for regression

# Bias-variance trade-off

Train error, test error and  
model complexity in  
**supervised regression**

30

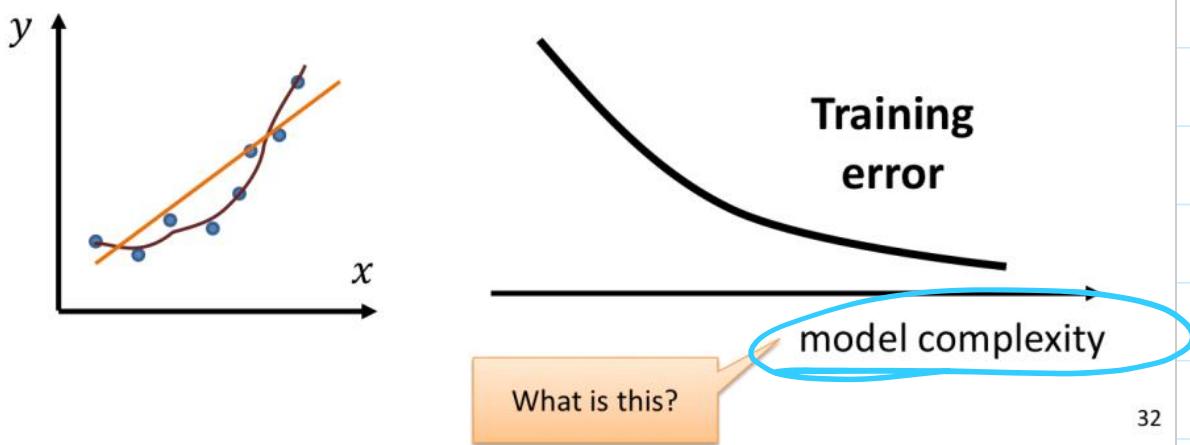
## Assessing generalisation

- Supervised learning: train the model on existing data, then make predictions on new data
- Training the model: ERM / minimisation of training error
- Generalisation capacity is captured by risk / test error
- Model complexity is a major factor that influences the ability of the model to generalise (**vague still**)
- In this section, our aim is to explore error in the context of supervised regression. One way to decompose it.

31

## Training error and model complexity

- More complex model → training error goes down
- Finite number of points → usually can reduce training error to 0 (is it always possible?)



32

## (Another) Bias-variance decomposition

- Squared loss for **supervised-regression** predictions

$$l(Y, \hat{f}(\mathbf{X}_0)) = (Y - \hat{f}(\mathbf{X}_0))^2$$

Classification  
later on

- Lemma: Bias-variance decomposition

$$\mathbb{E}[l(Y, \hat{f}(\mathbf{X}_0))] = (\mathbb{E}[Y] - \mathbb{E}[\hat{f}])^2 + \text{Var}[\hat{f}] + \text{Var}[Y]$$

Risk /  
test error  
for  $\mathbf{x}_0$

(bias)<sup>2</sup>

variance

irreducible  
error

\* Prediction randomness comes from randomness in test features AND training data

33

## Decomposition proof sketch

- Here  $(x)$  is omitted to de-clutter notation

$$\mathbb{E}[(Y - \hat{f})^2] = \mathbb{E}[Y^2 + \hat{f}^2 - 2Y\hat{f}]$$

$$= \mathbb{E}[Y^2] + \mathbb{E}[\hat{f}^2] - \mathbb{E}[2Y\hat{f}]$$

$$= Var[Y] + \mathbb{E}[Y]^2 + Var[\hat{f}] + (\mathbb{E}[\hat{f}]^2 - 2\mathbb{E}[Y]\mathbb{E}[\hat{f}])$$

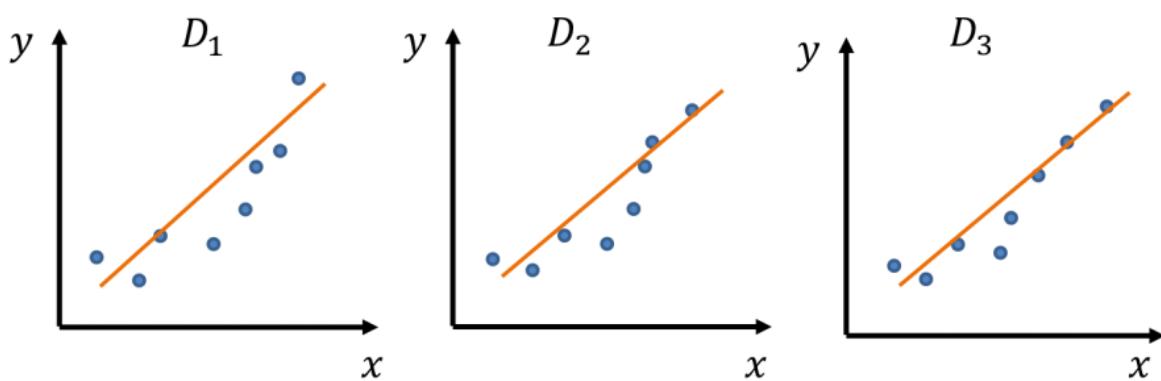
$$= Var[Y] + Var[\hat{f}] + (\mathbb{E}[Y]^2 - 2\mathbb{E}[Y]\mathbb{E}[\hat{f}] + \mathbb{E}[\hat{f}]^2)$$

$$= Var[Y] + Var[\hat{f}] + (\mathbb{E}[Y] - \mathbb{E}[\hat{f}])^2$$

$[\mathbb{E}(Y) - \mathbb{E}(\hat{f})]^2$

34

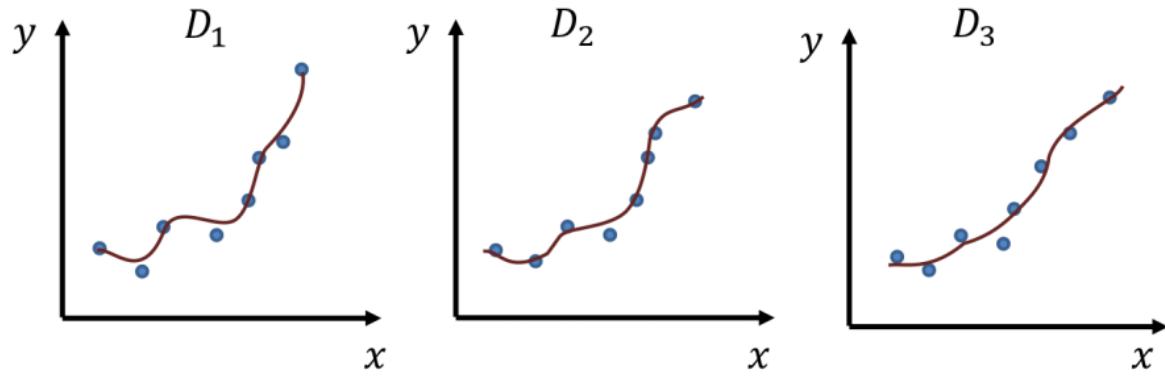
## Training data as a random variable



less change  
(low variance)

35

## Training data as a random variable

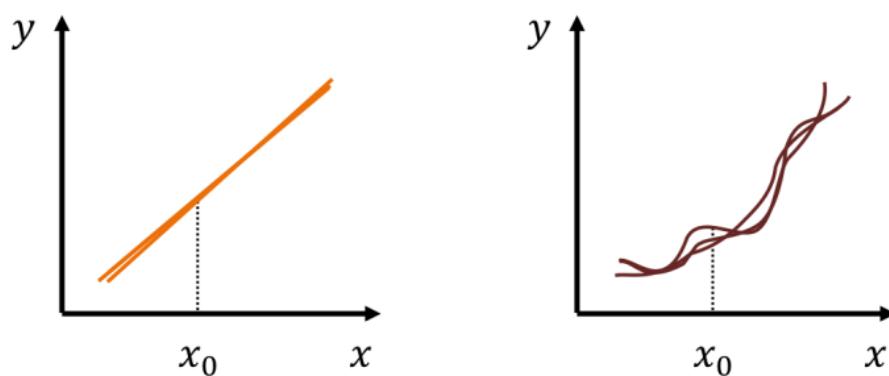


*more change  
(high variance)*

36

## Intuition: Model complexity and variance

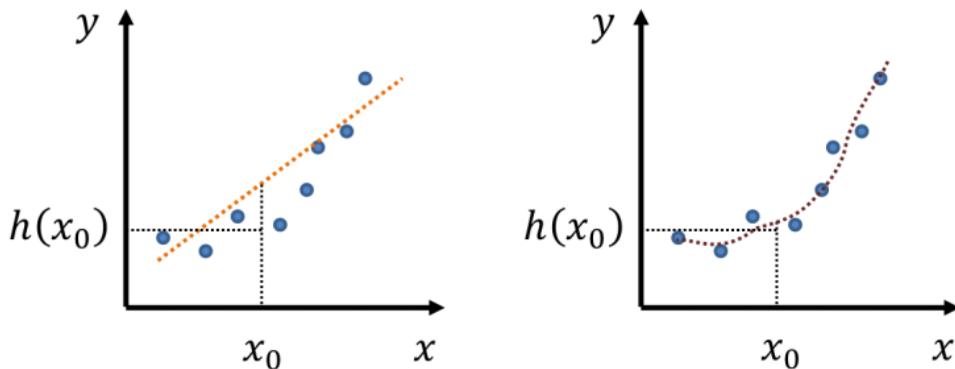
- simple model → low variance
- complex model → high variance



37

## Intuition: Model complexity and variance

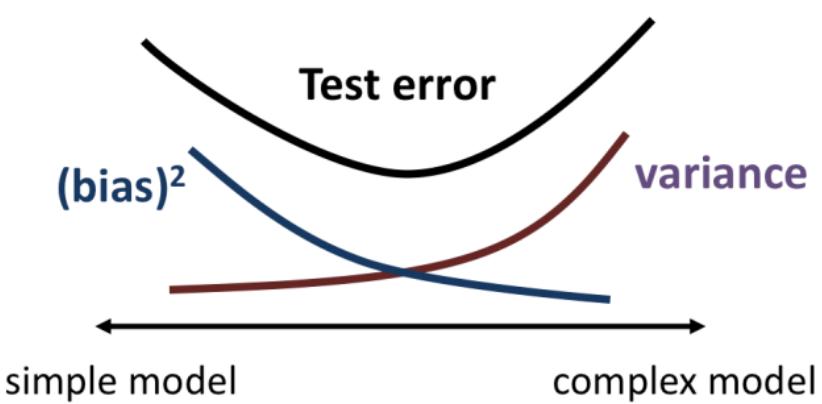
- simple model → high bias
- complex model → low bias



38

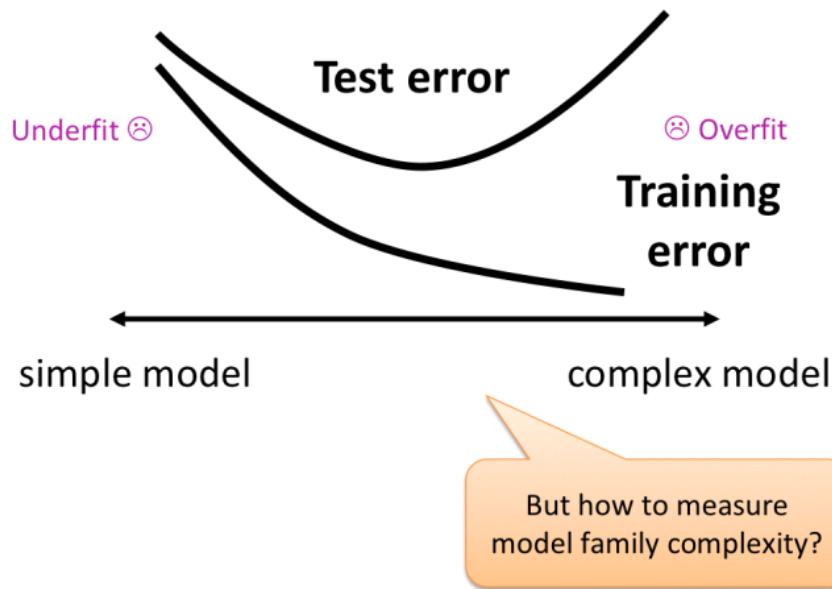
## Bias-variance trade-off

- simple model → high bias, low variance
- complex model → low bias, high variance



39

# Test error and training error



40

## Mini Summary

- Supervised regression: square-loss risk decomposes to bias, variance and irreducible terms
- This trade-off mirrors under/overfitting
- Controlled by “model complexity”
  - \* But we’ve been vague about what this means!?
- Next lectures: Bounding generalisation error in ML

41

## Lecture 06

2020年8月21日 星期五 09:23



06

# Lecture 6. PAC Learning Theory

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

COMP90051 Statistical Machine Learning

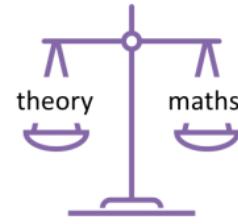
## This lecture

- Excess risk
  - \* Decomposition: Estimation vs approximation
  - \* Bayes risk – irreducible error
- Probably approximation correct learning
- Bounding generalisation error with high probability
  - \* Single model: Hoeffding's inequality
  - \* Finite model class: Also use the union bound
- Importance & limitations of uniform deviation bounds



## Generalisation and Model Complexity

- Theory we've seen so far (mostly statistics)
  - \* Asymptotic notions (consistency, efficiency)
  - \* Convergence could be really slow
  - \* Model complexity undefined
- Want: finite sample theory; convergence rates, trade-offs
- Want: define model complexity and relate it to test error
  - \* Test error can't be measured in real life, but it can be provably bounded!
  - \* Growth function, VC dimension
- Want: distribution-independent, learner-independent theory
  - \* A fundamental theory applicable *throughout ML*
  - \* Unlike bias-variance: distribution dependent, no model complexity,



3

## Probably Approximately Correct Learning

*The bedrock of machine learning theory in computer science.*

4

## Standard setup

- Supervised binary classification of  
\* data in  $\mathcal{X}$  into label set  $\mathcal{Y} = \{-1, 1\}$
- iid data  $\{(x_i, y_i)\}_{i=1}^m \sim D$  some fixed unknown distribution
- Single test example independent from same  $D$  when representing generalisation performance (risk)
- Learning from a class of function  $\mathcal{F}$  mapping (classifying)  $\mathcal{X}$  into  $\mathcal{Y}$
- What parts depend on the sample of data
  - \* Empirical risk  $\hat{R}[f]$  that averages loss over the sample
  - \*  $f_m \in \mathcal{F}$  the learned model (it could be same sample or different; theory is actually fully general here)

5

## Risk: The good, bad and ugly

*goal*  $\rightarrow$  *error in class  $\mathcal{F}$*

$$\underbrace{R[f_m] - R^*}_{\text{Excess risk}} = \underbrace{(R[f_m] - R[f^*])}_{\text{Estimation error}} + \underbrace{(R[f^*] - R^*)}_{\text{Approximation error}}$$

- **Good:** what we'd aim for in our class, with infinite data

\*  $R[f^*]$  true risk of best in class  $f^* \in \operatorname{argmin}_{f \in \mathcal{F}} R[f]$

- **Bad:** we get what we get and don't get upset

\*  $R[f_m]$  true risk of learned  $f_m \in \arg \min_{f \in \mathcal{F}} \hat{R}[f] + C \|f\|^2$  (e.g.)

- **Ugly:** we usually cannot even hope for perfection!

\*  $R^* \in \inf_f R[f]$  called the **Bayes risk**; noisy labels makes large

*best possible  
risk that ever  
get*

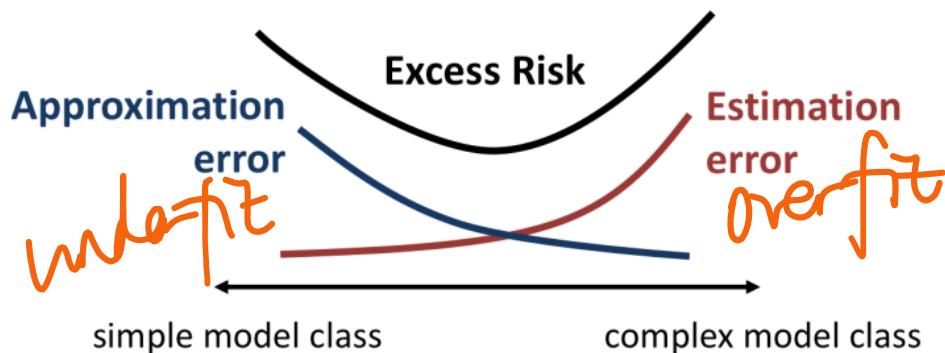
*min true risk in  $\mathcal{F}$*   
*regularizer*  
*min empirical risk in  $\mathcal{F}$*

## A familiar trade-off: More intuition

- simple family  $\rightarrow$  may underfit due to approximation error
- complex family  $\rightarrow$  may overfit due to estimation error

## A familiar trade-off: More intuition

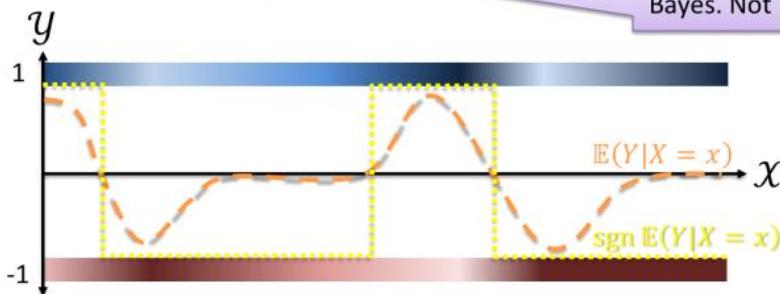
- simple family → may underfit due to approximation error
- complex family → may overfit due to estimation error



7

## About Bayes risk

Once again, named after Bayes. Not Bayesian ML.



- Bayes risk  $R^* \in \inf_f R[f]$ 
  - Best risk possible, ever; but can be large
  - Depends on distribution and loss function
- Bayes classifier achieves Bayes risk
  - $f_{\text{Bayes}}(x) = \text{sgn } \mathbb{E}(Y|X=x)$

8

## Let's focus on $R[f_m]$



Leslie Valiant  
CCA2.0 Renate Schmid

- Since we don't know data distribution, we need to bound generalisation to be small
  - \* Bound by test error  $\hat{R}[f_m] = \frac{1}{m} \sum_{i=1}^m f(X_i, Y_i)$
  - \* Abusing notation:  $f(X_i, Y_i) = l(Y_i, f(X_i))$
  - $$R[f_m] \leq \hat{R}[f_m] + \varepsilon(m, \mathcal{F})$$
- Unlucky training sets, no always-guarantees possible!
- With probability  $\geq 1 - \delta$ :  $R[f_m] \leq \hat{R}[f_m] + \varepsilon(m, \mathcal{F}, \delta)$
- Called Probably Approximately Correct (PAC) learning
  - \*  $\mathcal{F}$  called PAC learnable if  $m = O(\text{poly}(1/\varepsilon, 1/\delta))$  to learn  $f_m$  for any  $\varepsilon, \delta$
  - \* Won Leslie Valiant (Harvard) the 2010 Turing Award
- Later: Why this bounds estimation error.

Don't require exponential growth in training size  $m$

9

## Mini Summary

- Excess risk as the goal of ML
- Decomposition into approximation, estimation errors
- Probably Approximately Correct (PAC) learning
  - \* Like asymptotic theory in stats, but for finite sample size
  - \* Worst-case on distributions: We don't want to assume something unrealistic about where the data comes from
  - \* Worst-case on models: We don't want a theory that applies to narrow set of learners, but to ML in general
  - \* We want it to produce a useful measure of model complexity

Next: First step to PAC theory – bounding single model risk

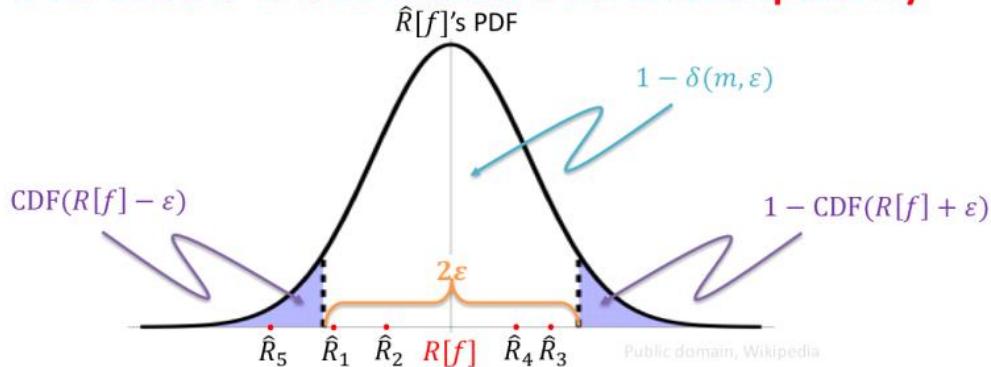
10

# Bounding true risk of one function

*One step at a time*

11

## We need a concentration inequality



- $\hat{R}[f]$  is an unbiased estimate of  $R[f]$  for any fixed  $f$  (*why?*)
- That means on average  $\hat{R}[f]$  lands on  $R[f]$
- What's the likelihood  $1 - \delta$  that  $\hat{R}[f]$  lands within  $\epsilon$  of  $R[f]$ ? Or more precisely, what  $1 - \delta(m, \epsilon)$  achieves a given  $\epsilon > 0$ ?
- Intuition: Just bounding CDF of  $\hat{R}[f]$ , independent of distribution!!

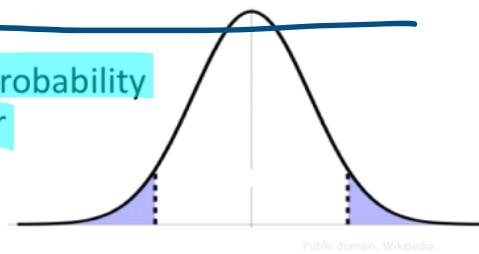
12

## Hoeffding's inequality

- Many such concentration inequalities; a simplest one...
- Theorem:** Let  $Z_1, \dots, Z_m, Z$  be iid random variables and  $h(z) \in [a, b]$  be a bounded function. For all  $\varepsilon > 0$

$$\left\{ \begin{array}{l} \Pr\left(\left|\mathbb{E}[h(Z)] - \frac{1}{m} \sum_{i=1}^m h(Z_i)\right| \geq \varepsilon\right) \leq 2 \exp\left(-\frac{2m\varepsilon^2}{(b-a)^2}\right) \\ \Pr\left(\mathbb{E}[h(Z)] - \frac{1}{m} \sum_{i=1}^m h(Z_i) \geq \varepsilon\right) \leq \exp\left(-\frac{2m\varepsilon^2}{(b-a)^2}\right) \end{array} \right.$$

- Two-sided case in words: The probability that the empirical average is far from the expectation is small.



13

## Common probability ‘tricks’

- Inversion:
  - For any event  $A$ ,  $\Pr(\bar{A}) = 1 - \Pr(A)$
  - Application:  $\Pr(X > \varepsilon) \leq \delta$  implies  $\Pr(X \leq \varepsilon) \geq 1 - \delta$
- Solving for, in high-probability bounds:
  - For given  $\varepsilon$  with  $\delta(\varepsilon)$  function  $\varepsilon: \Pr(X > \varepsilon) \leq \delta(\varepsilon)$
  - Given  $\delta'$  can write  $\varepsilon = \delta^{-1}(\delta')$ :  $\Pr(X > \delta^{-1}(\delta')) \leq \delta'$
  - Let's you specify either parameter
  - Sometimes sample size  $m$  a variable we can solve for too



14

## Et voila: A bound on true risk!

Result!  $R[f] \leq \hat{R}[f] + \sqrt{\frac{\log(1/\delta)}{2m}}$  with high probability (w.h.p.)  $\geq 1 - \delta$

*Proof*

- Take the  $Z_i$  as labelled examples  $(X_i, Y_i)$
- Take  $h(X, Y) = l(Y, f(X))$  zero-one loss for some fixed  $f \in \mathcal{F}$   
then  $h(X, Y) \in [0, 1]$
- Apply one-sided Hoeffding:  $\Pr(R[f] - \hat{R}[f] \geq \varepsilon) \leq \exp(-2m\varepsilon^2)$

$$\begin{aligned} \Pr(R - \hat{R} \leq \varepsilon) &\geq 1 - \delta \\ \delta &= \exp(-2m\varepsilon^2) \\ \Rightarrow \varepsilon &= \sqrt{\frac{\log(1/\delta)}{2m}} \end{aligned}$$

15

## Mini Summary

- Goal: Bound true risk of a classifier based on its empirical risk plus “stuff”
- Caveat: Bound is “with high probability” since we could be unlucky with the data
- Approach: Hoeffding’s inequality which bounds how far a mean is likely to be from an expectation

Next: PAC learning as uniform deviation bounds

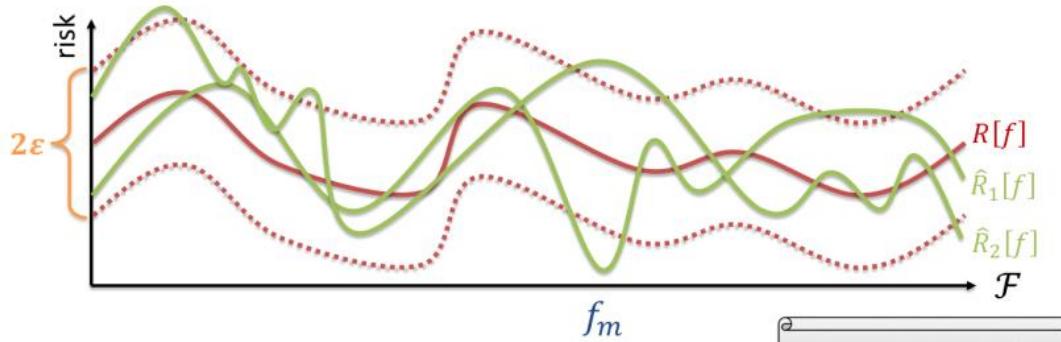
16

# Uniform deviation bounds

*Why we need our bound to **simultaneously** (or uniformly) hold over a family of functions.*

17

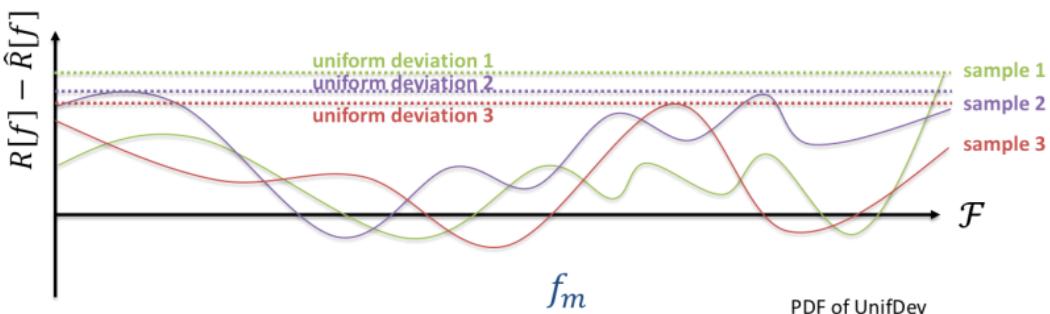
## Our bound doesn't hold for $f = f_m$



- Result says there's set  $S$  of good samples for which  $R[f] \leq \hat{R}[f] + \sqrt{\frac{\log(1/\delta)}{2m}}$  and  $\Pr(\mathbf{Z} \in S) \geq 1 - \delta$
- But for different functions  $f_1, f_2, \dots$  we might get very different sets  $S_1, S_2, \dots$
- $S$  observed may be bad for  $f_m$ . Learning minimises  $\hat{R}[f_m]$ , exacerbating this

18

# Uniform deviation bounds



- We could analyse risks of  $f_m$  from specific learner
  - \* But repeating for new learners? How to compare learners?
  - \* Note there are ways to do this, and data-dependently
- Bound uniform deviations across whole class  $\mathcal{F}$  –
 
$$R[f_m] - \hat{R}[f_m] \leq \sup_{f \in \mathcal{F}} (R[f] - \hat{R}[f]) \leq ?$$
  - \* Worst deviation over an entire class bounds learned risk!
  - \* Convenient, but could be much worse than the actual gap for  $f_m$

19

# Relation to estimation error?

- Recall **estimation error?** Learning part of excess risk!

$$R[f_m] - R^* = (R[f_m] - R[f^*]) - (R[f^*] - R^*)$$

**Theorem:** ERM's estimation error is at most twice the uniform divergence



- \* Proof: a bunch of algebra!

$$\begin{aligned} R[f_m] &\leq (\hat{R}[f^*] - \hat{R}[f_m]) + R[f_m] - R[f^*] + R[f^*] \\ &= \hat{R}[f^*] - R[f^*] + R[f_m] - \hat{R}[f_m] + R[f^*] \\ &\leq |R[f^*] - \hat{R}[f^*]| + |R[f_m] - \hat{R}[f_m]| + R[f^*] \\ &\leq 2 \sup_{f \in \mathcal{F}} |R[f] - \hat{R}[f]| + R[f^*] \end{aligned}$$

best  $f$

$f_m \in \mathcal{F}$

20

## Mini Summary

- Why Hoeffding doesn't cover a model  $f_m$  learned from data, only a fixed data-independent  $f$

## Mini Summary

- Why Hoeffding doesn't cover a model  $f_m$  learned from data, only a fixed data-independent  $f$
- Uniform deviation idea: Cover the worst case deviation between risk and empirical risk, across  $\mathcal{F}$
- Advantages: works for any learner, data distribution
- Connection back to bounding estimation error

Next: Next step for PAC learning – finite classes

21

## Error bound for finite function classes

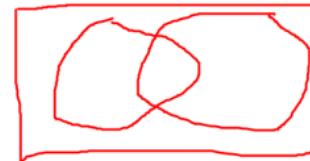
*Our first uniform deviation bound*

22

## The Union Bound

- If each model  $f$  having large risk deviation is a “bad event”, we need a tool to bound the probability that any bad event happens. I.e. the union of bad events!
- Union bound:** for a sequence of events  $A_1, A_2 \dots$

$$\Pr\left(\bigcup_i A_i\right) \leq \sum_i \Pr(A_i)$$



Proof:

Define  $B_i = A_i \setminus \bigcup_{j=1}^{i-1} A_j$  with  $B_1 = A_1$ .

- We know:  $\bigcup_i B_i = \bigcup_i A_i$  (could prove by induction)
- The  $B_i$  are disjoint (empty intersections)
- We know:  $B_i \subseteq A_i$  so  $\Pr(B_i) \leq \Pr(A_i)$  by monotonicity

$$\Pr\left(\bigcup_i A_i\right) = \Pr\left(\bigcup_i B_i\right) = \sum_i \Pr(B_i) \leq \sum_i \Pr(A_i)$$

23

## Bound for finite classes $\mathcal{F}$

- A uniform deviation bound over *any* finite class or distribution

**Theorem:** Consider any  $\delta > 0$  and **finite** class  $\mathcal{F}$ . Then w.h.p

at least  $1 - \delta$ : For all  $f \in \mathcal{F}$ ,  $R[f] \leq \hat{R}[f] + \sqrt{\frac{\log |\mathcal{F}| + \log(1/\delta)}{2m}}$

Proof:

- If each model  $f$  having large risk deviation is a “bad event”, we bound the probability that any bad event happens.
- $\Pr(\exists f \in \mathcal{F}, R[f] - \hat{R}[f] \geq \varepsilon) \leq \sum_{f \in \mathcal{F}} \Pr(R[f] - \hat{R}[f] \geq \varepsilon)$
- $\leq |\mathcal{F}| \exp(-2m\varepsilon^2)$  by the union bound
- Followed by inversion, setting  $\delta = |\mathcal{F}| \exp(-2m\varepsilon^2)$

finite

24

## Discussion

- Hoeffding's inequality only uses boundedness of the loss, not the variance of the loss random variables
  - \* Fancier concentration inequalities leverage variance
- Uniform deviation is worst-case, ERM on a very large over-parametrised  $\mathcal{F}$  may approach the worst-case, but learners generally may not
  - \* Custom analysis, data-dependent bounds, PAC-Bayes, etc.
- Dependent data?
  - \* Martingale theory
- Union bound is in general loose, as bad as if all the bad events were independent (not necessarily the case even though underlying data modelled as independent); and finite  $\mathcal{F}$ 
  - \* VC theory coming up next!

25

## Mini Summary

- More on uniform deviation bounds
- The union bound (generic tool in probability theory)
- Finite classes: Bounding uniform deviation with union+Hoeffding

Next time: PAC learning with infinite function classes!

26

## Lecture 07

2020年8月25日 星期二 12:00



07

# Lecture 7. VC Theory

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

COMP90051 Statistical Machine Learning

## This lecture

- PAC learning bounds:
  - \* Countably infinite case works as we've done so far
  - \* General infinite case? Needs new ideas!
- Growth functions for the general PAC case
  - \* Considering patterns of labels possible on a data set
  - \* Gives good PAC bounds provided possible patterns don't grow too fast in the data set size
- Vapnik-Chervonenkis (VC) dimension
  - \* Max number of points that can be labelled in all ways
  - \* Beyond this point, growth function is polynomial in data set size
  - \* Leads to famous, general PAC bound from VC theory
- Optional proofs at end (just for fun)

## Countably infinite $\mathcal{F}$ ?

- Hoeffding gave us for a single  $f \in \mathcal{F}$

$$\Pr\left(R[f] - \hat{R}[f] \geq \sqrt{\frac{\log(\frac{1}{\delta(f)})}{2m}}\right) \leq \delta(f)$$

...where we're free to choose (varying)  $\delta(f)$  in  $[0,1]$ .



Josh Staiger (CC-BY-NC-ND)

- Union bound "works" (sort of) for this case

$$\Pr\left(\exists f \in \mathcal{F}, R[f] - \hat{R}[f] \geq \sqrt{\frac{\log(\frac{1}{\delta(f)})}{2m}}\right) \leq \sum_{f \in \mathcal{F}} \delta(f)$$

Try this  
for finite  $\mathcal{F}$  with  
uniform  $p(f)$

- Choose confidences to sum to constant  $\delta$ , then this works
  - E.g.  $\delta(f) = \delta \cdot p(f)$  where  $1 = \sum_{f \in \mathcal{F}} p(f)$

- By inversion: w.h.p  $1 - \delta$ , for all  $f$ ,  $R[f] \leq \hat{R}[f] + \sqrt{\frac{\log(\frac{1}{p(f)}) + \log(\frac{1}{\delta})}{2m}}$

3

## Ok fine, but general case?

- Much of ML has continuous parameters
  - Countably infinite covers only discrete parameters 😞
- Our argument fails! 😞 😞
  - $p(f)$  becomes a density
  - Its zero for all  $f$ . No divide by zero!
  - Need a new argument!
- Idea introduced by VC theory: intuition
  - Don't focus on whole class  $\mathcal{F}$  as if each  $f$  is different
  - Focus on differences over sample  $Z_1, \dots, Z_m$

$$\sqrt{\frac{\log(\frac{1}{p(f)}) + \log(\frac{1}{\delta})}{2m}}$$

$f(a) - \bar{f}(a) = 0$

$2^m$

4

## Mini Summary

- Can eek out PAC bounds on countably infinite families using Hoeffding bound + union bound
- No good for general (uncountably infinite) cases
- Need another fundamentally new idea

Next: Organising analysis around patterns of labels possible on a data set, to avoid worst-case bad events

5

## Growth Function

*Focusing on the size of model families  
on data samples*

6

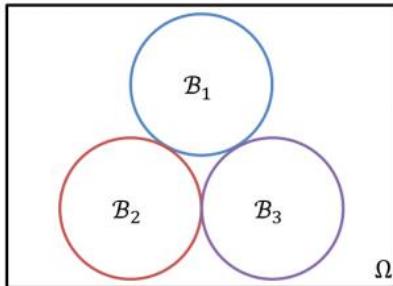
## Bad events: Unreasonably worst case?

- Bad event  $\mathcal{B}_i$  for model  $f_i$

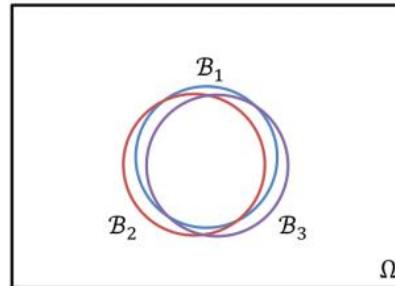
$$R[f_i] - \hat{R}[f_i] \geq \varepsilon \text{ with probability } \leq 2 \exp(-2m\varepsilon^2)$$

- Union bound: bad events don't overlap!?

$$\Pr(\mathcal{B}_1 \text{ or } \dots \text{ or } \mathcal{B}_{|\mathcal{F}|}) \leq \Pr(\mathcal{B}_1) + \dots + \Pr(\mathcal{B}_{|\mathcal{F}|}) \leq 2|\mathcal{F}| \exp(-2m\varepsilon^2)$$



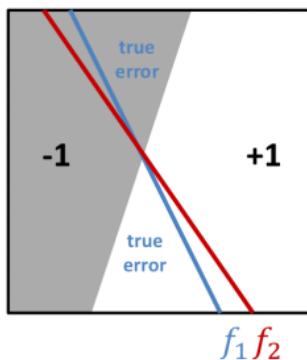
Tight bound: No overlaps



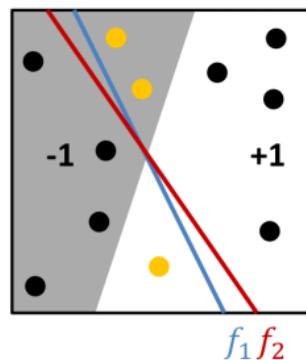
Loose bound: Overlaps

7

## How do overlaps arise?



Whole of population



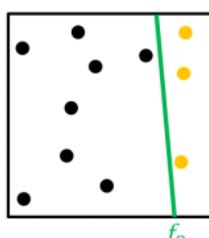
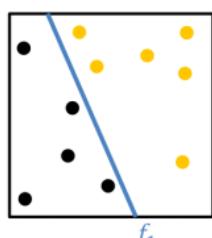
On a sample

Significantly overlapping events  $\mathcal{B}_1$  and  $\mathcal{B}_2$ 

$\mathcal{B}_1$  happens (error)  
 $\downarrow$  very likely  
 $\mathcal{B}_2$  happens (error)

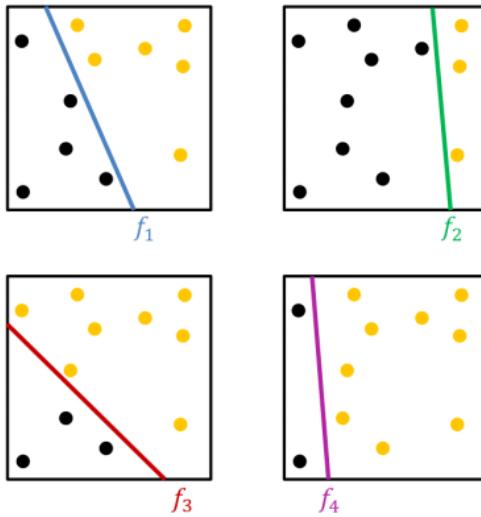
8

## How do overlaps arise?

VC theory focuses on the pattern of labels any  $f \in \mathcal{F}$  could make

## How do overlaps arise?

VC theory focuses on the pattern of labels any  $f \in \mathcal{F}$  could make



9

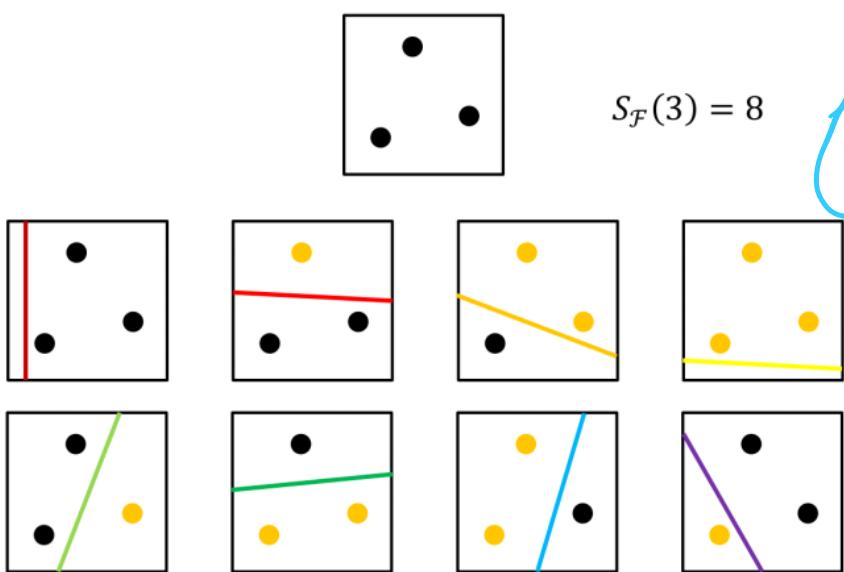
## Dichotomies and Growth Function

- **Definition:** Given sample  $x_1, \dots, x_m$  and family  $\mathcal{F}$ , a **dichotomy** is a  $(f(x_1), \dots, f(x_m)) \in \{-1, +1\}^m$  for some  $f \in \mathcal{F}$ .
- Unique dichotomies  $\mathcal{F}(\mathbf{x}) = \{(f(x_1), \dots, f(x_m)) : f \in \mathcal{F}\}$ , patterns of labels possible with the family
- Even when  $\mathcal{F}$  infinite,  $|\mathcal{F}(\mathbf{x})| \leq 2^m$  (why?)  $\begin{cases} + \\ - \end{cases}$
- And also (relevant for  $\mathcal{F}$  finite, tiny),  $|\mathcal{F}(\mathbf{x})| \leq |\mathcal{F}|$  (why?)
- Intuition:  $|\mathcal{F}(\mathbf{x})|$  might replace  $|\mathcal{F}|$  in union bound? How remove  $\mathbf{x}$ ?
- **Definition:** The **growth function**  $S_{\mathcal{F}}(m) = \sup_{\mathbf{x} \in \mathcal{D}^m} |\mathcal{F}(\mathbf{x})|$  is the max number of label patterns achievable by  $\mathcal{F}$  for any  $m$  sample.

as string (not them)

10

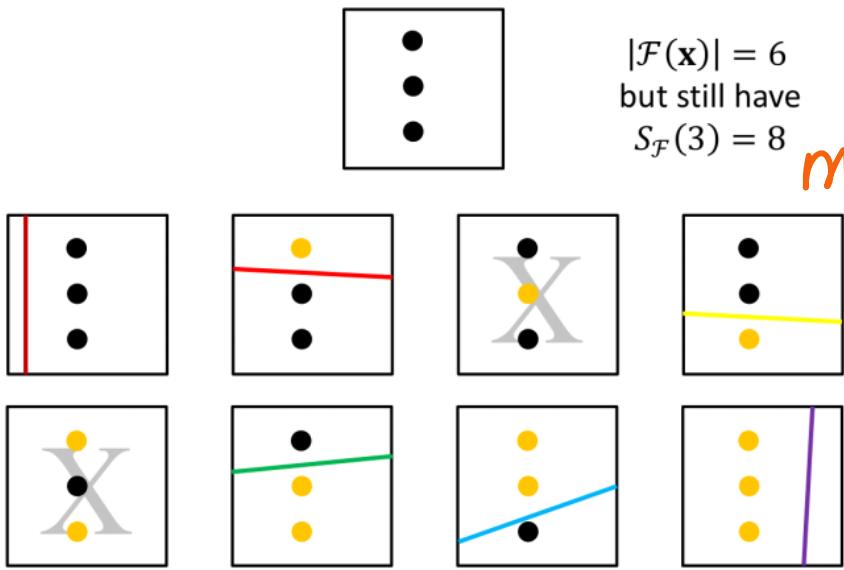
$S_{\mathcal{F}}(3)$  for  $\mathcal{F}$  linear classifiers in 2D?



Sup  
2<sup>3</sup>  
upper bound

11

$S_{\mathcal{F}}(3)$  for  $\mathcal{F}$  linear classifiers in 2D?

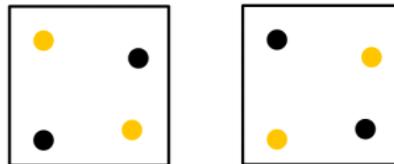


maximum  
number

12

## $S_{\mathcal{F}}(4)$ for $\mathcal{F}$ linear classifiers in 2D?

- What about  $m = 4$  points?
- Can never produce the criss-cross (XOR) dichotomy



- In fact  $S_{\mathcal{F}}(4) = 14 < 2^4$
- Guess/exercise: What about general  $m$  and dimension?

13

## PAC Bound with Growth Function

- Theorem: Consider any  $\delta > 0$  and any class  $\mathcal{F}$ . Then w.h.p. at least  $1 - \delta$ : For all  $f \in \mathcal{F}$

$$R[f] \leq \hat{R}[f] + 2 \sqrt{2 \frac{\log S_{\mathcal{F}}(2m) + \log(4/\delta)}{m}}$$

- Proof: out of scope (“only” 2-3pgs), optional reading.
- Compare to PAC bounds so far
  - \* A few negligible extra constants (the 2s, the 4)
  - \*  $\mathcal{F}$  has become  $\log S_{\mathcal{F}}(2m)$
  - \*  $S_{\mathcal{F}}(m) \leq |\mathcal{F}|$ , not “worse” than union bound for finite  $\mathcal{F}$
  - \*  $S_{\mathcal{F}}(m) \leq 2^m$ , very bad for big family with exponential growth function gets  $R[f] \leq \hat{R}[f] + \text{Big Constant}$ . Even  $R[f] \leq \hat{R}[f] + 1$  meaningless!!

14

## Mini Summary

- The previous PAC bound approach that organises bad events by model and applies uniform bound is only tight if bad events are disjoint
- In reality some models generate overlapping bad events
- Better to organise families by possible patterns of labels on a data set: the dichotomies of the family
- Counting possible dichotomies gives the growth function
- PAC bound with growth function potentially tackles general (uncountably infinite) families provided growth function is sub-exponential in data size

Next: VC dimension for a computable bound on growth functions, with the polynomial behaviour we need! Gives our final, general, PAC bound

15

## The VC dimension

*Computable, bounds growth function*

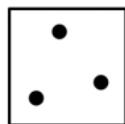
16

## Vapnik-Chervonenkis dimension

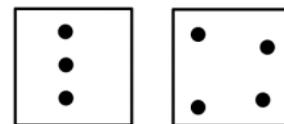
- **Definition:** The VC dimension  $\text{VC}(\mathcal{F})$  of a family  $\mathcal{F}$  is the largest  $m$  such that  $S_{\mathcal{F}}(m) = 2^m$ .

- \* Points  $\mathbf{x} = (x_1, \dots, x_m)$  are shattered by  $\mathcal{F}$  if  $|\mathcal{F}(\mathbf{x})| = 2^m$
- \* So  $\text{VC}(\mathcal{F})$  is the size of the largest set shattered by  $\mathcal{F}$

- Example: linear classifiers in  $\mathbb{R}^2$ ,  $\text{VC}(\mathcal{F}) = 3$



Shattered



Not shattered

- Guess: VC-dim of linear classifiers in  $\mathbb{R}^d$ ? d+1

17

## Example: $\text{VC}(\mathcal{F})$ from $\mathcal{F}(\mathbf{x})$ on whole domain?

$x_1$	$x_2$	$x_3$	$x_4$
0	0	0	0
0	1	1	0
1	0	0	1
1	1	0	1
0	1	0	0
1	0	1	0
1	1	1	1
0	0	1	1
0	1	0	1
1	1	1	0

Note we're using labels {0,1} instead of {-1,+1}. Why OK?

- Columns are all points in domain
- Each row is a dichotomy on entire input domain
- Obtain dichotomies on a subset of points  $\mathbf{x}' \subseteq \{x_1, \dots, x_4\}$  by: drop columns, drop dupe rows
- $\mathcal{F}$  shatters  $\mathbf{x}'$  if number of rows is  $2^{|\mathbf{x}'|}$

$x_1$	$x_2$	$x_4$
0	0	0
0	1	0
1	0	1
1	1	1
0	+	0
1	0	0
+	+	+
0	0	1
0	1	1
1	1	0

This example:

- Dropping column 3 leaves 8 rows behind:  $\mathcal{F}$  shatters  $\{x_1, x_2, x_4\}$
- Original table has  $< 2^4$  rows:  $\mathcal{F}$  doesn't shatter more than 3

VC( $\mathcal{F}$ ) = 3

18

## Sauer-Shelah Lemma

- Lemma (Sauer-Shelah):** Consider any  $\mathcal{F}$  with finite  $\text{VC}(\mathcal{F}) = k$ , any sample size  $m$ . Then  $S_{\mathcal{F}}(m) \leq \sum_{i=0}^k \binom{m}{i}$ .

- From basic facts of Binomial coefficients

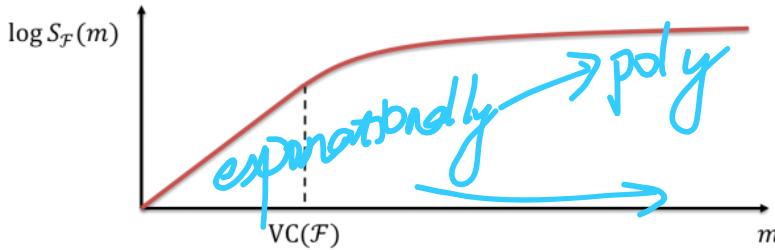
- \* Bound is  $O(m^k)$ : finite VC  $\Rightarrow$  eventually polynomial growth!
- \* For  $m \geq k$ , it is bounded by  $\left(\frac{em}{k}\right)^k$

- Theorem (VC bound):** Consider any  $\delta > 0$  and any VC- $k$  class  $\mathcal{F}$ . Then w.h.p. at least  $1 - \delta$ : For all  $f \in \mathcal{F}$

$$R[f] \leq \hat{R}[f] + 2 \sqrt{2 \frac{k \log \frac{2em}{k} + \log \frac{4}{\delta}}{m}}$$

19

## VC bound big picture



- (Uniform) difference between  $R[f], \hat{R}[f]$  is  $O\left(\sqrt{\frac{k \log m}{m}}\right)$  down from  $\infty$
- Limiting complexity of  $\mathcal{F}$  leads to better generalisation
- VC dim, growth function measure "effective" size of  $\mathcal{F}$
- VC dim doesn't count functions, but uses geometry of family: projections of family members onto possible samples
- Example: linear "gap-tolerant" classifiers (like SVMs) with "margin"  $\Delta$  have  $\text{VC} = O(1/\Delta^2)$ . Maximising "margin" reduces VC-dimension.

20

## Mini Summary

- VC-dim is the largest set size shattered by a family
  - \* It is  $d + 1$  for linear classifiers in  $\mathbb{R}^d$
  - \* Can calculate it on entire-domain dichotomies of a family by dropping columns and counting unique rows
- Sauer-Shelah: The growth function grows only polynomially in the set size beyond the VC-dim
- As a result, VC PAC bounds uniform risk and empirical risk deviation by  $O(\sqrt{VC(\mathcal{F}) \log m}/m)$

Next: Two selected proofs. Optional but beautiful.

21

## Two Selected Proofs

*Green slides: Not examinable.*

*Food for thought. Soul food.*

22

$$\text{VC}(\mathcal{F}) = d + 1$$

### Linear classifiers in $d$ -dim: $VC(\mathcal{F}) \geq d + 1$

- Goal: construct  $m = d + 1$  specific points in  $\mathbb{R}^d$  that are shattered by the linear classifier family

- Data in rows of  $\mathbf{X} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & 1 \end{pmatrix}$  is invertible!

- Data in rows of  $\mathbf{X} = \begin{pmatrix} 1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & 1 \end{pmatrix}$  is invertible!
- Any dichotomy  $y \in \{-1, 1\}^{d+1}$ , need  $\mathbf{w}$  with  $\text{sign}(\mathbf{X}\mathbf{w}) = \mathbf{y}$
- $\mathbf{w} = \mathbf{X}^{-1}\mathbf{y}$  works!!  $\text{sign}(\mathbf{X}\mathbf{w}) = \text{sign}(\mathbf{X}\mathbf{X}^{-1}\mathbf{y}) = \text{sign}(\mathbf{y}) = \mathbf{y}$
- We've shown that  $\mathcal{F}$  can shatter  $d + 1$  points:  $\text{VC}(\mathcal{F}) \geq d + 1$

I

23

COMP90051 Statistical Machine Learning

## Linear classifiers in $d$ -dim: $\text{VC}(\mathcal{F}) \leq d + 1$

- Goal: cannot shatter any set of  $d + 2$  points
- Any  $\mathbf{x}_1, \dots, \mathbf{x}_{d+2}$ , have more pts than dims: linear dependent  

$$\mathbf{x}_j = \sum_{i \neq j} a_i \mathbf{x}_i , \quad \text{for some } j, \text{ where not all } a_i \text{'s are zero}$$
- Possible dichotomy  $\mathbf{y}$ ?  $y_i = \begin{cases} \text{sign}(a_i), & \text{if } i \neq j \\ -1, & \text{if } i = j \end{cases}$ 
  - \* Suppose  $\mathbf{w}$  generated  $i \neq j$ :  $\text{sign}(a_i) = \text{sign}(\mathbf{w}' \mathbf{x}_i)$  so  $a_i \mathbf{w}' \mathbf{x}_i > 0$
  - \* Can  $\mathbf{w}$  generate  $i = j$ ??
  - \*  $\mathbf{w}' \mathbf{x}_j = \mathbf{w}' \sum_{i \neq j} a_i \mathbf{x}_i = \sum_{i \neq j} a_i \mathbf{w}' \mathbf{x}_i > 0$  so  $\text{sign}(\mathbf{w}' \mathbf{x}_j) \neq y_i$
- We've shown  $\text{VC}(\mathcal{F}) < d + 2$ , in other words  $\text{VC}(\mathcal{F}) = d + 1$

24

COMP90051 Statistical Machine Learning

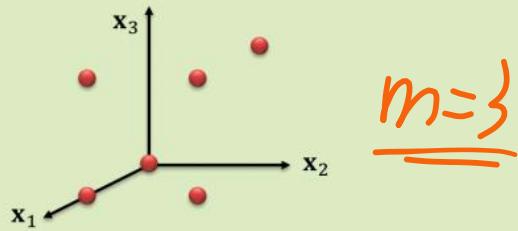
## Proof of Sauer-Shelah Lemma (by Haussler '95)

- To show that growth function  $S_{\mathcal{F}}(m) \leq \sum_{i=0}^k \binom{m}{i}$  we prove the bound for any dichotomies  $|\mathcal{F}(x_1, \dots, x_m)|$  since  $|\mathcal{F}(x_1, \dots, x_m)| \leq S_{\mathcal{F}}(m)$
- Write  $\mathbf{Y} = \mathcal{F}(x_1, \dots, x_m) \subseteq \{0, 1\}^m$ , where  $-1 \rightarrow 0$ .
- Definition: Consider any column  $1 \leq i \leq m$  and dichotomy  $\mathbf{y} \in \mathbf{Y}$ . The shift operator  $H_i(\mathbf{y}; \mathbf{Y})$  returns  $\mathbf{y}'$  if there exists some  $\mathbf{y}' \in \mathbf{Y}$  differing to  $\mathbf{y}$  only in the  $i^{\text{th}}$  coordinate; otherwise it returns  $\mathbf{y}$  with  $y_i = 0$ . Define  $H_i(\mathbf{Y}) = \{H_i(\mathbf{y}; \mathbf{Y}) : \mathbf{y} \in \mathbf{Y}\}$  the shifting all dichotomies.
  - \* Intuition: Shifting along a column drops a +1 to 0 in that column so long as no other row would become duplicated.
- Definition: A set of dichotomies  $\mathbf{V} \subseteq \{0, 1\}^m$  is called closed below if for all  $1 \leq i \leq m$ , shifting does nothing  $H_i(\mathbf{V}) = \mathbf{V}$ .
  - \* Intuition: Every  $\mathbf{v} \in \mathbf{V}$  has, for every  $1 \leq i \leq m$  for which  $v_i = 1$ , some  $\mathbf{u} \in \mathbf{V}$  the same as  $\mathbf{v}$  except with  $u_i = 1$ .

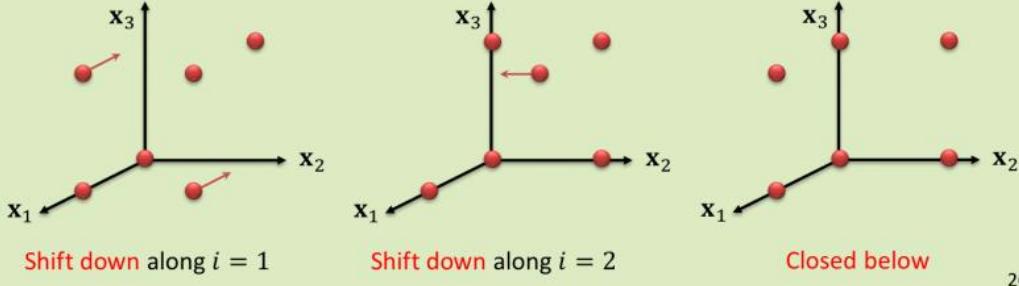
25

## Proof of Sauer-Shelah Lemma (by Haussler '95)

$x_1$	$x_2$	$x_3$
0	0	0
0	1	1
1	0	0
1	1	0
1	0	1
1	1	1



- Example set of 6 unique dichotomies on  $m = 3$  pts with  $\text{VC}=2$



26

## Proof of Sauer-Shelah Lemma (by Haussler '95)

- Goal: show that (1) shifting almost maintains VC dimension and cardinality all the way to a closed-below end, (2) closed-below sets have the desired Sauer-Shelah bound
- Shifting property 1:  $|H_i(\mathbf{Y})| = |\mathbf{Y}|$  for any  $\mathbf{Y}$ .
  - \* Proof: no two dichotomies in  $\mathbf{Y}$  shift to the same dichotomy
- Shifting property 2:  $\text{VC}(H_i(\mathbf{Y})) \leq \text{VC}(\mathbf{Y})$  for any  $i, \mathbf{Y}$ .
  - \* Proof sketch: If  $H_i(\mathbf{Y})$  shatters a subset of points, then so too does  $\mathbf{Y}$
- Shifting property 3: if  $\mathbf{Y}$  is closed below, then all dichotomies  $\mathbf{y} \in \mathbf{Y}$  have at most  $\text{VC}(\mathbf{Y})$ -many  $y_i = 1$  (the rest 0).
  - \* Therefore:  $|\mathbf{Y}| \leq \binom{m}{0} + \binom{m}{1} + \dots + \binom{m}{\text{VC}(\mathbf{Y})}$  by counting
  - \* Proof sketch: if a  $\mathbf{y} \in \mathbf{Y}$  had more 1s, all combinations would exist "below"
- Together: exists a shift sequence  $i_1, \dots, i_N$  to a closed below  $H_{i_N}(\mathbf{Y})$ :  

$$|\mathbf{Y}| = |H_{i_1}(\mathbf{Y})| = \dots = |H_{i_N}(\mathbf{Y})| \leq \sum_{i=0}^{\text{VC}(H_{i_N}(\mathbf{Y}))} \binom{m}{i} \leq \dots \leq \sum_{i=0}^{\text{VC}(\mathbf{Y})} \binom{m}{i}$$

27

## Mini Summary

- Linear classifiers in  $\mathbb{R}^d$  have VC dimension  $d + 1$ 
  - \* Lower bound VC-dim with specific points that are shattered
  - \* Upper bound VC-dim by lin. dependence of any  $d + 2$  points
- Sauer-Shelah lemma bounds a family's growth function by a polynomial in VC dimension.
  - \* Ingenious shifting operator transforms sets of dichotomies into boundable closed-below sets
  - \* Along the way keeps cardinality and VC-dim controlled

Next time: Support vector machines

# **Superman vs Batman**

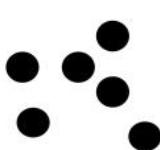


<https://www.dailymotion.com/video/x64zal>

3

<https://www.pinterest.com/pin/88702654667128852>

# **Superman vs Batman**



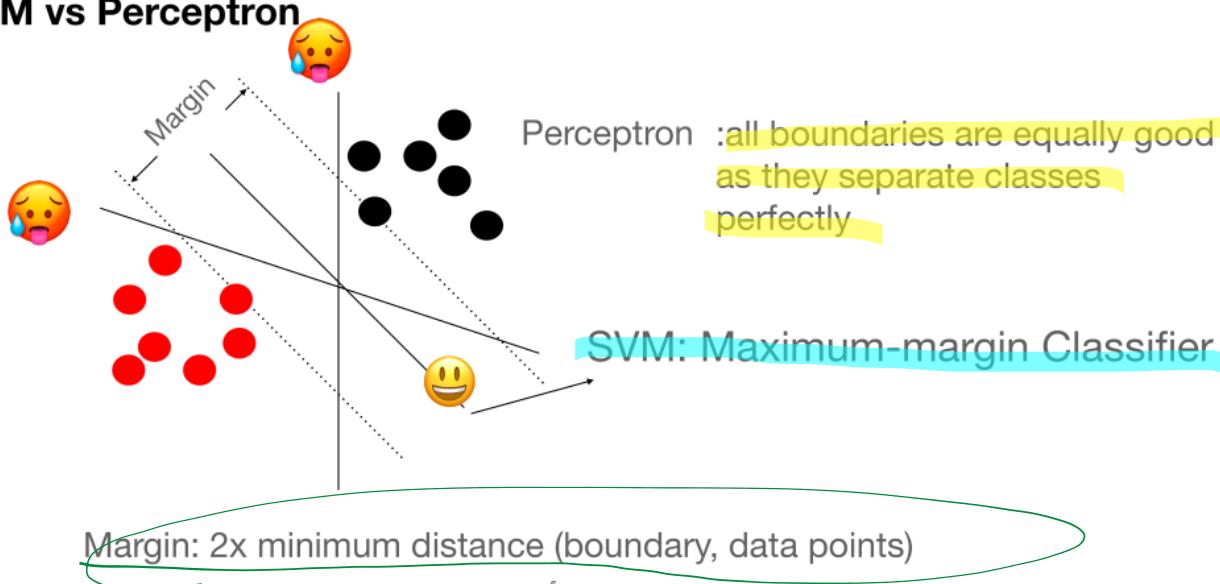
<https://5k3dnetworks.com.au/product/2-cartoon-for-kids-5-Silly-monday-1-Slimy-4-Slimey-6th-july-2020>  
<https://pregio.com/PNGs/5120-baby-superman-cartoon.html>  
<https://www.gutenberg.org.au/poetry/217268678/>  
<https://www.dailymotion.com/video/xdmzft>

4

<https://www.pagefaster.com/middle/?t=uhuman-cutie-png-transparent.png>  
[https://www.dipartimax.com/file/n281f7c3d3bf1d25\\_baby-batman-image-baby-batman/](https://www.dipartimax.com/file/n281f7c3d3bf1d25_baby-batman-image-baby-batman/)  
<https://www.pinterest.com/pin/232550343163174160/>

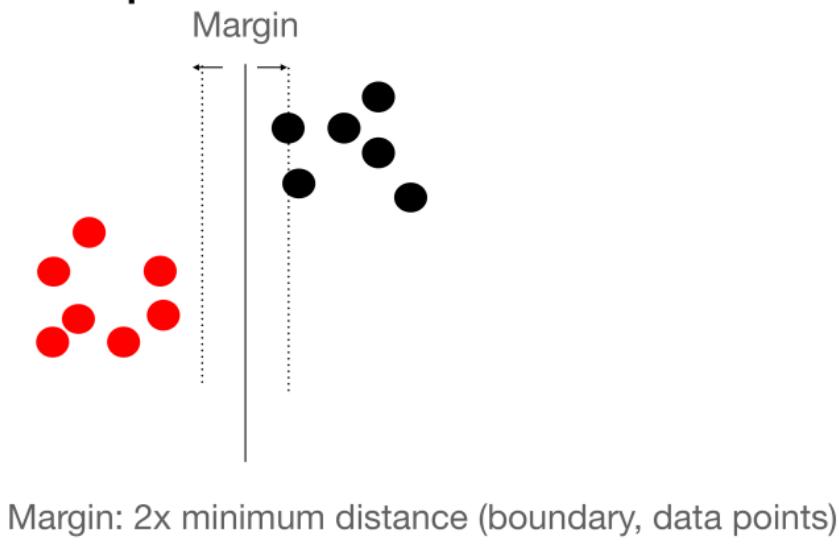
# Binary Linear Classifier

## SVM vs Perceptron



# Binary Linear Classifier

## SVM vs Perceptron



# Outline

- Margin
- Lagrange Duality
- Soft-margin SVM
- Kernels

7

Margin	Lagrange Duality	Soft-margin SVM	Kernels
--------	------------------	-----------------	---------

**Linear classifier**

$f(x) = w^T x + b$

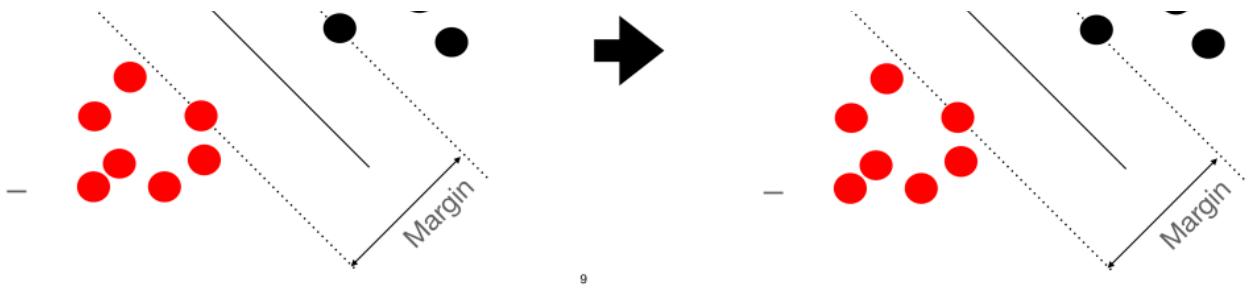
$x$  : Feature vector (column)  
 $w$  : Weight vector (column)  
 $T$  : Transpose  
 $b$  : Bias  
 $w^T x = \|w\| \|x\| \cos \theta$   
 $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$

$\hat{w}^T x + \hat{b} = 0$   
 $\hat{w}^T x + \hat{b} = -b_0$   
 $\hat{w}^T x + \hat{b} = +b_0$

$f(x) = 0$   
 $f(x) = +1$   
 $f(x) = -1$

Margin	Lagrange Duality	Soft-margin SVM	Kernels
--------	------------------	-----------------	---------



9

## Margin

## Lagrange Duality

## Soft-margin SVM

## Kernels

### Margin formula

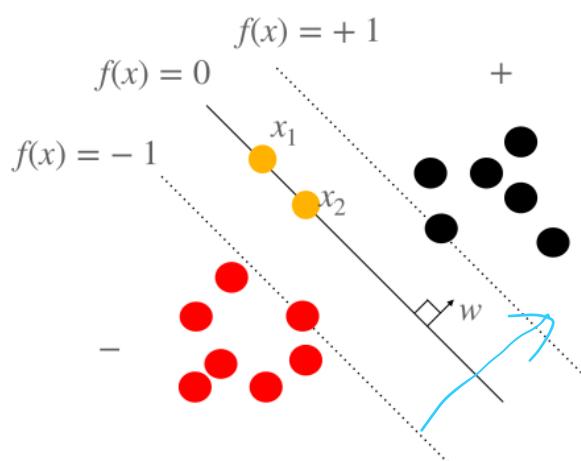
$$f(x) = w^T x + b$$

$$w^T x_1 + b = 0$$

$$w^T x_2 + b = 0$$

$$w^T(x_1 - x_2) = 0$$

$$\|w\| \|x_1 - x_2\| \cos\theta = 0$$



10

## Margin

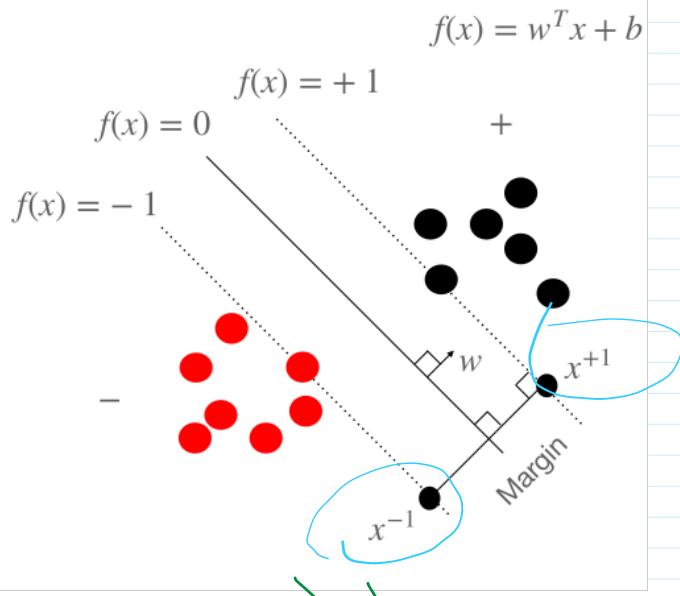
## Lagrange Duality

## Soft-margin SVM

## Kernels

### Margin formula

$$\begin{aligned} w^T x^{-1} + b &= -1 \\ w^T x^{+1} + b &= 1 \\ w^T(x^{+1} - x^{-1}) &= 2 \\ \|w\| \cdot \text{Margin} \cdot \cos\theta &= 2 \\ \text{Margin} &= \frac{2}{\|w\|} \end{aligned}$$



11

## Margin

## Lagrange Duality

## Soft-margin SVM

## Kernels

### SVM: Constrained optimisation problem

$$\min_w \frac{\|w\|^2}{2}$$

$$\max_w \frac{2}{\|w\|}$$

max margin

## SVM: Constrained optimisation problem

$$\min_w \frac{\|w\|^2}{2}$$

s.t.

$$1 - y^{(i)}(w^T x^{(i)} + b) \leq 0, i = 1, \dots, n \text{ (data points)}$$

$$f(x) = 0$$

$$f(x) = +1$$

$$f(x) = -1$$

$$-$$

$$+ \quad \quad \quad +$$

$$\begin{array}{c} f(x) = 0 \\ f(x) = +1 \\ f(x) = -1 \end{array}$$



$$\max_w \frac{2}{\|w\|}$$

subject to

$$\text{if } y^{(i)} = +1 : f(x^{(i)}) = w^T x^{(i)} + b \geq +1$$

$$\text{if } y^{(i)} = -1 : f(x^{(i)}) = w^T x^{(i)} + b \leq -1$$

( $i = 1, \dots, n$  data points)

max margin

$$y^{(i)} f(x^{(i)}) \geq 1$$

12

Margin

Lagrange  
Duality

Soft-margin  
SVM

Kernels

## Primal problem

$$\min_w \frac{\|w\|^2}{2}$$

s.t.

$$1 - y^{(i)}(w^T x^{(i)} + b) \leq 0, i = 1, \dots, n \text{ (data points)}$$

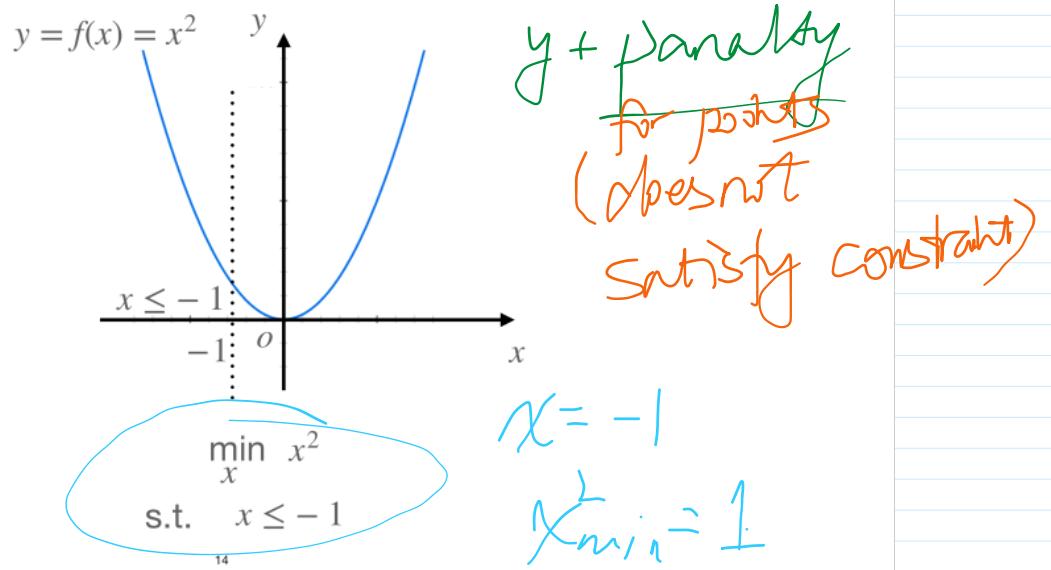


## Dual problem

What's the dual problem?

Why solving primal by  
solving dual problem?

13

**Simple example****Primal problem**

$$\min_x f(x)$$

$$\text{s.t. } g(x) = x + 1 \leq 0$$

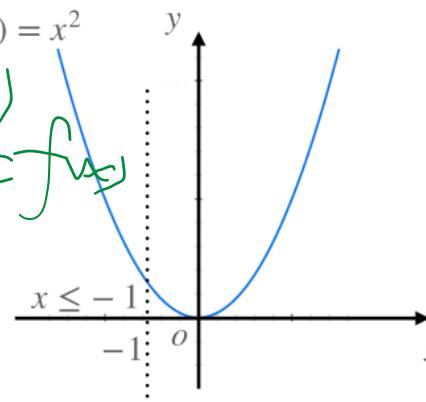
- Construct a function:  $L(x, \lambda) = f(x) + \lambda g(x)$
- Set  $\lambda \geq 0$ , calculate  $\max_{\lambda} L(x, \lambda)$

$g(x) > 0 : \max_{\lambda} L(x, \lambda) = \infty$  when  $\lambda = \infty$

$g(x) \leq 0 : \max_{\lambda} L(x, \lambda) = f(x)$  when  $\lambda = 0$



$$\max_{\lambda} L(x, \lambda) = f(x)$$



**Primal problem**

$$\min_x f(x)$$

$$\text{s.t. } g(x) = x + 1 \leq 0$$

- Construct a function

$$L(x, \lambda) = f(x) + \lambda g(x) : \text{Lagrangian function}$$

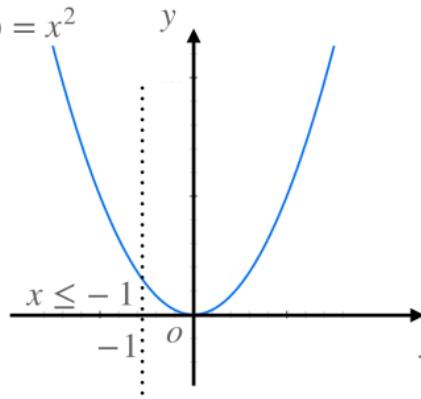
$\lambda \geq 0$  : Lagrange multiplier

- Primal function

$$\theta_p(x) = \max_{\lambda} L(x, \lambda) = \begin{cases} f(x) & \text{if } g(x) \leq 0 \\ -\infty & \text{otherwise} \end{cases}$$

$$\text{So: } \min_x f(x) = \min_x \theta_p(x) = \min_x \max_{\lambda} L(x, \lambda)$$

$$y = f(x) = x^2$$



$$\min_x x^2$$

$$\text{s.t. } x \leq -1$$

**From primal to dual problem**

- Primal problem:

$$\min_x f(x) = \min_x \max_{\lambda} L(x, \lambda)$$

- Dual problem:

$$\max_{\lambda} \min_x L(x, \lambda) = \max_{\lambda} \theta_d(\lambda)$$

$$\text{Dual function: } \theta_d(\lambda) = \min_x L(x, \lambda)$$

$$L(x, \lambda) = f(x) + \lambda g(x)$$

$$\lambda \geq 0 \quad g(x) \leq 0$$

so

$L(x, \lambda) \leq f(x)$

$$\theta_d(\lambda) = \min_x L(x, \lambda) \leq L(x, \lambda) = f(x) + \lambda g(x) \leq f(x)$$

so

## From primal to dual problem

- Primal problem:

$$\min_x f(x) = \min_x \max_{\lambda} L(x, \lambda)$$

- Dual problem:

$$\max_{\lambda} \min_x L(x, \lambda) = \max_{\lambda} \theta_d(\lambda)$$

$$\theta_d(\lambda) = \min_x L(x, \lambda) \leq L(x, \lambda) = f(x) + \lambda g(x) \leq f(x)$$

Solutions:

$x^*$  makes  $f(x)$  minimum :  $f(x^*) = p^*$

$\lambda^*$  makes  $\theta_d(\lambda)$  maximum :  $\theta_d(\lambda^*) = d^*$

18

## From primal to dual problem

- Primal problem:

$$\min_x f(x) = \min_x \max_{\lambda} L(x, \lambda)$$

- Dual problem:

$$\max_{\lambda} \min_x L(x, \lambda) = \max_{\lambda} \theta_d(\lambda)$$

$$\theta_d(\lambda) = \min_x L(x, \lambda) \leq L(x, \lambda) = f(x) + \lambda g(x) \leq f(x)$$

$$d^* = \theta_d(\lambda^*) = \min_x L(x, \lambda^*) \leq L(x^*, \lambda^*) = f(x^*) + \lambda^* g(x^*) \leq f(x^*) = p^*$$

Under some conditions:  $d^* = p^*$

Solutions:

$$f(x^*) = p^* = \min_x f(x)$$

$$\theta_d(\lambda^*) = d^* = \max_{\lambda} \theta_d(\lambda)$$

?

19

**From primal to dual problem**

$$d^* = \theta_d(\lambda^*) = \min_x L(x, \lambda^*) \leq L(x^*, \lambda^*) = f(x^*) + \lambda^* g(x^*) \leq f(x^*) = p^*$$

if  $\min_x L(x, \lambda^*) = L(x^*, \lambda^*)$  and  $f(x^*) + \lambda^* g(x^*) = f(x^*)$   
 $d^* = p^*$

KKT (Karush-Kuhn-Tucker) conditions :

$g(x) \leq 0$  (Primal feasibility)

$\lambda \geq 0$  (Dual feasibility)

$\lambda g(x) = 0$  (Complementary slackness)

$\frac{\partial L}{\partial x} = 0$  (Stationarity)

20

**Dual problem of SVM**

Primal problem  $\min_w \frac{\|w\|^2}{2}$  *n constraints*  
 s.t.  $g_i(w, b) = 1 - y^{(i)}(w^T x^{(i)} + b) \leq 0, i = 1, \dots, n$  data points

(1) Lagrangian function:

$$L(w, b, \lambda) = \frac{\|w\|^2}{2} + \sum_{i=1}^n \lambda_i (1 - y^{(i)}(w^T x^{(i)} + b))$$

(2) dual function  $\theta_d(\lambda) = \min_{w, b} L(w, b, \lambda)$ :

$$\begin{aligned} \frac{\partial L}{\partial w_j} &= 0 : w_j = \sum_{i=1}^n \lambda_i y^{(i)} x_j^{(i)} \\ \frac{\partial L}{\partial b} &= 0 : \sum_{i=1}^n \lambda_i y^{(i)} = 0 \end{aligned}$$

21

**Dual problem of SVM****Dual function**

$$\theta_d(\lambda) = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{k=1}^n \lambda_i \lambda_k y^{(i)} y^{(k)} (x^{(i)})^T x^{(k)}$$

**Dual problem**

$$\begin{aligned} &\max_{\lambda} \theta_d(\lambda) \\ \text{s.t. } &\lambda_i \geq 0 \quad \text{and} \quad \sum_{i=1}^n \lambda_i g_i(x) = 0 \end{aligned}$$

$$\text{s.t. } \lambda_i \geq 0 \quad \text{and} \quad \sum_{i=1}^n \lambda_i g_i(x) = 0$$

22

Margin

Lagrange  
DualitySoft-margin  
SVM

Kernels

**Support vectors**

$$\min \frac{\|w\|^2}{2} \quad \text{s.t. } g_i(w, b) = 1 - y^{(i)}(w^T x^{(i)} + b) \leq 0, i = 1, \dots, n \text{ data points}$$

$$\text{Lagrangian function: } L(w, b, \lambda) = \frac{\|w\|^2}{2} + \sum_{i=1}^n \lambda_i (1 - y^{(i)}(w^T x^{(i)} + b))$$

$$\lambda_i \geq 0$$

(Dual feasibility)

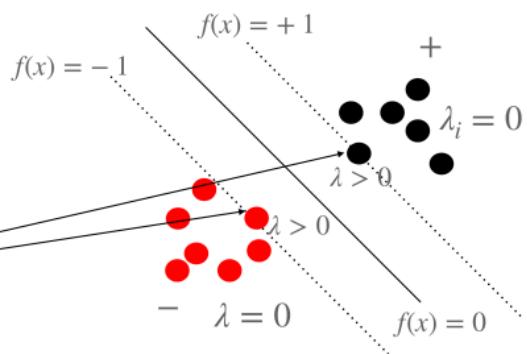
$$\lambda_i g_i(w, b) = 0$$

(Complementary slackness)

KKT condition

support vectors:

23



Margin

Lagrange  
DualitySoft-margin  
SVM

Kernels

**Primal vs Dual (Training)** ~~$\lambda \neq 0$~~ 

- Primal problem: solve  $d+1$  variables ( $w_j$  and  $b$ ) ( $d$ : dimension of weight vector  $w$ )

$$\min_w \frac{\|w\|^2}{2}$$

$$g_i(w, b) = 1 - y^{(i)}(w^T x^{(i)} + b) \leq 0, i = 1, \dots, n \text{ data points}$$

- Dual problem: solve  $n$  variables ( $\lambda_i$ )

$$\max_{\lambda} \theta_d(\lambda) \quad \theta_d(\lambda) = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{k=1}^n \lambda_i \lambda_k y^{(i)} y^{(k)} (x^{(i)})^T x^{(k)}$$

s.t.

$$\lambda_i \geq 0 \text{ and } \sum_{i=1}^n \lambda_i g_i(x) = 0$$

If data size  $n$  is large, ( $n \gg d$ ) solving dual problem is slower than solving primal problem, and vice versa.

24

## Primal vs Dual (Prediction)

- Primal form:

$$f(x) = w^T x + b \quad \begin{array}{l} f(x) > 0 : \text{positive class} \\ f(x) < 0 : \text{negative class} \end{array}$$

- Dual form:

$$w_j = \sum_{i=1}^n \lambda_i y^{(i)} x_j^{(i)}$$

$$f(x) = \sum_{i=1}^n \lambda_i y^{(i)} (x^{(i)})^T x + b$$

( $b$  can be solved using support vectors:  $f(x) = \pm 1$ )

25

## Why bother solving dual problem to solve primal problem

Training, solve:

$$\begin{aligned} \max_{\lambda} \theta_d(\lambda) \quad & \theta_d(\lambda) = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{k=1}^n \lambda_i \lambda_k y^{(i)} y^{(k)} (x^{(i)})^T x^{(k)} \\ \text{s.t.} \quad & \lambda_i \geq 0 \text{ and } \sum_{i=1}^n \lambda_i g_i(x) = 0 \end{aligned}$$

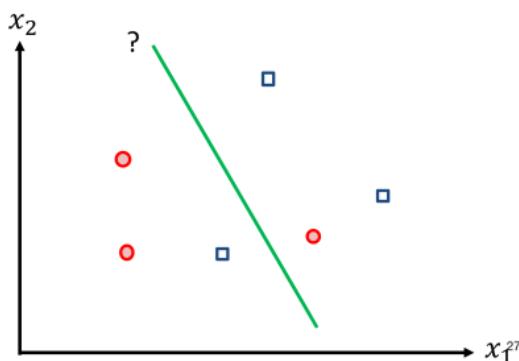
$$\text{Prediction: } f(x) = \sum_{i=1}^n \lambda_i y^{(i)} (x^{(i)})^T x + b$$

- Use only support vectors for prediction: Efficient in prediction
- Inner product: Kernel trick can be used to efficiently handle non-linearly separable data

26

**Data not linearly separable**

- Hard-margin loss is too stringent (hard!)
- Real data is unlikely to be linearly separable
- If the data is not separable, hard-margin SVMs are in trouble

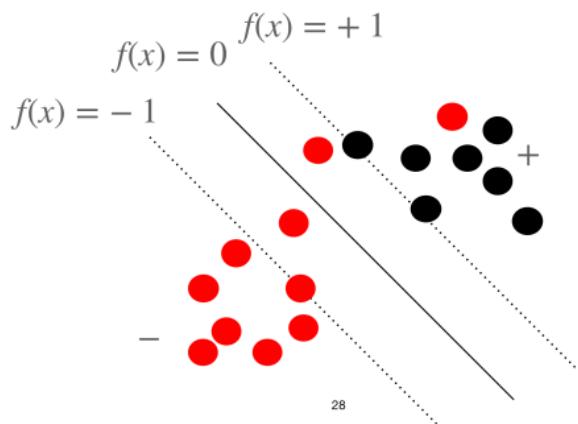


SVMs offer 3 approaches to address this problem:

1. Relax the constraints (soft-margin)
2. Still use hard-margin SVM, but transform the data (kernel)
3. The combination of 1 and 2 ☺

**Soft-margin SVM: 'soft' constraint**

- Relax constraints to allow points to be inside the margin or even on the wrong side of the boundary



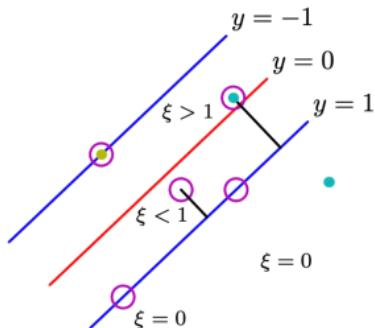
**Objective of soft-margin SVM**

$$\min_w \left( \frac{\|w\|^2}{2} + C \sum_{i=1}^n \xi_i \right) \quad \text{s.t.} \quad \begin{aligned} & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0 \quad (i = 1, \dots, n \text{ data points}) \end{aligned}$$

Use slack variable to 'soft' constraint:  
allow violation of the constraint

$$\xi_i = \begin{cases} 0, & y^{(i)}(w^T x^{(i)} + b) \geq 1, \\ 1 - y^{(i)}(w^T x^{(i)} + b), & \text{otherwise} \end{cases}$$

or  $\xi_i = \max(0, 1 - y^{(i)}(w^T x^{(i)} + b))$  hinge loss



29 Figure 7.3 in Pattern Recognition and Machine Learning by Chris Bishop

**Objective of soft-margin SVM**

$$\min_w \left( \frac{\|w\|^2}{2} + C \sum_{i=1}^n \xi_i \right) \quad \text{s.t.} \quad \begin{aligned} & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0 \quad (i = 1, \dots, n \text{ data points}) \end{aligned}$$

Slack penalty:  $C > 0$

If  $C = 0$ : data is ignored



Underfitting

If  $C = \infty$ : data has to be correctly classified



Overfitting

## Margin

## Lagrange Duality

## Soft-margin SVM

## Kernels

### KKT

$$L(w, b, \lambda, \beta, \xi) = \frac{\|w\|^2}{2} + C \sum_{i=1}^n \xi_i + \sum_{i=1}^n \lambda_i g_i(w, b, \xi) + \sum_{i=1}^n \beta_i (-\xi_i)$$

$$g_i(w, b, \xi) = 1 - \xi_i - y^{(i)}(w^T x^{(i)} + b) \leq 0 \quad -\xi_i \leq 0$$

**Primal feasibility:**  $\underline{g_i(w, b, \xi) \leq 0}$   $-\xi_i \leq 0$

**Dual feasibility**  $\lambda_i \geq 0 \quad \beta_i \geq 0$

**Complementary slackness**  $\lambda_i g_i(w, b, \xi) = 0 \quad \beta_i \xi_i = 0$

**Stationarity**

$\frac{\partial L}{\partial w_j} = 0 : w_j = \sum_{i=1}^n \lambda_i y^{(i)} x_j^{(i)}$	$\frac{\partial L}{\partial b} = 0 : \sum_{i=1}^n \lambda_i y^{(i)} = 0$
$\frac{\partial L}{\partial \xi_i} = 0 : C - \lambda_i - \beta_i = 0$	

## Margin

## Lagrange Duality

## Soft-margin SVM

## Kernels

### KKT

**Primal feasibility:**  $\underline{g_i(w, b, \xi) = 1 - \xi_i - y^{(i)}(w^T x^{(i)} + b) \leq 0}, \quad -\xi_i \leq 0$

**Dual feasibility**  $\underline{\lambda_i \geq 0 \quad \beta_i \geq 0}$

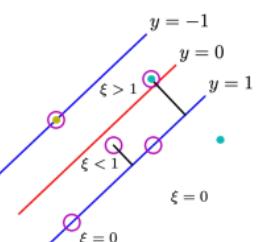
**Complementary slackness**  $\underline{\lambda_i g_i(w, b, \xi) = 0 \quad \beta_i \xi_i = 0}$

$$C - \lambda_i - \beta_i = 0 : 0 \leq \lambda_i \leq C$$

$$\text{if } \lambda_i = 0 : \beta = C, \xi_i = 0 \quad y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i = 1$$

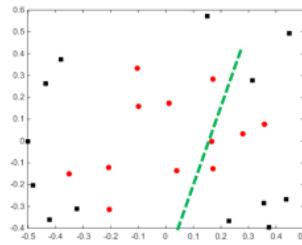
$$\text{if } \lambda_i = C : \beta_i = 0, -\xi_i \leq 0 \quad y^{(i)}(w^T x^{(i)} + b) = 1 - \xi_i \leq 1$$

$$\text{if } 0 < \lambda_i < C : \xi_i = 0 \quad g_i(w, b, \xi) = 0 \quad y^{(i)}(w^T x^{(i)} + b) = 1 - \xi_i = 1$$



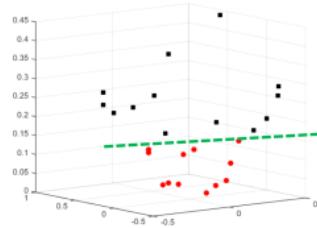
## Non-linearly separable data

- Consider a binary classification problem
- Each example has features  $[x_1, x_2]$
- Not linearly separable



- Now 'add' a feature  $x_3 = x_1^2 + x_2^2$
- Each point is now  $[x_1, x_2, x_1^2 + x_2^2]$
- Linearly separable!

*2D → 3D*



33

## Naïve workflow

- Choose/design a linear model
- Choose/design a high-dimensional transformation  $\varphi(x)$ 
  - Hoping that after adding a lot of various features some of them will make the data linearly separable
- For each training example, and for each new instance compute  $\varphi(x)$
- Train classifier/Do predictions

34

**Hard-margin SVM in feature space**

Training, solve:

$$\begin{aligned} \max_{\lambda} \theta_d(\lambda) \quad \theta_d(\lambda) &= \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{k=1}^n \lambda_i \lambda_k y^{(i)} y^{(k)} (x^{(i)})^T x^{(k)} \\ \text{s.t.} \quad \lambda_i \geq 0 \text{ and } \sum_{i=1}^n \lambda_i g_i(x) &= 0 \end{aligned}$$

$$\text{Prediction: } f(x) = \sum_{i=1}^n \lambda_i y^{(i)} (x^{(i)})^T x + b$$

We just need the dot product!

$$x \rightarrow \varphi(x)$$

$$(\varphi(x))^\top \varphi(x)$$

35

**Observation: Kernel representation**

- Both parameter estimation and computing predictions depend on data only in a form of a dot product
  - In original space  $\mathbf{u}'\mathbf{v} = \sum_{i=1}^m u_i v_i$
  - In transformed space  $\varphi(\mathbf{u})' \varphi(\mathbf{v}) = \sum_{i=1}^l \varphi(\mathbf{u})_i \varphi(\mathbf{v})_i$
- Kernel is a function that can be expressed as a dot product in some feature space  $K(\mathbf{u}, \mathbf{v}) = \varphi(\mathbf{u})' \varphi(\mathbf{v})$

Benefits:

- no need to find the mapping function.
- no need to do transformation.
- no need to do dot product.

36

**Kernel as shortcut**

- For some  $\varphi(x)$ 's, kernel is faster to compute directly than first mapping to feature space then taking dot product.
- For example, consider two vectors  $\mathbf{u} = [u_1]$  and  $\mathbf{v} = [v_1]$  and transformation  $\varphi(\mathbf{x}) = [x_1^2, \sqrt{2c}x_1, c]$ , some  $c$ 
  - \* So  $\varphi(\mathbf{u}) = [u_1^2, \sqrt{2c}u_1, c]^T$  and  $\varphi(\mathbf{v}) = [v_1^2, \sqrt{2c}v_1, c]^T$
  - \* Then  $\varphi(\mathbf{u})' \varphi(\mathbf{v}) = (u_1^2 v_1^2 + 2cu_1 v_1 + c^2)$
- This can be alternatively computed directly as  $\varphi(\mathbf{u})' \varphi(\mathbf{v}) = (u_1 v_1 + c)^2$ 
  - \* Here  $K(\mathbf{u}, \mathbf{v}) = (u_1 v_1 + c)^2$  is the corresponding kernel

37

**Hard-margin SVM in feature space**Training: solve

$$\max_{\lambda} L(\lambda) \quad L(\lambda) = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{k=1}^n \lambda_i \lambda_k y^{(i)} y^{(k)} \underbrace{(\varphi(x^{(i)}))^T \varphi(x^{(k)})}_{K(x^{(i)}, x^{(k)})}$$

Making predictions:

$$f(x) = w^T x + b = \sum_{i=1}^n \lambda_i y^{(i)} \underbrace{(\varphi(x^{(i)}))^T \varphi(x)}_{K(x^{(i)}, x)} + b$$

*kernel trick*

38

**Radial Basis Function (RBF) kernel**

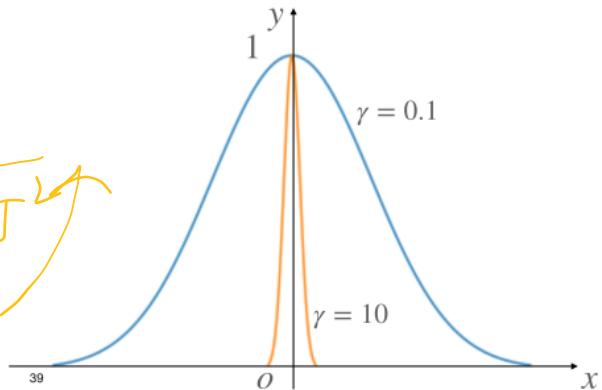
$$K(u, v) = \exp(-\gamma \|u - v\|^2)$$

$\gamma$  is too small : underfitting

$\gamma$  is too large : overfitting

$$\gamma = \frac{1}{2\sigma^2}$$

$$y = \exp(-\gamma x^2) = \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

**Identify new kernels****Mercer's theorem:**

Consider a finite sequences of vectors  $x_1, \dots, x_n$

Construct  $n \times n$  matrix A (Gram matrix) of pairwise values

$K(x_i, x_j)$  is a valid kernel if this matrix is positive semi-definite, and this holds for all possible sequences

PSD

$$A = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \dots & K(x_1, x_n) \\ K(x_2, x_1) & K(x_2, x_2) & \dots & K(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(x_n, x_1) & K(x_n, x_2) & \dots & K(x_n, x_n) \end{bmatrix}$$

**Identify new kernels**

$$A^T = A$$

Positive semi-definite matrix: a square symmetric matrix satisfies  $v^T A v \geq 0$   
 $v \in \mathbb{R}^{n \times 1}$  any non-zero vector (column),  $A \in \mathbb{R}^{n \times n}$ ,  $A = A^T$

$$A = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \dots & K(x_1, x_n) \\ K(x_2, x_1) & K(x_2, x_2) & \dots & K(x_2, x_n) \\ \vdots & \vdots & \vdots & \vdots \\ K(x_n, x_1) & K(x_n, x_2) & \dots & K(x_n, x_n) \end{bmatrix} \triangleq B^T B$$

41

**Identify new kernels**

Let  $K_1(u, v)$ ,  $K_2(u, v)$  be kernels,  $c > 0$  be a constant, and  $f(x)$  be a real-valued function.

Then each of the following is also a kernel:

1)  $K(u, v) = K_1(u, v) + K_2(u, v)$

2)  $K(u, v) = c K_1(u, v)$

3)  $K(u, v) = f(u)^T \varphi_1(u) f(v)$

$$\sqrt{A} V = \sqrt{A} \sqrt{V} + \sqrt{A} \sqrt{V} \geq 0 \iff \geq 0 \geq 0$$

$$\begin{aligned} & f(u)^T \varphi_1(u) f(v) \\ &= (f(u)^T \varphi_1(u))^T (\varphi_1(v)^T f(v)) \end{aligned}$$

→

**Summary**

- What are the objective and constraints of hard-margin, soft-margin SVM
- What are KKT conditions?
- What are support vectors?
- What are Slack variables & slack penalty of soft-margin SVM?
- What is Kernel?

- What is Kernel?

- How do parameters  $\gamma, C$  influence performance of SVM?

- How to identify new kernels?



# Introduction to Deep Learning — — Recap of Neural network

COMP90051 Statistical Machine Learning

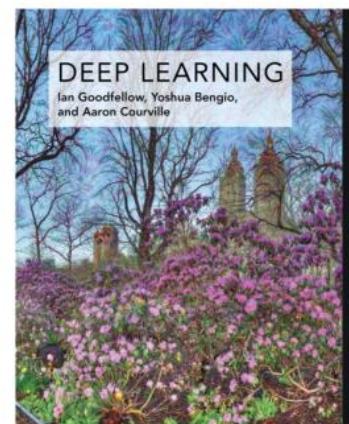
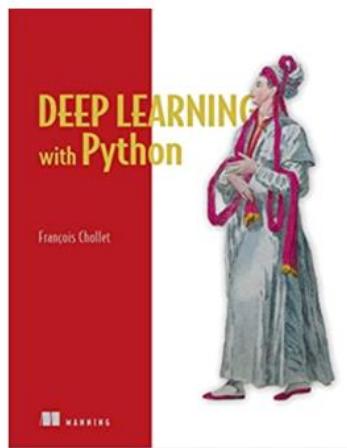
QiuHong Ke

Copyright: University of Melbourne

## Before we start

### Books & resources

- Deep Learning with Python, by Francois Chollet, available in unimelb library
- Deep Learning, by Ian Goodfellow and Yoshua Bengio and Aaron Courville, <https://www.deeplearningbook.org/>



# Angry birds



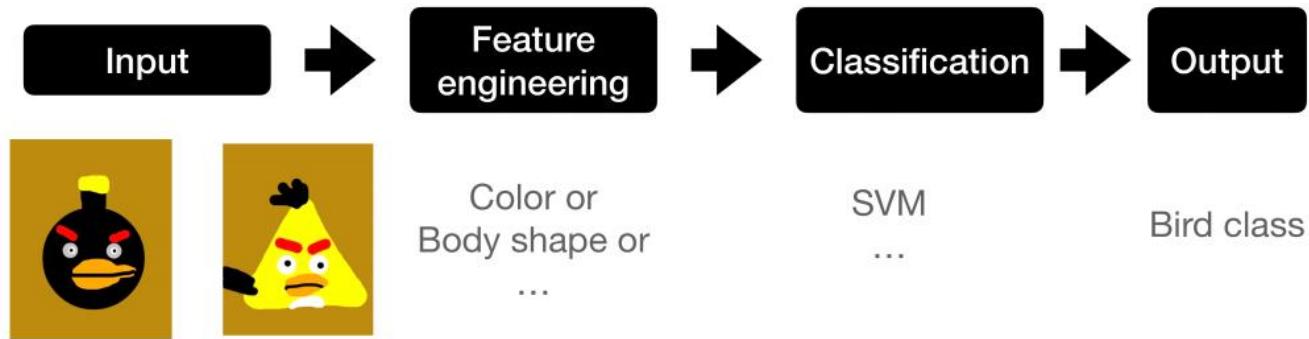
VS



Source: by Papachan (CC BY)  
<https://www.sketchport.com/share/45242868978976/bomb-and-chick>

3

## Bird classification



Source: by Papachan (CC BY)  
<https://www.sketchport.com/share/45242868978976/bomb-and-chick>

4

## In real life...

Boat tailed Grackle



Pomarine Jaeger



Bobolink



Prairie Warbler



Bohemian Waxwing



Prothonotary Warbler



Brandt Cormorant



Purple Finch



Brewer Blackbird



Red bellied Woodpecker



Source: Welinder, Peter, et al. "Caltech-UCSD birds 200." (2010).

5

## More features for real birds...

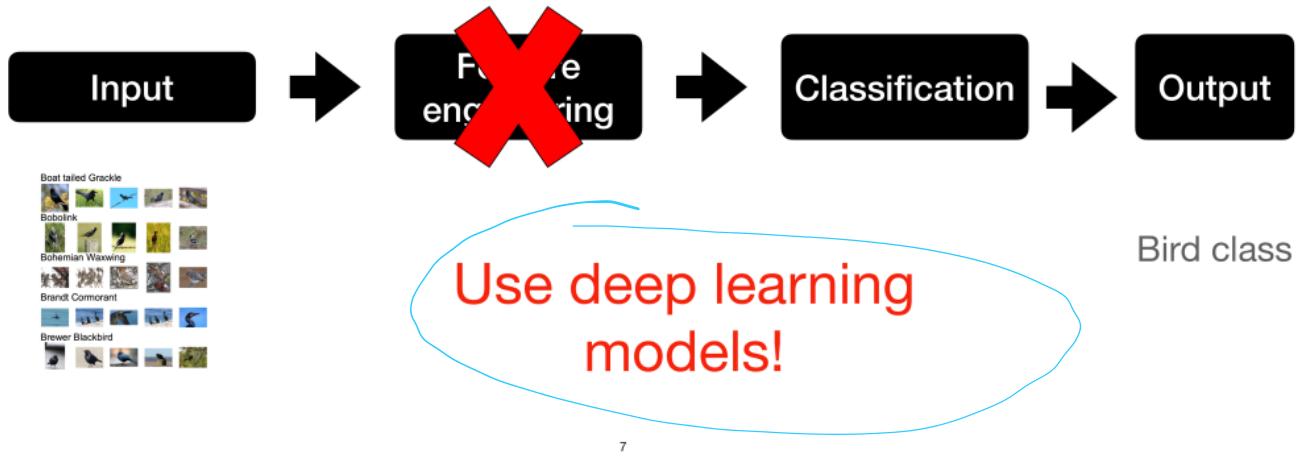


forehead_color	black	black	black
breast_pattern	solid	solid	solid
breast_color	white	white	white
head_pattern	plain	capped	plain
back_color	white	white	black
wing_color	grey/white	grey	white
leg_color	orange	orange	orange
size	medium	large	medium
bill_shape	needle	dagger	dagger
wing_shape	pointed	tapered	long
...	...	...	...
primary_color	white	white	white

Source: Welinder, Peter, et al. "Caltech-UCSD birds 200." (2010).

6

# 'REAL Bird' classification



## Outline

- Introduction
- Neural network (Perceptron and Multi-layer Perceptron)
- Gradient descent algorithm

**Deep learning**

Input

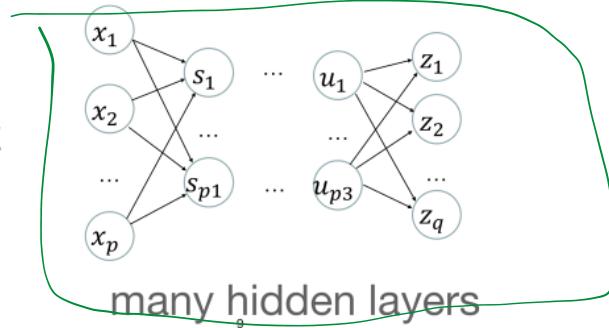


Feature learning + classification



Output

Deep neural network

**Deep learning is everywhere**

Near-human-level  
image classification  
speech recognition  
handwriting transcription  
autonomous driving  
....

## Why now deep learning is popular?

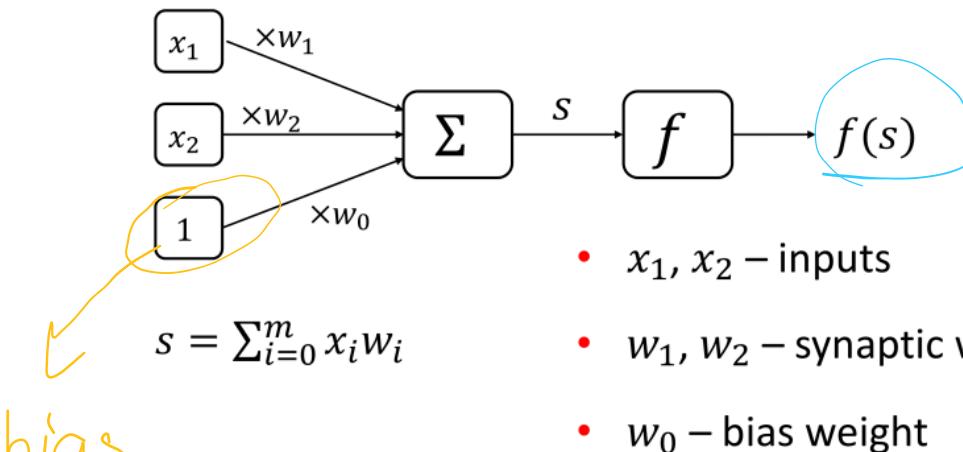
- Large dataset
- Powerful computing resources
- Better algorithms:
  - Weight-initialization schemes
  - Optimisation schemes
  - Activation functions
  - ...



Network: deeper than deeper

11

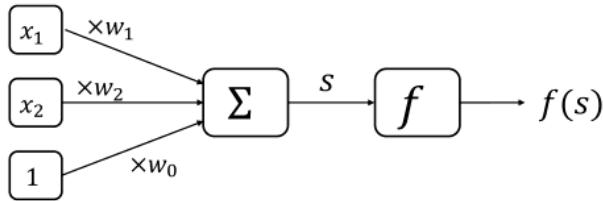
## Perceptron



- $x_1, x_2$  – inputs
- $w_1, w_2$  – synaptic weights
- $w_0$  – bias weight
- $f$  – activation function

12

## Perceptron is a linear binary classifier



if  $s \geq 0 : f(s)$  is positive class

if  $s < 0 : f(s)$  is negative class

Step function

$$f(s) = \begin{cases} 1, & \text{if } s \geq 0 \\ 0, & \text{if } s < 0 \end{cases}$$

$\circ \rightarrow$  negative class

Sign function

$$f(s) = \begin{cases} 1, & \text{if } s \geq 0 \\ -1, & \text{if } s < 0 \end{cases}$$

$-1 \rightarrow$  negative class

13

## Simple example

$$x_1 + x_2$$

Exercise: find weights of a perceptron capable of perfect classification of the following dataset

$$w_1 = w_2 = 1$$

$$w_0 = -1.5$$

$$\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

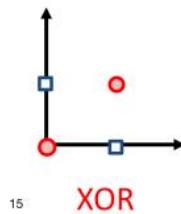
$x_1$	$x_2$	$y$
0	0	Class B
0	1	Class B
1	0	Class B
1	1	Class A

$$w_0 + w_1 x_1 + w_2 x_2$$

14

## Limitations of perceptron learning

- If the data is linearly separable, the perceptron training algorithm will converge to a correct solution
  - \* It will converge to some solution (separating boundary), one of infinitely many possible ← bad!
- However, if the data is not linearly separable, the training will fail completely rather than give some approximate solution
  - \* Ugly ☹

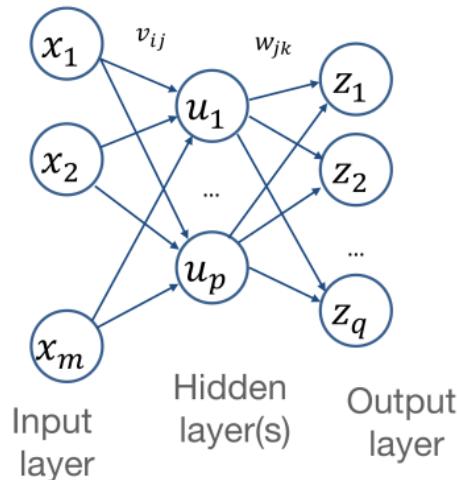


## Multi-layer perceptron: function composition

$$u_j = g(r_j)$$

$$r_j = \sum_{i=0}^m x_i v_{ij}$$

$g(x)$ : activation function



$$z_k = h(s_k)$$

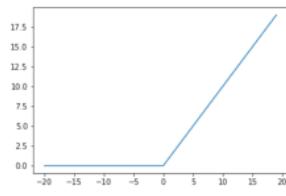
$$s_k = \sum_{j=0}^p u_j w_{jk}$$

$h(x)$ : activation function

## Common non-linear activation functions

Relu:  $f(x) = \max(0, x)$

~~binary classification~~



Sigmoid:  $f(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$

TanH  $f(x) = \frac{2}{1 + e^{-2x}} - 1 = \frac{2e^{2x}}{e^{2x} + 1} - 1$

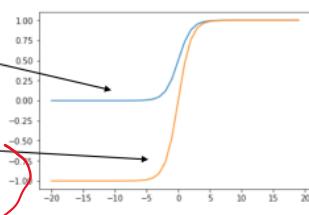
Softmax:  $f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^C e^{x_j}}$   $i = 1, \dots, C$   
 (Multi-class classification)

17

(0, 1)

(-1, 1)

(0, 1)  
range



probability

## How to train your dragon network?

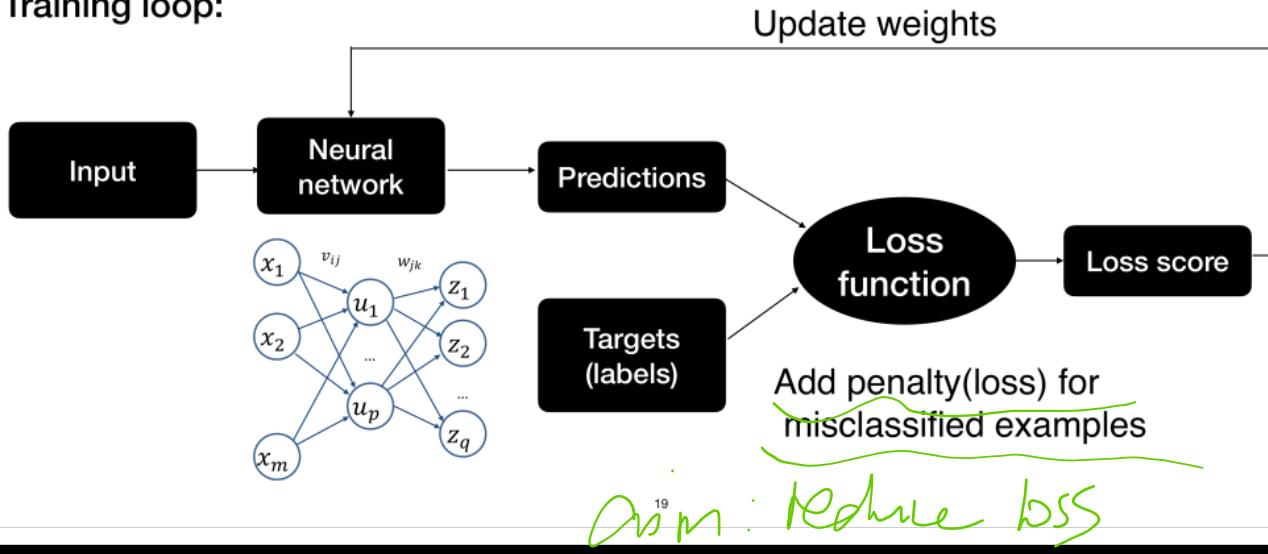


Adapted from Movie Poster from  
Flickr user jdxyw (CC BY-SA 2.0)

**“Training”: adjust weights to minimise loss.**

**How?**

Training loop:



## Derivative

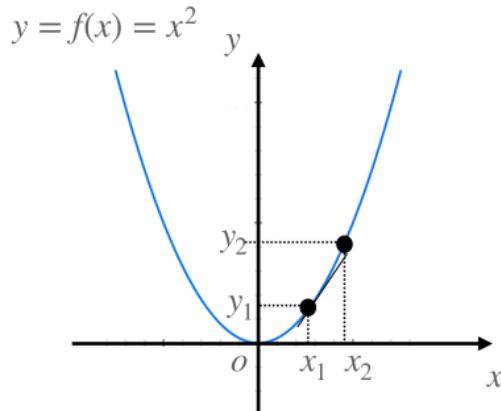
Input  $x$ , output  $y$ :  $f(x)$

$$\Delta x \rightarrow 0 : f(x_1 + \Delta x) - f(x_1) = a\Delta x$$

*a : rate of change (derivative) at  $x_1$*

**derivative: a vector from origin.**

Move  $x$  along OPPOSITE direction of the vector: decrease  $f(x)$



**Gradient**

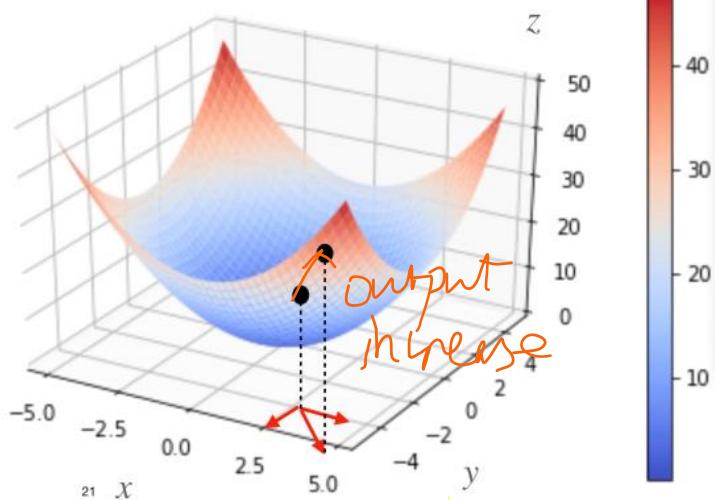
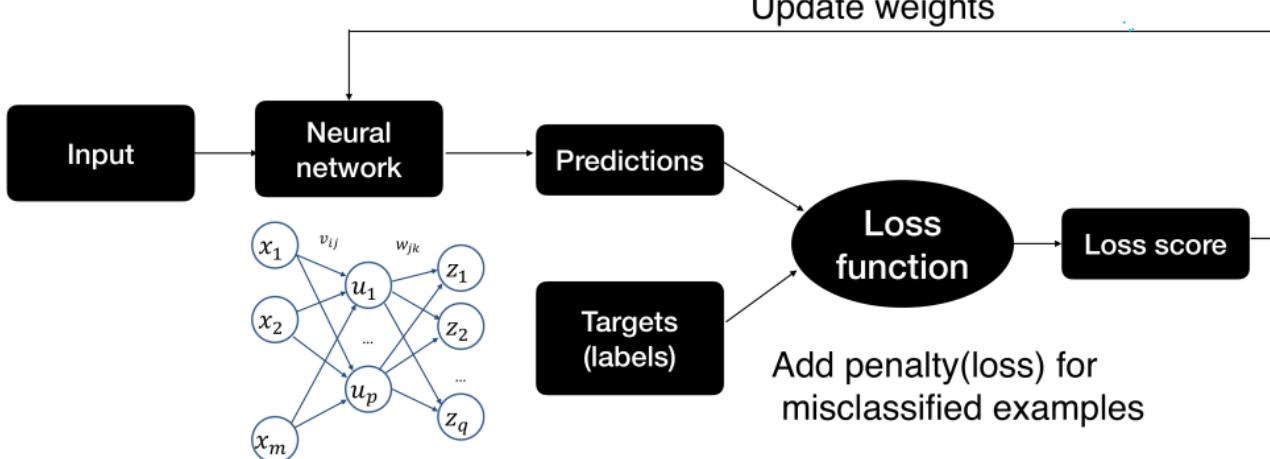
$$G = \left[ \frac{\partial z}{\partial x}, \frac{\partial z}{\partial y} \right]$$

$$(2x, 2y)$$

Move input  $(x, y)$  along  
OPPOSITE direction of  
gradient vector:  
decrease output  $f(x, y)$

*partial derivatives*

$$z = f(x, y) = x^2 + y^2$$

**Loss = function (weights)**

**Loss = function (weights)**

only parameter is weights

To reduce loss, update weights:

$$w_i^{new} = w_i^{old} - \eta * \Delta L(w_i)$$

$$\Delta L(w_i) = \frac{\partial L}{\partial w_i}$$

(*i*-th weight)

$\eta$  : learning rate

opposite direction

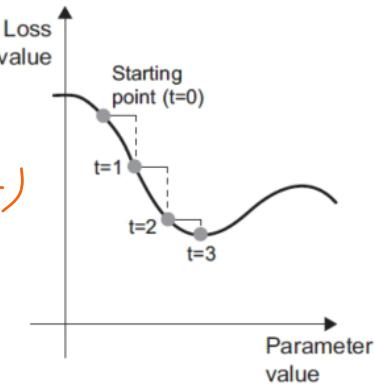


Figure 2.11 in Deep learning with python by Francois Chollet

23

**Loss = function (weights)**

$\eta$  : learning rate

more very slow

Small  $\eta$ : local optimal value

Large  $\eta$ : random location

more very fast  
(may miss the global mini)

Loss value

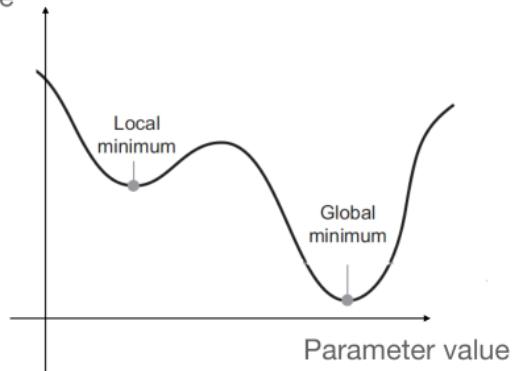


Figure 2.13 in Deep learning with python by Francois Chollet

24

## Gradient descent Algorithm

- Randomly shuffle/split all training examples in  $B$  batches
- Choose initial  $\theta^{(1)}$
- For  $i$  from 1 to  $T$
- For  $j$  from 1 to  $B$
- Do gradient descent update using data from batch  $j$   
(one batch)
- Advantage of such an approach: computational feasibility for large datasets

Iterations over the entire dataset are called epochs



25

## Stochastic gradient descent for perceptron

Choose initial guess  $w^{(0)}$ ,  $k = 0$

For  $i$  from 1 to  $T$  (epochs)

    For  $j$  from 1 to  $N$  (training examples)

        Consider example  $\{x_j, y_j\}$

        Update\*:  $w^{(k++)} = w^{(k)} - \eta \nabla L(w^{(k)})$

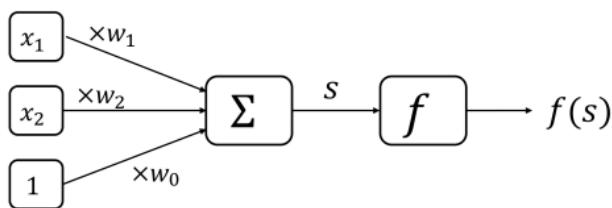
SGD

$B=N$

26

## Simple example: Perceptron Model

Encode class: A: +1, B: -1



*if  $s \geq 0$  : prediction  $f(s) = 1$*

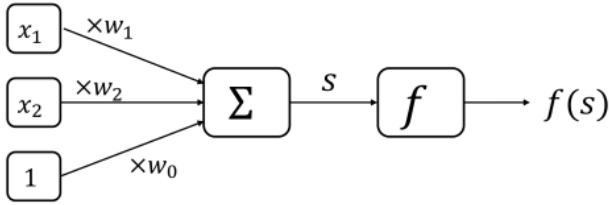
*if  $s < 0$  : prediction  $f(s) = -1$*

where  $s = \sum_{i=0}^m x_i w_i$

$x_1$	$x_2$	$y$
0	0	-1
0	1	-1
1	0	-1
1	1	1

*27*  $\begin{cases} y > 0 & s > 0 \\ y \leq 0 & s \leq 0 \end{cases}$  come it  $\rightarrow$   $y \cdot s > 0$

## Simple example: Loss



$$L(s, y) = \max(0, -ys)$$

$$= \max(0, -y \cdot \sum_{i=0}^m x_i w_i)$$

$x_1$	$x_2$	$y$
0	0	-1
0	1	-1
1	0	-1
1	1	1

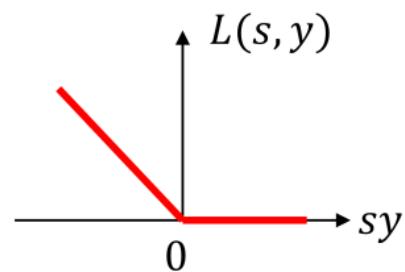
## Simple example: gradient

$$L(w) = \max(0, -ys) = \max(0, -y \sum_{i=0}^m x_i w_i)$$

$$\frac{\partial L}{\partial w_i} = -yx_i \text{ when } sy < 0$$

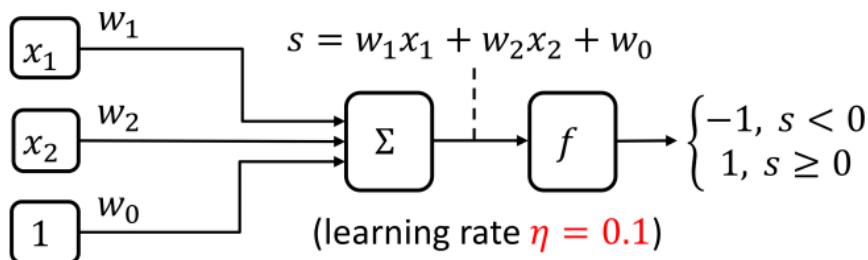
$$\frac{\partial L}{\partial w_i} = 0 \text{ when } sy > 0$$

no update if correctly classified



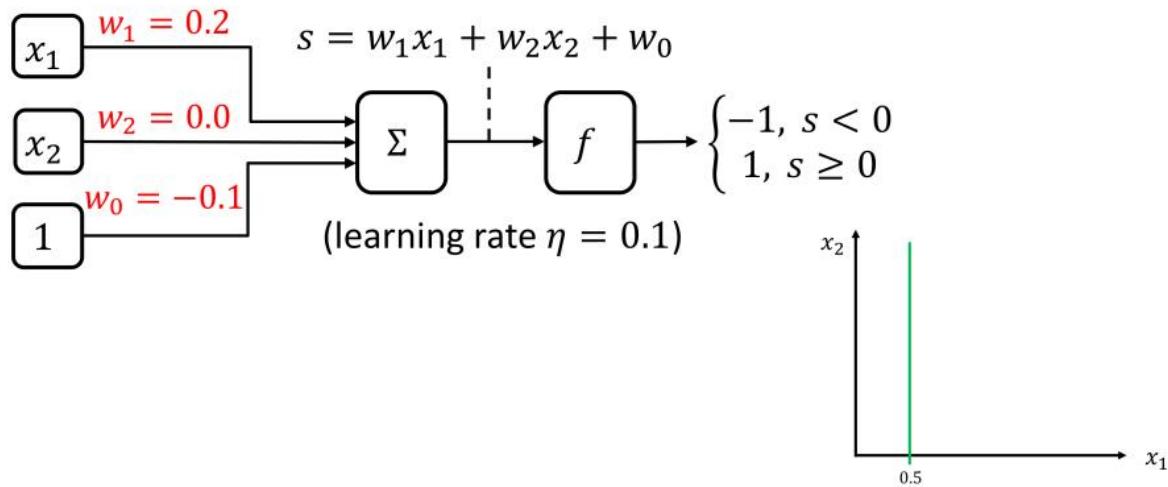
29

## Learn perceptron on the simple example: basic setup



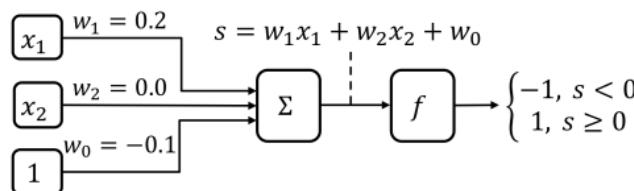
30

### Learn perceptron on the simple example: Start with random weights



31

### Learn perceptron on the simple example: epoch 1, data point 1



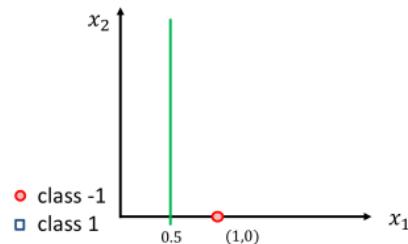
$x_1$	$x_2$	$y$
0	0	-1
0	1	-1
1	0	-1
1	1	1

Prediction  $s$  on  $(1,0)$ :

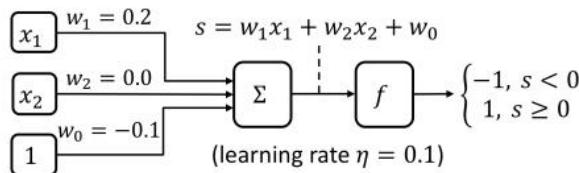
$$w_1 \times 1 + w_2 \times 0 + w_0 \times 1 = \underline{0.1} > 0$$

misclassify

32



### Learn perceptron on the simple example: epoch 1, data point 1



$x_1$	$x_2$	$y$
0	0	-1
0	1	-1
1	0	-1
1	1	1

Gradient on (1,0):  $x_0 = 1, x_1 = 1, x_2 = 0$

$$s > 0, y = -1, sy < 0: \frac{\partial L}{\partial w_i} = -yx_i \quad \left. \frac{\partial L}{\partial w_i} \right|_{i=1} = -1$$

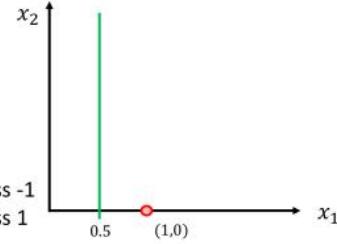
$$w_1 \leftarrow w_1 - \eta \cdot 1 = 0.1$$

$$w_2 \leftarrow w_2 - \eta \cdot 0 = 0$$

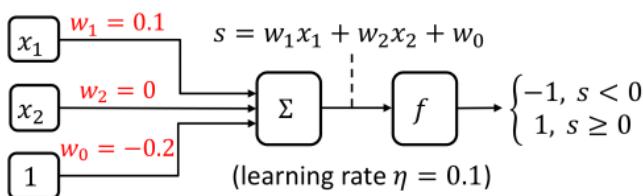
$$w_0 \leftarrow w_0 - \eta \cdot 1 = -0.2$$

$$\left. \frac{\partial L}{\partial w_i} \right|_{i=2} = 1$$

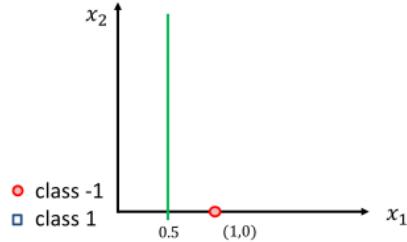
$$\left. \frac{\partial L}{\partial w_i} \right|_{i=3} = 1$$



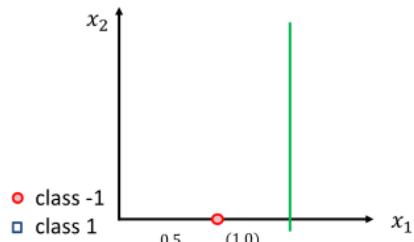
### Learn perceptron on the simple example: update weights



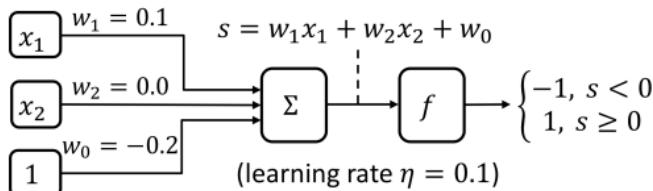
$x_1$	$x_2$	$y$
0	0	-1
0	1	-1
1	0	-1
1	1	1



34



### Learn perceptron on the simple example: epoch 1, data point 2



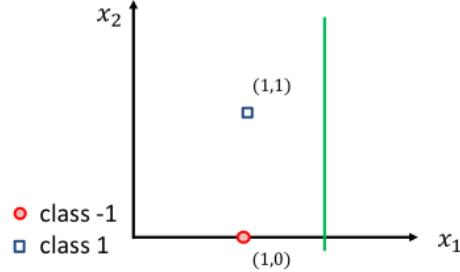
$x_1$	$x_2$	$y$
0	0	-1
0	1	-1
1	0	-1
1	1	1

Prediction  $s$  on  $(1,1)$ :

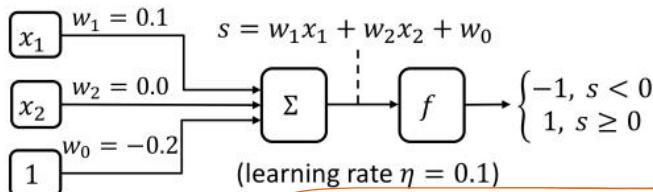
$$w_1 \times 1 + w_2 \times 1 + w_0 \times 1 = -0.1 < 0$$

*misclassified*

35



### Learn perceptron on the simple example: epoch 1, data point 2



$x_1$	$x_2$	$y$
0	0	-1
0	1	-1
1	0	-1
1	1	1

Gradient on  $(1,1)$ :

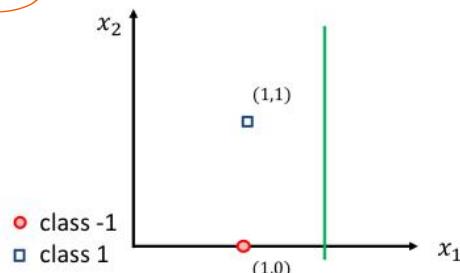
$$s < 0, y=1, sy < 0: \frac{\partial L}{\partial w_i} = -yx_i$$

$$w_1 \leftarrow w_1 - \eta \cdot (-1) = 0.2$$

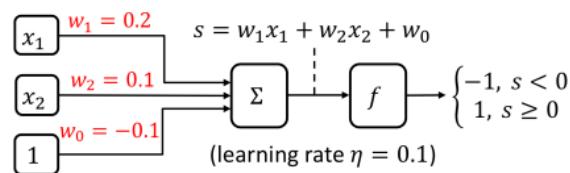
$$w_2 \leftarrow w_2 - \eta \cdot (-1) = 0.1$$

$$w_0 \leftarrow w_0 - \eta \cdot (-1) = -0.1$$

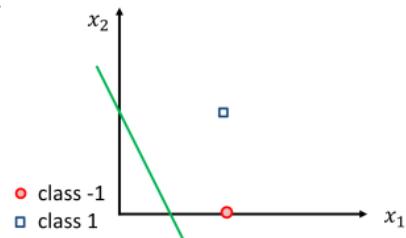
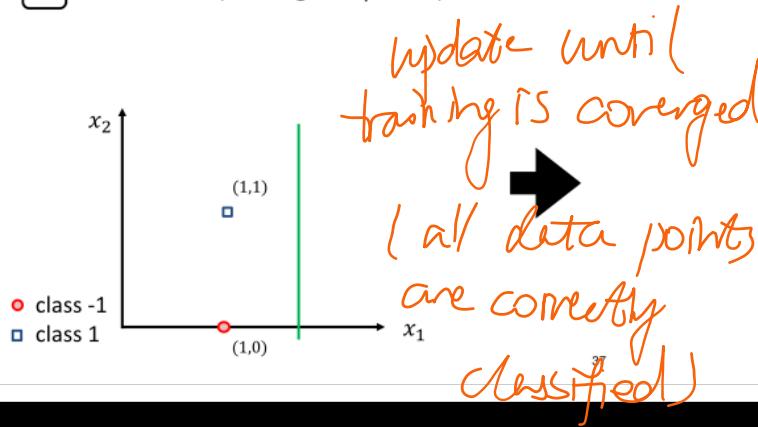
36



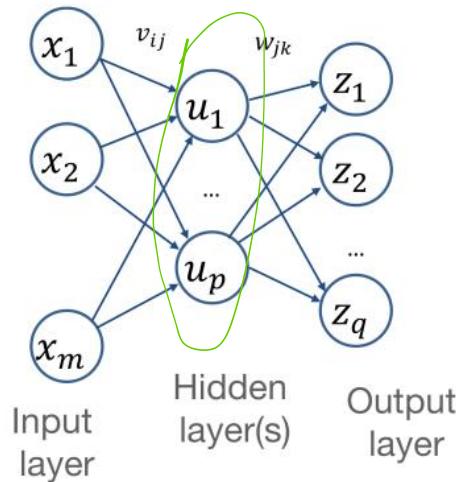
## Learn perceptron on the simple example: Update weights



$x_1$	$x_2$	$y$
0	0	-1
0	1	-1
1	0	-1
1	1	1



## Stochastic gradient descent for multi-layer perceptron



## Stochastic gradient descent for multiple-layer perceptron

Choose initial guess  $\theta^{(0)}$ ,  $k = 0$

Here  $\theta$  is a set of all weights from all layers

For  $i$  from 1 to  $T$  (epochs)

For  $j$  from 1 to  $N$  (training examples)

Consider example  $\{x_j, y_j\}$

Update:  $\theta^{(k+1)} = \theta^{(k)} - \eta \nabla L(\theta^{(k)})$ ;  $k \leftarrow k+1$

Need to compute partial derivatives  $\frac{\partial L}{\partial v_{ij}}$  and  $\frac{\partial L}{\partial w_j}$

39

## Chain rule

Given  $z = g(u)$     $u = f(x)$



$$\frac{dz}{dx} = \frac{dz}{du} \frac{du}{dx}$$

Example :  $z = \sin(x^2)$

$$\begin{array}{l} z = \sin(u) \\ \uparrow \\ u = x^2 \end{array}$$

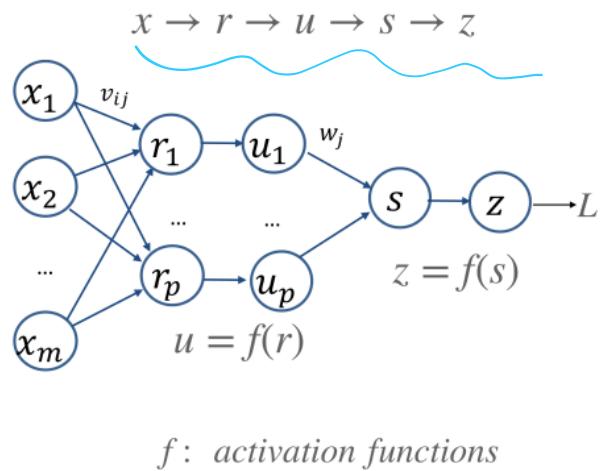
$$\begin{aligned} \frac{dz}{du} &= \cos(u) \\ \downarrow \\ \frac{dz}{dx} &= \frac{dz}{du} \frac{du}{dx} = 2x \cos(u) \end{aligned}$$

40

## Multi-layer perceptron: Function composition

Forward prediction

$$\begin{aligned} z &= f(s) = \text{sigmoid}(s) = \frac{1}{1 + e^{-s}} \\ s &= \sum_{j=0}^p w_j u_j \\ u_j &= f(r_j) = \text{sigmoid}(r_j) = \frac{1}{1 + e^{-r_j}} \\ r_j &= \sum_{i=0}^m x_i v_{ij} \end{aligned}$$

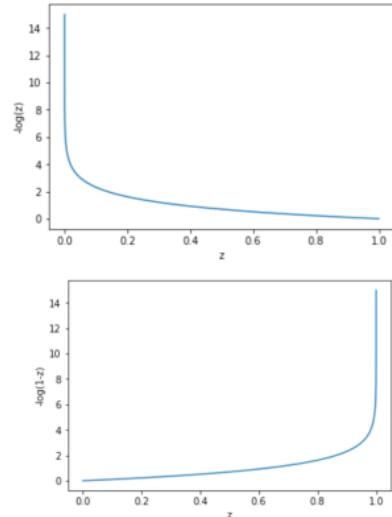


41

## Binary cross-entropy

$$L = -[y \log(z) + (1 - y) \log(1 - z)]$$

y: Labels of the data  
y=1 for positive class or 0 for negative class



42

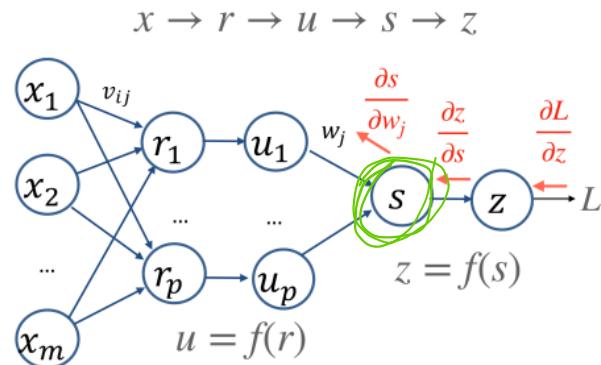
## Multi-layer perceptron: Chain rule

Backward propagation:

$$\frac{\partial L}{\partial w_j} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial s} \frac{\partial s}{\partial w_j}$$

$$\begin{aligned} L &= -[y \log(z) + (1-y)\log(1-z)] \\ z &= \text{sigmoid}(s) = \frac{1}{1+e^{-s}} \\ s &= \sum_{j=0}^p u_j w_j \end{aligned}$$

Forward



43

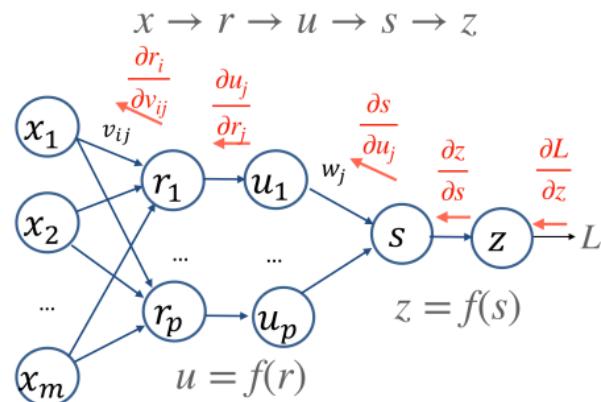
## Multi-layer perceptron: Chain rule

Backward propagation:

$$\frac{\partial L}{\partial v_{ij}} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial s} \frac{\partial s}{\partial u_j} \frac{\partial u_j}{\partial r_j} \frac{\partial r_j}{\partial v_{ij}}$$

$$\begin{aligned} s &= \sum_{j=0}^p u_j w_j \\ u_j &= \text{sigmoid}(r_j) = \frac{1}{1+e^{-r_j}} \\ r_j &= \sum_{i=0}^m x_i v_{ij} \end{aligned}$$

Forward



44

## Forward prediction

$$L = -[y \log(z) + (1 - y) \log(1 - z)]$$

↑

$$z = \text{sigmoid}(s) = \frac{1}{1 + e^{-s}}$$

$$s = \sum_{j=0}^p u_j w_j$$

$$u_j = \text{sigmoid}(r_j) = \frac{1}{1 + e^{-r_j}}$$

$$r_j = \sum_{i=0}^m x_i v_{ij}$$

## Backward propagation

$$\frac{\partial L}{\partial z} = -\frac{y}{z} + \frac{1-y}{1-z}$$

$$\frac{\partial L}{\partial s} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial s} = z - y, \quad \frac{\partial z}{\partial s} = z(1-z)$$

$$\frac{\partial L}{\partial w_j} = \frac{\partial L}{\partial s} \frac{\partial s}{\partial w_j} = \frac{\partial L}{\partial s} u_j \quad \frac{\partial L}{\partial u_j} = \frac{\partial L}{\partial s} \frac{\partial s}{\partial u_j} = \frac{\partial L}{\partial s} w_j$$

$$\frac{\partial L}{\partial r_j} = \frac{\partial L}{\partial u_j} \frac{\partial u_j}{\partial r_j} = \frac{\partial L}{\partial u_j} (u_j)(1-u_j)$$

$$\frac{\partial L}{\partial v_{ij}} = \frac{\partial L}{\partial r_j} \frac{\partial r_j}{\partial v_{ij}} = \frac{\partial L}{\partial r_j} x_i$$

45

## Summary

- What is deep learning & difference with traditional machine learning?
- How to train neural network using gradient descent algorithm?
- How to train a perceptron using stochastic gradient descent?
- What is chain rule?
- How to perform forward prediction and back propagation in multilayer perceptron?

Next: Convolutional neural network

46

# References

- Deep learning with python by Francois Chollet, chapter 1 & 2.4
- Pattern Recognition and Machine Learning by Chris Bishop, chapter 4.1.7, 5.1-5.3.



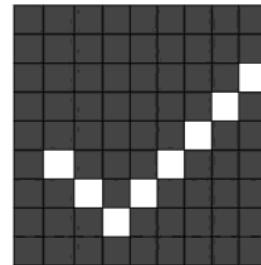
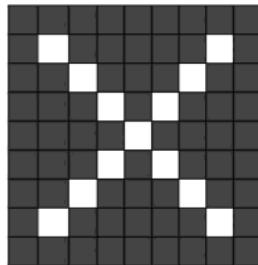
# Convolutional Neural Networks

COMP90051 Statistical Machine Learning

Qiuhong Ke

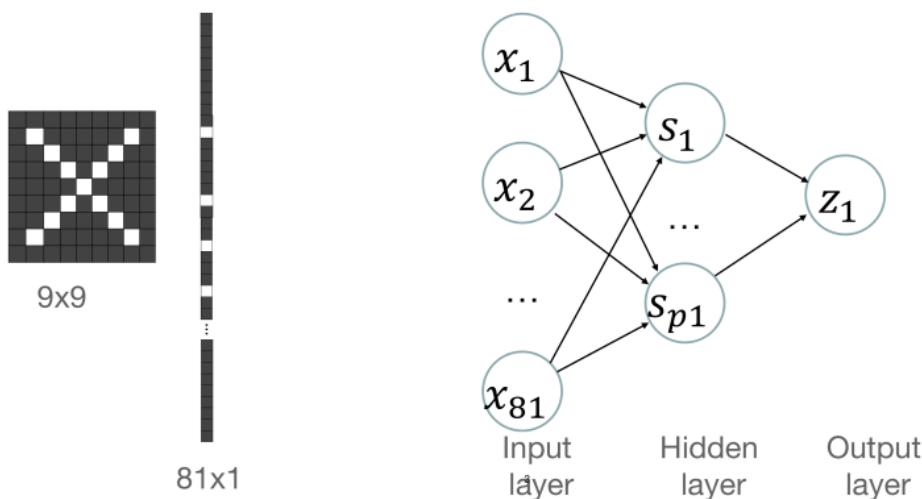
Copyright: University of Melbourne

✗ vs ✓



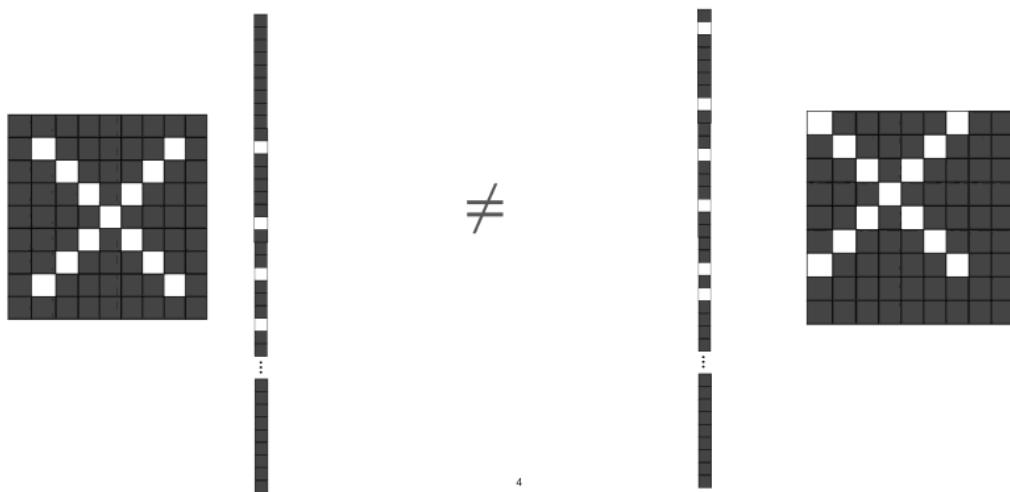
## Multi-layer perceptron: A fully connected network

Consists of only fully connected (FC) layers



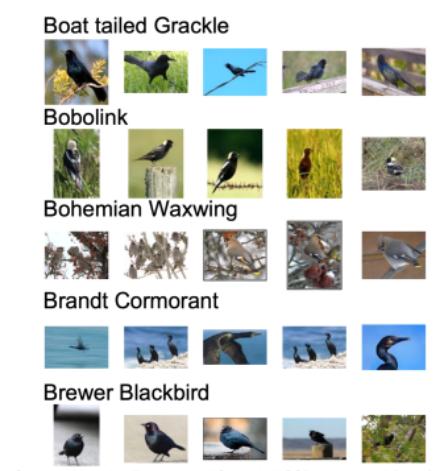
## Multi-layer perceptron: A fully connected network

Disadvantage: Not spatial invariant

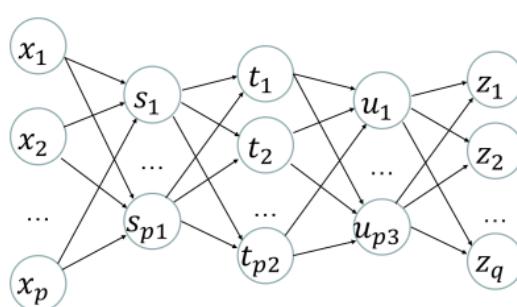


## Multi-layer perceptron: A fully connected network

Disadvantage: more parameters with more hidden layers



Source: Welinder, Peter, et al. "Caltech-UCSD birds 200." (2010).

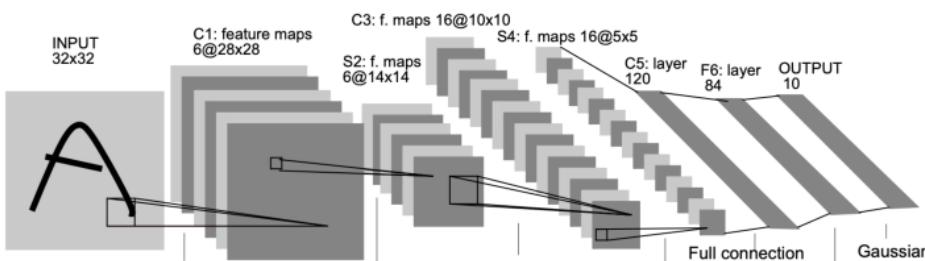


Suffer from overfitting problem

5

# Convolutional Neural Network (CNN)

## Convolution, Max-Pooling, and Fully Connected (FC) layers



LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

6

## Outline

- Convolutional layer
- Max-Pooling layer
- Additional notes in training neural network
  - Batch size
  - Optimisation algorithms
  - Activation function
  - How to prevent overfitting

7

## Tool: Keras

**Easy, simple and powerful**

- Build the architecture (add layers from input to output. eg. FC layer, convolution layer...)

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

8

# Tool: Keras

## Easy, simple and powerful

- Select an optimisation algorithm (eg. SGD, more in this lecture)
- Select the loss function
- Compile the model and train the model

```
opt = tf.keras.optimizers.SGD(learning_rate=0.1)
model.compile(loss="categorical_crossentropy",
              optimizer=opt, metrics=[ "accuracy"])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs)
```

9

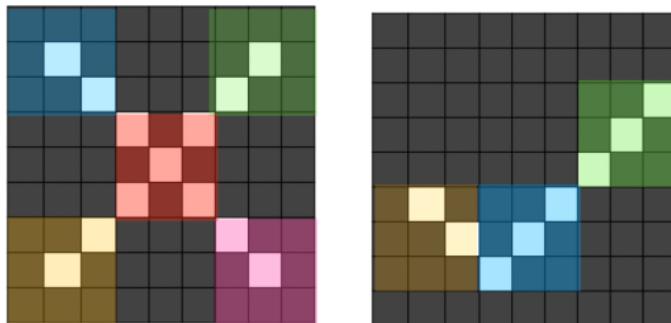
## Convolutional layer

## Max-pooling layer

## Additional Training notes

### An image can be decomposed into local patches

- Different local patches could have different patterns
- To do classification, we can first extract local features(: Identify local patterns) and then combine the local features for classification



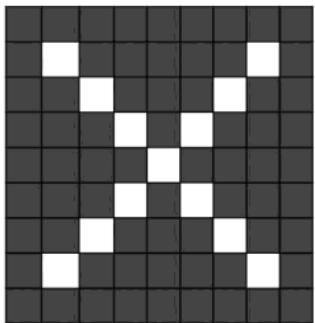
10

## Convolutional layer

## Max-pooling layer

## Additional Training notes

### Identify different patterns at local patches



0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0	0
0	0	0	1	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0
0	0	1	0	0	0	1	0	0	0
0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0

11

## Convolutional layer

### Identify different patterns

0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0
0	0	1	0	0	0	1	0	0
0	0	0	1	0	1	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0
0	0	0	1	0	1	0	0	0
0	0	1	0	0	0	1	0	0
0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0

## Max-pooling layer

## Additional Training notes

Element-wise multiplication

$$\text{Sum} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 2$$

$$\text{Sum} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 1$$

Filter (kernel)

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}_{12}$$

weight matrix

element-wise multi

## Convolutional layer

## Max-pooling layer

## Additional Training notes

### Input and kernel have the same pattern: high response

0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0
0	0	1	0	0	0	1	0	0
0	0	0	1	0	1	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0
0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0

Element-wise multiplication

$$\text{Sum} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = 1$$

$$\text{Sum} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = 2$$

Filter (kernel)

13

## Convolutional layer

## Max-pooling layer

## Additional Training notes

### Identify different patterns

0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0
0	0	1	0	0	0	1	0	0
0	0	0	1	0	1	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0
0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0

Element-wise multiplication

$$\text{Sum} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 2$$

$$\text{Sum} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 2$$

Filter (kernel)



14

## Convolutional layer

## Max-pooling layer

## Additional Training notes

### Different kernels identify different patterns

0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0
0	0	1	0	0	0	1	0	0
0	0	0	1	0	1	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0
0	0	0	1	0	1	0	0	0
0	0	1	0	0	0	1	0	0
0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0

Element-wise multiplication

Sum (  $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$  )  $\times$   $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$  ) = 2

Sum (  $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$  )  $\times$   $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$  ) = 5

Filter (kernel)

Separate

15

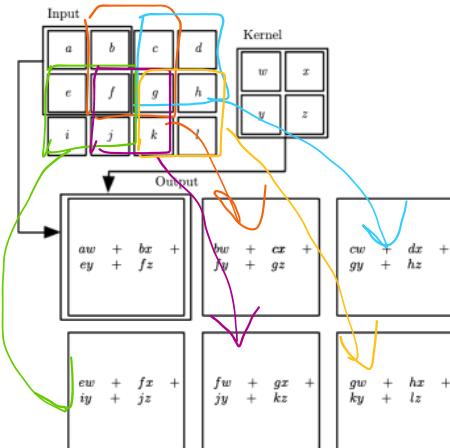
## Convolutional layer

## Max-pooling layer

## Additional Training notes

### Convolution on 2D

Use kernel to perform element-wise multiplication and sum for every local patch



16

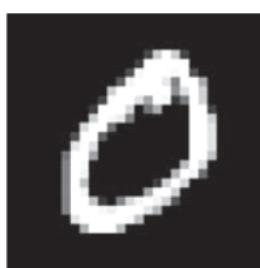
Figure 9.1 in Deep learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville

## Convolutional layer

## Max-pooling layer

## Additional Training notes

### Response map (Feature map)



Input



kernel



Feature map: 2D map of the presence of a pattern at different locations in an input

Same pattern  $\rightarrow$  response higher  
different pattern  $\rightarrow$  response lower

Figure 5.3 in Deep learning with python by Francois Chollet

17

## Convolutional layer

## Max-pooling layer

## Additional Training notes

Different kernels identify different patterns: use multiple filters in each layer

0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0	0
0	0	0	1	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0
0	0	1	0	0	0	1	0	0	0
0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0

1	0	0
0	1	0
0	0	1

0	1	0
1	0	1
0	0	0

0	0	1
0	1	0
1	0	0

The number of filters decides the number of output feature maps

18

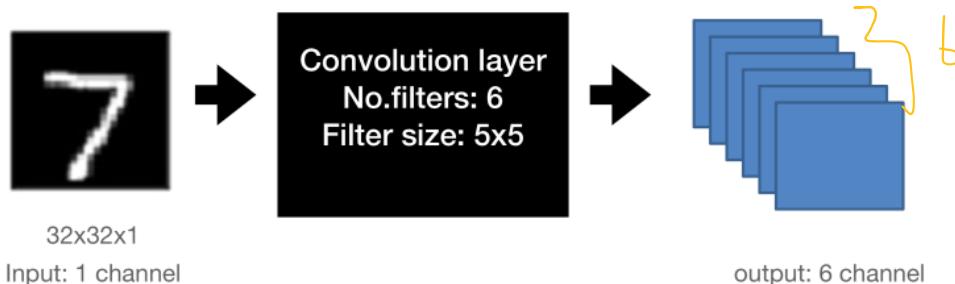
## Convolutional layer

## Max-pooling layer

## Additional Training notes

### Two key parameters in Convolution

Filter (kernel) size: Size of the patches extracted from the inputs  
Number of filters: Depth (channel) of the output feature map



RGB  $\rightarrow$  3 channels

19

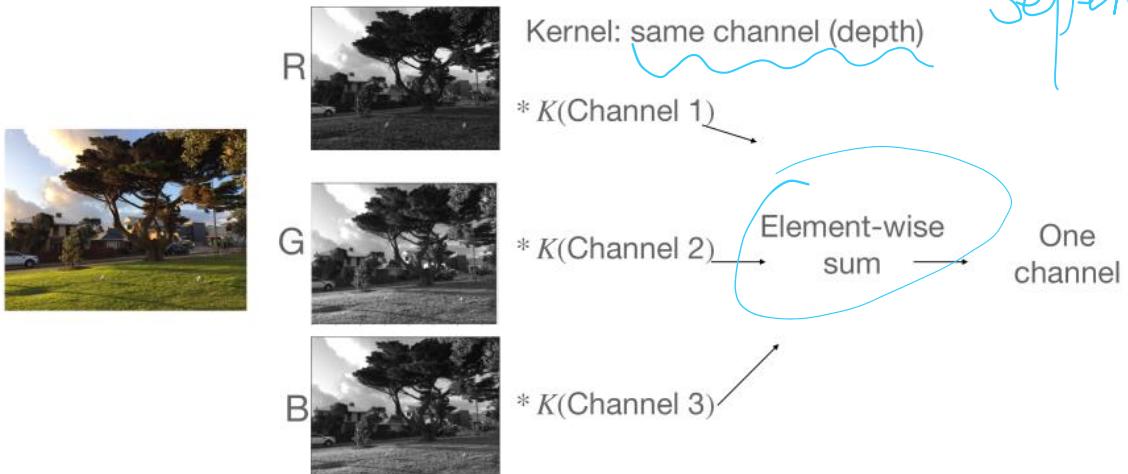
## Convolutional layer

## Max-pooling layer

## Additional Training notes

operate  
Separately

### Convolution on Multiple-channel input

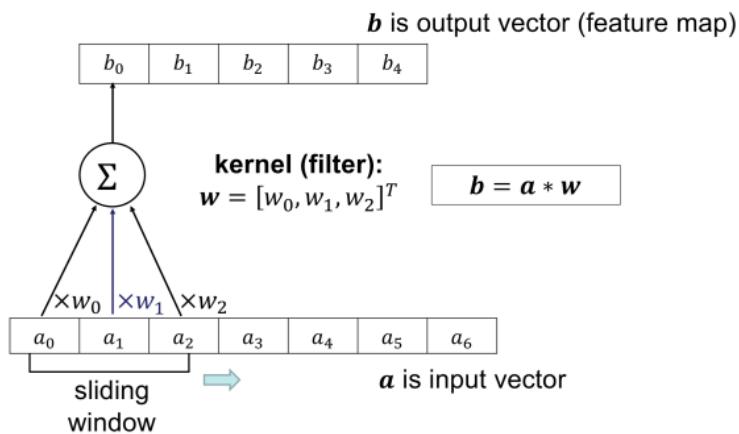


## Convolutional layer

## Max-pooling layer

## Additional Training notes

### Convolution on 1D



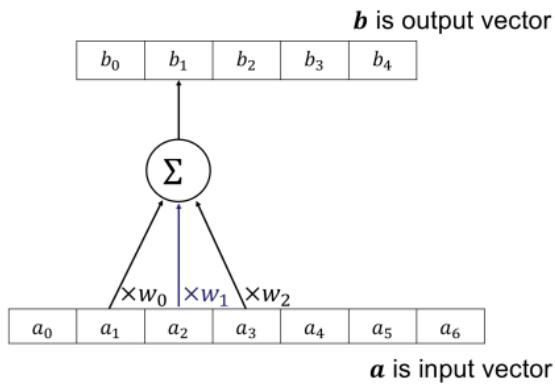
21

## Convolutional layer

## Max-pooling layer

## Additional Training notes

### Convolution on 1D



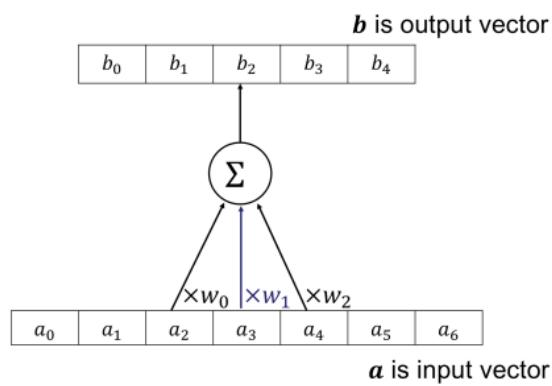
22

## Convolutional layer

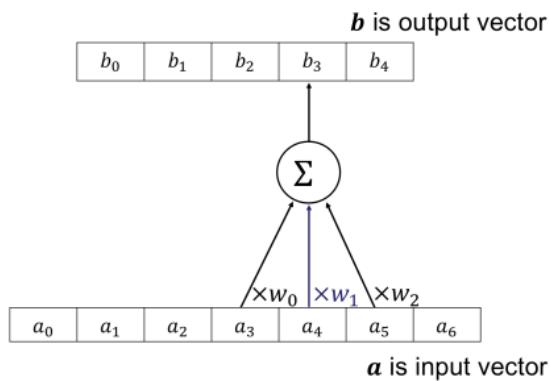
## Max-pooling layer

## Additional Training notes

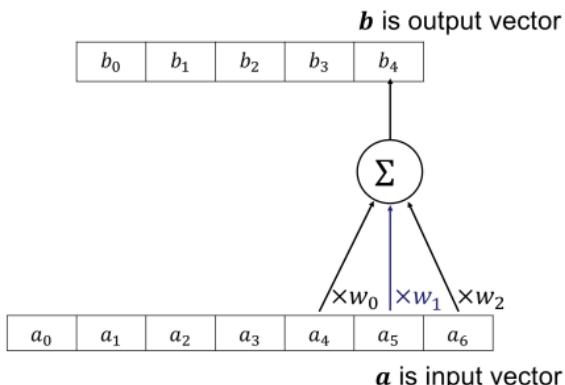
### Convolution on 1D



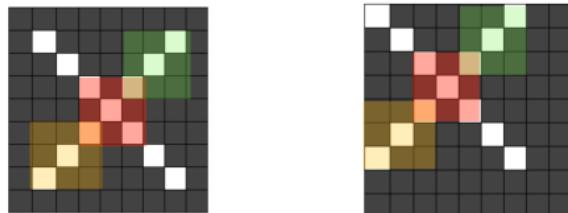
23

**Convolution on 1D**

24

**Convolution on 1D**

25

**Advantage:** learn translation-invariant pattern

26

## Convolutional layer

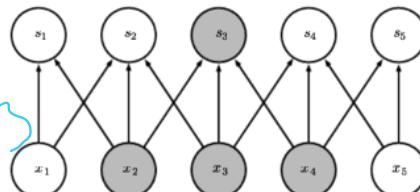
## Max-pooling layer

## Additional Training notes

**Advantage: weight sharing and sparse connection**

Convolutional layer:

*Shared weights (kernel)*



Fully connected layer:

Each arrow is a weight (no sharing)

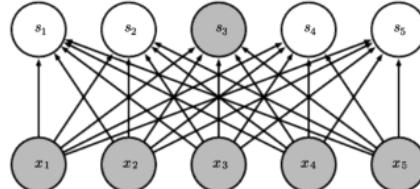


Figure 9.3 in Deep learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville

27

## Convolutional layer

## Max-pooling layer

## Additional Training notes

**Advantage: learn hierarchical pattern**

More layers: larger size of receptive field  
(larger window of the input is seen)

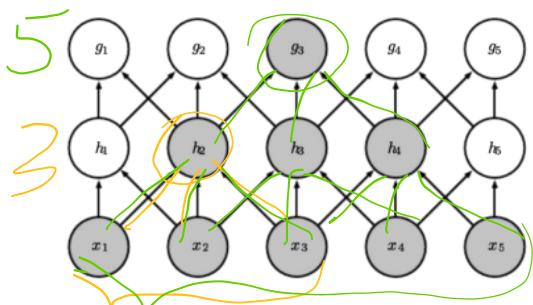


Figure 9.4 in Deep learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville

28

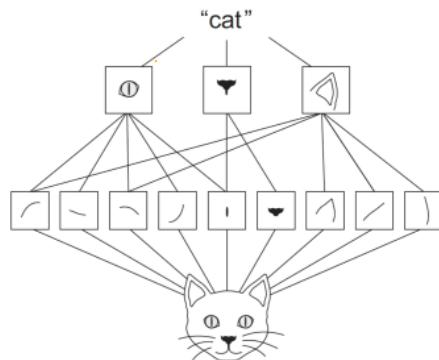


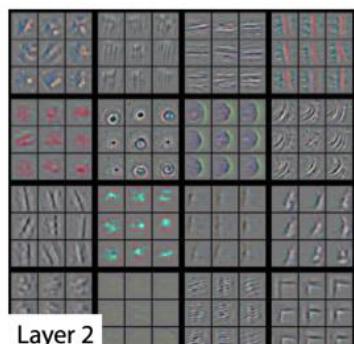
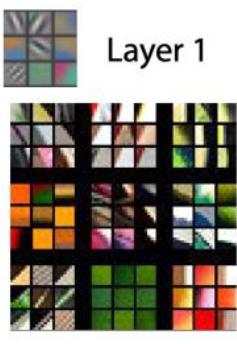
Figure 5.2 in Deep learning with python by Francois Chollet

1.

## Convolutional layer

## Max-pooling layer

## Additional Training notes



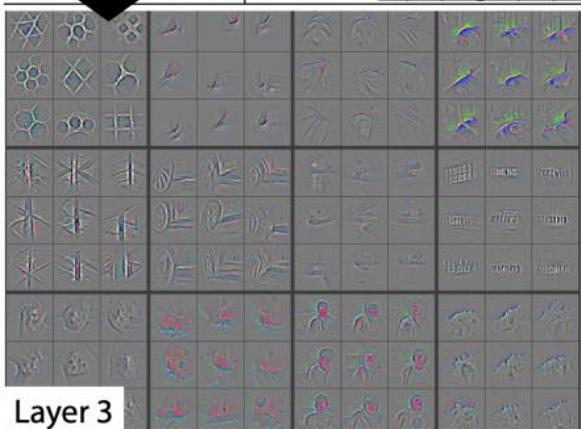
Zisser, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." European conference on computer vision. Springer, Cham, 2014.

29

## Convolutional layer

## Max-pooling layer

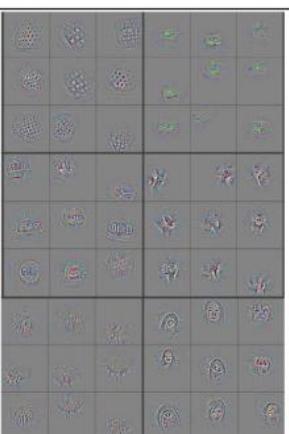
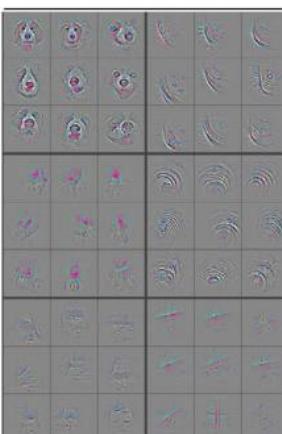
## Additional Training notes



## Convolutional layer

## Max-pooling layer

## Additional Training notes

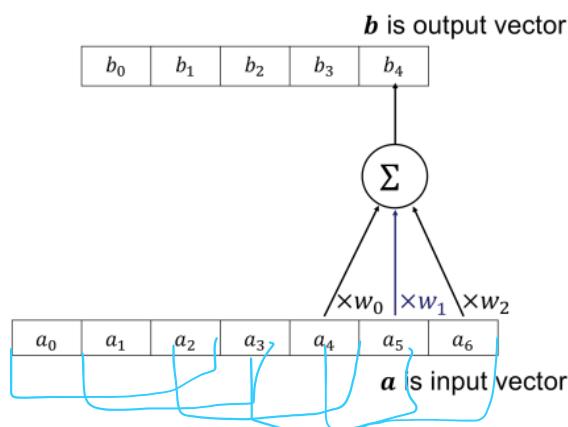


## Convolutional layer

## Max-pooling layer

## Additional Training notes

**output size  $\neq$  input size**



## Convolutional layer

## Max-pooling layer

## Additional Training notes

## Convolutional layer

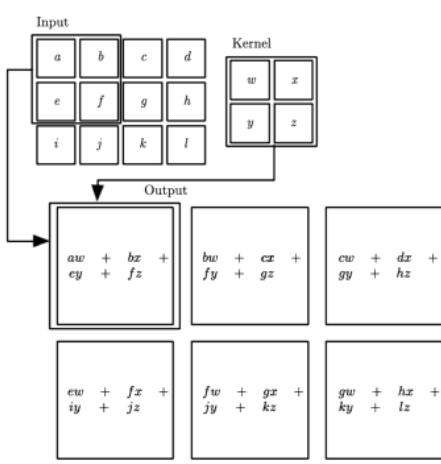
**output size  $\neq$  input size**

$3 \times 4$

## Max-pooling layer

## Additional Training notes

UFTY  
Local patches



33

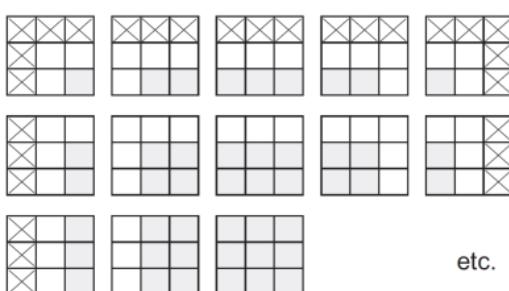
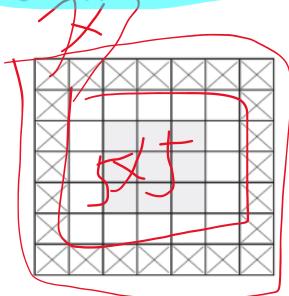
Figure 9.1 in Deep learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville

## Convolutional layer

**Padding**

$\rightarrow$  make sure size of output = size of input

adding an appropriate number of rows and columns on each side of the input feature map



## Additional Training notes

padding

$7 \times 7 \xrightarrow{K=3 \times 3} 3 \times 3$

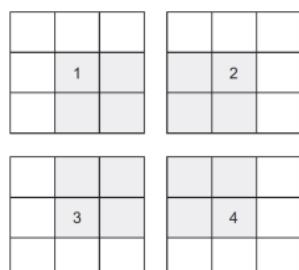
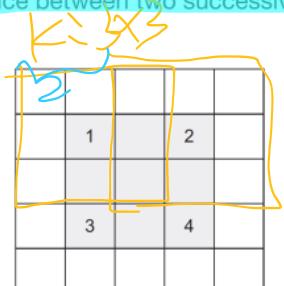
Figure 5.6 in Deep learning with python by Francois Chollet

34

## Convolutional layer

**Stride**

the distance between two successive windows



## Additional Training notes

Stride  $\Rightarrow$  output size smaller

No padding:  $\text{output\_size} = \text{ceiling}((\text{input\_size} - \text{kernel\_size} + 1) / \text{stride})$

padding:  $\text{output\_size} = \text{ceiling}(\text{input\_size} / \text{stride})$

## Convolutional layer

## Max-pooling layer

## Additional Training notes

## Convolutional layer

## Max-pooling layer

## Additional Training notes

### Convolttional layer

filters: the number of filters in the convolution  
kernel\_size: the height and width of the 2D convolution window  
padding: one of "valid" or "same"  
stride: the strides of the convolution along the height and width

```
layer=keras.layers.Conv2D(  
    32,  
    (3, 3),  
    padding='same',  
    strides=1,  
    activation='relu')
```

36

## Convolutional layer

## Max-pooling layer

## Additional Training notes

0.1	0.2	0.5	0.8
0.7	0.3	0.4	0.2
1	0.2	0.3	0.5
0.6	0.9	0.1	0.2

$2 \times L$   
stride : 2

37

## Convolutional layer

## Max-pooling layer

## Additional Training notes

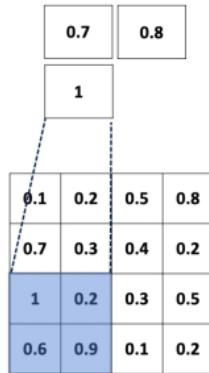
0.1	0.2	0.5	0.8
0.7	0.3	0.4	0.2
1	0.2	0.3	0.5
0.6	0.9	0.1	0.2

38

## Convolutional layer

## Max-pooling layer

## Additional Training notes

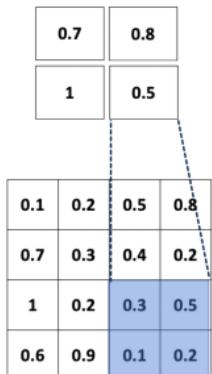


39

## Convolutional layer

## Max-pooling layer

## Additional Training notes



```
tf.keras.layers.MaxPooling2D(  
    pool_size=(2, 2), strides=None, padding="valid")
```

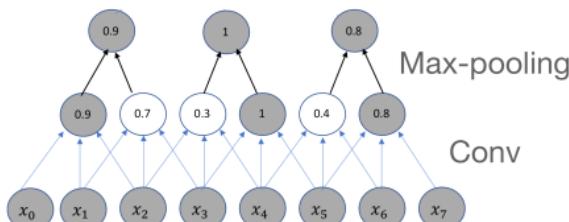
→ 0 padding

## Convolutional layer

## Max-pooling layer

## Additional Training notes

**Advantage:** downsample feature map, reduce computational burden



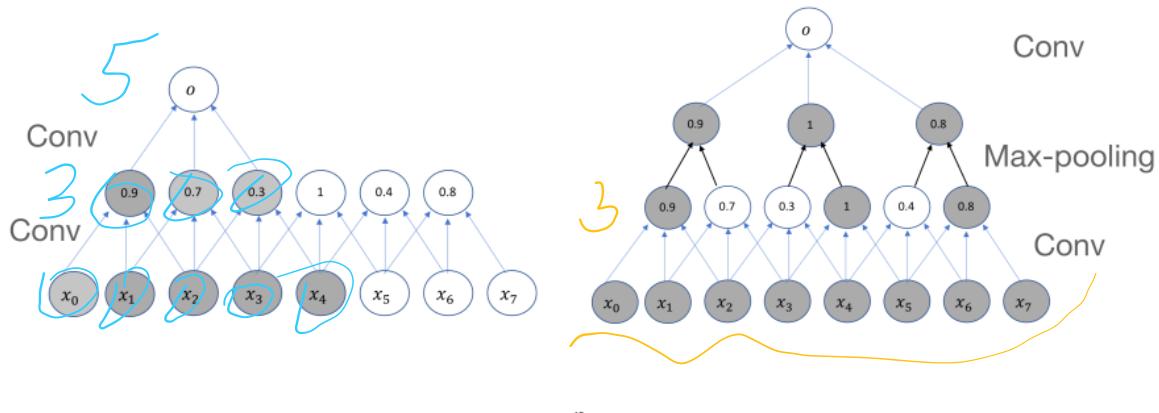
41

## Convolutional layer

## Max-pooling layer

## Additional Training notes

**Advantage:** increase size of receptive field (window of the input is seen)



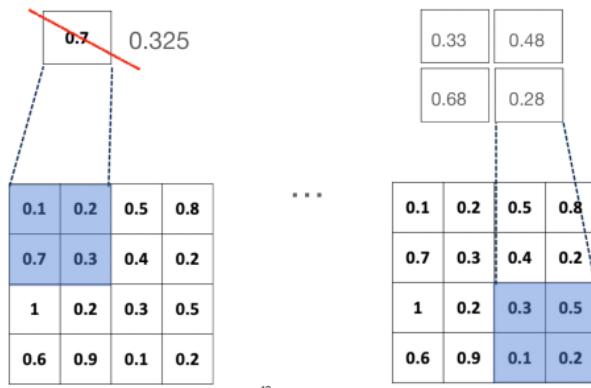
## Convolutional layer

## Max-pooling layer

## Additional Training notes

**Other pooling method: Average pooling**

taking the average value over the patch



## Convolutional layer

## Max-pooling layer

## Additional Training notes

**Why max-pooling to downsample feature map?**

Average pooling: dilute feature-presence information

Convolution with stride>1: miss feature-presence information

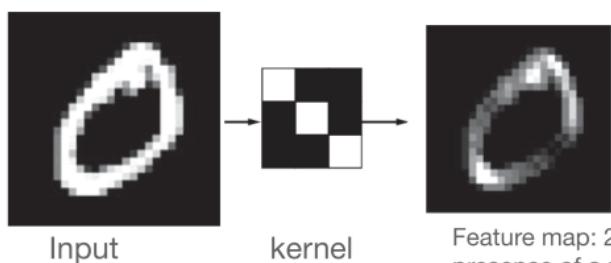


Figure 5.3 in Deep learning with python by Francois Chollet

## Outline

- Batch size
- Other optimisation methods (optimiser)
  - Momentum
  - Adaptive gradient (AdaGrad)
  - Root Mean Square Propagation (Rmsprop)
  - Adaptive moment estimation (Adam)
- Activation function
- How to prevent overfitting

45

## Gradient descent Algorithm

- Randomly shuffle/split all training examples in  $B$  batches
- Choose initial  $\theta^{(1)}$
- For  $i$  from 1 to  $T$ 
  - For  $j$  from 1 to  $B$
  - Do gradient descent update using data from batch  $j$

Iterations over the entire dataset are called *epochs*

46

## Stochastic gradient descent: $B=1$

Consider only one example

Choose initial guess  $\theta^{(0)}$ ,  $k = 0$

Here  $\theta$  is a set of all weights form all layers

For  $i$  from 1 to  $T$  (epochs)

For  $j$  from 1 to  $N$  (training examples)

Consider example  $\{x_j, y_j\}$

Update:  $\theta^{(k+1)} = \theta^{(k)} - \eta \nabla L(\theta^{(k)})$ ;  $k \leftarrow k+1$

47

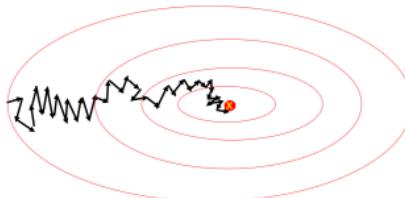
**Stochastic gradient descent (SGD)**

Batch number==N (Batch size==1)

Quick update each step, but

- high variance in gradient
- update model too often

*error surface*

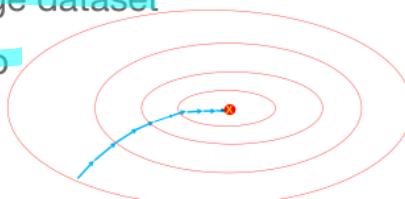


48

**Batch SGD: Batch number==1 (Batch size==N)**

Stable update, but

- not computational feasible for large dataset
- takes long time to move each step



49

**Mini-batch SGD**

mini-batch:  $1 < \text{batch size} < N$

- compared to SGD (batch size==1), more stable update
- compared to batch SGD (batch size==N), more computational feasible



50

**Gradient descent: carefully choose learning rate**

To reduce loss, update weights:

$$w_{t+1} = w_t - \eta * \Delta L(w_t)$$

$$\Delta L(w_t) = \frac{\partial L}{\partial w_t}$$

$\eta$  : learning rate

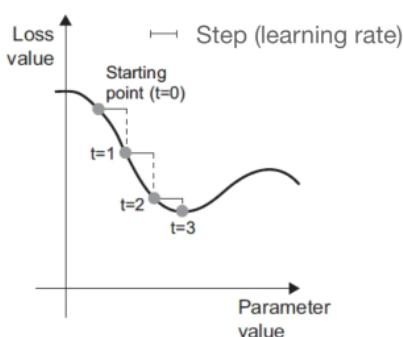


Figure 2.11 in Deep learning with python by Francois Chollet

51

**Gradient descent: carefully choose learning rate**

$\eta$  : learning rate

Large  $\eta$ : random location

Small  $\eta$ : local optimal value

$$w_{t+1} = w_t - \eta * \Delta L(w_t)$$

$\Delta L(w_t) = 0$  : no update

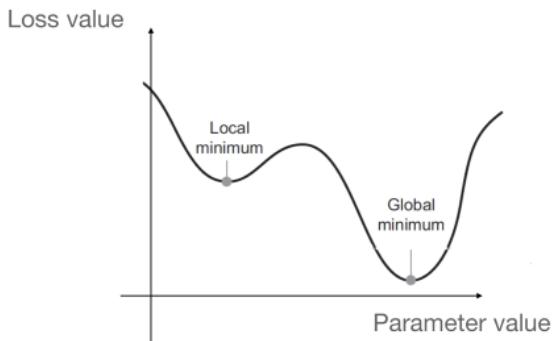


Figure 2.13 in Deep learning with python by Francois Chollet

52

**Gradient descent: carefully choose learning rate**

Rolling Ball

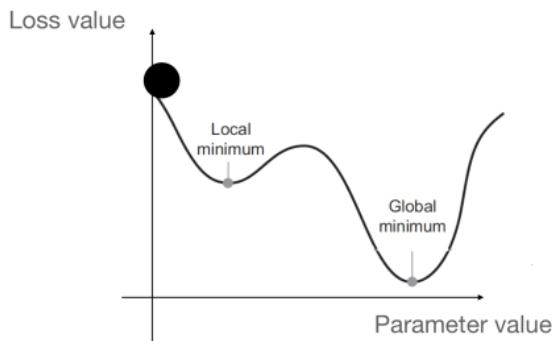
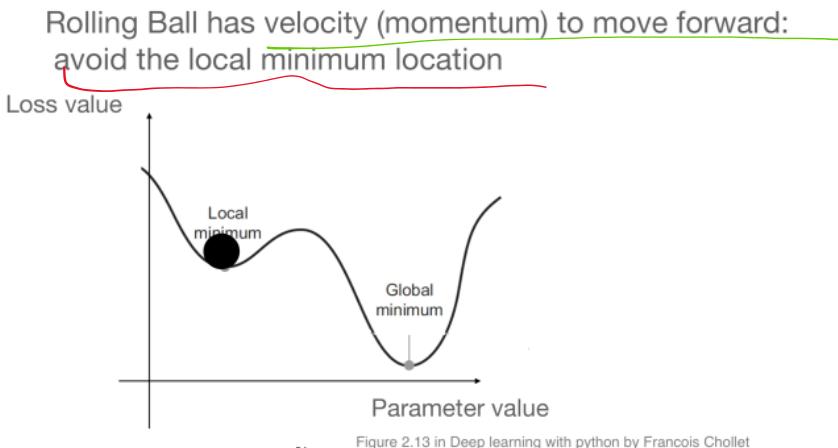


Figure 2.13 in Deep learning with python by Francois Chollet

53

**Gradient descent: carefully choose learning rate****Gradient descent: carefully choose learning rate**

Rolling Ball has velocity (momentum) to move forward:  
avoid the local minimum location

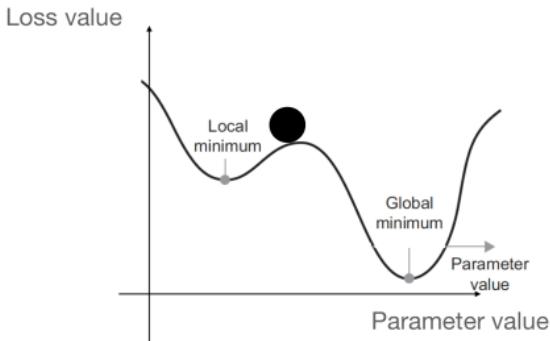


Figure 2.13 in Deep learning with python by Francois Chollet

**Update weights with momentum**

$$w_{t+1} = w_t - \eta \Delta L(w_t) + v_t$$

$$v_t = \alpha * v_{t-1} + (-\eta \Delta L(w_t))$$

$\alpha$  : momentum (decay the previous velocity, e.g. 0.9)

`opt = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9)`

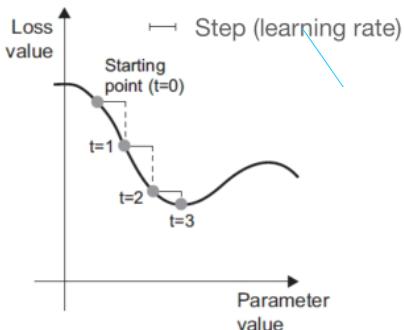


Figure 2.11 in Deep learning with python by Francois Chollet

Advantages:

- fast convergence speed
- avoid local minimum

**A single fix learning rate?**

SGD maintains a single learning rate, and does not change, but

Magnitudes of gradient

- Different for different weights
- change during training

Problem: difficult to find a single global learning rate

57

**Other optimisation algorithm: Adaptive gradient (AdaGrad)**

- Keep a accumulative sum of the squared gradient

$$r_t = r_{t-1} + (\Delta L(w_t))^2 = \sum_{i=1}^t \Delta(L(w_i))^2$$

- Divide the gradient with squared root of  $r_t$

$$w_{t+1} = w_t - \eta \frac{\Delta L(w_t)}{\sqrt{r_t + \epsilon}}$$

$\epsilon$  : A small constant for numerical stability (1e-7)

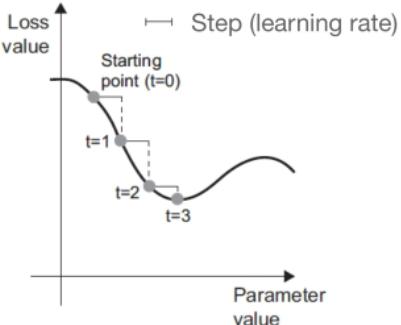


Figure 2.11 in Deep learning with python by Francois Chollet

58

**Other optimisation algorithm: Root Mean Square Propagation (Rmsprop)**

- Keep a moving average of the squared gradient

$$r_t = \rho r_{t-1} + (1 - \rho)(\Delta L(w_t))^2$$

- Divide the gradient with squared root of  $r_t$

$$w_{t+1} = w_t - \eta \frac{\Delta L(w_t)}{\sqrt{r_t + \epsilon}}$$

$\rho$  : exponential decay parameter (0.9)

$\epsilon$  : A small constant for numerical stability (1e-7)

```
opt = tf.keras.optimizers.RMSprop(learning_rate=0.001,
    rho=0.9,
    momentum=0.0,
    epsilon=1e-07)
```

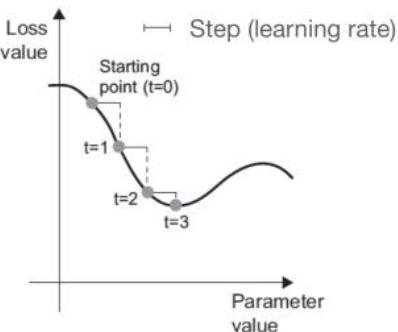


Figure 2.11 in Deep learning with python by Francois Chollet

59

## Convolutional layer

## Max-pooling layer

## Additional Training notes

### Other optimisation algorithm: Adaptive moment estimation (Adam)

- Keep a moving average of the gradient and squared gradient

$$m_t = \frac{\beta_1 m_{t-1} + (1 - \beta_1)(\Delta L(w_t))}{1 - \beta_1^{t+1}}$$

$$r_t = \frac{\beta_2 r_{t-1} + (1 - \beta_2)(\Delta L(w_t))^2}{1 - \beta_2^{t+1}}$$

Bias correction

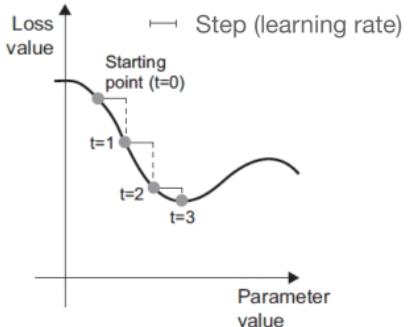


Figure 2.11 in Deep learning with python by Francois Chollet

- Divide the  $m_t$  with squared root of  $r_t$

$$w_{t+1} = w_t - \eta \frac{m_t}{\sqrt{r_t + \epsilon}}$$

$\beta_1, \beta_2$ : exponential decay parameter (0.9 and 0.999)  
 $\epsilon$ : A small constant for numerical stability (1e-7) <sup>60</sup>

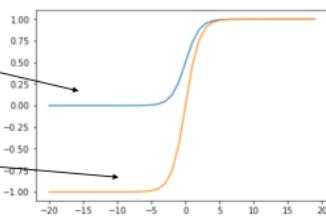
## Convolutional layer

## Max-pooling layer

## Additional Training notes

### Activation function

Sigmoid:  $f(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$



TanH  $f(x) = \frac{2}{1 + e^{-2x}} - 1 = \frac{2e^{2x}}{e^{2x} + 1} - 1$

Problem: output saturates if  $x$  is very large or very small

→ Gradient is killed (vanishing gradient problem)

*Vanishing gradient* <sup>61</sup>

## Convolutional layer

## Max-pooling layer

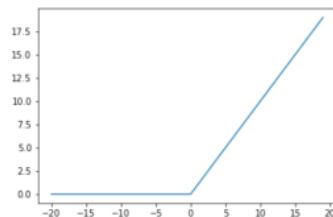
## Additional Training notes

### Activation function

Use Relu (Rectified Linear Unit) to handle vanishing gradient problem:

$$f(x) = \max(0, x)$$

Problem: no gradient for inactive unit (Dying Relu)



Use Leaky Relu to allow a small gradient for inactive unit:

$$\begin{cases} f(x) = x, & \text{if } x \geq 0 \\ f(x) = \alpha x, & \text{if } x < 0 \end{cases}$$

## Prevent overfitting

**Early stopping:** Stop the training if generalisation error (validation dataset) increases

Explicit regularisation:

- Instead of minimising the loss  $L$ , minimise regularised function  $L + \lambda \left( \sum_{i=0}^m \sum_{j=1}^p v_{ij}^2 + \sum_{j=0}^p w_j^2 \right)$
- This will simply add  $2\lambda v_{ij}$  and  $2\lambda w_j$  terms to the partial derivatives

63

## Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Randomly dropping out (setting to zero) a number of output features of the layer during training

e.g., normal output: [1.3, 0.4, 0.5, 1.5, 3.2, 0.9]  
After dropout: [1.3, 0, 0, 1.5, 3.2, 0]

Dropout rate: the fraction of the units that are zeroed out (0.2-0.5).

No units are dropped out during testing

64

## Summary

- How to perform convolution? How to compute output size after convolution?  
Advantages & comparison between convolutional layer and fully connected layer?
- How to perform max-pooling? Advantages of max-pooling?
- What problem does momentum solve in SGD? How do RMSprop and Adam update weights?
- What's the advantage of (Leaky) ReLU compared to sigmoid and tanh?
- How to prevent overfitting problem in network training?

Next: CNN Architectures

65

# References

- Hinton: [https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture\\_slides\\_lec6.pdf](https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf)
- Deep learning with python by Francois Chollet, chapter 2.4.3, 5.1
- Deep Learning, by Ian Goodfellow and Yoshua Bengio and Aaron Courville, chapter 9.2-9.3
- Pattern Recognition and Machine Learning by Chris Bishop, chapter 5.5.6



# CNN Architectures

COMP90051 Statistical Machine Learning

QiuHong Ke

Copyright: University of Melbourne

## Outline

- LeNet5
- AlexNet
- VGG
- GoogleNet
- ResNet

LeNet5

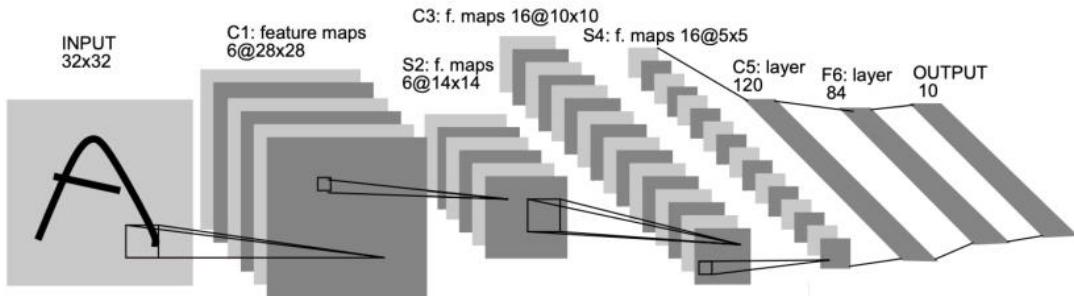
AlexNet

VGG

GoogleNet

ResNet

## Architecture



LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

3

LeNet5

AlexNet

VGG

GoogleNet

ResNet

1st layer

How many parameters?

$$11 \times 11 \times 3 \times 96 = 34,848$$



32x32x1

Convolutional layer  
No.filters: 6  
Filter size: 5x5  
Padding: 0  
Stride:1



? 28x28x6

No padding: `output_size=ceiling( (input_size-kernel_size+1)/stride )`

4

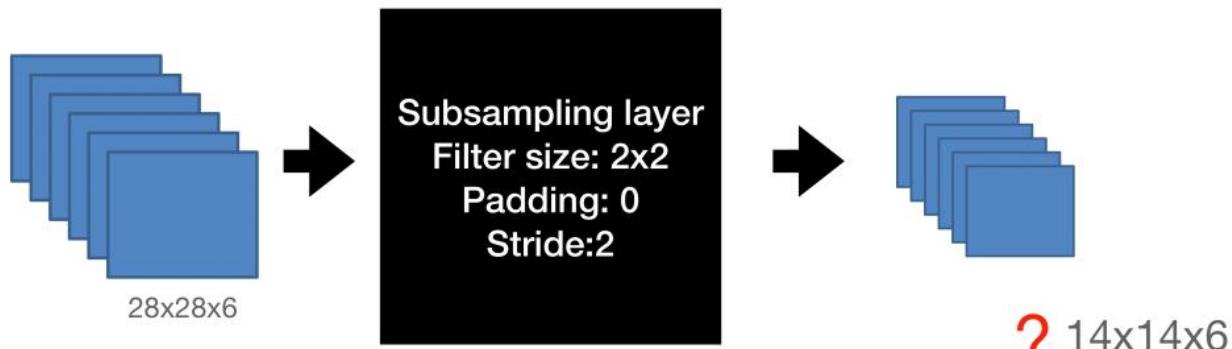
LeNet5

AlexNet

VGG

GoogleNet

ResNet

**2nd layer**

- Take the sum of all units in the 2x2 window
  - output of each patch = sum\*coefficient (trainable) + bias (trainable)
- No padding:  $\text{output\_size} = \text{ceiling}(\frac{\text{input\_size} - \text{kernel\_size} + 1}{\text{stride}})$

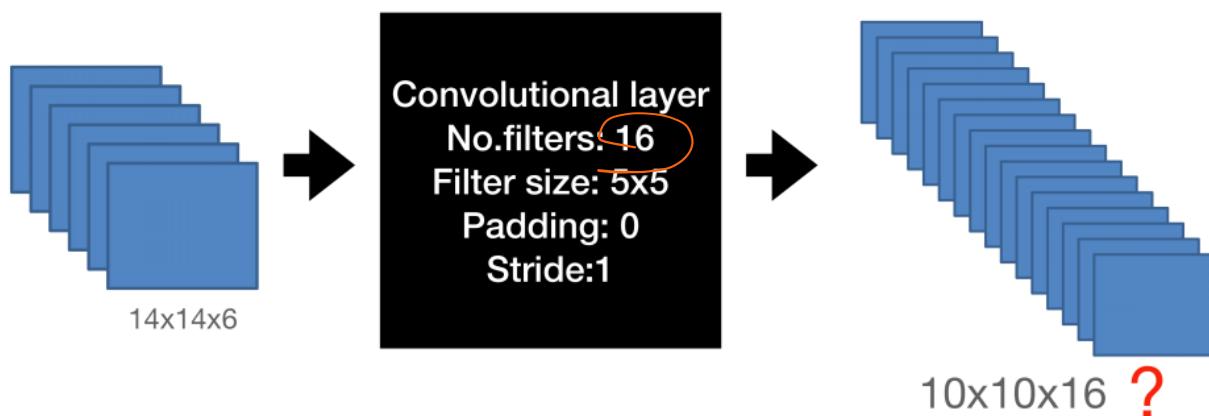
LeNet5

AlexNet

VGG

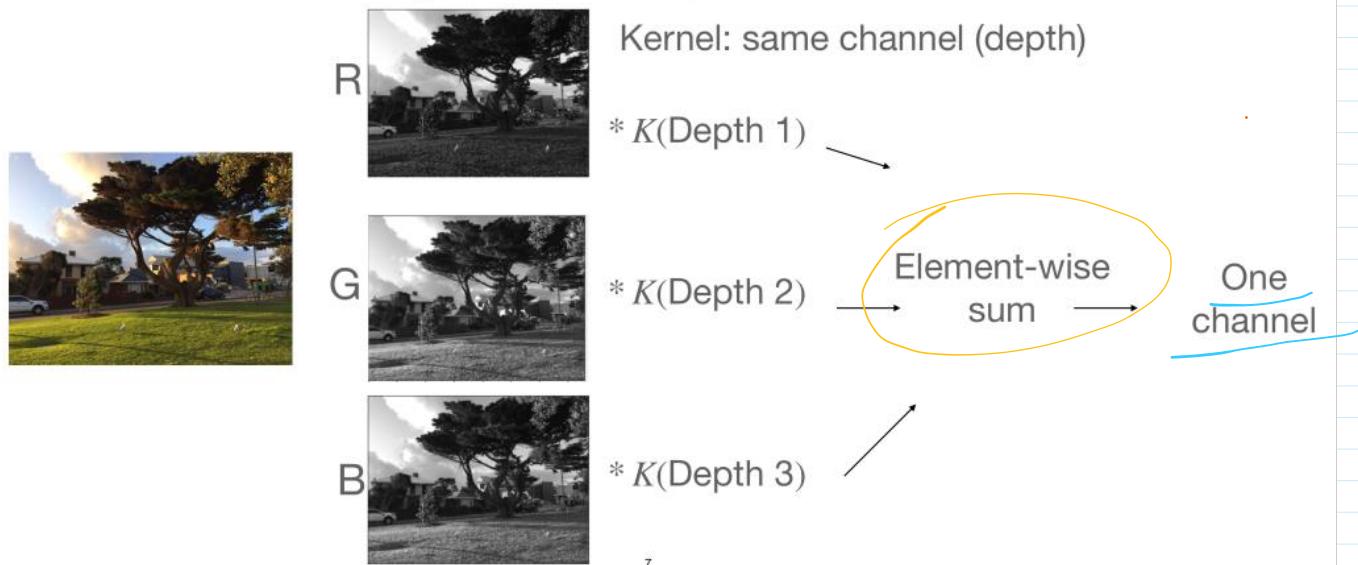
GoogleNet

ResNet

**3rd layer**

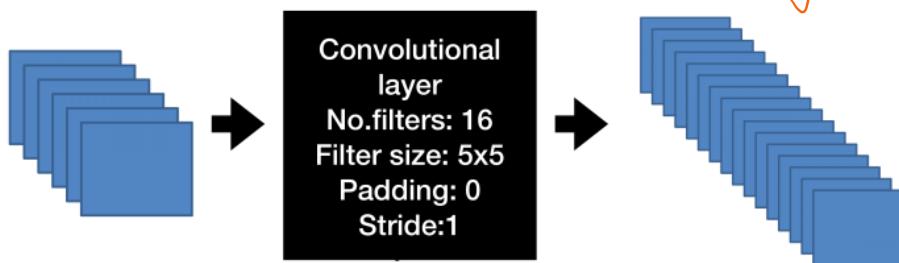
No padding:  $\text{output\_size} = \text{ceiling}(\frac{\text{input\_size} - \text{kernel\_size} + 1}{\text{stride}})$

## Convolution on Multiple-channel input



## 3rd layer: Non-complete connection scheme

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X			X	X	X			X	X	X	X		X	X	
1	X	X			X	X	X			X	X	X	X		X	
2	X	X	X			X	X	X			X		X	X	X	
3		X	X	X		X	X	X	X			X	X	X		
4		X	X	X		X	X	X	X		X	X		X		
5			X	X	X		X	X	X	X		X	X	X		



LeNet5

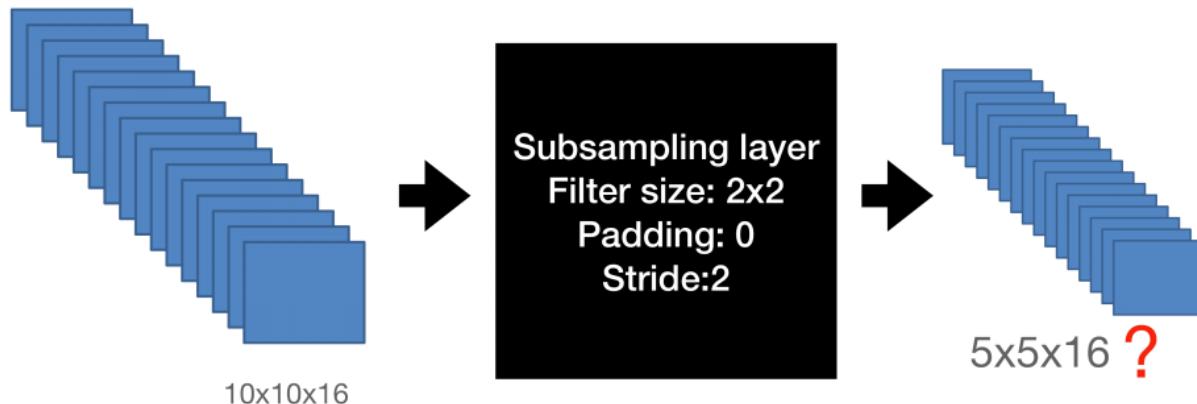
AlexNet

VGG

GoogleNet

ResNet

4th layer



No padding:  $\text{output\_size} = \text{ceiling}(\frac{\text{input\_size} - \text{kernel\_size} + 1}{\text{stride}})$

LeNet5

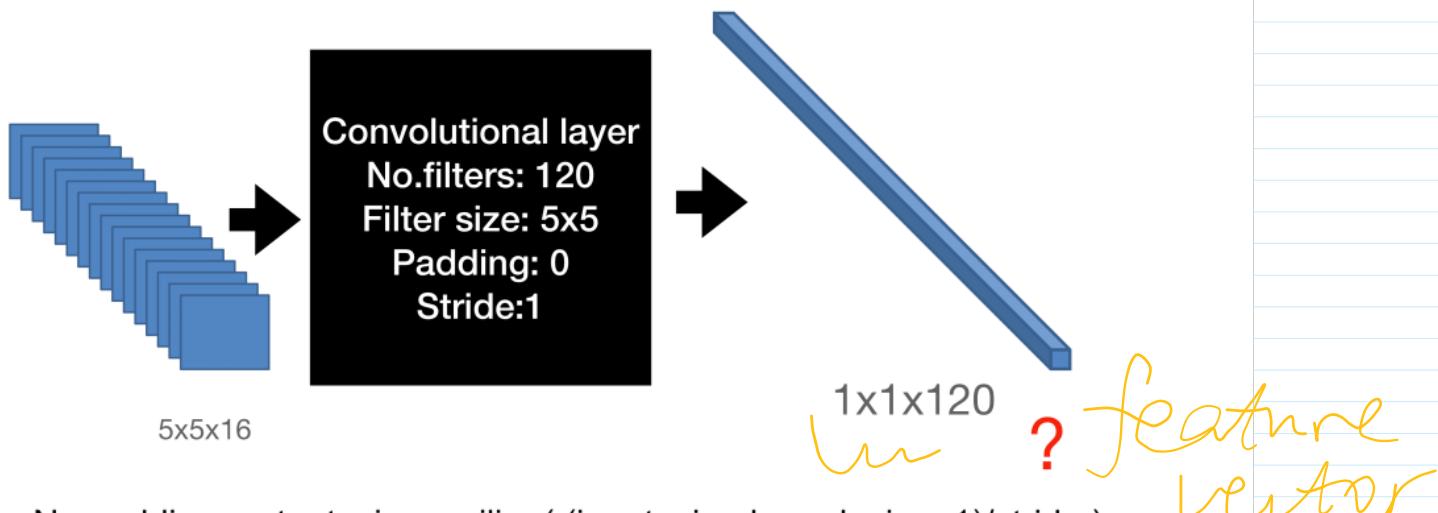
AlexNet

VGG

GoogleNet

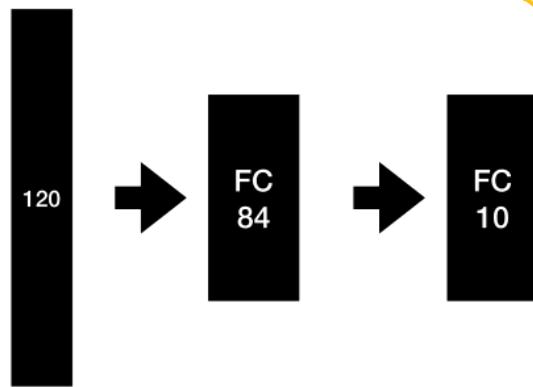
ResNet

5th layer



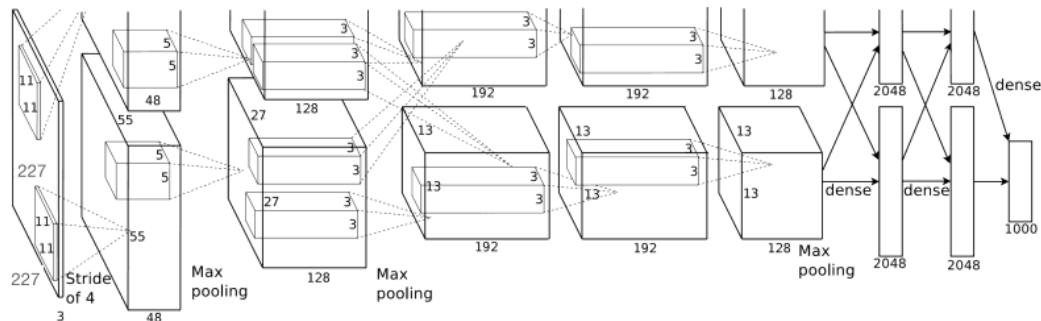
No padding:  $\text{output\_size} = \text{ceiling}(\frac{\text{input\_size} - \text{kernel\_size} + 1}{\text{stride}})$

**Following: Fully connected layers**



11

**Architecture**



Krizhevsky, Alex., Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

12

LeNet5

AlexNet

VGG

GoogleNet

ResNet

**1st layer****How many parameters?**

$$11 \times 11 \times 3 \times 96 = 34,848$$

Consider filters

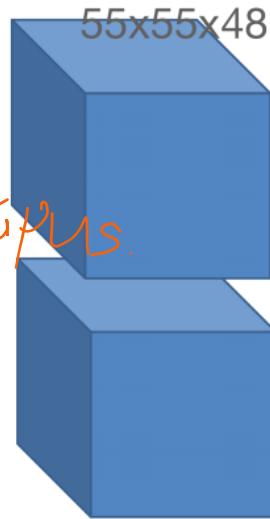


Channel

**Convolutional layer**  
 No. filters: 96  
 Filter size: 11x11  
 Padding: 0  
 Stride: 4



55x55x48

No padding:  $\text{output\_size} = \text{ceiling}((\text{input\_size} - \text{kernel\_size} + 1) / \text{stride})$ 

LeNet5

AlexNet

VGG

GoogleNet

ResNet

**Convolution on Multiple-channel input**

R



G



B



Kernel: same channel (depth)

\* K(Depth 1)

Element-wise sum

\* K(Depth 2)

\* K(Depth 3)

One channel

LeNet5

AlexNet

VGG

GoogleNet

ResNet

**2nd layer****How many parameters?**

0



Max-pooling  
Pool size: 3x3  
Padding: 0  
Stride: 2

27x27x48



27x27x48 ?

No padding:  $\text{output\_size} = \text{ceiling}(\frac{\text{input\_size} - \text{kernel\_size} + 1}{\text{stride}})$ 

LeNet5

AlexNet

VGG

GoogleNet

ResNet

**3rd layer****How many parameters?**

$$5 \times 5 \times 48 \times 256 = 614,400$$

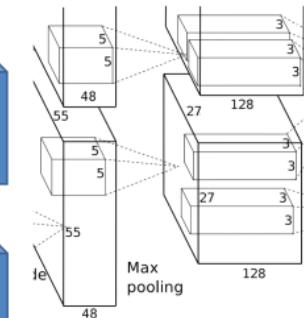


Convolutional layer  
No. filters: 256  
Filter size: 5x5  
Padding: 2  
Stride: 1

27x27x128



27x27x128

padding:  $\text{output\_size} = \text{ceiling}(\frac{\text{input\_size}}{\text{stride}})$ 

?

LeNet5

AlexNet

VGG

GoogleNet

ResNet

**4th layer****How many parameters?**

0



$$\text{output\_size} = \text{ceiling}\left(\frac{\text{input\_size} - \text{kernel\_size} + 1}{\text{stride}}\right)$$

LeNet5

AlexNet

VGG

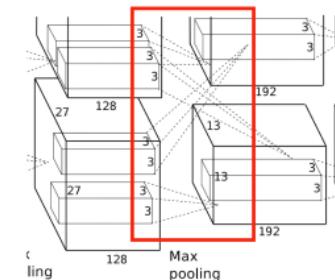
GoogleNet

ResNet

**5th layer****How many parameters?**

$$3 \times 3 \times 256 \times 384 = 884,736$$

**Convolution layer**  
No. filters: 384  
Filter size: 3x3  
Padding: 1  
Stride: 1



LeNet5

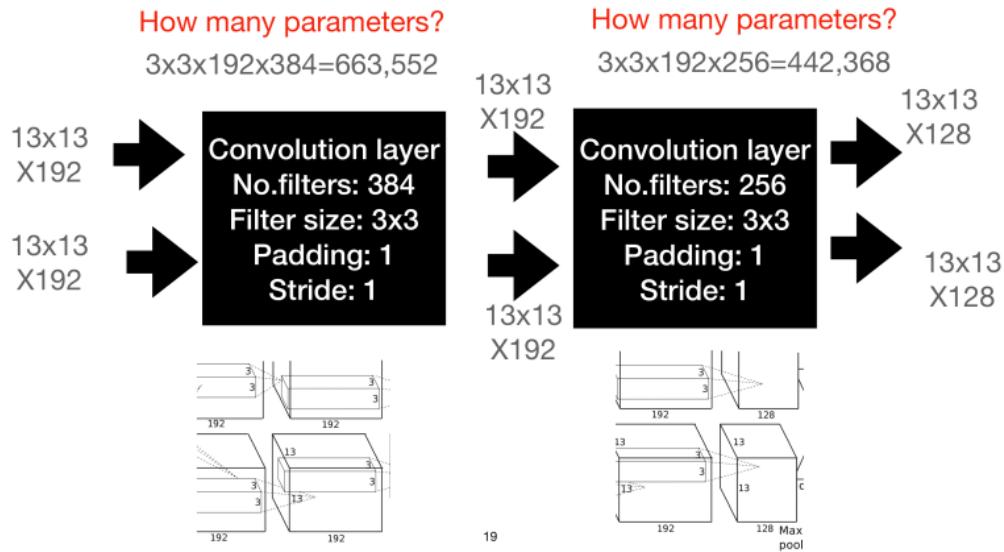
AlexNet

VGG

GoogleNet

ResNet

## Following convolutional layers



LeNet5

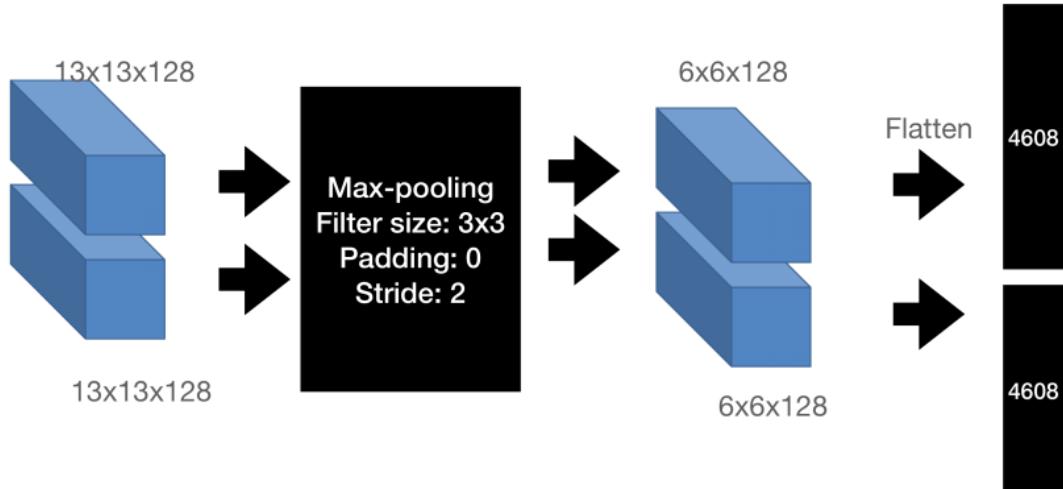
AlexNet

VGG

GoogleNet

ResNet

## Max-pooling and flatten



LeNet5

AlexNet

VGG

GoogleNet

ResNet

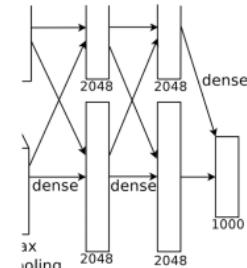
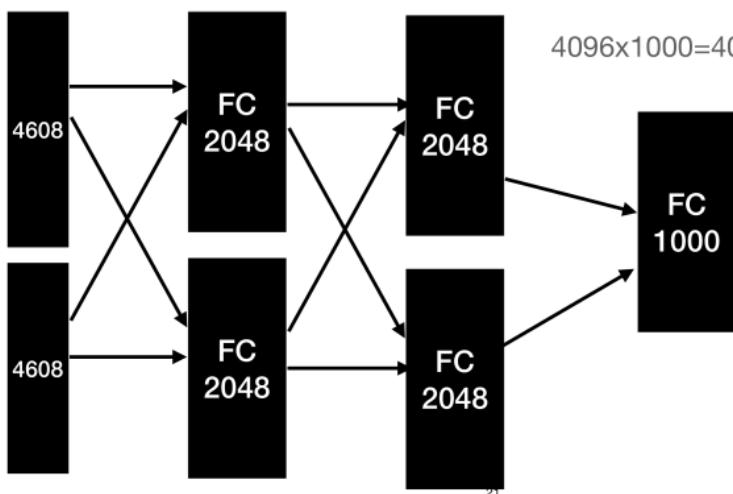
**Following fully connected layers**

How many parameters?

$$9216 \times 4096 = 37,748,736$$

$$4096 \times 4096 = 16,777,216$$

$$4096 \times 1000 = 4096000$$



LeNet5

AlexNet

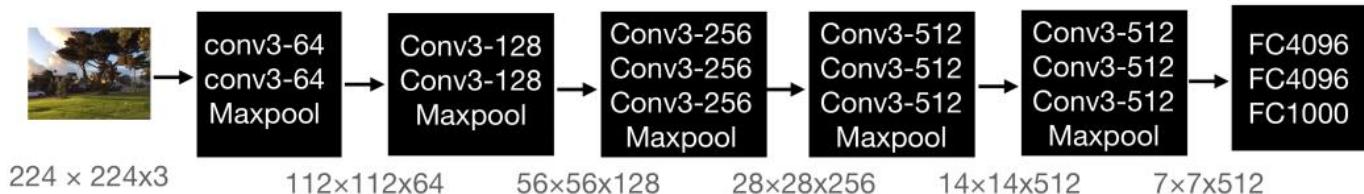
VGG

GoogleNet

ResNet

**Architecture**

- VGG16: 16 weight layers



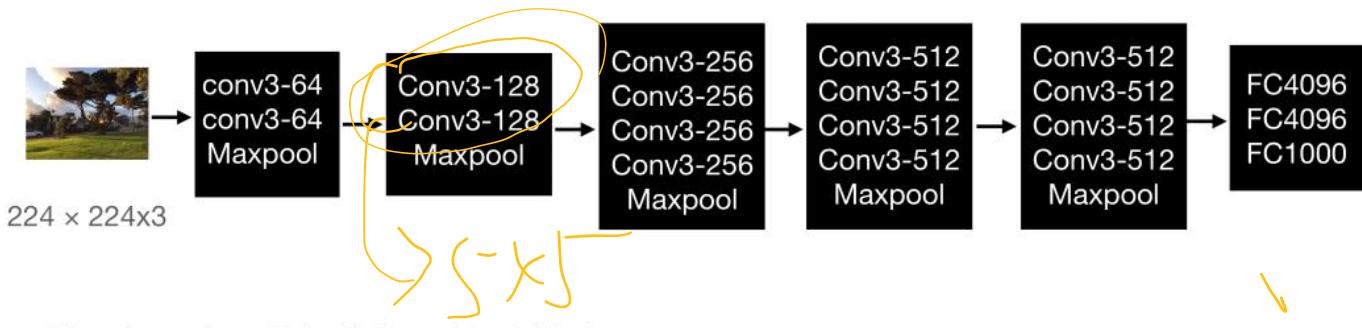
Conv layer: kernel size 3x3, pad 1, stride 1

Maxpooling layer: 2x2, stride 2

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

**Architecture**

- VGG19: 19 weight layers**



Conv layer: kernel size 3x3, pad 1, stride 1  
Maxpooling layer: 2x2, stride 2

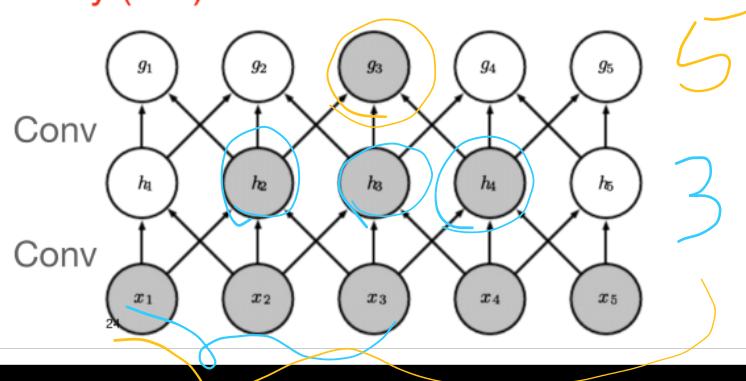
Simonyan, Karen, and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556 (2014).

23

**Stacking multiple 3x3 conv layers**

- a stack of two 3x3 conv. layers has an effective receptive field of  $5 \times 5$
  - a stack of 3 3x3 conv. layers has an effective receptive field of  $7 \times 7$
- If you add an additional convolutional layer with kernel size K, the receptive field is increased by (K-1)

More layers: larger size of receptive field  
(larger window of the input is seen)

**Why Stacking multiple 3x3 conv layers instead of large filter size?**

- Reduce parameter:
  - $5 \times 5 = 25$ ,  $3 \times 3 \times 2 = 18$
  - $7 \times 7 = 49$ ,  $3 \times 3 \times 3 = 27$
- Each conv use ReLU as the activation function. More layers, more non-linear rectification layers → More powerful network

non-linear rectification layers → More powerful network

25

LeNet5

AlexNet

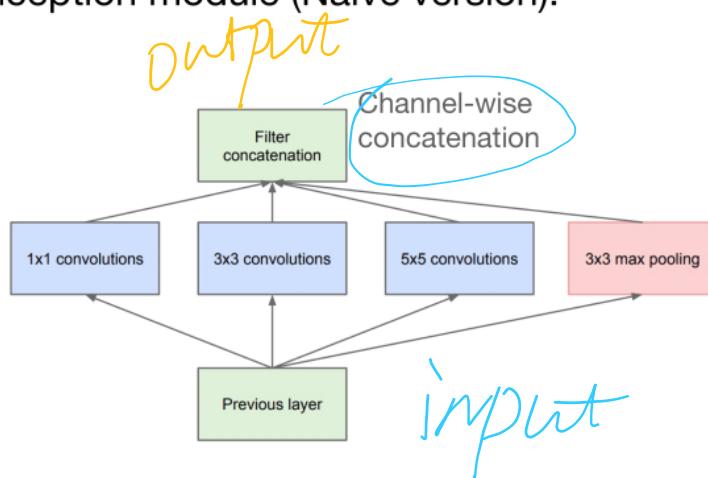
VGG

GoogleNet

ResNet

## Architecture

Inception module (Naive version):



Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition, 2015.

26

type	patch size/stride	output size	depth
convolution	$7 \times 7 / 2$	$112 \times 112 \times 64$	1
max pool	$3 \times 3 / 2$	$56 \times 56 \times 64$	0
convolution	$3 \times 3 / 1$	$56 \times 56 \times 192$	2
max pool	$3 \times 3 / 2$	$28 \times 28 \times 192$	0
inception (3a)		$28 \times 28 \times 256$	2
inception (3b)		$28 \times 28 \times 480$	2
max pool	$3 \times 3 / 2$	$14 \times 14 \times 480$	0
inception (4a)		$14 \times 14 \times 512$	2
inception (4b)		$14 \times 14 \times 512$	2
inception (4c)		$14 \times 14 \times 512$	2
inception (4d)		$14 \times 14 \times 528$	2
inception (4e)		$14 \times 14 \times 832$	2
max pool	$3 \times 3 / 2$	$7 \times 7 \times 832$	0
inception (5a)		$7 \times 7 \times 832$	2
inception (5b)		$7 \times 7 \times 1024$	2
avg pool	$7 \times 7 / 1$	$1 \times 1 \times 1024$	0
dropout (40%)		$1 \times 1 \times 1024$	0
linear		$1 \times 1 \times 1000$	1
softmax		$1 \times 1 \times 1000$	0

LeNet5

AlexNet

VGG

GoogleNet

ResNet

Different scales of data require different convolutional filter sizes



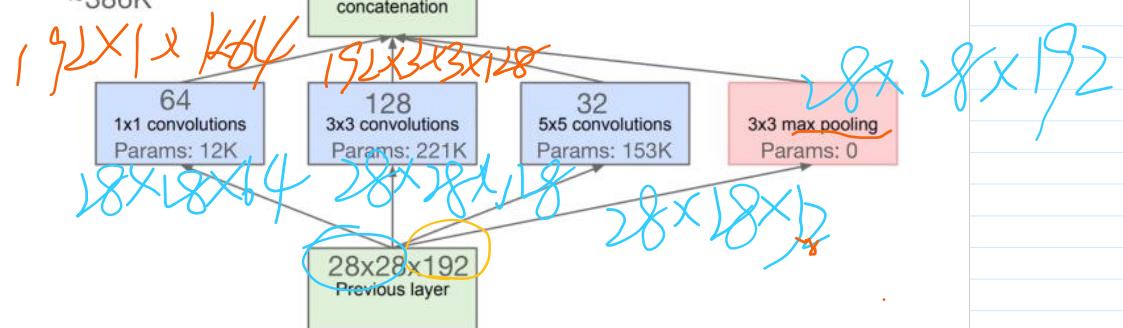
27

## Inception module (Naive version)

How many parameters?

Input\_channel x  
Kernel\_size x  
NO.filters  
(output\_channel)

Total parameters:  
~386K



Channel-wise concatenation

$$28 \times 28 \times (64 + 128 + 32 + 192) \\ = 28 \times 28 \times 416$$

Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015

28

## Inception module (Naive version)

More modules →

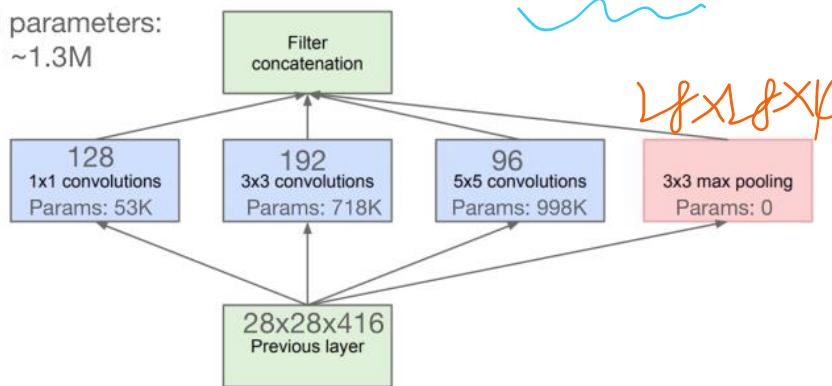
More parameters

More computations

Total parameters:  
~1.3M

Channel-wise concatenation

$$28 \times 28 \times (128 + 192 + 96 + 256) \\ = 28 \times 28 \times 672$$

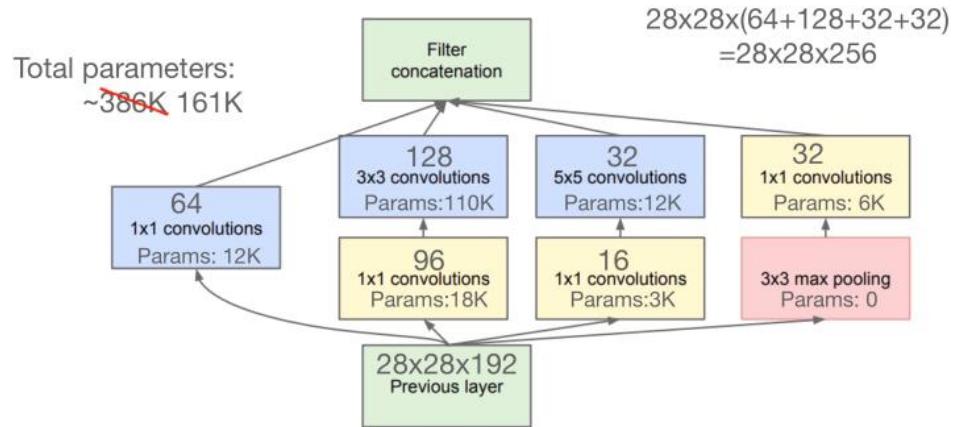


Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015

29

## Inception module with dimensionality reduction

Use 1x1 convolution to reduce channels

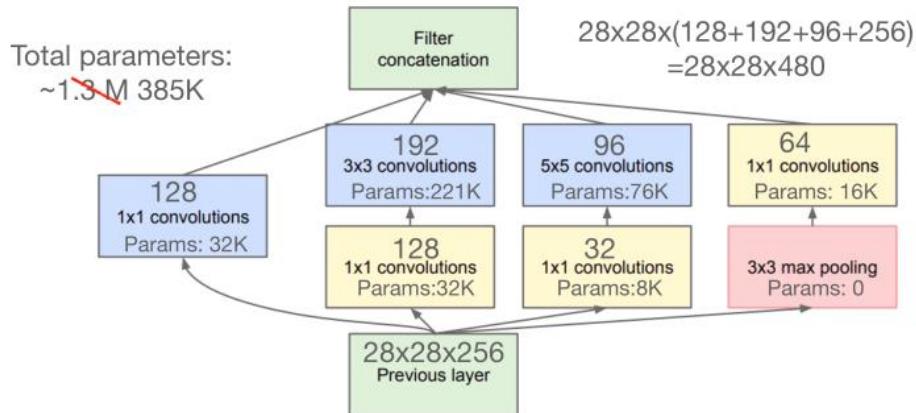


Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

30

## Inception module with dimensionality reduction

Use 1x1 convolution to reduce channels



Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

31

# Large Scale Visual Recognition Challenge

## Comparison of different architectures

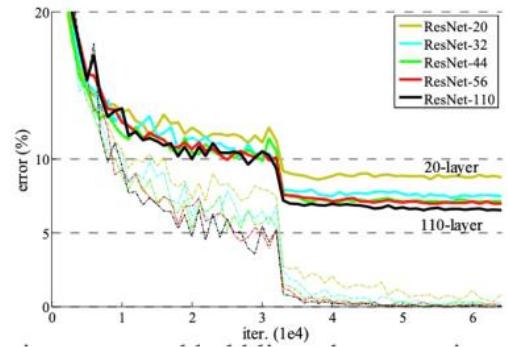
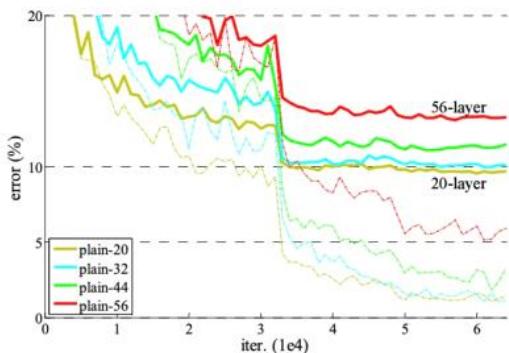
- Top-5 error: the proportion of images that the ground-truth category is outside the top-5 predicted categories of the model.

CNN Architecture	Layers	Top-5 error
AlexNet	8	16.4%
VGG-19	19	7.3%
GoogleNet	22	6.7%

32

## More layers?

Residual network: more layers & better performance



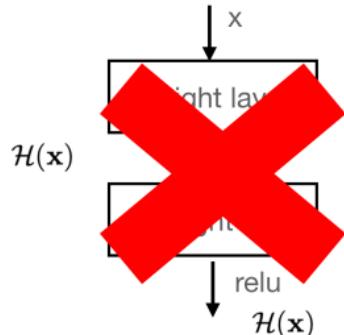
Training on CIFAR-10. Dashed lines denote training error, and bold lines denote testing error

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.

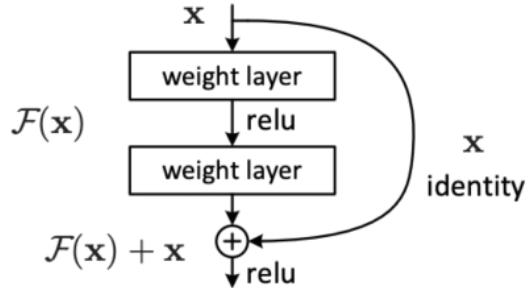
33

## Residual network

Hypothesis: residual mapping  $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$  is easier to optimise



Unreferenced mapping:  
directly fit the desired underlying mapping

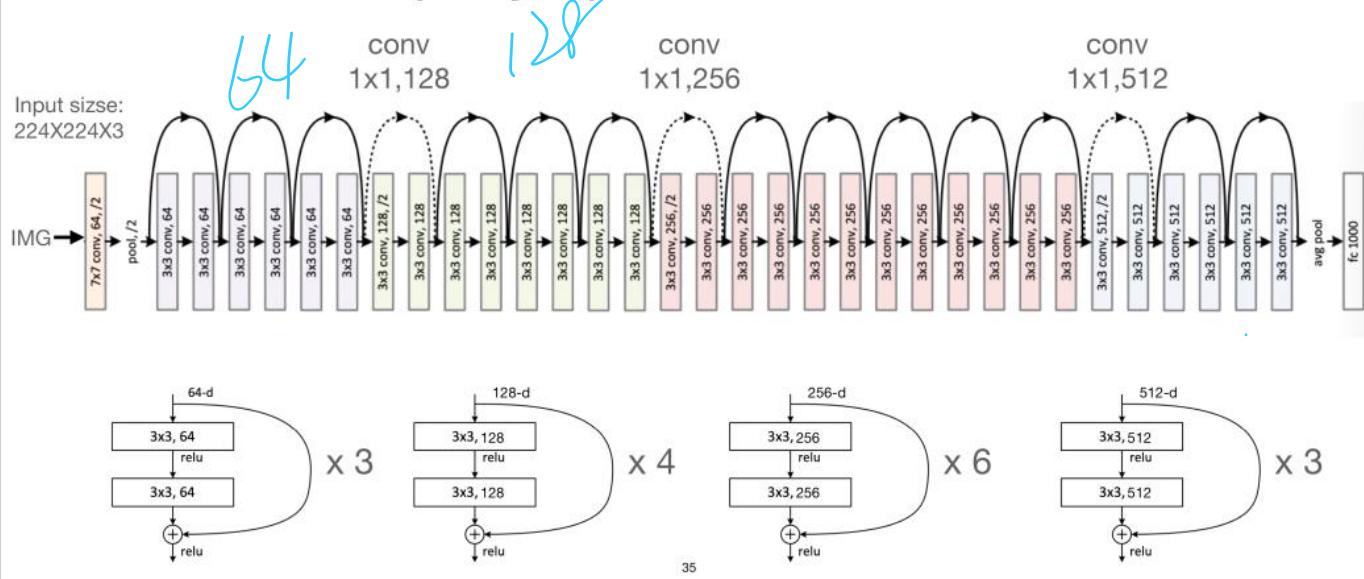


Residual learning:  
let these layers fit a residual mapping

34

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

## Residual network (34 layers)



35

## Residual network

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112			7×7, 64, stride 2		
				3×3 max pool, stride 2		
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1			average pool, 1000-d fc, softmax		

Downsampling is performed by conv3 1, conv4 1, and conv5 1 with a stride of 2

36

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

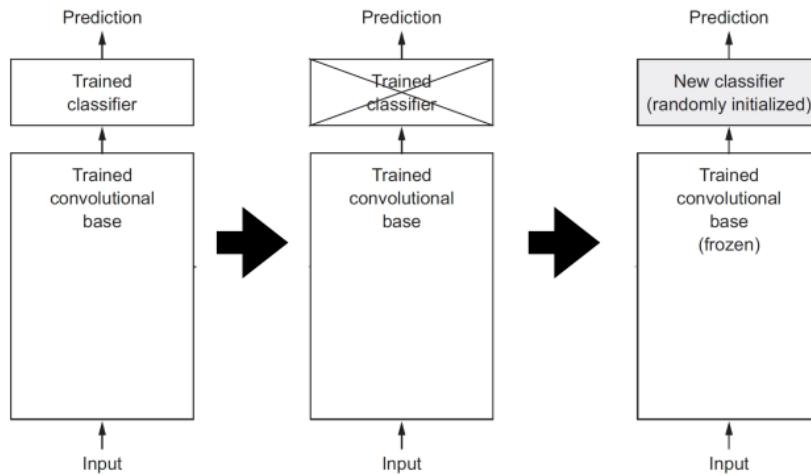
## Large Scale Visual Recognition Challenge Comparison of different architectures

- Top-5 error: the proportion of images that the ground-truth category is outside the top-5 predicted categories of the model.

CNN Architecture	Layers	Top-5 error
AlexNet	8	16.4%
VGG-19	19	7.3%
GoogleNet	22	6.7%
ResNet	152	3.57%

# Use the pretrained CNN model as feature extractor

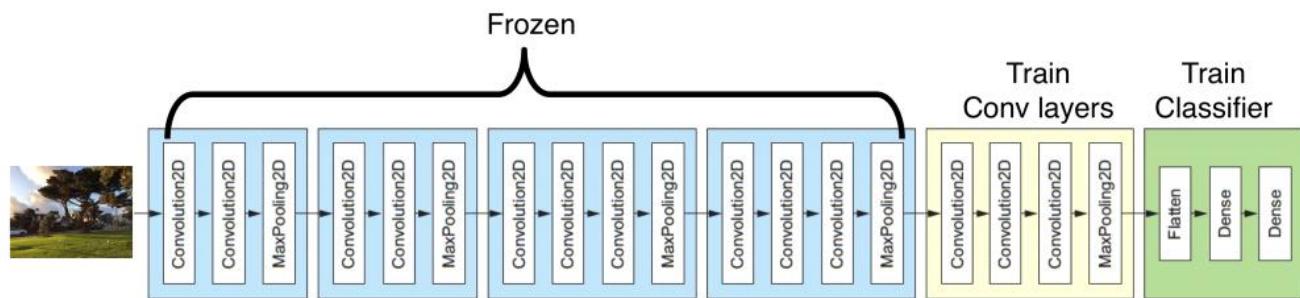
Train a new classifier for output



38

## If you have quite a lot of data: fine-tuning

Slightly train a few more top layers



39

# Summary

- How to calculate the NO. parameters & size of output feature map ?
- Difference of the architectures
- Key idea of VGG: how to increase receptive field?
- Key idea of GoogleNet: how to reduce parameters?
- Key idea of ResNet: how to increase layers with better performance?



# Generative models

COMP90051 Statistical Machine Learning

*"What I Cannot Create, I Do Not Understand" ——Richard Feynman*

QiuHong Ke

Copyright: University of Melbourne

## So far..

Classifier:

- SVM
- Perceptron
- Multi-layer perceptron
- CNN

Feature extraction:

- Pretrained CNN model

# Using pretrained model in Keras

## Feature extraction before the last classifier

```
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input
import numpy as np

model = ResNet50(weights='imagenet', include_top=False)

img_path = '1.jpeg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

features = model.predict(x)
```

3

# Using pretrained model in Keras

## Feature extraction from arbitrary layer

```
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input
import numpy as np
from tensorflow.keras.models import Model

model = ResNet50(weights='imagenet')
layers=model.layers #get name of the layers
model = Model(inputs=model.input, outputs=model.get_layer('conv5_block3_out').output)

img_path = '1.jpeg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

features = model.predict(x)
```

4

# Using pretrained model in Keras

## Perform classification

```
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np

model = ResNet50(weights='imagenet')

img_path = '1.png'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)

print('Predicted:', decode_predictions(preds, top=3)[0])
```

5

## Predict class using a pretrained ResNet 50

Pretrained on ImageNet (1000 classes)



class	probability
espresso	0.9995716
cup	0.00022115342
coffee_mug	0.00020262193



class	probability
daisy	0.79781777
pot	0.10065505
picket_fence	0.05316206



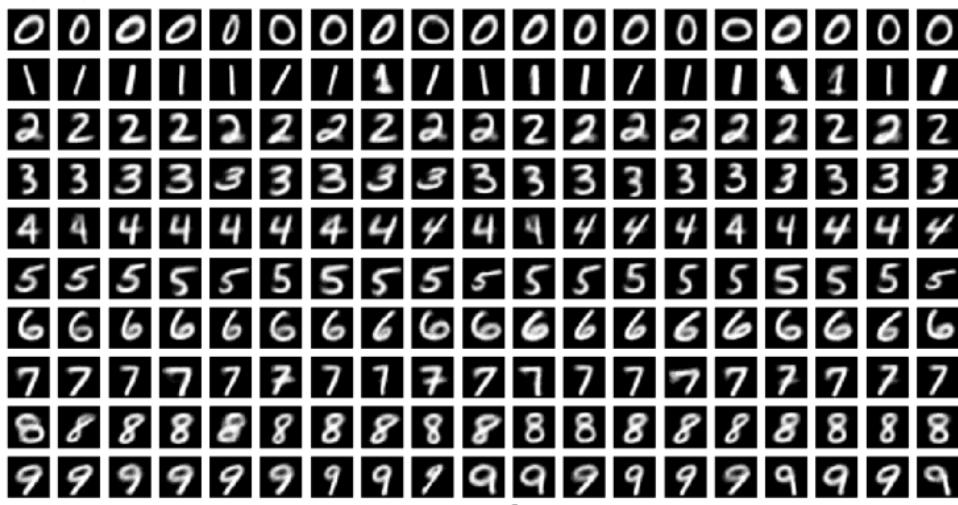
class	probability
alp (mountain)	0.7066206
geyser	0.106300876
valley	0.056517705



class	probability
rubber_eraser	0.53692865
envelope	0.1995637
paper_towel	0.040610846

# Beyond classification

Image generation



7

# Beyond classification

Image generation



Figure 8.10 in Deep learning with python by Francois Chollet

# Beyond classification

## Image editing



(a) *Glasses*: remove and add the glasses



(b) *Mouth\_open*: close and open the mouth



(c) *No\_beard*: add and remove the beard

Shen, Wei, and Ruijie Liu. "Learning residual images for face attribute manipulation." *CVPR* 2017.

# Generative Models

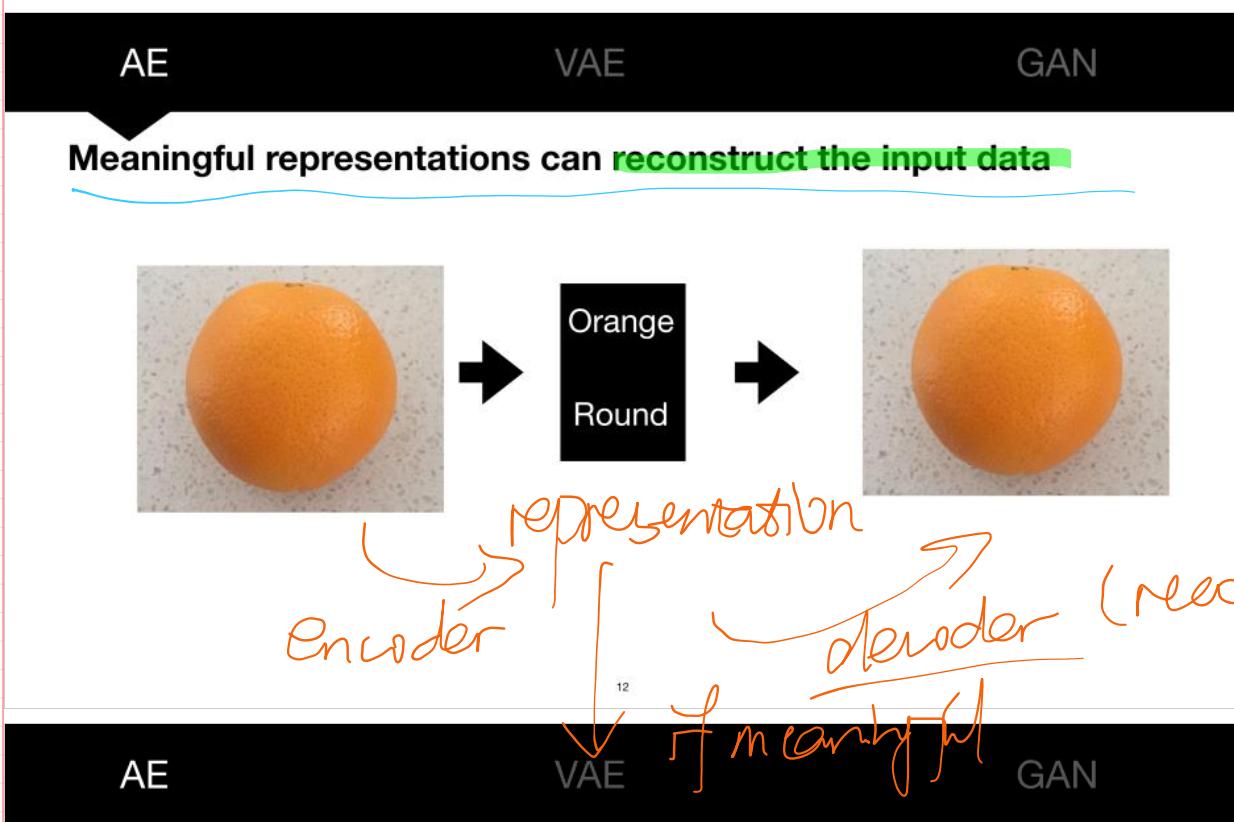
## Cope with all of above tasks

- Variational Autoencoder (VAE)
- Generative Adversarial Network (GAN)

# Outline

- Autoencoder (AE)
- Variational Autoencoder (VAE)
- Generative Adversarial Model (GAN)

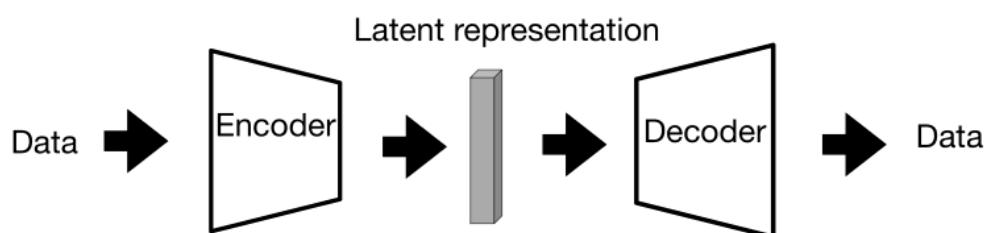
11



Encoder  
Latent representation  
Decoder  
meaningful

## Encoder - decoder

Extract meaningful representation to reconstruct the input data

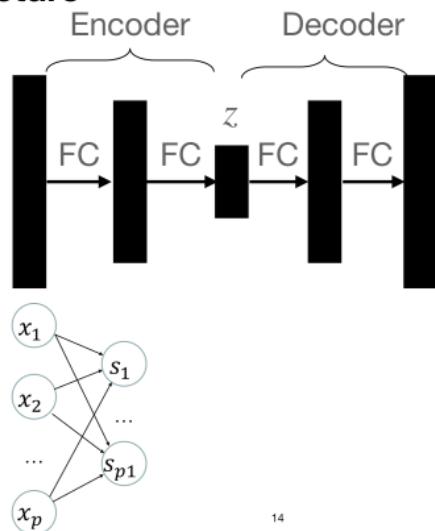


13

AE

VAE

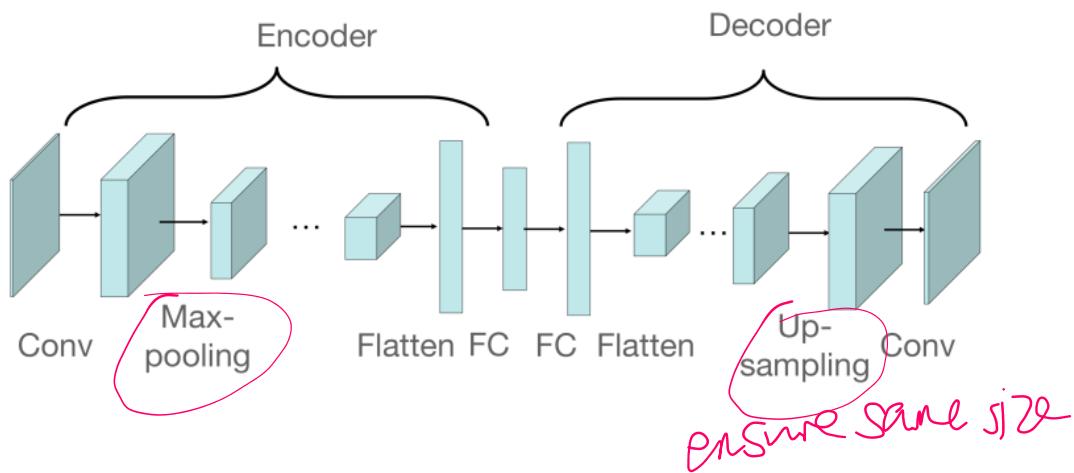
GAN

**Network architecture**

AE

VAE

GAN

**Network architecture**

AE

VAE

GAN

## Upsampling

```
y = tf.keras.layers.UpSampling2D(size=(2, 2))(x)
```

Size: The upsampling factors for rows and columns

1	2
3	4

Upsampling



1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

16

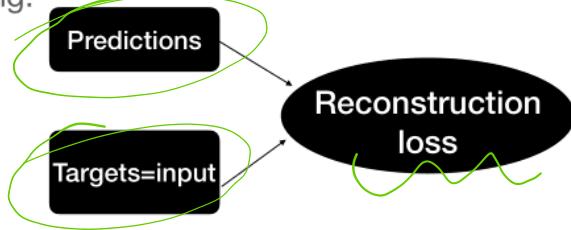
AE

VAE

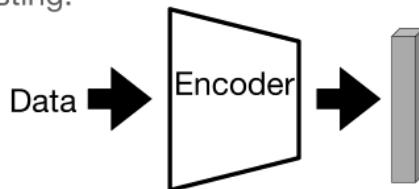
GAN

## Training and testing

Training:



Testing:



Loss:

'mse':  $L = E[(X - y_{pre})^2]$

'binary\_crossentropy': Input is between 0~1

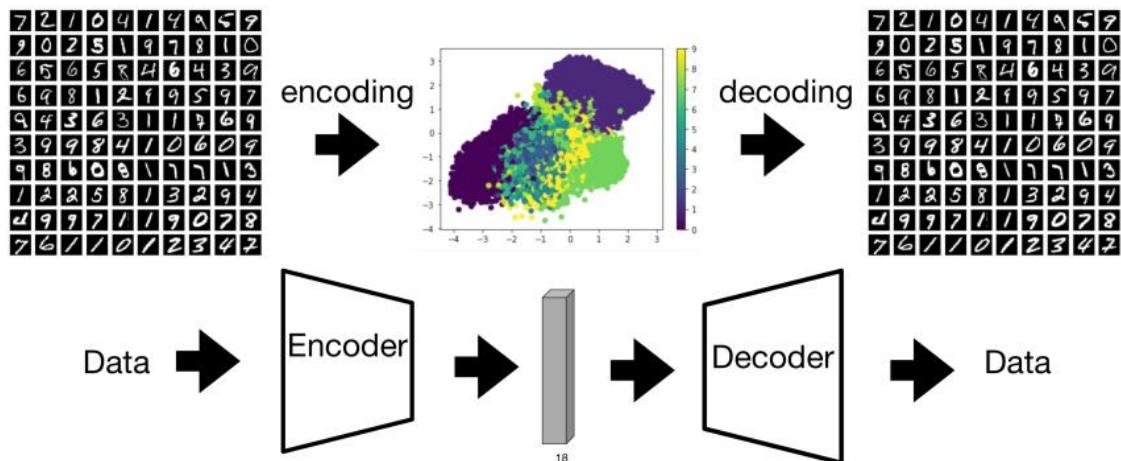
17

AE

VAE

GAN

### Dimension reduction

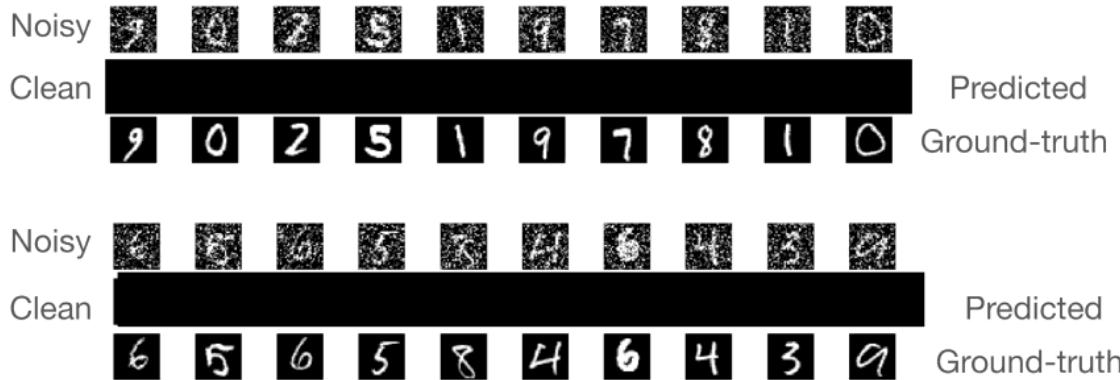


AE

VAE

GAN

### Image denoising



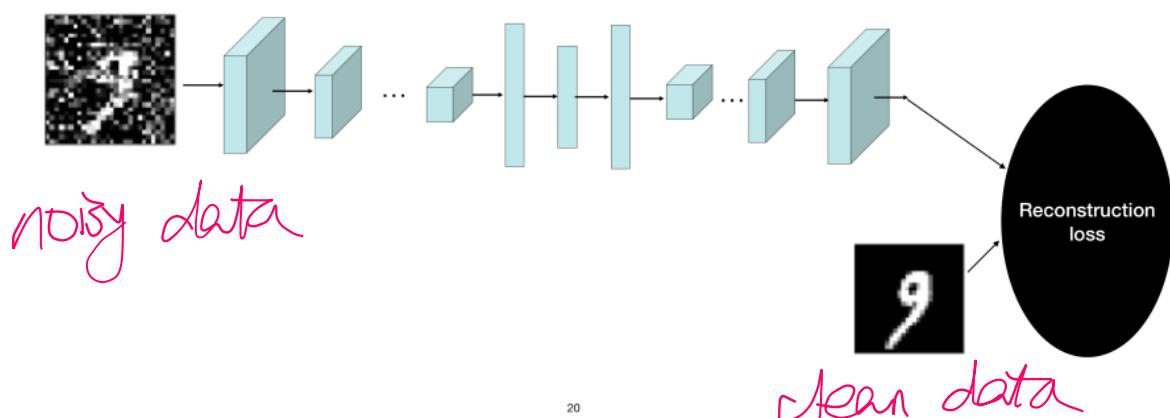
AE

VAE

GAN

### Denoising Autoencoder

Training (have access to clean data):



20

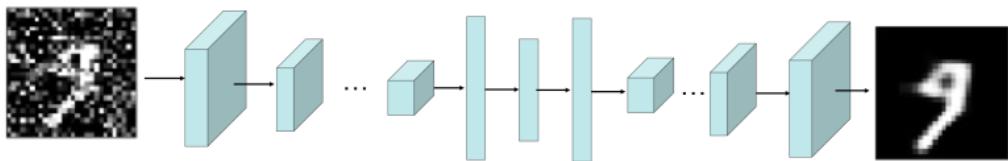
AE

VAE

GAN

### Denoising Autoencoder

Testing:

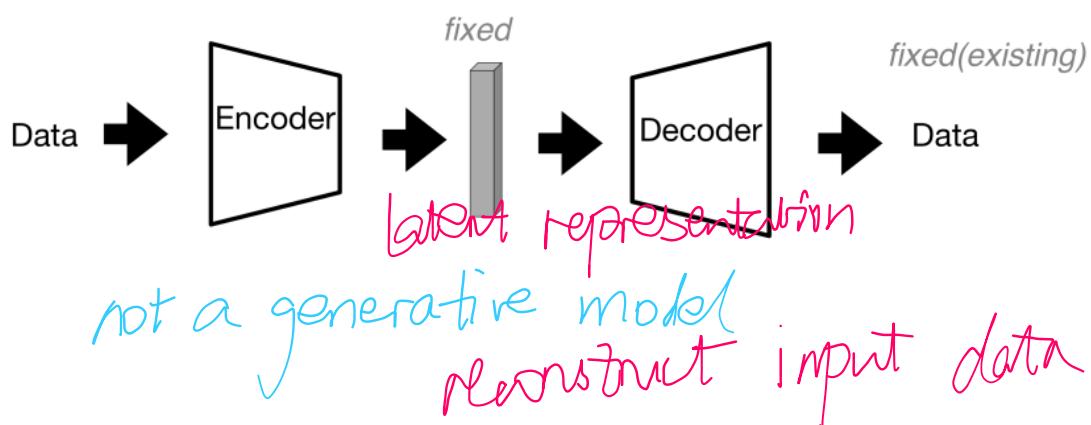


21

AE

VAE

GAN

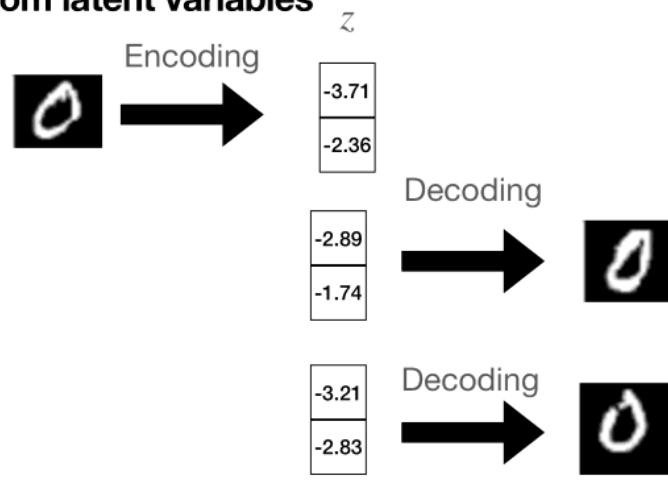
**Generate new data? NO**

22

AE

VAE

GAN

**Data generation from latent variables**

23 \*\*\*

\*\*\*

AE

VAE

GAN

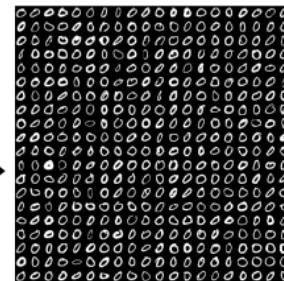
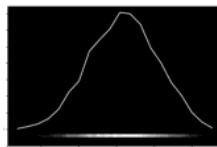
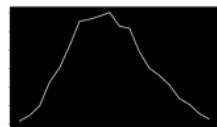
## Data generation from latent variable

$$p(x) = \int p(x|z)p(z)dz$$

$p(x)$ : probability of the data  $x$

$p(z)$ : probability of the latent variable  $z$

$p(x|z)$ : probability of  $x$  given  $z$



24

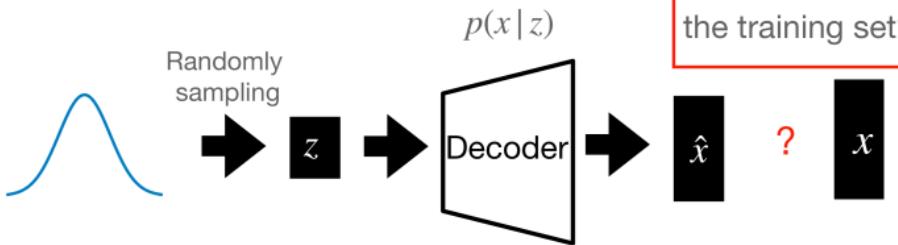
AE

VAE

GAN

## Turn latent variable into data using a decoder (network)

Assume:  $p(z) = N(z|0,I)$  I: identity matrix



25

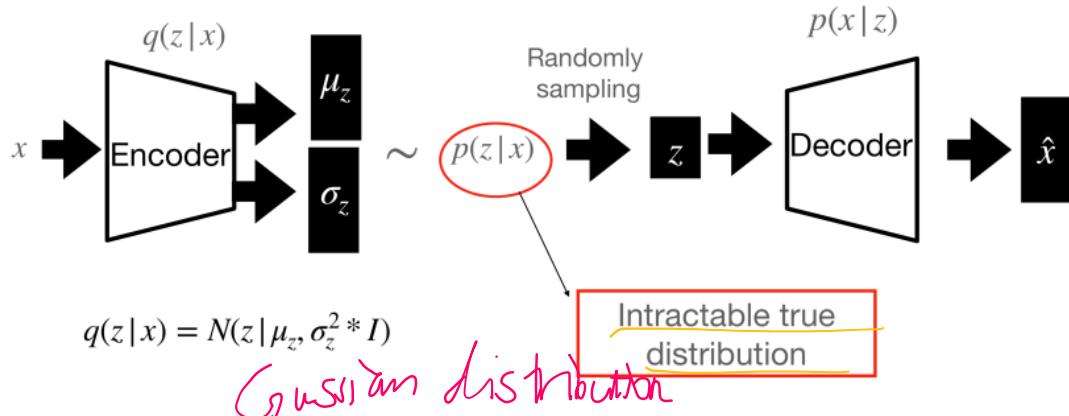
AE

VAE

GAN

### Encoder: Reduce sampling space

Use posterior distribution to sample  $z$  is more likely to produce  $x$



26

AE

VAE

GAN

Math is coming...



27

## Network optimisation: Maximize log likelihood and gradient ascend

*constant*

$$\begin{aligned}
 \log p(x^{(i)}) &= E_{z \sim q(z|x)} [\log p(x^{(i)})] \\
 &= E_{z \sim q(z|x)} \left[ \log \frac{p(x^{(i)}|z)p(z)}{p(z|x^{(i)})} \right] = E_{z \sim q(z|x)} \left[ \log \frac{p(x^{(i)}|z)p(z)}{p(z|x^{(i)})} \frac{q(z|x^{(i)})}{q(z|x^{(i)})} \right] \\
 &= E_{z \sim q(z|x)} [\log p(x^{(i)}|z)] - E_{z \sim q(z|x)} \left[ \log \frac{q(z|x^{(i)})}{p(z)} \right] + E_{z \sim q(z|x)} \left[ \log \frac{q(z|x^{(i)})}{p(z|x^{(i)})} \right] \\
 &= E_{z \sim q(z|x)} [\log p(x^{(i)}|z)] - D_{KL} [q(z|x^{(i)}) || p(z)] + D_{KL} [q(z|x^{(i)}) || p(z|x^{(i)})]
 \end{aligned}$$

KL Divergence

28

## Maximize lower bound

$$\begin{aligned}
 \log p(x^{(i)}) &= E_{z \sim q(z|x)} [\log p(x^{(i)}|z)] - D_{KL} [q(z|x^{(i)}) || p(z)] + D_{KL} [q(z|x^{(i)}) || p(z|x^{(i)})] \\
 &\geq E_{z \sim q(z|x)} [\log p(x^{(i)}|z)] - D_{KL} [q(z|x^{(i)}) || p(z)]
 \end{aligned}$$

*maximize lower bound*

lower bound

Unknown true posterior

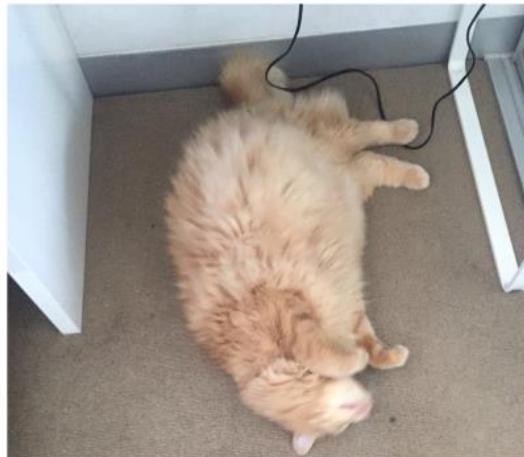
$\max E_{z \sim q(z|x)} [\log p(x^{(i)}|z)]$  : Minimize reconstruction loss between output and input

$q(z|x)$  : Output of encoder  $q(z) = N(z|\mu, \sigma^2 I)$

$p(z)$  : Prior distribution, assumed to be standard Gaussian distribution  $p(z) = N(z|0, I)$

$D_{KL} [q(z|x^{(i)}) || p(z)]$  : KL divergence of two gaussian distributions

Math is coming AGAIN...



30

**Warning of math: KL divergence**

Gaussian Distribution

$$\begin{aligned}
 KL(p_1 || p_2) &= \int p_1(x) \log \frac{p_1(x)}{p_2(x)} dx \\
 &= \int [-\log \sigma_1 - \frac{1}{2}(\log(2\pi)) - \frac{1}{2} \left( \frac{x - \mu_1}{\sigma_1} \right)^2 - (-\log \sigma_2 - \frac{1}{2}(\log(2\pi)) - \frac{1}{2} \left( \frac{x - \mu_2}{\sigma_2} \right)^2] p_1(x) dx \\
 &= -\log \sigma_1 + \log \sigma_2 - \frac{1}{2\sigma_1^2} E_1[(x - \mu_1)^2] + \frac{1}{2\sigma_2^2} E_1[(x - \mu_2)^2] \\
 &= -\log \sigma_1 + \log \sigma_2 - \frac{1}{2} + \frac{1}{2\sigma_2^2} E_1[(x - \mu_2)^2]
 \end{aligned}$$

31

**Warning of math: KL divergence**

$$KL(p_1 || p_2) = \int p_1(x) \log \frac{p_1(x)}{p_2(x)} dx = -\log \sigma_1 + \log \sigma_2 - \frac{1}{2} + \frac{1}{2\sigma_2^2} E_1[(x - \mu_2)^2]$$

$$p_1(x) \sim N(\mu, \sigma^2) \quad p_2(x) \sim N(0, 1)$$

$$KL(p_1, p_2) = -\log \sigma - \frac{1}{2} + \frac{1}{2}(\sigma^2 + \mu^2)$$

$$= \frac{1}{2}(-\log \sigma^2 + \sigma^2 + \mu^2 - 1)$$

32

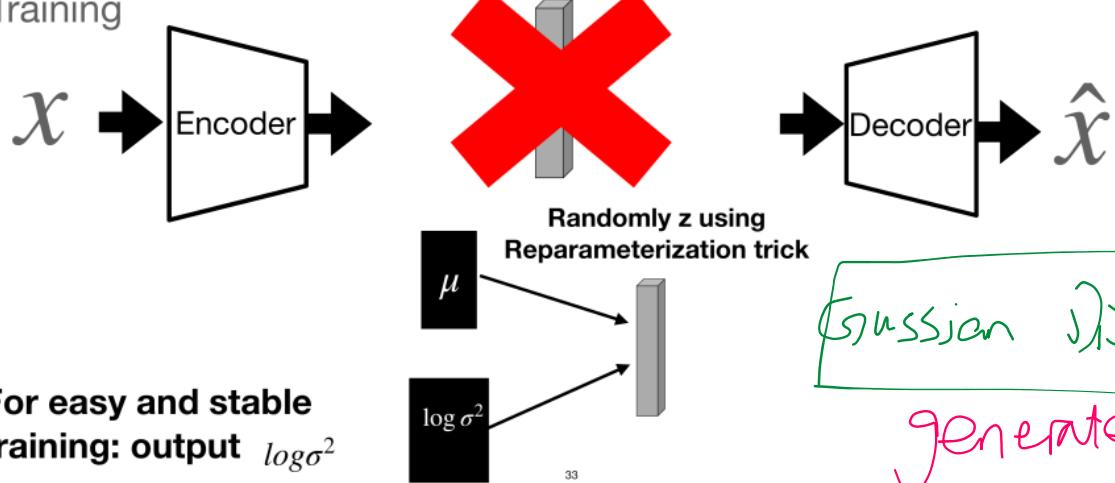
$$-\log \sigma - \frac{1}{2} + \frac{1}{2} E_1[(x^2)]$$

$$x^2 = (x - \mu)^2 + 2x\mu - \mu^2$$

$$\begin{aligned} E(x^2) &= E(x\mu) + 2E(\mu) - \mu^2 \\ &= \sigma^2 + 2\mu^2 - \mu^2 = \sigma^2 + \mu^2 \end{aligned}$$

**(Summary) Different from AE: estimate the distribution of z**

Training



33

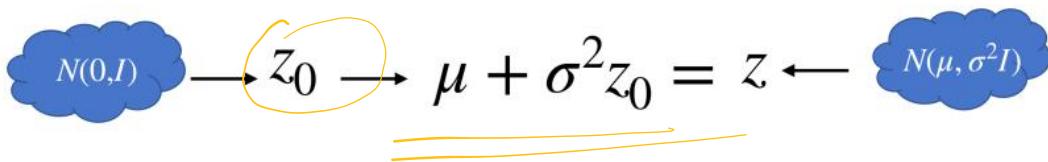
Gaussian distribution  
generate

AE

VAE

GAN

### Sample z: Reparameterization trick to make network differentiable



```
z0 = K.random_normal(shape=(batch_size, z_dim), mean=0., stddev=1.0)
z=mu + K.exp(log_var/2) * z0
```

34

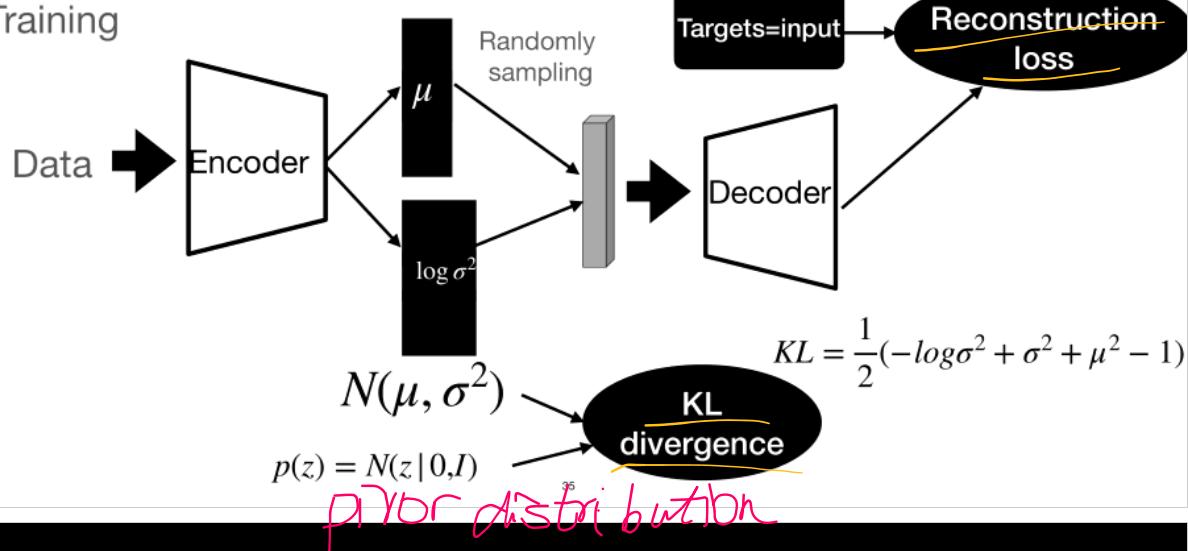
AE

VAE

GAN

### (Summary) Different from AE: Two losses

Training



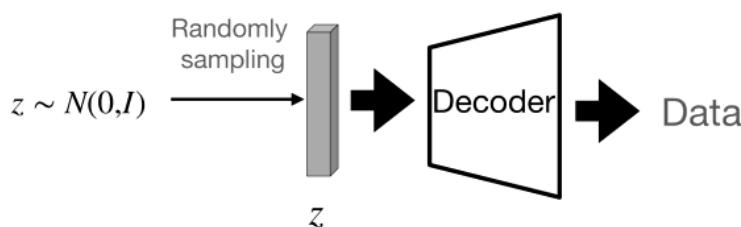
35

AE

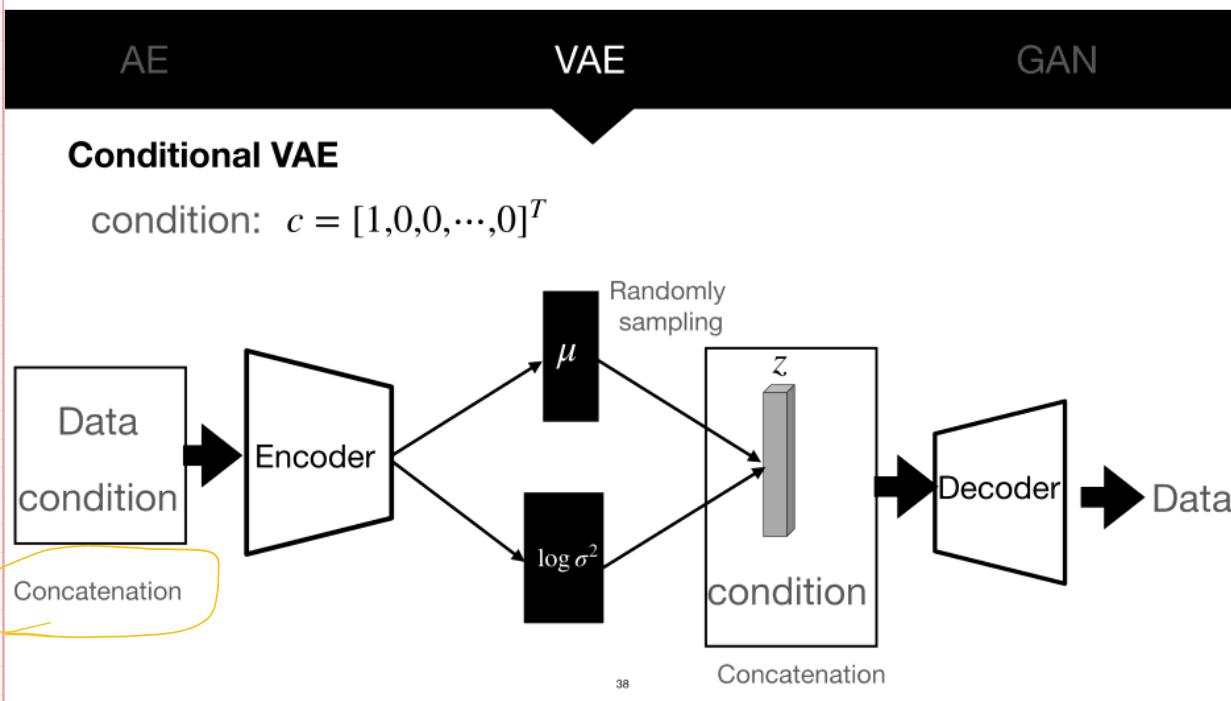
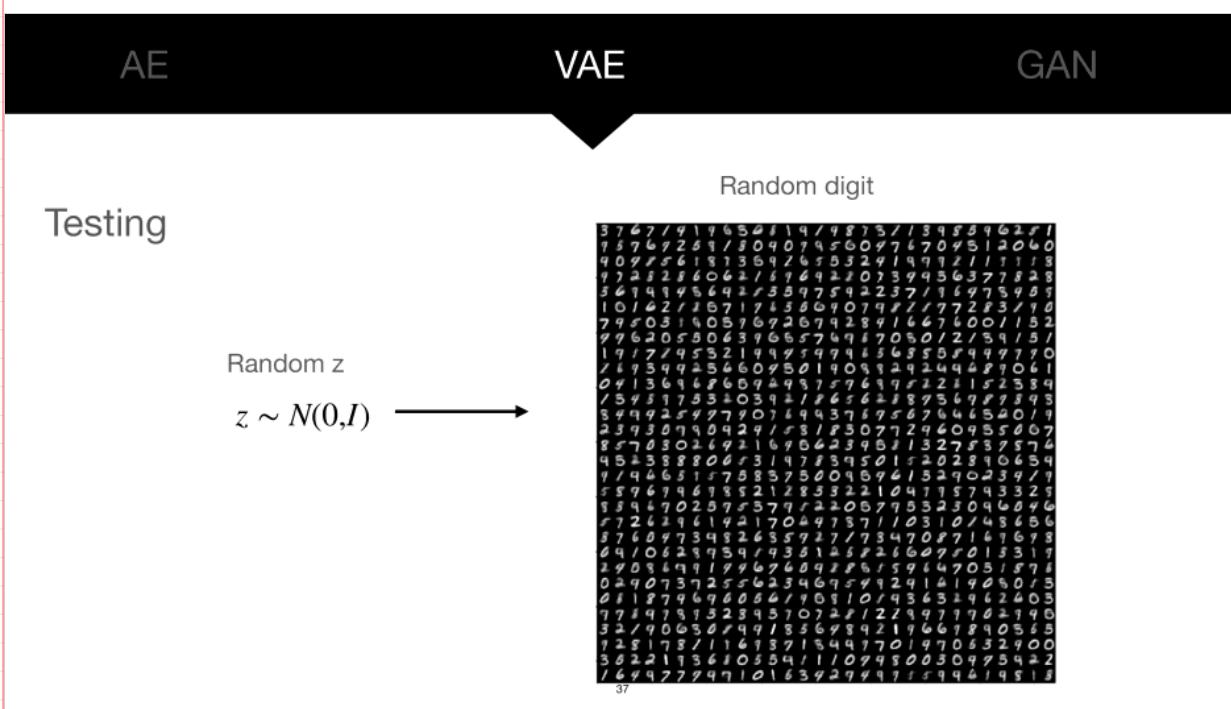
VAE

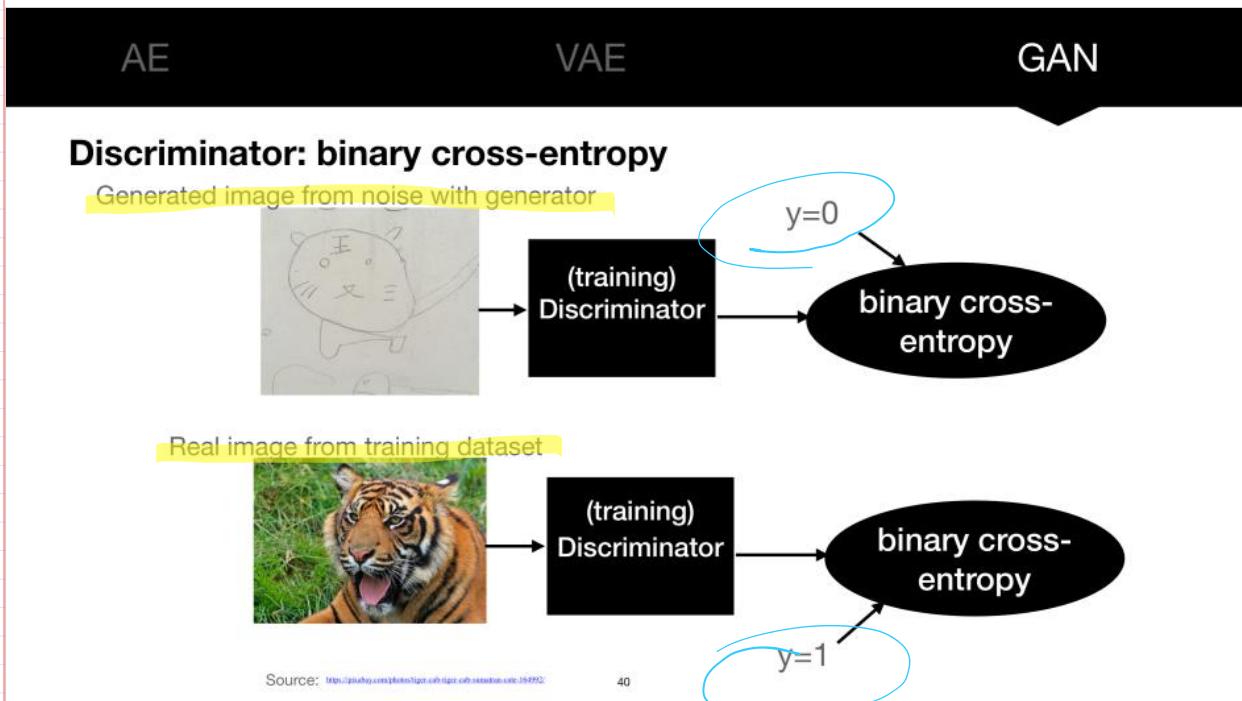
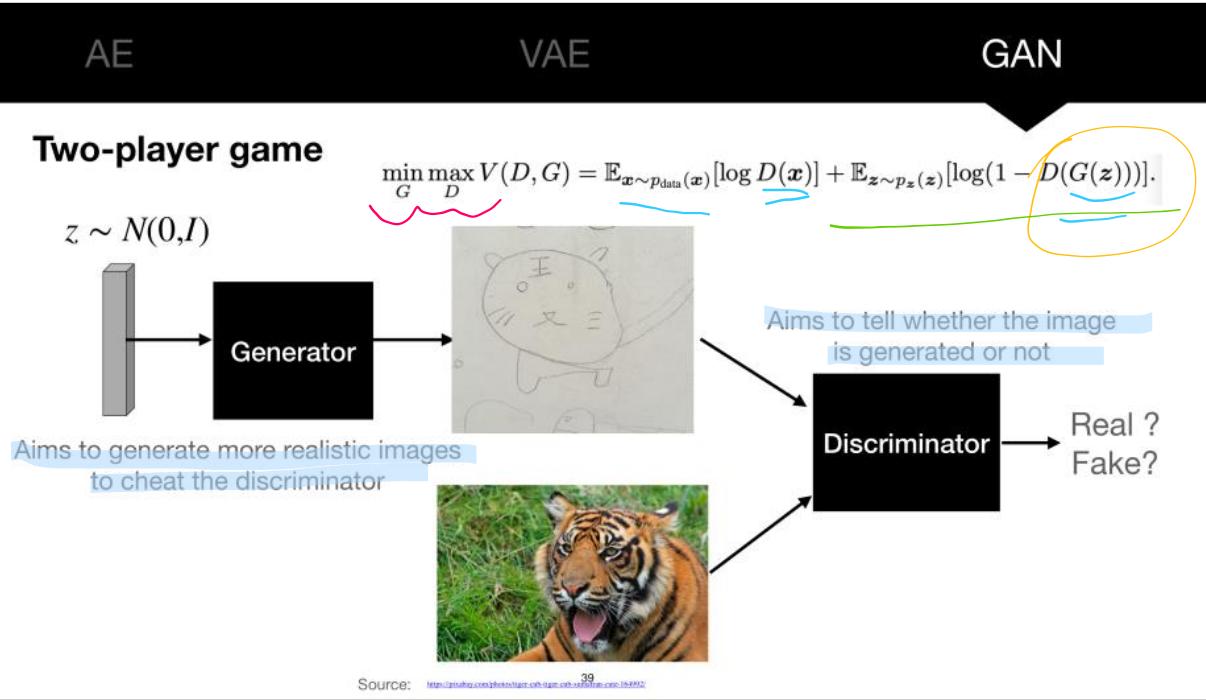
GAN

Testing



36





```
d_loss_real = discriminator.train_on_batch(real_imgs, valid)
d_loss_fake = discriminator.train_on_batch(gen_imgs, fake)
```

41

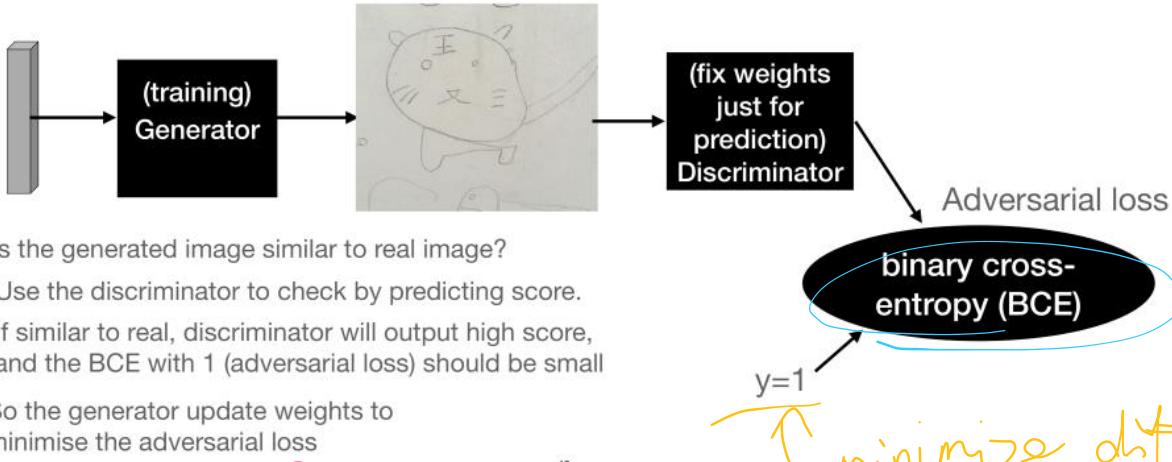
0

AE

VAE

GAN

## Generator: Fool discriminator by adversarial loss



42

AE

VAE

GAN

## Keras code for generator

```
# generate image
z = Input(shape=(self.latent_dim,))
gen_img = generator(z)

# fix weights of discriminator for prediction only
discriminator.trainable = False

# output score for generated image
validity = discriminator(gen_img)

# The combined model: input noise, output prediction of discriminator to calculate loss
combined_model = Model(z, validity)
combined_model.compile(loss='binary_crossentropy', optimizer=optimizer)

valid = np.ones((batch_size, 1))
noise = np.random.normal(0, 1, (batch_size, self.latent_dim))
# update generator to make the output of discriminator to be similar to 1
g_loss = combined_model.train_on_batch(noise, valid)
```

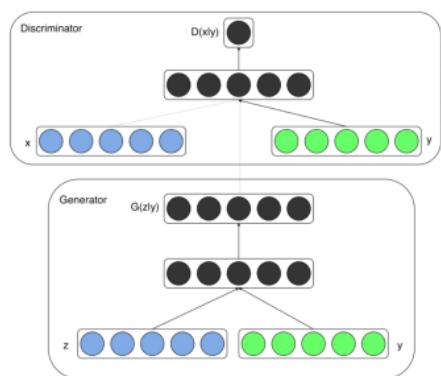
43

AE

VAE

GAN

## Conditional generative adversarial nets



Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).

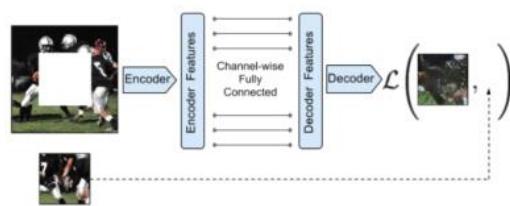
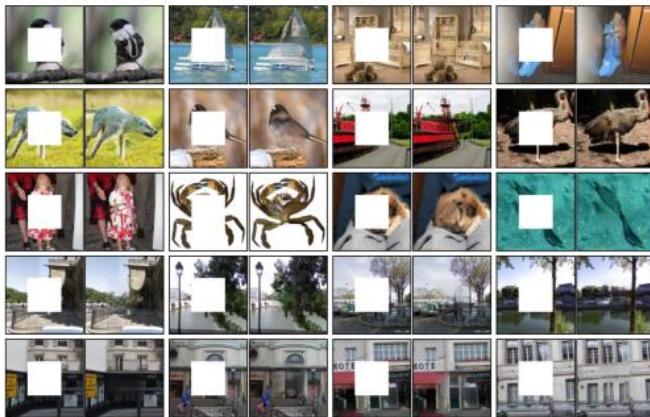
44

AE

VAE

GAN

## Context encoder



Pathak, Deepak, et al. "Context encoders: Feature learning by inpainting." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

45

AE

VAE

GAN

## Context encoder

Generator



(training)  
Generator



Reconstruction  
loss (L2)

(Frozen)  
Discriminator

y=1

Adversarial loss

binary cross-  
entropy

Discriminator is the same as the original gan

Pathak, Deepak, et al. "Context encoders: Feature learning by inpainting." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

46

AE

VAE

GAN

## Other applications



Jin, Yanghua, et al. "Towards the automatic anime characters creation with generative adversarial networks." arXiv preprint arXiv:1708.05509 (2017)

## Other applications

Turning a horse video into a zebra video (by CycleGAN)



[http://www.youtube.com/watch?time\\_continue=1&v=GfTIiR2vM4k&list=PLhjwJLgk](http://www.youtube.com/watch?time_continue=1&v=GfTIiR2vM4k&list=PLhjwJLgk)

48

## Summary

- How to train AE to learn meaningful representation and denoisy image?
- How to train VAE?
- How to train GAN?



[http://www.youtube.com/watch?time\\_continue=1&v=GfTIiR2vM4k&list=PLhjwJLgk](http://www.youtube.com/watch?time_continue=1&v=GfTIiR2vM4k&list=PLhjwJLgk)

Next: Temporal Convolutional Network & Recurrent Neural Network

49

# Temporal Convolutional Network & Recurrent Neural Network

COMP90051 Statistical Machine Learning

QiuHong Ke

Copyright: University of Melbourne

## Sit down or stand up?

We need to learn the temporal information to understand the action



...



...



temporal information: temporal evolution (changes) of the human pose

[https://www.youtube.com/watch?time\\_continue=227&v=tIN3PujXW9p&feature=emb\\_logo](https://www.youtube.com/watch?time_continue=227&v=tIN3PujXW9p&feature=emb_logo)

# Temporal information does matter

Change the order: different actions



[https://www.youtube.com/watch?time\\_continue=227&v=tN3PjIXW9g&feature=emb\\_logo](https://www.youtube.com/watch?time_continue=227&v=tN3PjIXW9g&feature=emb_logo)

3

## What is the next word ?

This game is really **fun**

This cookie is really **tasty**

4

# Like?Dislike? Sentiment analysis

I bought this charger in Jul 2003 and it worked OK for a while. The design is nice and convenient. 😊 However, after about a year, the batteries would not hold a charge. Might as well just get alkaline disposables, or look elsewhere for a charger that comes with batteries that have better staying power. 😞

Amazon product data <http://jmcnally.ucsd.edu/data/amazon/>

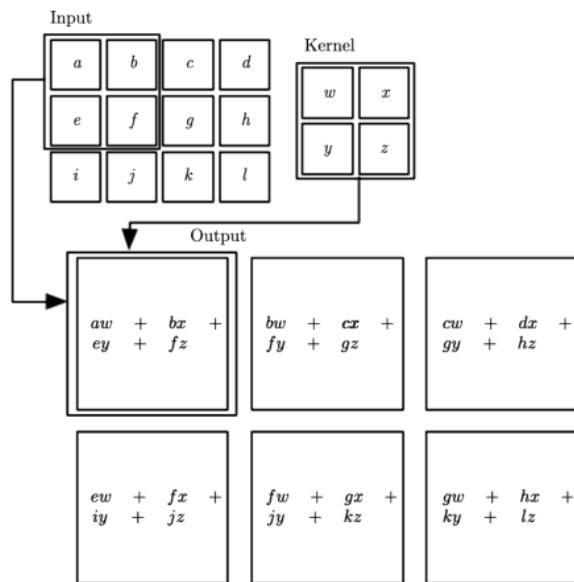
5

## Outline

- Temporal convolutional network (TCN)
- Recurrent neural network (RNN)

6

## Recap: 2D Convolution

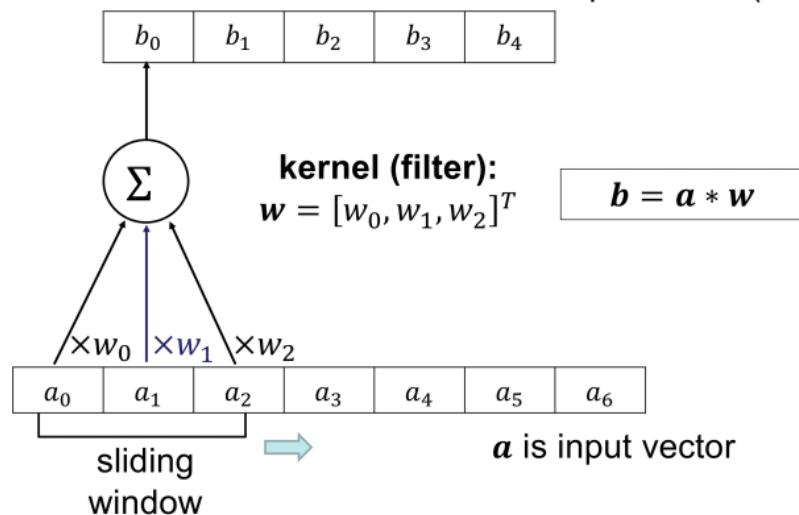


7

Figure 9.1 in Deep learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville

## Recap: 1D Convolution

$\mathbf{b}$  is output vector (feature map)

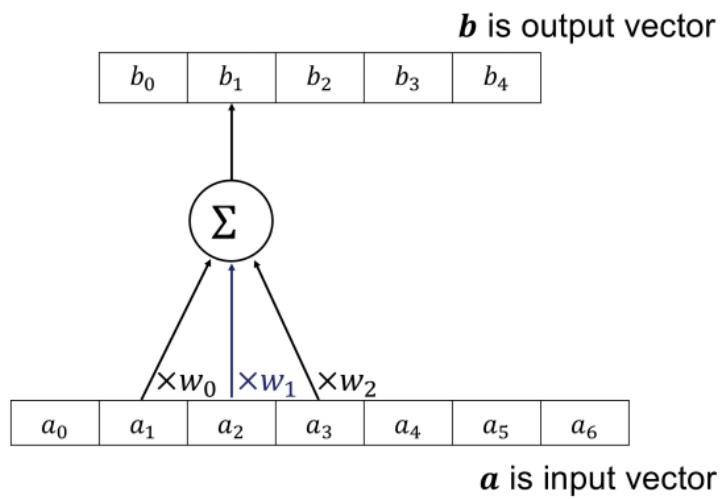


8

TCN

RNN

## Recap: 1D Convolution

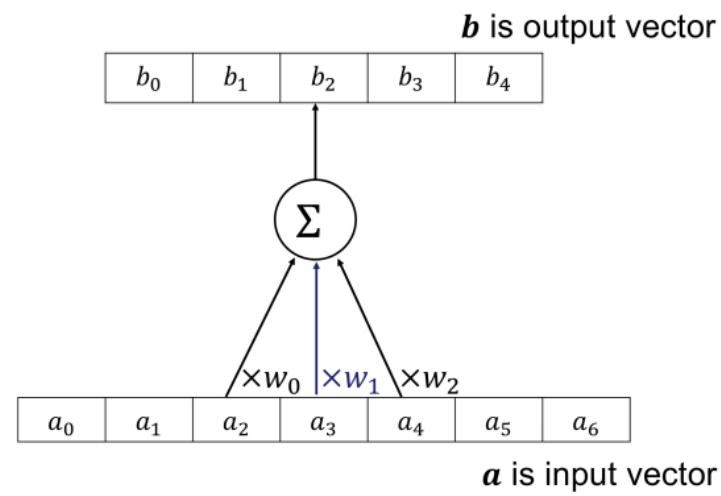


9

TCN

RNN

## Recap: 1D Convolution

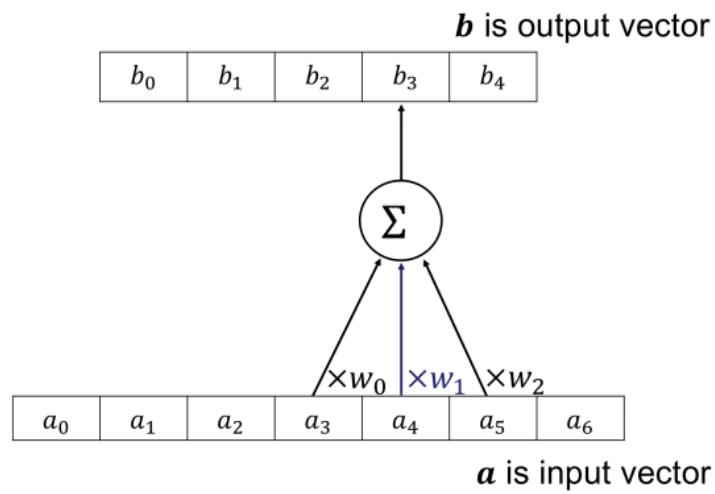


10

TCN

RNN

## Recap: 1D Convolution

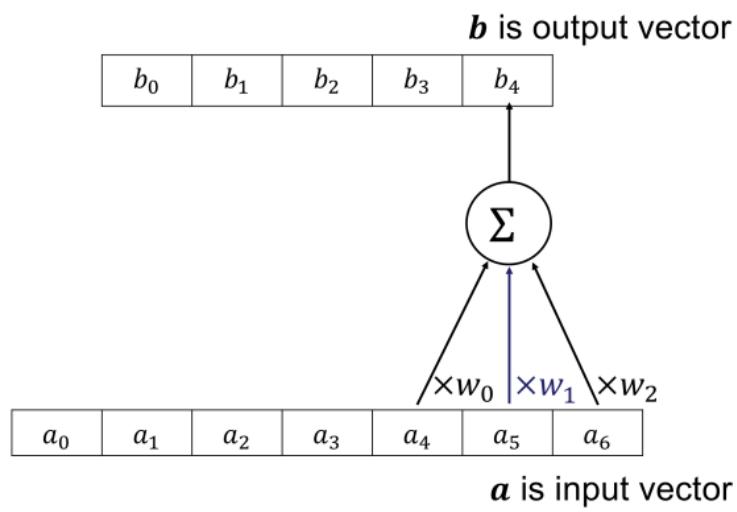


11

TCN

RNN

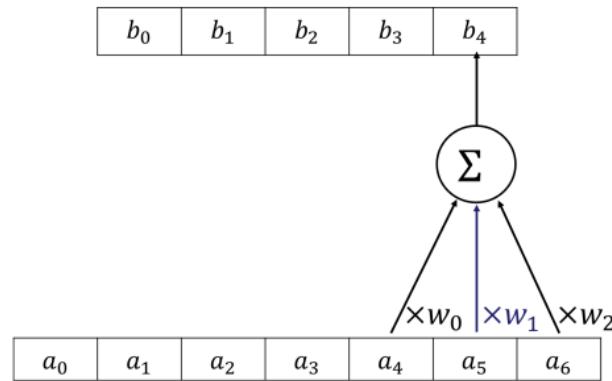
## Recap: 1D Convolution



12

## Temporal convolution: 1D convolution (on the temporal dimension)

1D input feature vector:  
special case of sequence data



A sequence that consists of 7 time-steps:  
One feature in each step

13

## Temporal convolution on a sequence of feature vectors

Input:

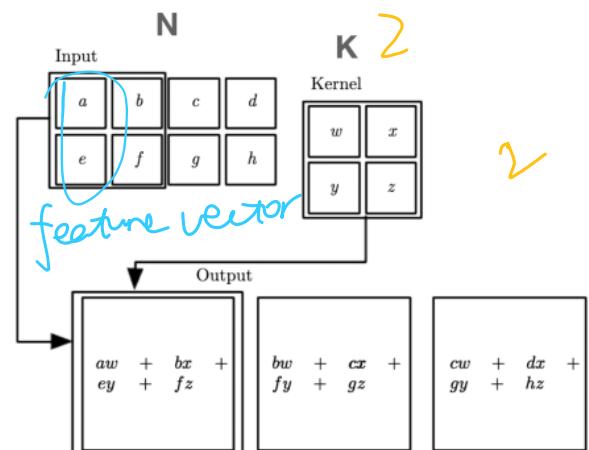
- a sequence of  $N=4$  time-steps
- Each time-step is a feature vector (dimension:  $d=2$  (2 features))

Kernel: a weight matrix ( $dxK$ )

- One dimension is the same as  $d$
- Another dimension: size of temporal convolutional window  $K$

Output: A sequence

- Each time-step is a **scalar (1 kernel)**
- The number of time-steps (same as 2D):
  - **No padding: ceiling  $(N-K+1)/\text{stride}$**
  - **With padding: ceiling  $(N/\text{stride})$**



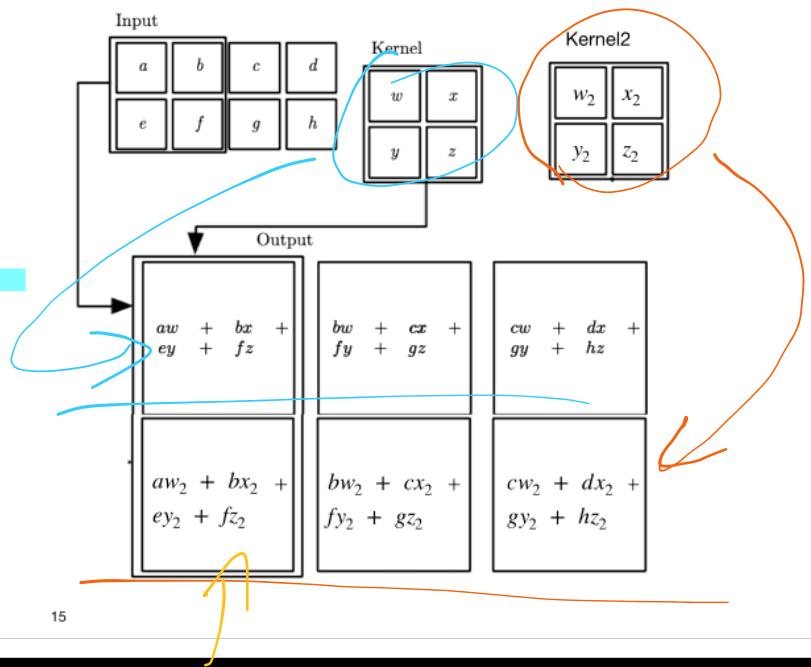
14

## Temporal convolution on a sequence of feature vectors

If there are m kernels in the layer  
 Output: m sequences ==

a sequence of feature vectors:

- The number of time-steps:
  - No padding: ceiling (N-K+1)/stride
  - With padding: ceiling (N/stride)
- Each time-step is a vector
  - Dimension of each vector == kernel number m



## Temporal convolution for text

I bought this charger in Jul 2003 and it worked OK for a while. The design is nice and convenient. However, after about a year, the batteries would not hold a charge. Might as well just get alkaline disposables, or look elsewhere for a charger that comes with batteries that have better staying power.

How to represent words at each time-step before temporal convolution

- One-hot encoding
- Word embedding

encode sequence

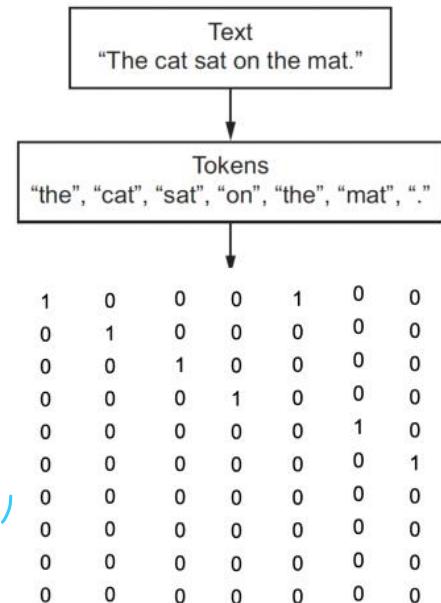
TCN

RNN

## One-hot encoding

- Create a feature vector
  - Dimension of the vector == size of vocabulary
  - If the word is the  $i^{th}$  word in the vocabulary, then the  $i^{th}$  element of the vector is 1, all the others are 0
  - **Sparse and high-dimensional**

```
dic={"the","cat","sat","on","mat",".",  
"these","are","other","words"}
```



TCN

RNN

## Word embedding: learn low-dimensional embedding from data

## One-hot word vectors:

- Sparse
  - High-dimensional
  - Hardcoded



## Word embeddings:

- Dense
  - Lower-dimensional
  - Learned from data

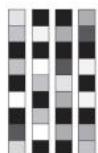


Figure 6.3 in Deep learning with python by Francois Chollet

Figure 6.1 in Deep learning with python by Francois Chollet

## Word embedding: learn low-dimensional embedding from data

```
Emb_layer = tf.keras.layers.Embedding(vocabulary_size, emb_dim)
```

Word index → Embedding layer → Corresponding word vector

Embedding layer:

- Create a weight matrix of d (emb\_dim) rows, N (vocabulary\_size) columns
- Return the  $i^{th}$  column as the feature vector for the  $i^{th}$  word
- Learn word embeddings jointly with the main task or load pretrained word embeddings

Figure 6.4 in Deep learning with python by Francois Chollet

19

## Example

Input:

	I	love	deep	learning
Word embedding	0.1	0.4	0.2	-0.5
	0.3	0.3	0.1	0.3

Kernel:

1	0	0
0	0	1

Output:

element-wise  
multiplication  
and sum

0.2

20

# TCN

# RNN

## Example

Input:

	I	love	deep	learning
Word embedding	0.1	0.4	0.2	-0.5
	0.3	0.3	0.1	0.3

Kernel:

1	0	0
0	0	1

Output:

element-wise  
multiplication  
and sum

0.2    0.7

21

# TCN

# RNN

## Example

Input:

	I	love	deep	learning
Word embedding	0.1	0.4	0.2	-0.5
	0.3	0.3	0.1	0.3

2 Kernels:

0	0	1
0	1	0

Output:

element-wise  
multiplication  
and sum

0.0    0.4  
0.5

22

# TCN

# RNN

## Example

Input:

	I	love	deep	learning
Word embedding	0.1	0.4	0.2	-0.5
	0.3	0.3	0.1	0.3

2 Kernels:

1	2	3
0	0	1
0	1	0

Output:

element-wise multiplication and sum

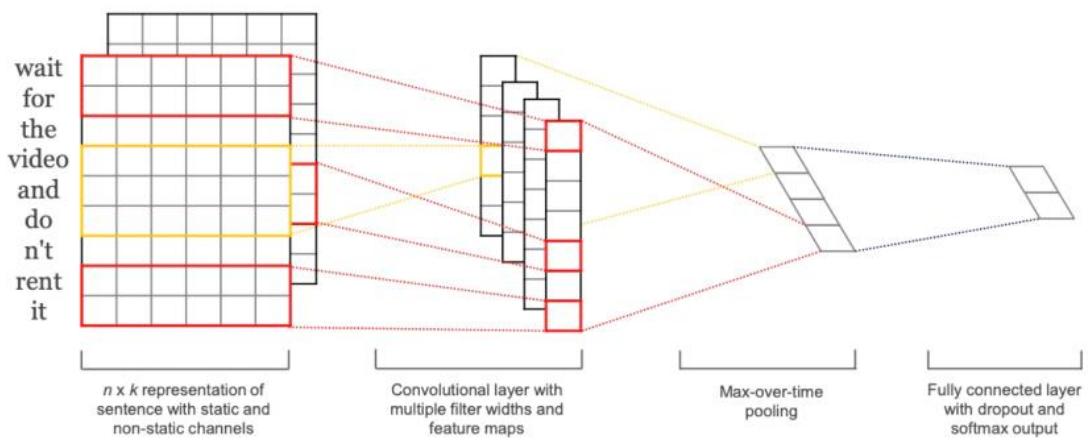
0.0	0.4
0.5	-0.4

23

# TCN

# RNN

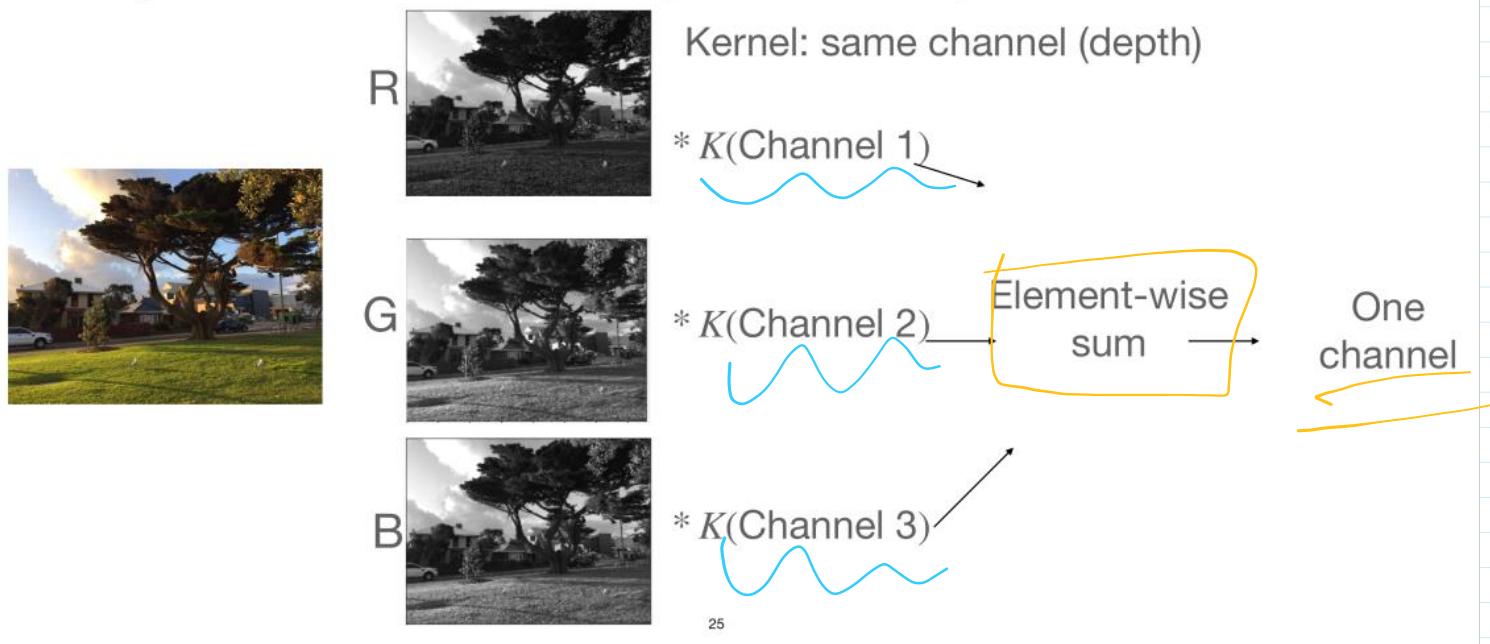
## TCN for Sentence Classification



Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014).

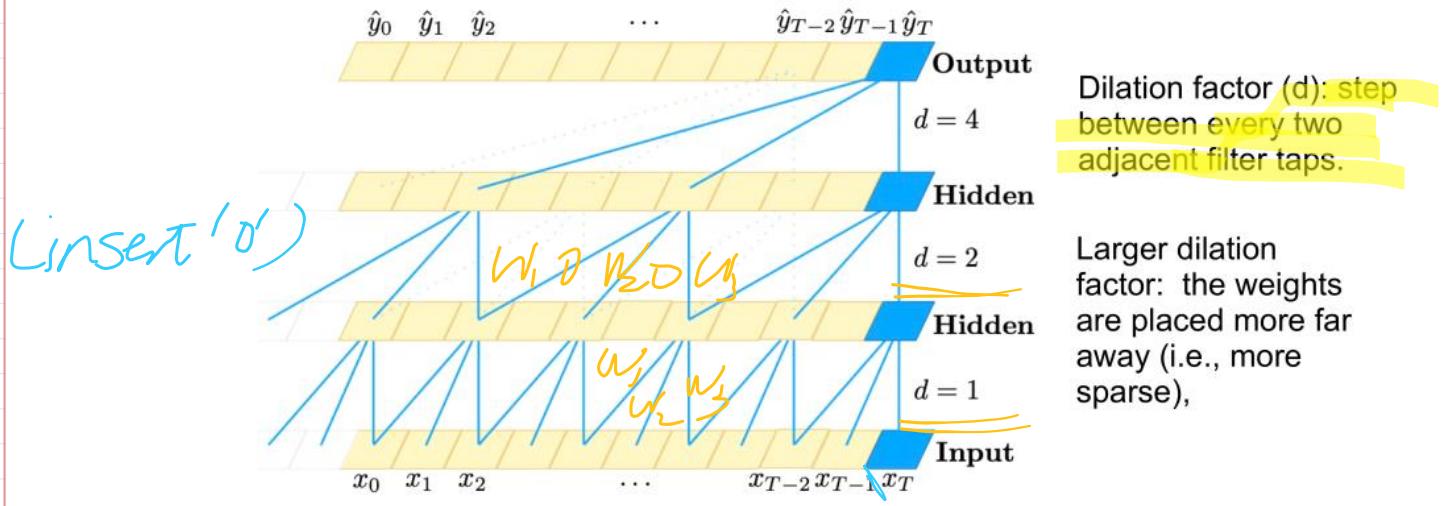
24

## Recap of Convolution on Multiple-channel input



## A special version: Dilated casual convolution

Dilated: expand the alignment of the kernel weights by dilation factor

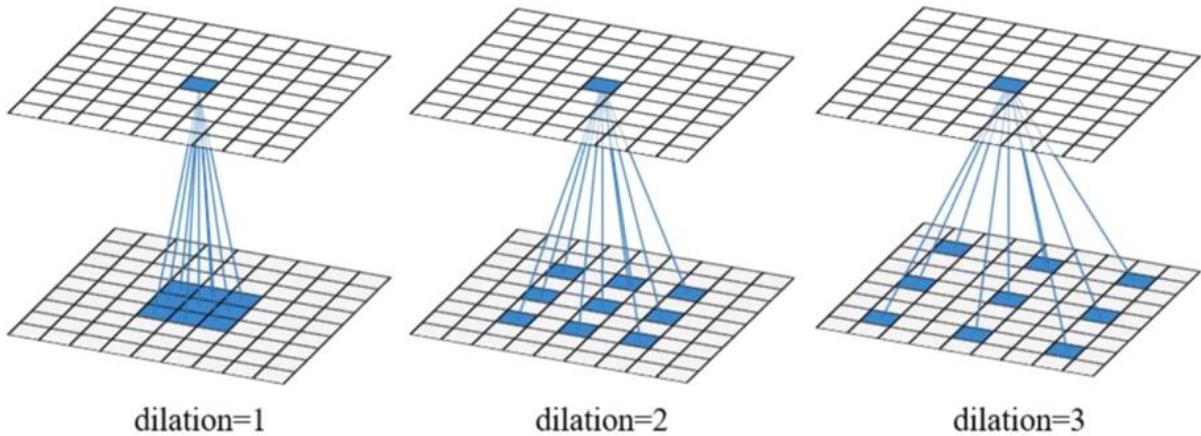


Bai, J. Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." arXiv preprint arXiv:1803.01271 (2018).

TCN

RNN

## Dilated convolution in 2D



27

TCN

RNN

## Increasing receptive fields

- Stacking multiple convolutional layers
- max-Pooling layer
- Dilated convolution

28

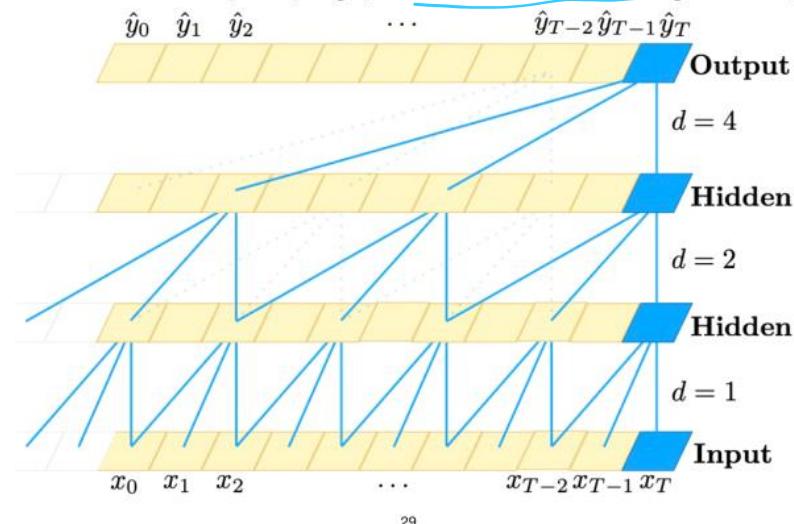
TCN

RNN

## Dilated casual convolution

(no future)

**Causal:** The output at time-step  $t$  does not depend on the input information after time-step  $t$  (e.g., real-time recognition, prediction)



Bai, J. Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." arXiv preprint arXiv:1803.01271 (2018).

29

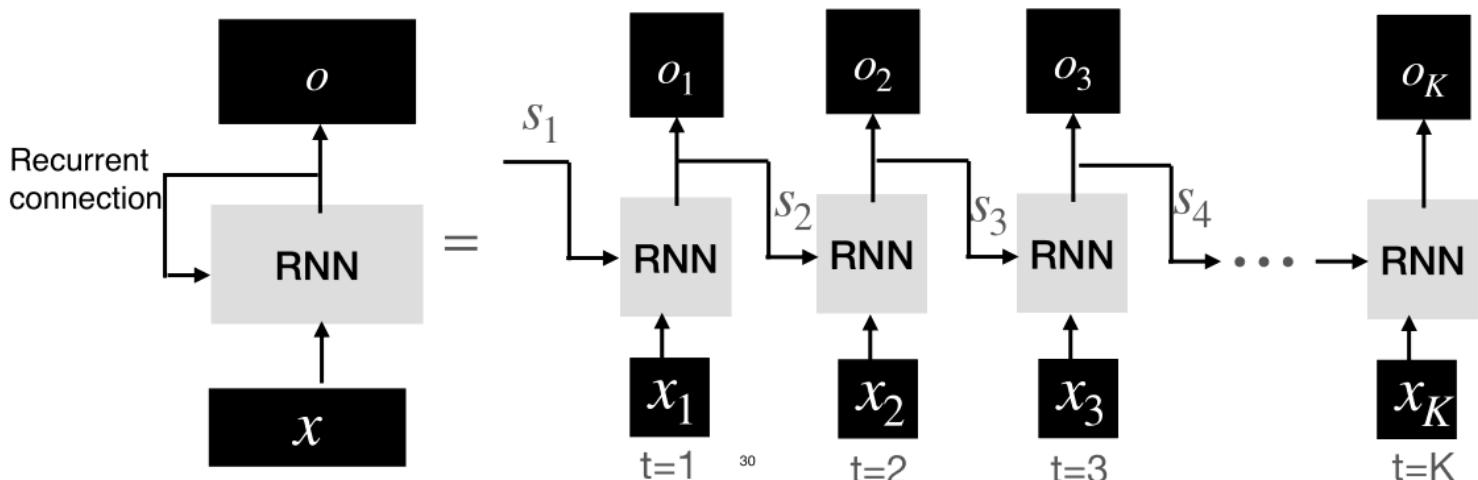
padding  
TCN

RNN

## Networks with loops

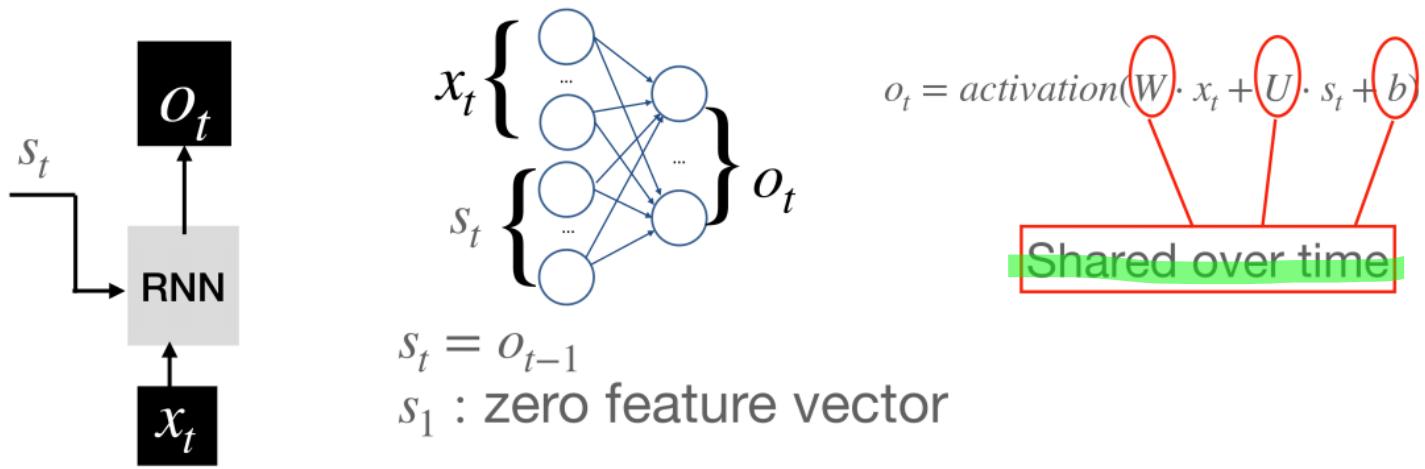
Process sequence step by step

Recurrent connection: the output hidden feature vector (state) of each step is connected to the next step



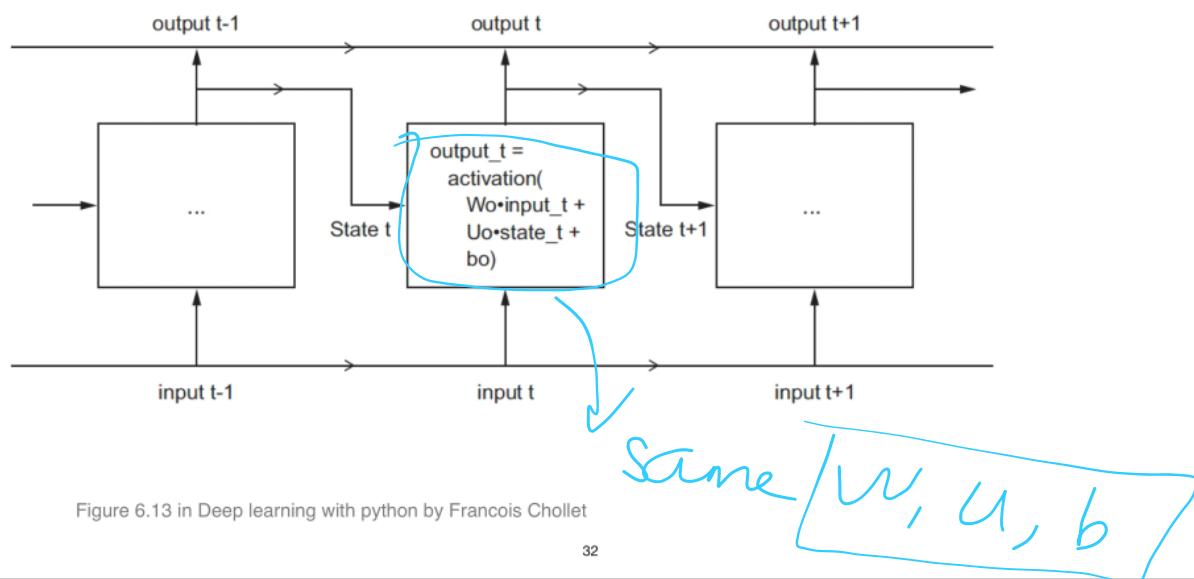
## RNN: Process sequence step by step

At each step  $t$ : combine current timestep  $x_t$  (feature vector of input sequence at timestep  $t$ ) and historical information, i.e., state  $s_t$  (feature vector) to generate  $o_t$



31

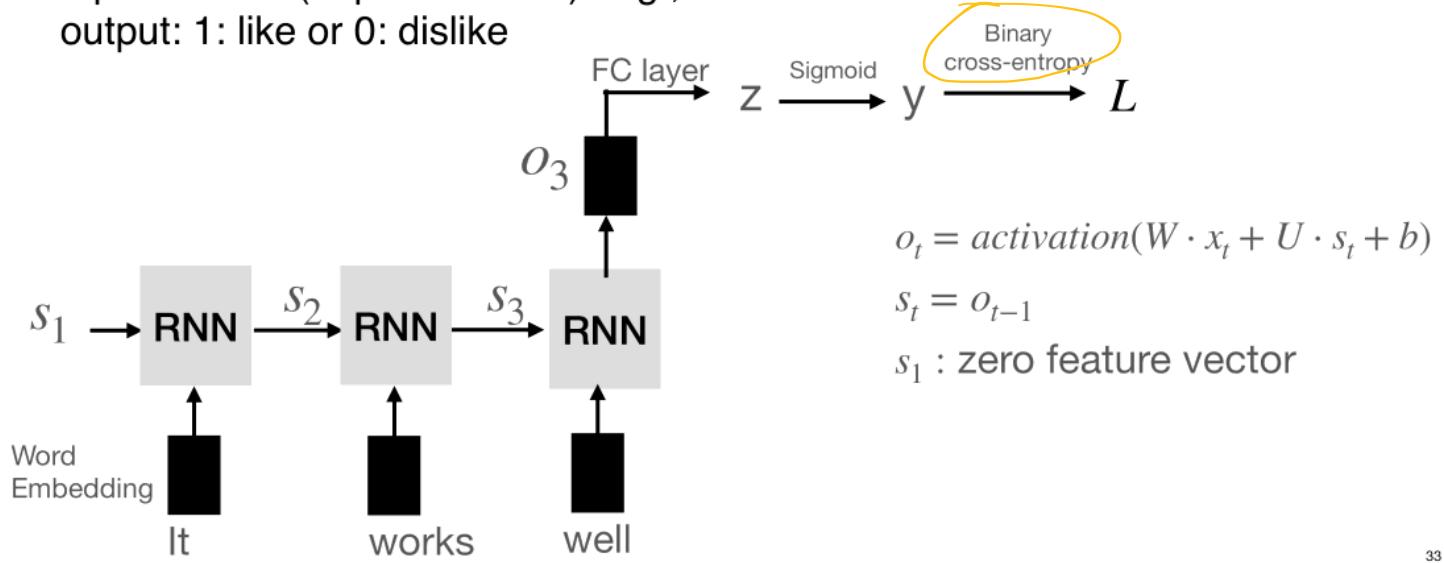
## A simple RNN



## Example: RNN for sentiment analysis

Input: review ( a piece of text) e.g., "It works well"

output: 1: like or 0: dislike



33

## Recap of Chain rule

Given  $z = g(u)$     $u = f(x)$



$$\frac{dz}{dx} = \frac{dz}{du} \frac{du}{dx}$$

Example :  $z = \sin(x^2)$

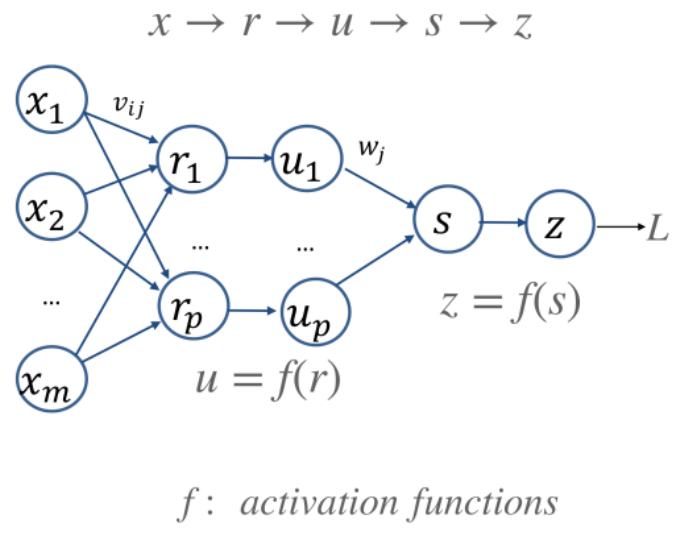
$$\begin{aligned}
 z &= \sin(u) & \frac{dz}{du} &= \cos(u) \\
 u &= x^2 & \downarrow & \\
 & & \frac{dz}{dx} &= \frac{dz}{du} \frac{du}{dx} = 2x \cos(u)
 \end{aligned}$$

34

## Recap of Multi-layer perceptron: Function composition

Forward prediction

$$\begin{aligned} z &= f(s) = \text{sigmoid}(s) = \frac{1}{1 + e^{-s}} \\ s &= \sum_{j=0}^p w_j u_j \\ u_j &= f(r_j) = \text{sigmoid}(r_j) = \frac{1}{1 + e^{-r_j}} \\ r_j &= \sum_{i=0}^m x_i v_{ij} \end{aligned}$$



35

Introduction

Neural network

Gradient descent

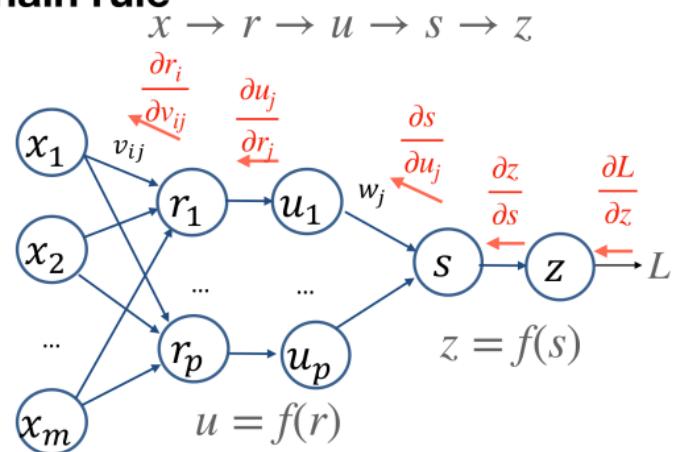
## Recap of Multi-layer perceptron: Chain rule

Backward propagation:

$$\frac{\partial L}{\partial v_{ij}} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial s} \frac{\partial s}{\partial u_j} \frac{\partial u_j}{\partial r_j} \frac{\partial r_j}{\partial v_{ij}}$$

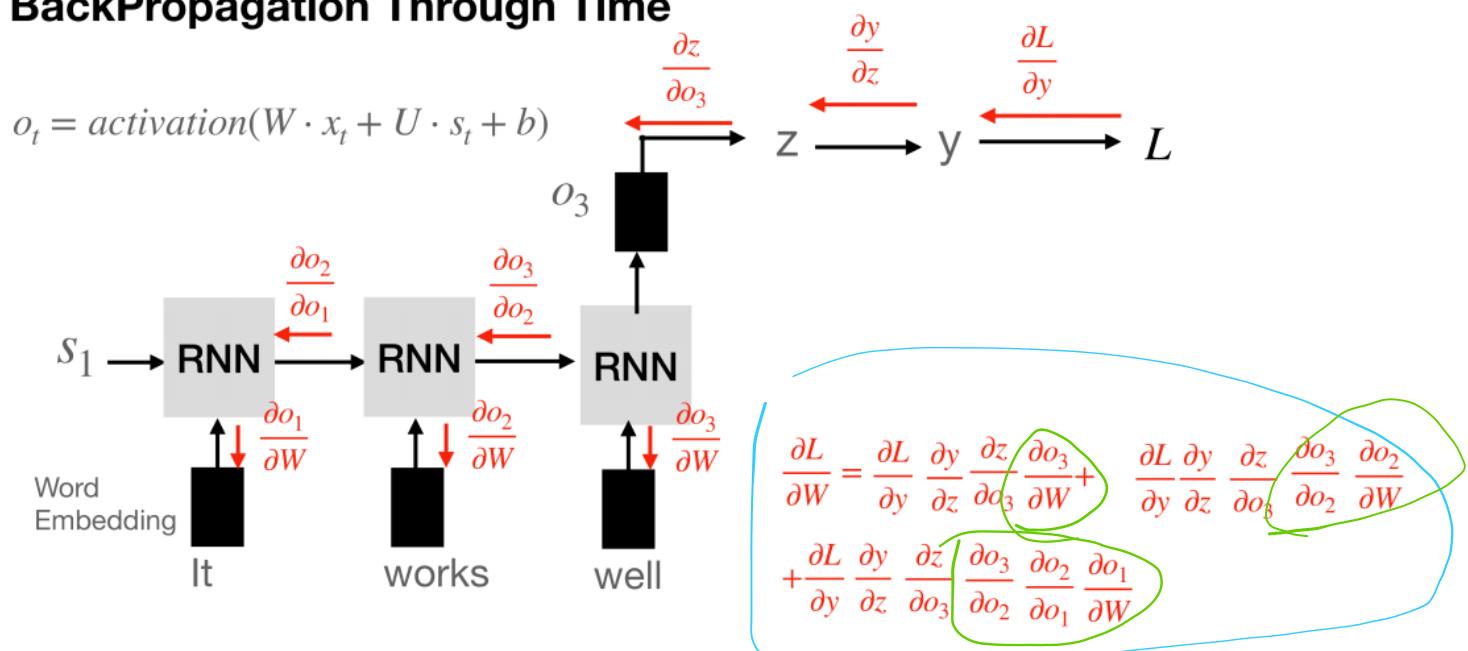
$$\begin{aligned} s &= \sum_{j=0}^p u_j w_j \\ u_j &= \text{sigmoid}(r_j) = \frac{1}{1 + e^{-r_j}} \\ r_j &= \sum_{i=0}^m x_i v_{ij} \end{aligned}$$

Forward



36

## BackPropagation Through Time



37

## A special RNN: Long short-term memory (LSTM)

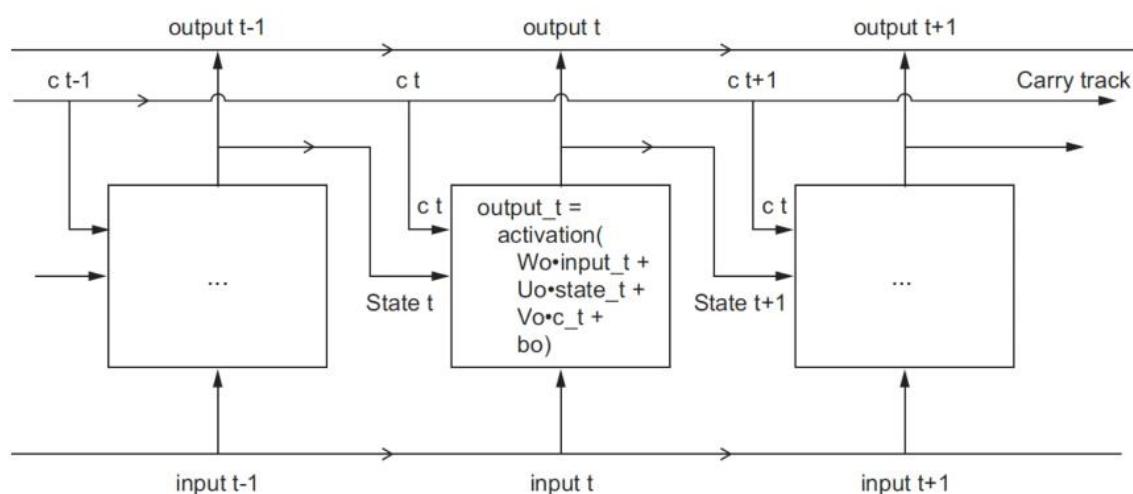


Figure 6.14 in Deep learning with python by Francois Chollet

## A special RNN: Long short-term memory (LSTM)

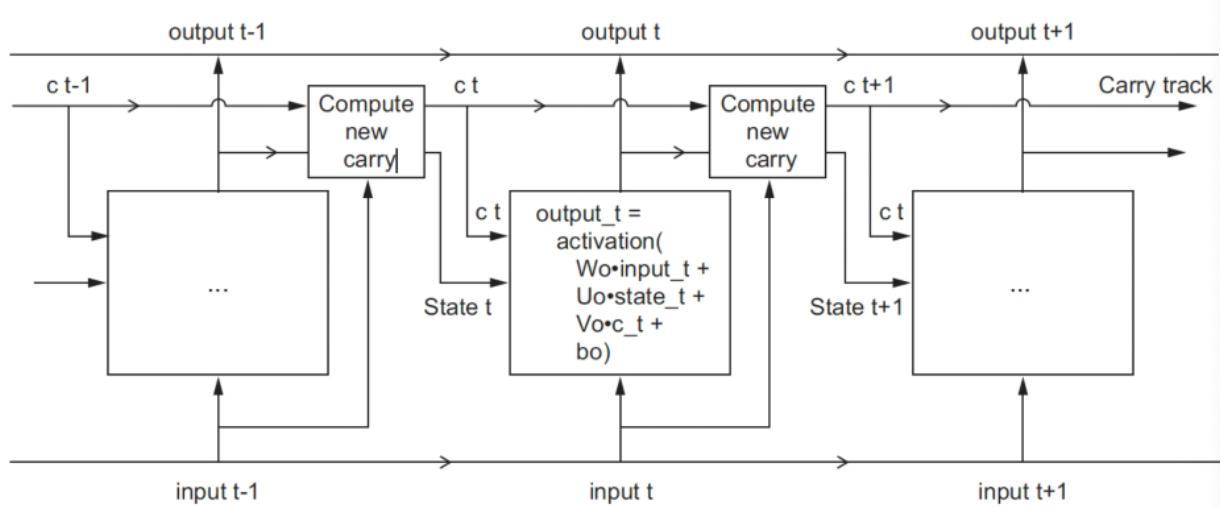
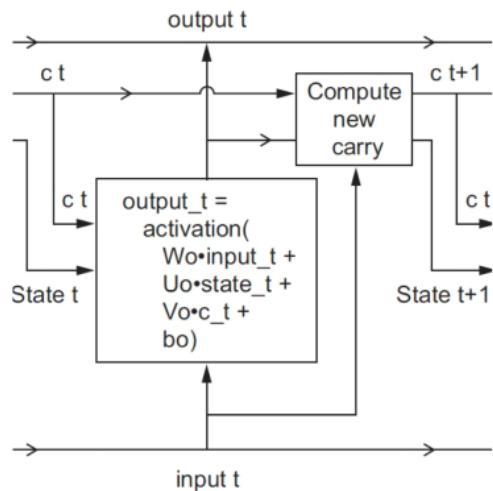


Figure 6.15 in Deep learning with python by Francois Chollet

39

## A special RNN: Long short-term memory (LSTM)



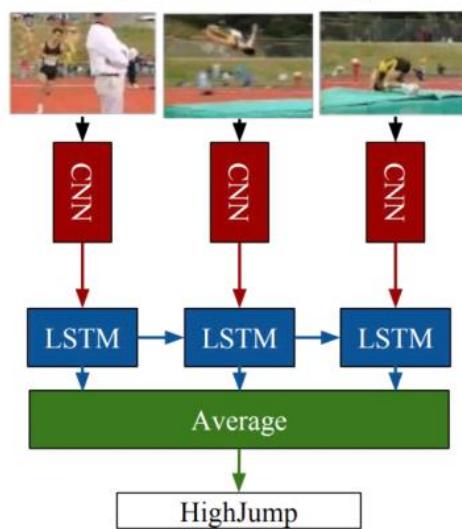
$$\text{Forget gate: } f_t = \text{sigmoid}(W_f \cdot x_t + U_f \cdot s_t + b_f)$$

$$\text{Input gate: } i_t = \text{sigmoid}(W_i \cdot x_t + U_i \cdot s_t + b_i)$$

$$\tilde{c}_{t+1} = \tanh(W_c \cdot x_t + U_c \cdot s_t + b_c)$$

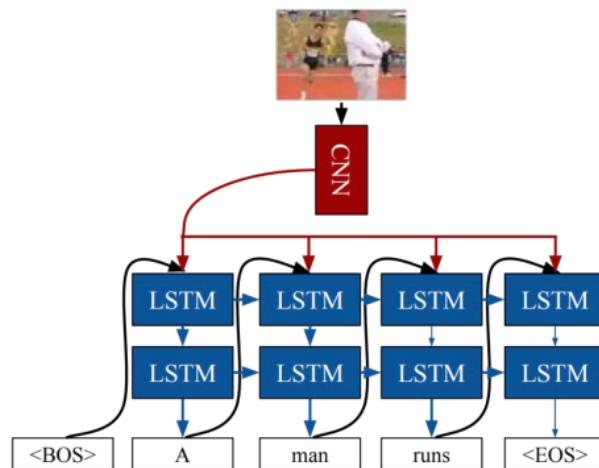
$$c_{t+1} = c_t * f_t + \tilde{c}_{t+1} * i_t$$

40

**Applications**
**Activity Recognition**  
Sequences in the Input


Jeff Donahue et al., Long-term Recurrent Convolutional Networks for Visual Recognition and Description

41

**Applications**
**Image Captioning**  
Sequences in the Output


Jeff Donahue et al., Long-term Recurrent Convolutional Networks for Visual Recognition and Description

42

# Summary

- How does temporal convolution work?
- How does dilated causal convolution work?
- How does recurrent neural network work?
- How does LSTM work?

Next: Graph Convolution Network (Guest lecture)

# Lecture 14

2020年9月15日 星期二 14:24



## 14\_GraphC NN

**Graph Convolution Networks**  
(Deep Learning After You Drop The Camera)

Andrew Cullen  
Copyright: University of Melbourne

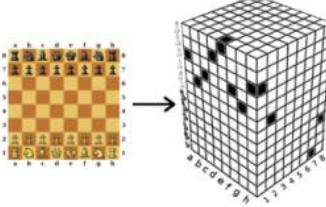


This block contains a presentation slide for Graph Convolution Networks. It features the title 'Graph Convolution Networks' and a subtitle '(Deep Learning After You Drop The Camera)'. Below this is the name 'Andrew Cullen' and the copyright notice 'Copyright: University of Melbourne'. At the bottom left is the University of Melbourne logo.

**CONVOLUTIONAL NEURAL NETWORKS**

CNNs, as you've seen are

- 1. Flexible
- 2. Computationally Efficient
- 3. Expressive
- 4. Give great results.



2/26

This block contains a presentation slide for Convolutional Neural Networks. It starts with the title 'CONVOLUTIONAL NEURAL NETWORKS'. Below it is a statement 'CNNs, as you've seen are' followed by a bulleted list: '1. Flexible', '2. Computationally Efficient', '3. Expressive', and '4. Give great results.'. To the right of the list is a diagram showing a 2D input grid (resembling a chessboard) being processed by a 3D convolutional volume. The volume has axes labeled with letters (a-h) and numbers (1-8). The bottom right corner of the slide shows the page number '2/26'.

## ASSUMPTIONS

But what is assumed when you use a CNN?

The data should be:

1. Regular
2. Dense (or non-sparse)
3. Have some local relational properties
4. Consistent
5. Suitable for taking layers of filtering

3/26

## ASSUMPTIONS

But what is assumed when you use a CNN?

The data should be:

1. Regular
2. Dense (or non-sparse)
3. Have some local relational properties
4. Consistent
5. Suitable for taking layers of filtering

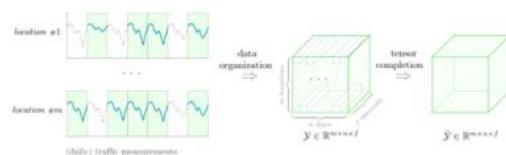
3/26

## BREAKING THESE ASSUMPTIONS?

What happens if these assumptions don't hold?

You can always try to map data into a more regular structure, or downsample your data—Uber does this for demand estimation.

Imputation is another option too, where we try and fill in missing data to make a regular structure



And even if the assumptions hold, it doesn't mean that there's not a better way.

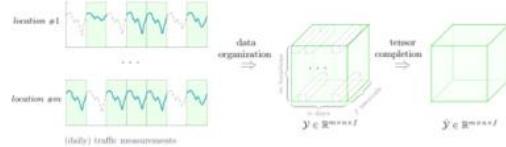
4/26

## BREAKING THESE ASSUMPTIONS?

What happens if these assumptions don't hold?

You can always try to map data into a more regular structure, or downsample your data—Uber does this for demand estimation.

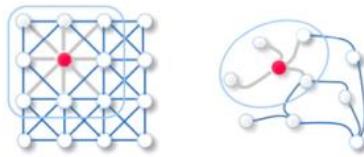
Imputation is another option too, where we try and fill in missing data to make a regular structure



And even if the assumptions hold, it doesn't mean that there's not a better way.

4/26

## GRAPH CONVOLUTIONAL NETWORKS



But what happens if the data isn't nicely structured, like you'd get with a CNN? How do we manage real world data sets?

5/26

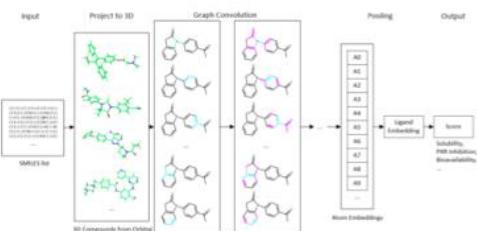
## REAL WORLD DATA

Real world data doesn't often exist on nice grids!

1. Traffic Graphs
2. Sensor network data
3. The relationship between your limbs as you move
4. Taxi demand
5. Social networks

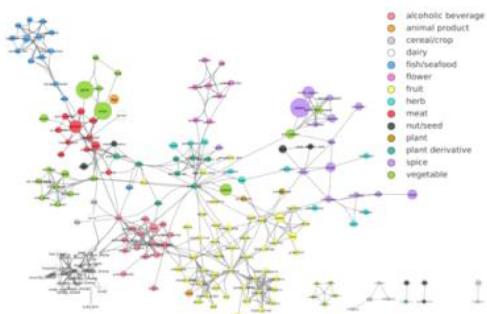
6/26

## REAL WORLD DATA



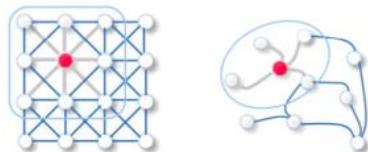
7/26

## REAL WORLD DATA



8/26

## GRAPHS

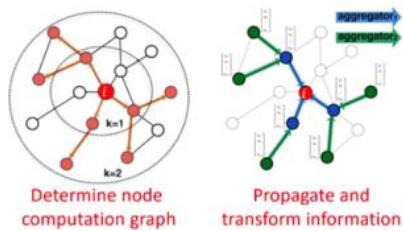


- ▶ Graphs are a collection of Vertices (or Nodes)  $V$  and Edges  $E$ . For deep learning, we presume that the Graph Nodes have attributes  $\mathbf{X} \in \mathbb{R}^{n \times d}$ .
- ▶ It is also possible that the edges have attributes  $\mathbf{X}^e \in \mathbb{R}^{m \times c}$ . These properties can be constant or vary with time.
- ▶ With a Graph Convolution Network, we want to aggregate information from the neighbours of a node.
- ▶ A neighbourhood can be more than just the nodes immediate neighbours.

9/26

## GRAPH NEURAL NETWORKS

With a graph neural network, we want to learn how to aggregate and propagate information across the graph, in a way that helps us extract **local** (node specific) or **global** (graph specific) features.



10/26

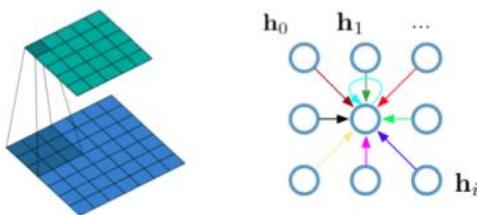
## WHAT EXACTLY IS CONVOLUTION AGAIN?

$$(f * g)(x) = \int_{\mathbb{R}^d} f(y)g(x - y)dy = \int_{\mathbb{R}^d} f(x - y)g(y)dy.$$

- ▶ In general, a convolution is the distortion of one function by another, so they take the properties.
- ▶ In a CNN, we project the data onto the convolution kernel, and extract properties.

11/26

## CNN NETWORKS



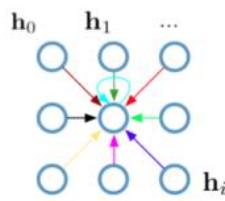
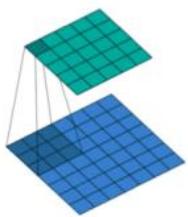
$\mathbf{h}_i \in \mathcal{R}^F$  are the hidden layer activations of each pixel.

Update the hidden layer by

$$\mathbf{h}_j^{(l+1)} = \sigma \left( \sum_{i,j} \mathbf{W}_j^{(l)} \mathbf{h}_i^{(l)} \right)$$

12/26

## CNN NETWORKS



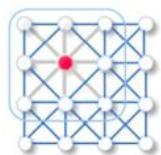
$h_i \in \mathcal{R}^F$  are the hidden layer activations of each pixel.  
Update the hidden layer by

$$h_j^{(l+1)} = \sigma \left( \sum_{i \in N_j} W_i^{(l)} h_i^{(l)} \right)$$

*function weighted sum*

12/26

## GRAPH CONVOLUTIONAL NETWORKS



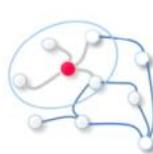
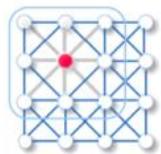
To update:

$$h_0^{(l+1)} = \sigma \left( W_0^{(l)} h_0^{(l)} + W_1^{(l)} h_1^{(l)} + W_2^{(l)} h_2^{(l)} + W_3^{(l)} h_3^{(l)} + W_4^{(l)} h_4^{(l)} \right)$$

Note how each node has a single weight  $W_i$ , rather than a unique weight for each link.

13/26

## GRAPH CONVOLUTIONAL NETWORKS



But how do we account for the fact that some nodes have fewer connections?

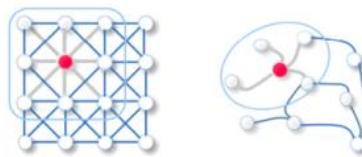
Weight the update, so that:

$$h_i^{(l+1)} = \sigma \left( W_i h_i^{(l)} + \sum_{j \in N_i} f(i, |N_i|) h_j^{(l)} W_i^{(l)} \right)$$

One possible weighting is  $f(i, |N_i|) = \frac{1}{|N_i|}$

14/26

## GRAPH CONVOLUTIONAL NETWORKS



But how do we account for the fact that some nodes have fewer connections?

Weight the update, so that:

$$\mathbf{h}_i^{(l+1)} = \sigma \left( \mathbf{W}_i \mathbf{h}_0^{(l)} + \sum_{j \in \mathcal{N}_i} f(i, |\mathcal{N}_j|) \mathbf{h}_j^{(l)} \mathbf{W}_j^{(l)} \right)$$

One possible weighting is  $f(i, |\mathcal{N}_i|) = \frac{1}{|\mathcal{N}_i|}$

weighting function

14/26

## GRAPH NETWORK EXTENSIONS

$$\mathbf{H}^{(l+1)} = \sigma \left( \mathbf{H}^{(l)} \mathbf{W}_0^{(l)} + \tilde{\mathbf{A}} \mathbf{H}^{(l)} \mathbf{W}_1^{(l)} \right)$$

adjacent matrix

Can be generalised as

$$H^{(l+1)} = \sigma \left( H^{(l)} W_0^{(l)} + \text{Agg} \left( \{h_j^{(l)}, \forall j \in \mathcal{N}_i\} \right) \right)$$

Where the Agg function can be. Examples include max-pooling

$$\text{Agg} = \gamma(\{\mathbf{Q}\mathbf{h}\})$$

where  $\gamma$  is a mean, max, or min function. We can also apply LSTMs to the output (or even, in some cases, on the hidden layers too)

$$\text{Agg} = \text{LSTM}(\mathbf{h})$$

15/26

## EFFICIENT GRAPH UPDATES

Just summing over all the connecting nodes is neither efficient, nor tensor-like. The update procedure can instead be framed as

$$\mathbf{H}^{(l+1)} = \sigma \left( \mathbf{H}^{(l)} \mathbf{W}_0^{(l)} + \tilde{\mathbf{A}} \mathbf{H}^{(l)} \mathbf{W}_1^{(l)} \right)$$

where  $\tilde{\mathbf{A}} = \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{1/2}$  is the Laplacian operator.

$\mathbf{H}^{(l)} = [\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, \dots, \mathbf{h}_N^{(l)}]$ . In this  $\mathbf{A}$  is the graph adjacency matrix, for which  $\mathbf{A}_{i,j} = 1$  if there's a link from node  $i$  to  $j$ ,  $\mathbf{D}$  is the diagonal degree matrix of  $\mathbf{A}$ , where

$$\mathbf{D}_{ii} = \sum_i \mathbf{A}_{ij}$$

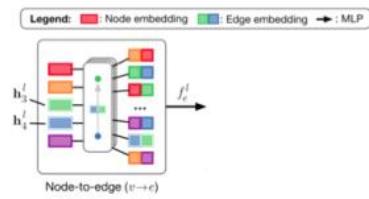
Instead of using  $\tilde{\mathbf{A}} = \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{1/2}$ , can use the modified Laplacian  $\mathbf{L} = \mathbf{I}_N + \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{1/2}$ , so that

$$\mathbf{H}^{(l+1)} = \sigma \left( \mathbf{L} \mathbf{H}^{(l)} \mathbf{W}^{(l)} + \mathbf{b}^{(l)} \right)$$

16/26

## GRAPH EDGE NETWORKS

But real data of interest might exist on the nodes and the edges.

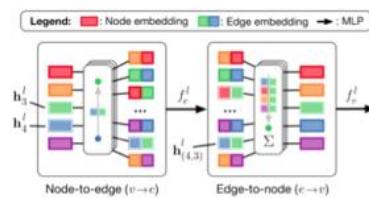


Edge hidden weights can be assigned as

$$\mathbf{h}_{(i,j)}^{(l)} = f_e^{(l)} \left( \mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}; \mathbf{x}_{(i,j)} \right)$$

17/26

## GRAPH EDGE NETWORKS



Edge hidden weights can be assigned as

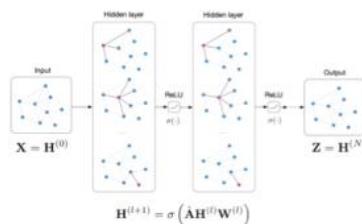
$$\mathbf{h}_{(i,j)}^{(l)} = f_e^{(l)} \left( \mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}; \mathbf{x}_{(i,j)} \right)$$

Vertex hidden weights are then

$$\mathbf{h}_j^{(l+1)} = f_v^{(l)} \left( \sum_{\forall i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^{(l)}; \mathbf{x}_i \right)$$

18/26

## THE FINAL LAYER



The final layer can be processed in a number of ways.  $\text{softmax}(\mathbf{z}_i)$  can be used for node classification, and  $\text{softmax}(\sum \mathbf{z}_i)$  for graph classification. The importance of links can be predicted by  $\sigma(\mathbf{z}_i^T \mathbf{z}_j)$ . Other activation functions can be used for regression.

19/26

## TRAINING

- ▶ With a CNN, all information is loaded into memory. If our data is an image, then we load the entire image at once.
- ▶ In a GCN, we can do the same thing. But a Graph  $G = (E, V)$  has a lot more information to load in, and there's no implicit structure.
- ▶ But with a GCN, we don't need global information to train or run the model. So can randomly select a node, expand over its neighbours, and train over a subset of the graph. Changing this subset over training allows global behaviour to be learned.

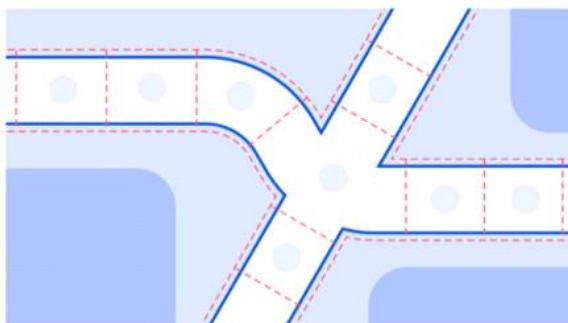
20/26

## TRAINING

- ▶ With a CNN, all information is loaded into memory. If our data is an image, then we load the entire image at once.
- ▶ In a GCN, we can do the same thing. But a Graph  $G = (E, V)$  has a lot more information to load in, and there's no implicit structure.
- ▶ But with a GCN, we don't need global information to train or run the model. So can randomly select a node, expand over its neighbours, and train over a subset of the graph. Changing this subset over training allows global behaviour to be learned.

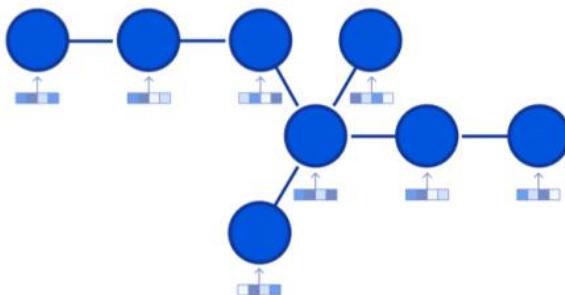
20/26

## CASE STUDY: GOOGLE MAPS



21/26

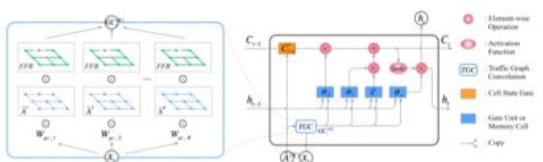
## CASE STUDY: GOOGLE MAPS



Create the graph network, and then at each node enter a sequence of time series data.

22/26

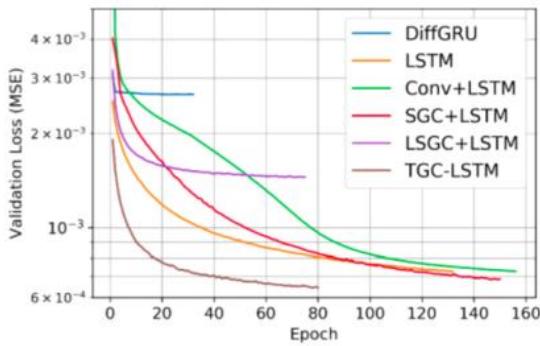
## CASE STUDY: GOOGLE MAPS



Behaviour across the whole network can be described by a Graph Convolution Network, embedded within an LSTM-like structure.

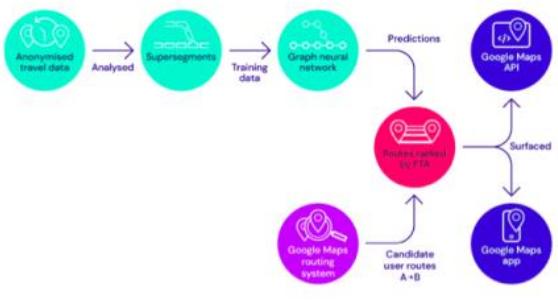
23/26

## CASE STUDY: GOOGLE MAPS



24/26

## CASE STUDY: GOOGLE MAPS



25 / 26

## OVERVIEW

The biggest feature of Graph Neural Networks is their flexibility. A GNN can perform almost any operation that you could see with a traditional CNN, but with extra flexibility in how you design the network.

This flexibility in turn allows ML practitioners to both approach a greater range of problems, and to tackle traditional problems with the potential for greater accuracy.

With careful design, even with the additional overhead of managing the graph, they also hold the potential to be more scalable to large data sets than other NN approaches.

26 / 26

## Lecture 15

2020年9月22日 星期二 14:02



15\_multipli  
cative\_we...

# Lecture 15. Learning with expert advice

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



THE UNIVERSITY OF  
MELBOURNE

Copyright: University of Melbourne

## This lecture

- Learning from expert advice / multiplicative weights
  - \* Learner listens to some/all experts making predictions
  - \* True outcomes are ADVERSARIAL!
  - \* Learner updates weights over experts based on their losses
  - \* Algorithms all forms of “multiplicative weights”
  - \* Nice clean bounds on total mistakes/loss: by “potential function” technique
- Infallible expert (one always perfect)
  - \* Majority Vote Algorithm
- Imperfect experts (none guaranteed perfect) – increasingly better
  - \* Weighted Majority Vote Algorithm by Halving
  - \* Weighted Majority Voting by General Multiplicative Weights
  - \* Probabilistic Experts Algorithm

2

## An infallible expert and the Majority Algorithm

*Warming up example*

3

## Warm-up: Case of the infallible expert

- Experts  $E_1, \dots, E_n$  predict the stock market daily
  - \* Each expert prediction is binary: stocks will go up/down
- Learner's game, daily:
  - \* Observe predictions of all experts
  - \* Make a prediction of its own
  - \* Observe outcome (could be anything!)
  - \* Goal: minimise number total mistakes
- Infallible expert assumption:
  - \* 1 or more experts makes no mistakes



NYSE. CCA: skeeze

4

*by assumption*

## Infallible Expert Algorithm: Majority Vote

1. Initialise set of experts who haven't made mistakes  $E = \{1, \dots, n\}$
2. Repeat per round
  - a) Observe predictions  $E_i$  for all  $i \in \{1, \dots, n\}$
  - b) Make majority prediction  $\arg \max_{y \in \{-1,1\}} \sum_{i \in E} 1[E_i = y]$
  - c) Observe correct outcome
  - d) Remove mistaken experts from  $E$



CCA3.0: Krisada, Noun Project

5

# Mistake Bound for Majority Vote

**Proposition:** Under infallible expert assumption, majority vote makes total mistakes  $M \leq \log_2 n$

Intuition: Halving (e.g. tree data structures!)? Expect to see log

## Proof

- Loop invariant: If algorithm makes a mistake, then at least  $|E|/2$  experts must have been wrong
- I.e. for every incorrect prediction,  $|E|$  reduced by at least half. I.e. after  $M$  mistakes,  $|E| \leq n/2^M$
- By infallibility, at all times  $1 \leq |E|$
- Combine to  $1 \leq |E| \leq n/2^M$ , then solve for  $M$ .

$|E|$  is the potential function

6

# How is this “online learning”?

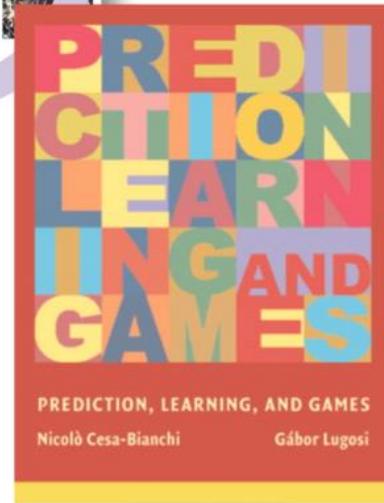
## Learning

- Weights on which experts are worth listening to
- (Infallible case: 0/1 weights)
- Making predictions/taking actions
- Incurring loss (so far 0/1)
- IID “Distribution” replaced by adversarial outcomes



## Online

A repeated game



<http://cesa-bianchi.di.unimi.it/predbook/>

7

## Mini Summary

- Learning with expert advice paradigm
  - \* Abstraction of online learning problem
  - \* Adversarial feedback
  - \* Later: Applications abound
- Bounds on mistakes (later losses) “easy”
  - \* Involve “potential function” technique
  - \* Later: interested in scaling with best expert performance

Next: Imperfect experts. Down weight don't drop bad experts

8

## Imperfect experts and the Halving Algorithm

*Similar proof technique; similar algorithm;  
much more interesting setting*

9

## No one's perfect

- No more guarantee of an infallible expert
- What breaks?
  - \* We could end up with  $E = \emptyset$ , how to predict then?
  - \* No sense: "Zero tolerance" dropping experts on a mistake
- Very general setting / very few assumptions
  - \* Not assuming anything about expert error rates
  - \* Not assuming anything about correlation of expert errors
  - \* Not assuming anything about outcome observations. Not even stochastic (could be adversarial!)

10

## Imperfect experts: Halving Algorithm

1. Initialise  $w_i = 1$  weight of expert  $E_i$
2. Repeat per round
  - a) Observe predictions  $E_i$  for all  $i \in \{1, \dots, n\}$
  - b) Make weighted majority prediction  

$$\arg \max_{y \in \{-1, 1\}} \sum_{i \in E} w_i \mathbf{1}[E_i = y]$$
  - c) Observe correct outcome
  - d) Downweigh each mistaken expert  $E_i$   

$$w_i \leftarrow w_i / 2$$



CC-BY-SA: Krisada, Noun Project

11

# Mistake Bound for Halving

Proposition: If the best expert makes  $m$  mistakes, then weighted majority vote makes  $M \leq 2.4(m + \log_2 n)$  mistakes.

## Proof

- Invariant: If algorithm makes a mistake, then weight of wrong experts is at least half the total weight  $W = \sum_{i=1}^n w_i$
- Weight of wrong experts reduced by  $1/2$ , therefore total weight reduced by at least  $3/4$ . i.e. after  $M$  mistakes,  $W \leq n(3/4)^M$
- Best expert  $E_i$  has  $w_i = (1/2)^m$
- Combine to  $(1/2)^m = w_i \leq W \leq n(3/4)^M$
- Taking logs  $-m \leq \log_2 n + M \log_2 (3/4)$ , solving  $M \leq \frac{m + \log_2 n}{\log_2 (4/3)}$

12

# Compare, compare: What's going on?

- Price of imperfection (vs. infallibility) is  $\mathcal{O}(m)$ 
  - \* Infallible case:  $M \in \mathcal{O}(\log n)$
  - \* Imperfect case:  $M \in \mathcal{O}(m + \log n)$
- Scaling to many experts is no problem
- Online learning vs. PAC frameworks

	Modelling of losses	Ultimate goal
PAC	i.i.d. losses (due to e.g. Hoeffding)	(For ERM; L6c) Small estimation error $R[f_m] - R[f^*]$ . Bounded in terms of family's VC dimension
Online learning	Adversarial/arbitrary losses	Small $M - m$ . Bounded in terms of number of experts.

13

## Mini Summary

- Imperfect expert setting
  - \* Don't drop bad experts, just halve their weight
  - \* Predict by weighted majority, not simply majority
  - \* Mistake bound follows similar "potential function" pattern!
- Learning with expert advice paradigm
  - \* Key difference to PAC is adversarial feedback
  - \* Similarity: Also concerned with performance relative to "best in class"

Next: Imperfect experts continued. Generalising halving.

14

## From Halving to Multiplying weights by $1 - \varepsilon$

*Generalising weighted majority.*

15

## Useful (but otherwise boring) inequalities

- Lemma 1: For any  $\varepsilon \in [0, 0.5]$ , we have  

$$-\varepsilon - \varepsilon^2 \leq \log_e(1 - \varepsilon) < -\varepsilon$$

- Proof:
  - \* Upper bound by Taylor expansion, dropping all but first term (as they're negative)
  - \* Lower bound by convexity of  $\exp(-\varepsilon - \varepsilon^2)$

- Lemma 2: For all  $\varepsilon \in [0, 1]$  we have,

$$1 - \varepsilon x > \begin{cases} (1 - \varepsilon)^x, & \text{if } x \in [0, 1] \\ (1 + \varepsilon)^{-x}, & \text{if } x \in [-1, 0] \end{cases}$$

- Proof: by convexity of the RHS functions

16

## Weighted Majority Vote Algorithm

1. Initialise  $w_i = 1$  weight of expert  $E_i$

2. Repeat per round

a) Observe predictions  $E_i$

for all  $i \in \{1, \dots, n\}$

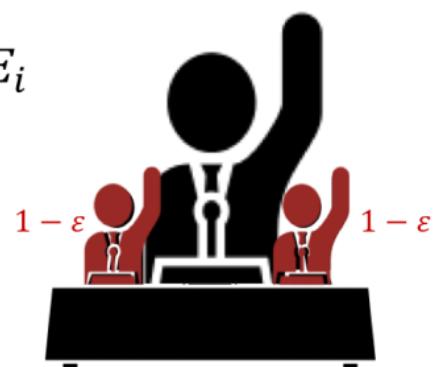
b) Make weighted majority prediction

$$\arg \max_{y \in \{-1, 1\}} \sum_{i \in E} w_i \mathbf{1}[E_i = y]$$

c) Observe correct outcome

d) Downweigh each mistaken expert  $E_i$

$$w_i \leftarrow (1 - \varepsilon) w_i$$



CC-BY-SA: Krisada, Noun Project

$$0 < \varepsilon < 1$$

17

↓  
hyper-parameter

## Mistake Bound

Proposition: If the best expert makes  $m$  mistakes, then  $(1 - \varepsilon)$ -weighted majority makes  $M \leq 2(1 + \varepsilon)m + (2\log_e n)/\varepsilon$  mistakes.

### Proof

Bound improves dependence on  $m$  compared to halving. Why?

- Whenever learner mistakes, at least half of total weight reduced by factor of  $1 - \varepsilon$ . So after  $M$  mistakes,  $W \leq n(1 - \varepsilon/2)^M$
- Best expert  $E_i$  has  $w_i = (1 - \varepsilon)^m$
- Combine to  $(1 - \varepsilon)^m = w_i \leq W \leq n(1 - \varepsilon/2)^M$
- Taking logs:  $m\log_e(1 - \varepsilon) \leq \log_e n + M\log_e(1 - \varepsilon/2)$
- Lemma 1 replaces both  $\log_e(1 - \varepsilon)$ :  $-m(\varepsilon + \varepsilon^2) \leq \log_e n - M\varepsilon/2$
- Solving for  $M$  proves the bound.

18

## Dependence in $m$ provably near optimal!

- New to lower bounds? example shows an analysis or even an algorithm can't do better than some limit
- Weighted majority almost achieves  $2m$  dependence, with  $2(1 + \varepsilon)m$  (considering no. experts fixed)
- Example with  $M = 2m$ 
  - \* Consider  $n = 2$  with  $E_1$  ( $E_2$ ) correct on odd (even) days
  - \* Then best expert makes mistakes half the time
  - \* But after 1<sup>st</sup> round, for any  $\varepsilon$ , majority vote is wrong all the time, as incorrect expert gets more than half weight
- Consequence? Can't improve the constant 2 factor in  $m$

19

## Mini Summary

- Imperfect expert setting continued...
- From halving to multiplicative weights!
  - \* Mistake bound proved as usual via “potential function” trick
  - \* Bound’s dependence on best expert improved to  $2 + \varepsilon$  factor
- Lower bound / impossibility result
  - \* Factor of 2 is optimal for (deterministic) multiplicative weights!

Next: Imperfect experts continued. Randomise!!

20

## The probabilistic experts algorithm

*wherein randomisation helps us do better!*

21

## Probabilistic experts algorithm

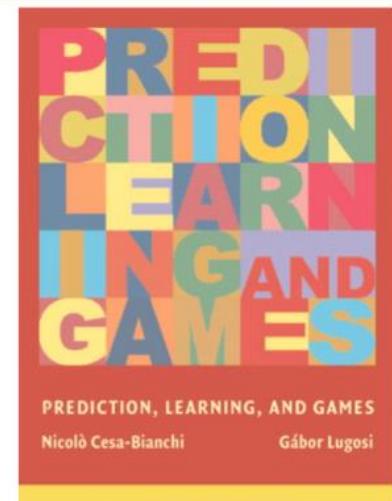
- Change 1 from mistakes: Loss  $\ell_i^{(t)} \in [0,1]$  of  $E_i$ , round  $t$
  - Change 2: Randomised algorithm means, bounding expected losses (sound familiar? It should.... a risk!)
1. Initialise  $w_i = 1$  weight of expert  $E_i$
  2. Repeat per round
    - a) Observe predictions  $E_i$  for all  $i \in \{1, \dots, n\}$
    - b) Predict  $E_i$  of expert  $i$  with probability  $\frac{w_i}{W}$  where  $W = \sum_{i=1}^n w_i$
    - c) Observe losses
    - d) Update each weight  $w_i \leftarrow (1 - \varepsilon)^{\ell_i^{(t)}} w_i$

22

## Probabilistic experts: Expected loss bound

- Proposition: Expected loss of the probabilistic experts algorithm is  $L \leq \frac{\log n}{\varepsilon} + (1 + \varepsilon)L^*$  where  $L^*$  is the minimum loss over experts.

- Proof: next, follows similar “potential” pattern
- Beats deterministic! Shaves off optimal constant 2
- Generalises in many directions. Active area of research in ML, control, economics, in top labs.



<http://cesa-bianchi.di.unimi.it/predbook/>

23

## Proof: Upper bounding potential function

- Learner's round  $t$  expected loss:  $L_t = \frac{\sum_{i=1}^n w_i^{(t)} \ell_i^{(t)}}{W(t)}$

- By Lemma 2, since losses are  $[0, 1]$ :  
updated  $w_i^{(t+1)} \leftarrow (1 - \varepsilon) \ell_i^{(t)} w_i^{(t)} \leq (1 - \varepsilon \ell_i^{(t)}) w_i^{(t)}$

- Rearrange to obtain recurrence relation:

$$\begin{aligned} W(t+1) &\leq \sum_{i=1}^n (1 - \varepsilon \ell_i^{(t)}) w_i^{(t)} = \sum_{i=1}^n w_i^{(t)} \left( 1 - \varepsilon \frac{\sum_{i=1}^n w_i^{(t)} \ell_i^{(t)}}{\sum_{i=1}^n w_i^{(t)}} \right) \\ &= W(t) (1 - \varepsilon L_t) \end{aligned}$$

- Initialisation gave  $W(0) = n$ , so telescoping we get:

$$W(T) \leq n \prod_{t=1}^T (1 - \varepsilon L_t)$$

24

## Proof: Lower bounding potential, Wrap up

- Proved upper bound:  $W(T) \leq n \prod_{t=1}^T (1 - \varepsilon L_t)$

- Lower bound from best expert total loss  $L^*$ :

$$W(T) \geq (1 - \varepsilon)^{L^*}$$

- Combining bounds and taking log's:

$$L^* \log_e (1 - \varepsilon) \leq \log_e n + \sum_{t=1}^T \log_e (1 - \varepsilon L_t)$$

- By Lemma 1:  $-L^*(\varepsilon + \varepsilon^2) \leq \log_e n - \varepsilon \sum_{t=1}^T L_t$

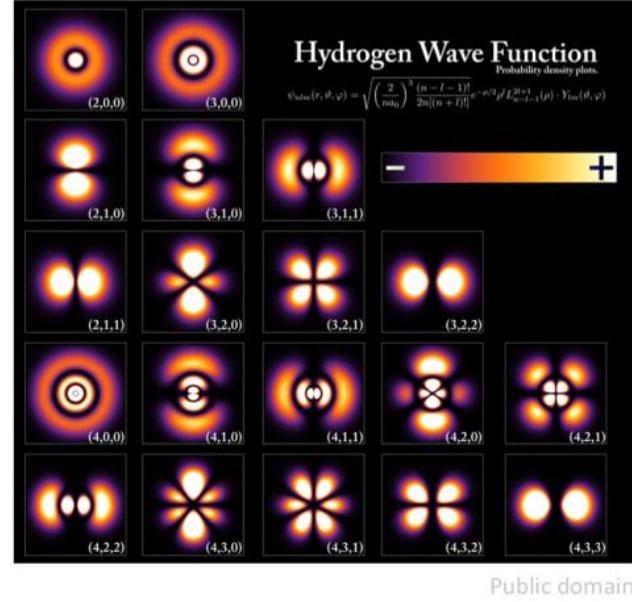
- Linearity of expectation  $L = \sum_{t=1}^T L_t$ , rearranging:

$$L \leq \frac{\log_e n}{\varepsilon} + (1 + \varepsilon)L^*$$

25

## Applications of multiplicative weights [Kale thesis 2007]

- Learning quantum states from noisy measurements
- Derandomising algorithms
- Solving certain zero-sum games
- Fast graph partitioning
- Fast solving of semidefinite programming problems
- Portfolio optimisation
- A basis for boosting
- Sparse vector technique in differential privacy



Public domain

26

## Mini Summary

- Introducing randomisation to learning with experts
  - \* Algorithm choosing a random expert to follow
  - \* Weights become probabilities
  - \* Mistakes generalise to losses
- Loss bound
  - \* Have to bound expected loss (hey, risk!!)
  - \* Shaves off that 2 factor. Proves that randomisation really helps!

Next: Only observe reward of chosen expert → bandits!

27

## Lecture 16

2020年9月25日 星期五 09:25



16\_bandits

# Lecture 16. Multi-armed bandits

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

# This lecture

- Bandit setting vs Learning with experts
  - \* Have to pick an expert (aka. arm)
  - \* Observe rewards only for chosen arm
- Aka. Sequential decision making under uncertainty
  - \* Simplest explore-vs-exploit setting
  - \* Incredibly rich area with heaps of industrial applications
- Basic algorithms
  - \* Greedy
  - \*  $\varepsilon$ -Greedy
  - \* Upper Confidence Bound (UCB)
- More: Contextual bandits, RL, ...

# Multi-Armed Bandits

*Where we learn to take actions; we receive only  
indirect supervision in the form of rewards; and we  
only observe rewards for actions taken – the simplest  
setting with an explore-exploit trade-off.*

# Exploration vs. Exploitation



CC0

# Exploration vs. Exploitation

- “Multi-armed bandit” (MAB)
  - \* Simplest setting for balancing exploration, exploitation
  - \* Same family of ML tasks as reinforcement learning
- Numerous applications
  - \* Online advertising
  - \* Caching in databases
  - \* Stochastic search in games (e.g. AlphaGo!)
  - \* Adaptive A/B testing
  - \* ....



CCO

# Stochastic MAB setting

- Possible actions  $\{1, \dots, k\}$  called “arms”
    - \* Arm  $i$  has distribution  $P_i$  on bounded rewards with mean  $\mu_i$
  - In round  $t = 1 \dots T$ 
    - \* Play action  $i_t \in \{1, \dots, k\}$  (possibly randomly)
    - \* Receive reward  $R_{i_t}(t) \sim P_{i_t}$
  - Goal: minimise cumulative regret
    - \*  $\mu^*T - \sum_{t=1}^T E[R_{i_t}(t)]$ 
      - where  $\mu^* = \max_i \mu_i$
      - \* Intuition: Do as well as a rule that is simple but has knowledge of the future
- Expected cumulative reward of bandit  
Best expected cumulative reward with hindsight

# Greedy

- At round  $t$

\* Estimate value of each arm  $i$  as average reward observed

$$Q_{t-1}(i) = \begin{cases} \frac{\sum_{s=1}^{t-1} R_i(s) \mathbf{1}[i_s = i]}{\sum_{s=1}^{t-1} \mathbf{1}[i_s = i]}, & \text{if } \sum_{s=1}^{t-1} \mathbf{1}[i_s = i] > 0 \\ Q_0, & \text{otherwise} \end{cases}$$

... some init constant  $Q_0(i) = Q_0$  used until arm  $i$  has been pulled

- \* Exploit, baby, exploit!

$$i_t \in \arg \max_{1 \leq i \leq k} Q_{t-1}(i)$$

- \* Tie breaking randomly

- What do you expect this to do? Effect of initial Qs?

## $\varepsilon$ -Greedy

- At round  $t$

\* Estimate value of each arm  $i$  as average reward observed

$$Q_{t-1}(i) = \begin{cases} \frac{\sum_{s=1}^{t-1} R_i(s) \mathbf{1}[i_s = i]}{\sum_{s=1}^{t-1} \mathbf{1}[i_s = i]}, & \text{if } \sum_{s=1}^{t-1} \mathbf{1}[i_s = i] > 0 \\ Q_0, & \text{otherwise} \end{cases}$$

... some init constant  $Q_0(i) = Q_0$  used until arm  $i$  has been pulled

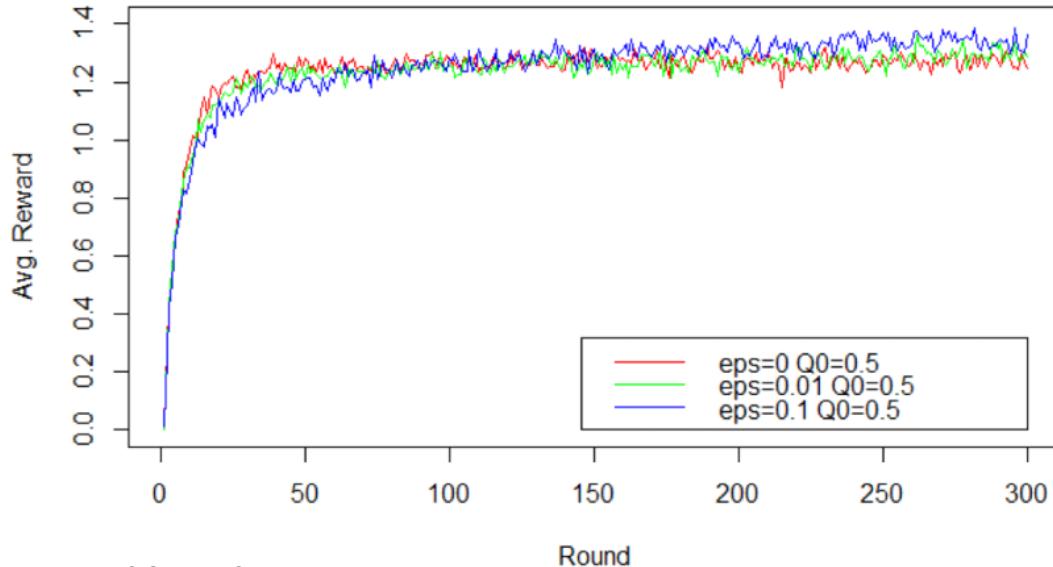
\* Exploit, baby exploit... probably; or possibly explore

$$i_t \sim \begin{cases} \arg \max_{1 \leq i \leq k} Q_{t-1}(i) & \text{w.p. } 1 - \varepsilon \\ \text{Unif}(\{1, \dots, k\}) & \text{w.p. } \varepsilon \end{cases}$$

\* Tie breaking randomly

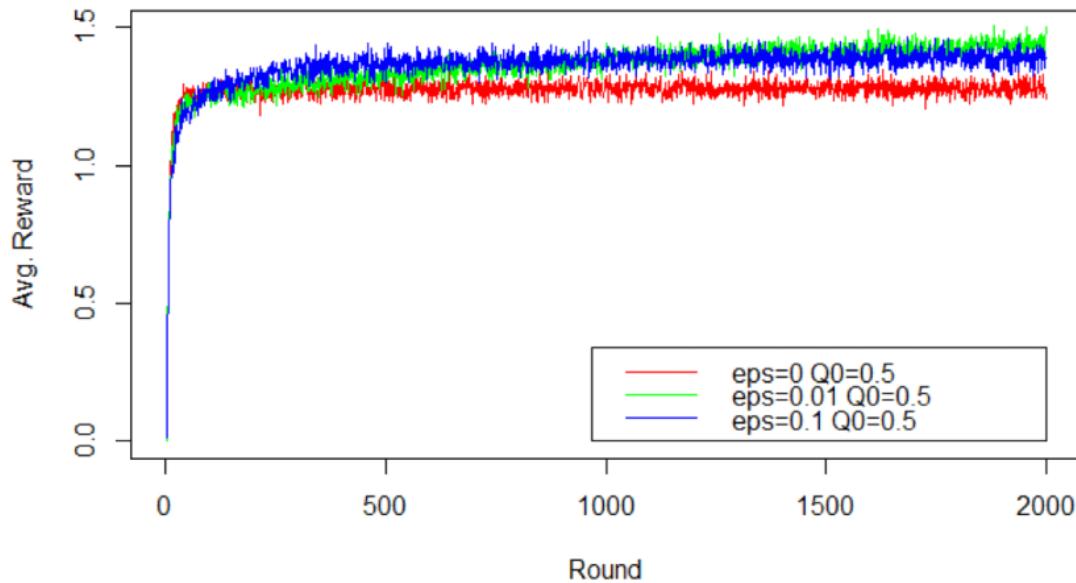
- Hyperparam.  $\varepsilon$  controls exploration vs. exploitation

# Kicking the tyres



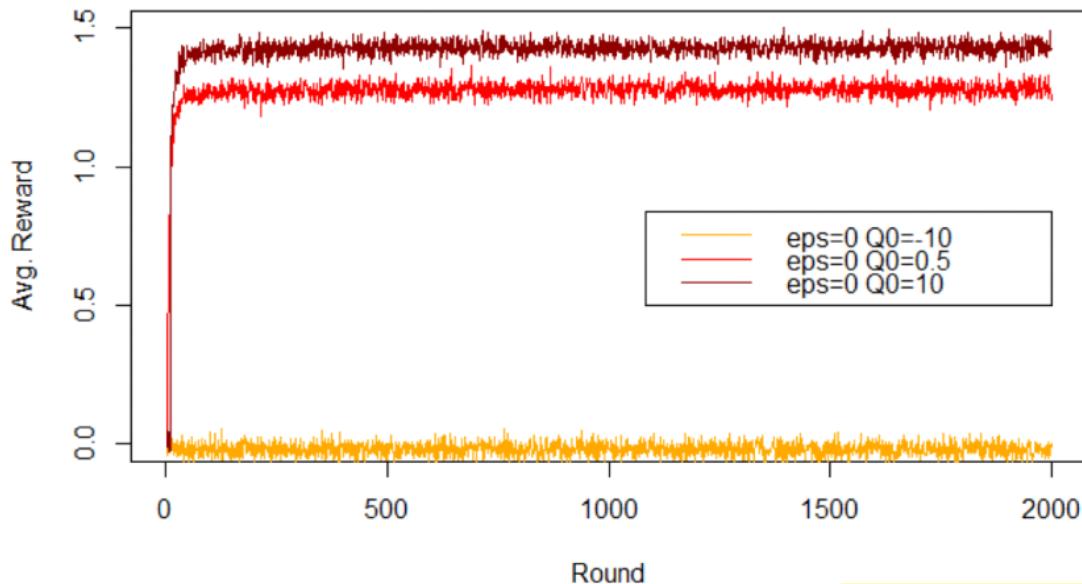
- 10-armed bandit
- Rewards  $P_i = \text{Normal}(\mu_i, 1)$  with  $\mu_i \sim \text{Normal}(0, 1)$
- Play game for 300 rounds
- Repeat 1,000 games, plot average per-round rewards

## Kicking the tyres: More rounds



- Greedy increases fast, but levels off at low rewards
- $\epsilon$ -Greedy does **better long-term by exploring**
- 0.01-Greedy initially slow (little explore) but eventually superior to 0.1-Greedy (**exploits after enough exploration**)

## Optimistic initialisation improves Greedy



- **Pessimism:** Init Q's below observable rewards → Only try one arm
- **Optimism:** Init Q's above observable rewards → Explore arms once
- Middle-ground init Q → Explore arms at most once

But pure greedy never explores an arm more than once

## Limitations of $\varepsilon$ -Greedy

- While we can improve on basic Greedy with optimistic initialisation and decreasing  $\varepsilon$ ...
- Exploration and exploitation are too “distinct”
  - \* Exploration actions completely blind to promising arms
  - \* Initialisation trick only helps with “cold start”
- Exploitation is blind to confidence of estimates
- These limitations are serious in practice

## Mini Summary

- Multi-armed bandit setting
  - \* Simplest instance of an explore-exploit problem
  - \* Greedy approaches cover exploitation fine
  - \* Greedy approaches overly simplistic with exploration (if have any!)
- Compared to: learning with experts
  - \* Superficial changes: Experts → Arms; Losses → Rewards
  - \* Choose one arm (like probabilistic experts algorithm)
  - \* Big difference: Only observe rewards on chosen arm

Next: A better way: optimism under uncertainty principle

# Upper-Confidence Bound (UCB)

*Optimism in the face of uncertainty;*  
A smarter way to balance exploration-exploitation.

## (Upper) confidence interval for Q estimates

- **Theorem: Hoeffding's inequality**

- \* Let  $R_1, \dots, R_n$  be i.i.d. random variables in  $[0,1]$  mean  $\mu$ , denote by  $\bar{R}_n$  their sample mean
- \* For any  $\varepsilon \in (0,1)$  with probability at least  $1 - \varepsilon$

$$\mu \leq \bar{R}_n + \sqrt{\frac{\log(1/\varepsilon)}{2n}}$$

- Application to  $Q_{t-1}(i)$  estimate – also i.i.d. mean!!

- \* Take  $n = N_{t-1}(i) = \sum_{s=1}^{t-1} 1[i_s = i]$  number of  $i$  plays
- \* Then  $\bar{R}_n = Q_{t-1}(i)$
- \* Critical level  $\varepsilon = 1/t$  (Lai & Robbins '85), take  $\varepsilon = 1/t^4$

# Upper Confidence Bound (UCB) algorithm

- At round  $t$

- \* Estimate value of each arm  $i$  as average reward observed

$$Q_{t-1}(i) = \begin{cases} \hat{\mu}_{t-1}(i) + \sqrt{\frac{2\log(t)}{N_{t-1}(i)}}, & \text{if } \sum_{s=1}^{t-1} 1[i_s = i] > 0 \\ Q_0, & \text{otherwise} \end{cases}$$

...some constant  $Q_0(i) = Q_0$  used until arm  $i$  has been pulled; where:

$$N_{t-1}(i) = \sum_{s=1}^{t-1} 1[i_s = i] \quad \hat{\mu}_{t-1}(i) = \frac{\sum_{s=1}^{t-1} R_i(s) 1[i_s = i]}{\sum_{s=1}^{t-1} 1[i_s = i]}$$

- \* “Optimism in the face of uncertainty”

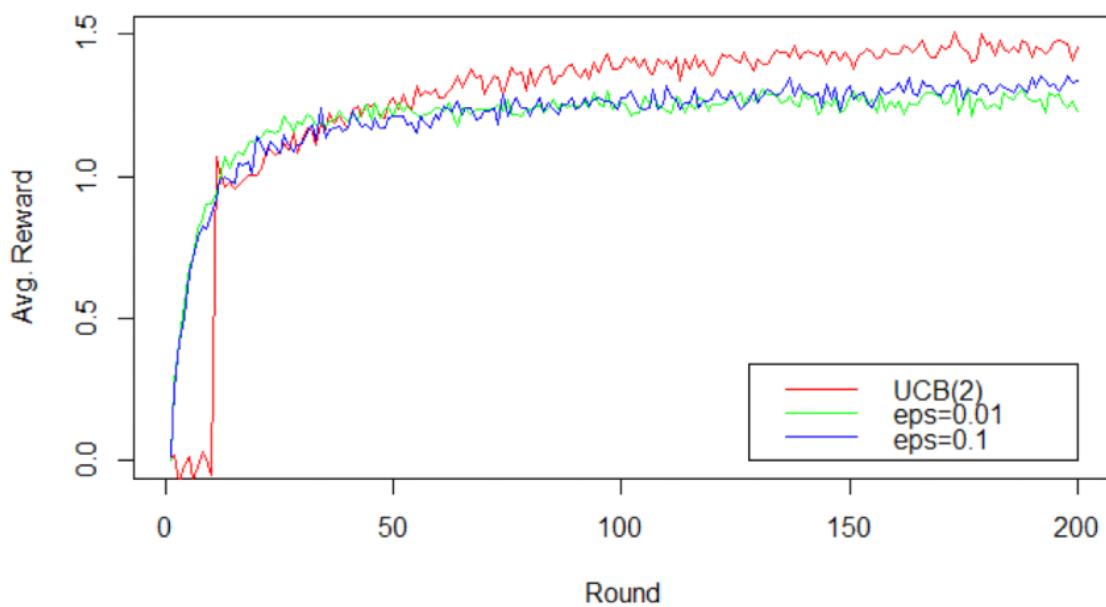
$$i_t \sim \arg \max_{1 \leq i \leq k} Q_{t-1}(i)$$

...tie breaking randomly

- Addresses several limitations of  $\epsilon$ -greedy
- Can “pause” in a bad arm for a while, but eventually find best

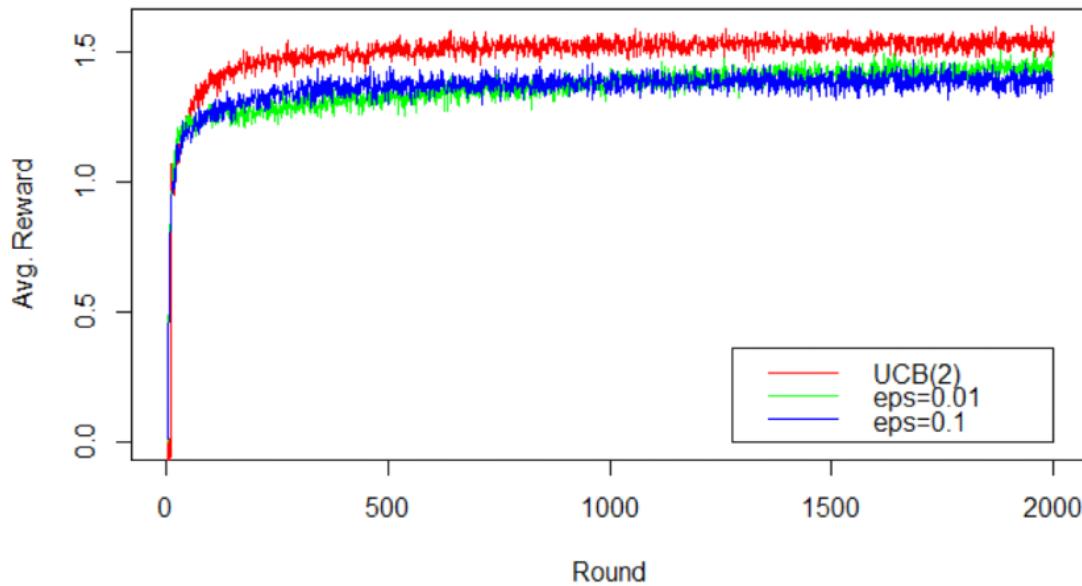
*blind explore*  
*exploit greedy*

## Kicking the tyres: How does UCB compare?



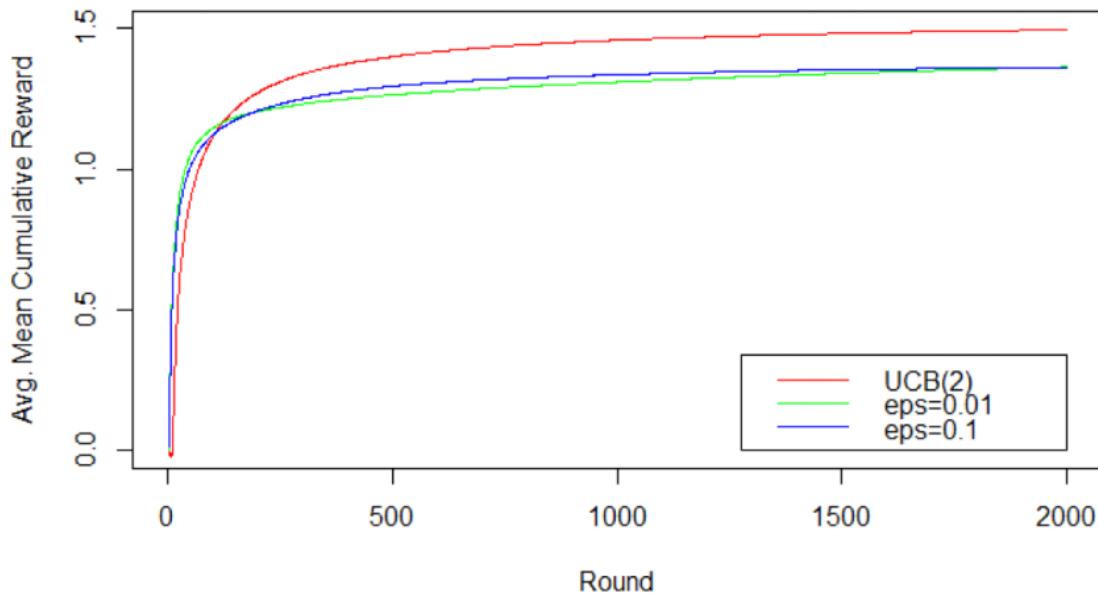
- UCB quickly overtakes the  $\varepsilon$ -Greedy approaches

## Kicking the tyres: How does UCB compare?



- UCB quickly overtakes the  $\epsilon$ -Greedy approaches
- Continues to outpace on per round rewards for some time

## Kicking the tyres: How does UCB compare?



- UCB quickly overtakes the  $\epsilon$ -Greedy approaches
- Continues to outpace on per round rewards for some time
- More striking when viewed as mean cumulative rewards

# Notes on UCB

- Theoretical regret bounds, optimal up to multiplicative constant

\* Grows like  $O(\log t)$  i.e. averaged regret goes to zero!

- Tunable  $\rho > 0$  exploration hyperparam. replaces “2”

$$Q_{t-1}(i) = \begin{cases} \hat{\mu}_{t-1}(i) + \sqrt{\frac{\rho \log(t)}{N_{t-1}(i)}}, & \text{if } \sum_{s=1}^{t-1} 1[i_s = i] > 0 \\ Q_0, & \text{otherwise} \end{cases}$$

\* Captures different  $\varepsilon$  rates & bounded rewards outside [0,1]

- Many variations e.g. different confidence bounds
- Basis for Monte Carlo Tree Search used in AlphaGo!

## Mini Summary

- Addressing limitations of greedy
  - \* Exploration blind to how good an arm is
  - \* Exploration/exploitation blind to confidence of arm estimates
- Upper-confidence bound (UCB) algorithm
  - \* Exploration boost: Upper-confidence bound (like PAC bound!)
  - \* Example of: Optimism in the face of uncertain principle
  - \* Achieves practical performance and good regret bounds

Next: Wrap-up with a few related directions

# Beyond basic bandits

*Adding state with contextual bandits;  
State transitions/dynamics with reinforcement learning.*

## But wait, there's more!! Contextual bandits

- Adds concept of “state” of the world
  - \* Arms’ rewards now depend on state
  - \* E.g. best ad depends on user and webpage
- Each round, observe arbitrary context (feature) vector representing state  $X_i(t)$  per arm
  - \* Profile of web page visitor (state)
  - \* Web page content (state)
  - \* Features of a potential ad (arm)
- Reward estimation
  - \* Was unconditional:  $E[R_i(t)]$
  - \* Now conditional:  $E[R_i(t)|X_i(t)]$
- A **regression problem!!!**

Still choose arm with maximizing UCB.

But UCB is not on a mean, but a regression prediction given context vector.

# MABs vs. Reinforcement Learning

- Contextual bandits introduce state
  - \* But don't model actions as causing state transitions
  - \* New state arrives "somehow"
- RL has rounds of states, actions, rewards too
- But (state,action) determines the next state
  - \* E.g. playing Go, moving a robot, planning logistics
- Thus, RL still learns value functions w regression, but has to "roll out" predicted rewards into the future

## Mini Summary

- Lecture: Stochastic multi-armed bandits
  - \* Sequential decision making under uncertainty
  - \* Simplest explore-vs-exploit setting
  - \*  $(\varepsilon)$ -greedy, UCB, LinUCB
- Related directions:
  - \* Contextual bandits: adding state; regression estimates rewards
  - \* Reinforcement learning: introducing state transitions

Next lecture: It's Bayesian time!

## Lecture 17

2020年9月30日 星期三 14:55



17\_bayesia  
n\_regress...

# Lecture 17. Bayesian regression

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



THE UNIVERSITY OF  
MELBOURNE

Copyright: University of Melbourne

COMP90051 Statistical Machine Learning

## This lecture

- Uncertainty not captured by point estimates
- Bayesian approach preserves uncertainty
- Sequential Bayesian updating
- Conjugate prior (Normal-Normal)
- Using posterior for Bayesian predictions on test

## Training == optimisation (?)

Stages of learning & inference:

- Formulate model

Regression

$$p(y|\mathbf{x}) = \text{sigmoid}(\mathbf{x}'\mathbf{w})$$

$$p(y|\mathbf{x}) = \text{Normal}(\mathbf{x}'\mathbf{w}; \sigma^2)$$

- Fit parameters to data

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\text{argmax}} p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w}) \quad \text{ditto}$$

- Make prediction

$$p(y_*|\mathbf{x}_*) = \text{sigmoid}(\mathbf{x}'_*\hat{\mathbf{w}})$$

$$E[y_*] = \mathbf{x}'_*\hat{\mathbf{w}}$$

$\hat{\mathbf{w}}$  referred to as a '*point estimate*'

*average*

3

## Bayesian Alternative

Nothing special about  $\hat{\mathbf{w}}$ ... use more than one value?

- Formulate model

Regression

$$p(y|\mathbf{x}) = \text{sigmoid}(\mathbf{x}'\mathbf{w})$$

$$p(y|\mathbf{x}) = \text{Normal}(\mathbf{x}'\mathbf{w}; \sigma^2)$$

- Consider the space of likely parameters – those that fit the training data well

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y})$$

*a whole range  
(not point estimate)*

- Make 'expected' prediction

$$p(y_*|\mathbf{x}_*) = E_{p(\mathbf{w}|\mathbf{X}, \mathbf{y})} [\text{sigmoid}(\mathbf{x}'_*\mathbf{w})]$$

*weighted average*

$$p(y_*|\mathbf{x}_*) = E_{p(\mathbf{w}|\mathbf{X}, \mathbf{y})} [\text{Normal}(\mathbf{x}'_*\mathbf{w}, \sigma^2)]$$

4



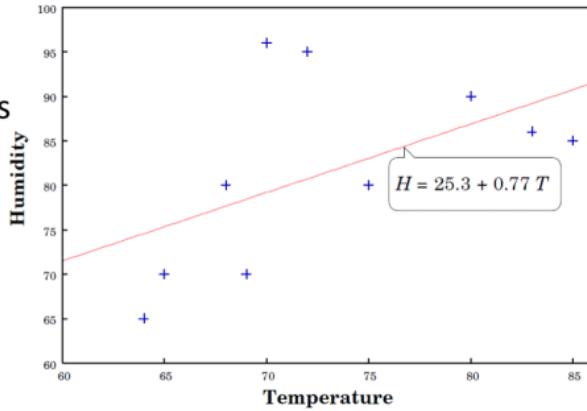
# Uncertainty

From small training sets, we rarely have complete confidence in any models learned. Can we quantify the uncertainty, and use it in making predictions?

5

## Regression Revisited

- Learn model from data
  - \* minimise error residuals by choosing weights  
 $\hat{\mathbf{w}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$
- But... how confident are we
  - \* in  $\hat{\mathbf{w}}$ ?
  - \* in the predictions?



Linear regression:  $y = w_0 + w_1 x$   
 (here  $y$  = humidity,  $x$  = temperature)

Bayesian ML  
 wants to address

6

## Do we trust point estimate $\hat{\mathbf{w}}$ ?

- How *stable* is learning?
  - \*  $\hat{\mathbf{w}}$  highly sensitive to noise
  - \* how much uncertainty in parameter estimate?
  - \* more *informative* if neg log likelihood objective highly peaked
- Formalised as *Fisher Information matrix*
  - \*  $E[2^{\text{nd}} \text{ deriv of NLL}]$
  - $$\mathcal{I} = \frac{1}{\sigma^2} \mathbf{X}' \mathbf{X}$$
  - \* measures *curvature of objective about  $\hat{\mathbf{w}}$*

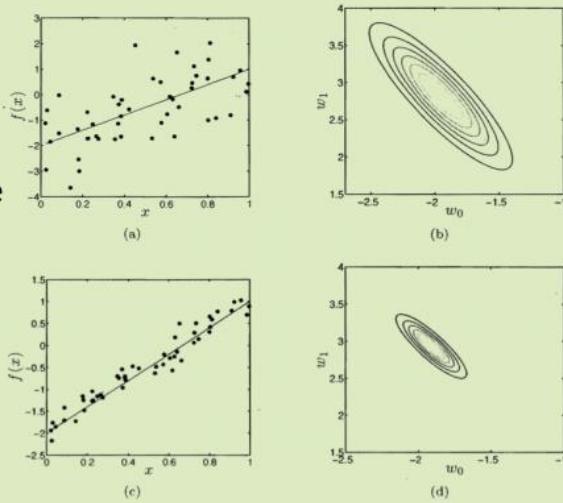


Figure: Rogers and Girolami p81

7

## Mini Summary

- Uncertainty not captured by point estimates (MLE, MAP)
- Uncertainty might capture range of plausible parameters
- (Frequentist) idea of Fisher information as likelihood sensitivity at point estimates

Next time: The Bayesian view (reminder)

8

# The Bayesian View

*quantify*

*Retain and model all unknowns (e.g., uncertainty over parameters) and use this information when making inferences.*

9

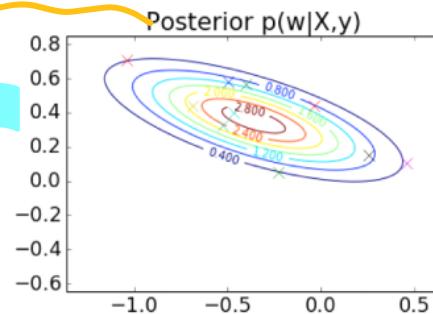
## A Bayesian View

- Could we reason over all parameters that are consistent with the data?
  - \* weights with a better fit to the training data should be more probable than others (not equal)
  - \* make predictions with all these weights, scaled by their probability (weighted predictions)
- This is the idea underlying Bayesian inference

10

## Uncertainty over parameters

- Many reasonable solutions to objective
  - \* why select just one?
- Reason under *all* possible parameter values
  - \* weighted by their *posterior probability*
- More robust predictions
  - \* less sensitive to overfitting, particularly with small training sets
  - \* can give rise to more expressive model class  
(Bayesian logistic regression becomes non-linear!)



11

## Frequentist vs Bayesian “divide”

- **Frequentist:** learning using *point estimates*, *regularisation*, *p-values* ...
  - \* backed by sophisticated theory on simplifying assumptions
  - \* mostly simpler algorithms, characterises much practical machine learning research
- **Bayesian:** maintain *uncertainty*, marginalise (sum) out unknowns during inference
  - \* some theory
  - \* often more complex algorithms, but not always
  - \* often (not always) more computationally expensive

12

## Mini Summary

- Frequentist's central preference of point estimates don't capture uncertainty
- Bayesian view is to quantify belief in prior, update it to posterior using observations

Next time: Bayesian approach to linear regression

13

## Bayesian Regression

*Application of Bayesian inference  
to linear regression, using  
Normal prior over  $w$*

14

## Revisiting Linear Regression

- Recall probabilistic formulation of linear regression

$$\mathbf{I}_D = D \times D \text{ identity matrix}$$

- Bayes rule:

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$

*posterior likelihood prior constant*

$$\max_{\mathbf{w}} p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \max_{\mathbf{w}} p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})$$

- Gives rise to penalised objective (ridge regression)

point estimate taken here, avoids computing marginal likelihood term

15

## Bayesian Linear Regression

- Rewind one step, consider full posterior

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma^2) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2)p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X}, \sigma^2)}$$

$$= \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2)p(\mathbf{w})}{\int p(\mathbf{y}, |\mathbf{X}, \mathbf{w}, \sigma^2)p(\mathbf{w})d\mathbf{w}}$$

*p(y, w | X)*

- Can we compute the denominator (**marginal likelihood** or **evidence**)?
  - \* if so, we can use the full posterior, not just its mode

16

## Bayesian Linear Regression (cont)

- We have two Normal distributions
  - normal likelihood  $\times$  normal prior
- Their product is also a Normal distribution
  - conjugate prior: when product of likelihood  $\times$  prior results in the same distribution as the prior**
  - evidence can be computed easily using the normalising constant of the Normal distribution

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma^2) \propto \text{Normal}(\mathbf{w}|\mathbf{0}, \gamma^2 \mathbf{I}_D) \text{Normal}(\mathbf{y}|\mathbf{X}\mathbf{w}, \sigma^2 \mathbf{I}_N)$$

$$\propto \text{Normal}(\mathbf{w}|\mathbf{w}_N, \mathbf{V}_N)$$

closed form solution for posterior!

17

## Bayesian Linear Regression (cont)

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma^2) \propto \text{Normal}(\mathbf{w}|\mathbf{0}, \gamma^2 \mathbf{I}_D) \text{Normal}(\mathbf{y}|\mathbf{X}\mathbf{w}, \sigma^2 \mathbf{I}_N)$$

$$\propto \text{Normal}(\mathbf{w}|\mathbf{w}_N, \mathbf{V}_N)$$

where

$$\mathbf{w}_N = \frac{1}{\sigma^2} \mathbf{V}_N \mathbf{X}' \mathbf{y}$$

$$\mathbf{V}_N = \sigma^2 (\mathbf{X}' \mathbf{X} + \frac{\sigma^2}{\gamma^2} \mathbf{I}_D)^{-1}$$

**Advanced:** verify by expressing product of two Normals, gathering exponents together and ‘completing the square’ to express as squared exponential (i.e., Normal distribution).

$$P(\theta|X=1) = \frac{P(X=1|\theta)P(\theta)}{P(X=1)}$$

$$\propto P(X=1|\theta)P(\theta)$$

Name of the game is to get posterior into a recognisable form.  
exp of quadratic must be a Normal

Discard constants w.r.t  $\theta$  =  $\left[ \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(1-\theta)^2}{2}\right) \right] \left[ \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\theta^2}{2}\right) \right]$

Collect exp's  $\propto \exp\left(-\frac{(1-\theta)^2 + \theta^2}{2}\right)$

Want leading numerator term to be  $\theta^2$  by moving coefficient from denominator

$\approx \exp\left(-\frac{2\theta^2 - 2\theta + 1}{2}\right)$

Complete the square in numerator: move last terms constants

$\propto \exp\left(-\frac{\theta^2 - \theta + \frac{1}{2}}{2}\right) \cdot \exp\left(-\frac{1}{2}\right)$

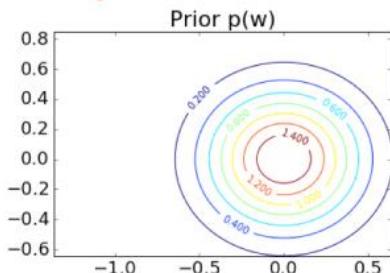
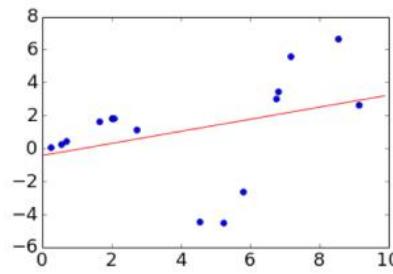
Factorise =  $\frac{1}{2} \exp\left(-\frac{(\theta - \frac{1}{2})^2 + \frac{1}{4}}{2}\right)$

Recognise as (unnormalised) Normal  $\propto N(0.5, 0.5)$

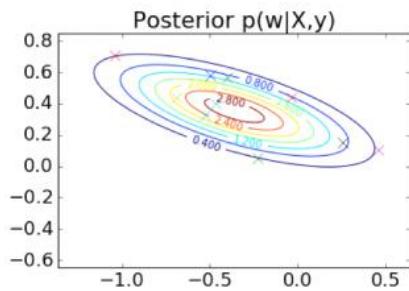
Slide 2.26 – completing square

18

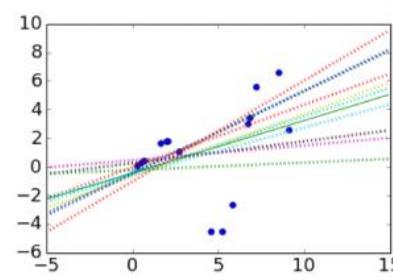
## Bayesian Linear Regression example

Step 1: select prior, here spherical about  $\mathbf{0}$ 

Step 2: observe training data



Step 3: formulate posterior, from prior &amp; likelihood



Samples from posterior

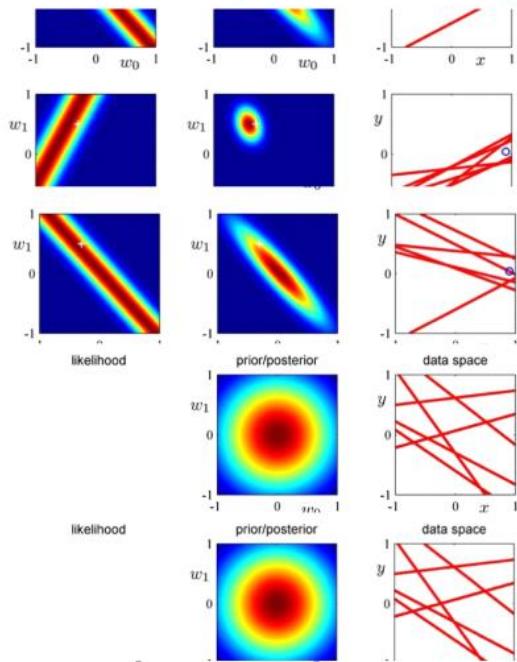
19

## Sequential Bayesian Updating

- Can formulate  $p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma^2)$  for given dataset
- What happens as we see more and more data?
  1. Start from prior  $p(\mathbf{w})$
  2. See new labelled datapoint
  3. Compute posterior  $p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma^2)$
  4. The **posterior now takes role of prior**  
& repeat from step 2

20

## Sequential Bayesian Updating



- Initially know little, many regression lines licensed
- Likelihood constrains possible weights such that regression is close to point
- Posterior becomes more refined/peaked as more data introduced
- Approaches a point mass

Bishop Fig 3.7, p155

21

## Stages of Training

- Decide on model formulation & prior
- Compute posterior over parameters,  $p(\mathbf{w}|\mathcal{X}, \mathcal{y})$

### MAP

- Find mode for  $\mathbf{w}$
- Use to make prediction on test

### approx. Bayes

- Sample many  $\mathbf{w}$
- Use to make ensemble average prediction on test

### exact Bayes

- Use all  $\mathbf{w}$  to make expected prediction on test

22

## Prediction with uncertain $w$

- Could predict using sampled regression curves
  - \* sample  $S$  parameters,  $\mathbf{w}^{(s)}, s \in \{1, \dots, S\}$
  - \* for each sample compute prediction  $y_*^{(s)}$  at test point  $\mathbf{x}_*$
  - \* compute the mean (and var.) over these predictions
  - \* this process is known as **Monte Carlo integration**
- For Bayesian regression there's a simpler solution
  - \* integration can be done analytically, for

$$p(\hat{y}_* | \mathbf{X}, \mathbf{y}, \mathbf{x}_*, \sigma^2) = \int p(\mathbf{w} | \mathbf{X}, \mathbf{y}, \sigma^2) p(y_* | \mathbf{x}_*, \mathbf{w}, \sigma^2) d\mathbf{w}$$

*bussian* *Gaussian*

23

*Gaussian*

## Prediction (cont.)

- Pleasant properties of Gaussian distribution means integration is tractable

$$\begin{aligned} p(y_* | \mathbf{x}_*, \mathbf{X}, \mathbf{y}, \sigma^2) &= \int p(\mathbf{w} | \mathbf{X}, \mathbf{y}, \sigma^2) p(y_* | \mathbf{x}_*, \mathbf{w}, \sigma^2) d\mathbf{w} \\ &= \int \text{Normal}(\mathbf{w} | \mathbf{w}_N, \mathbf{V}_N) \text{Normal}(y_* | \mathbf{x}'_* \mathbf{w}, \sigma^2) d\mathbf{w} \\ &= \text{Normal}(y_* | \mathbf{x}'_* \mathbf{w}_N, \sigma_N^2(\mathbf{x}_*)) \end{aligned}$$

$$\sigma_N^2(\mathbf{x}_*) = \sigma^2 + \mathbf{x}'_* \mathbf{V}_N \mathbf{x}_*$$

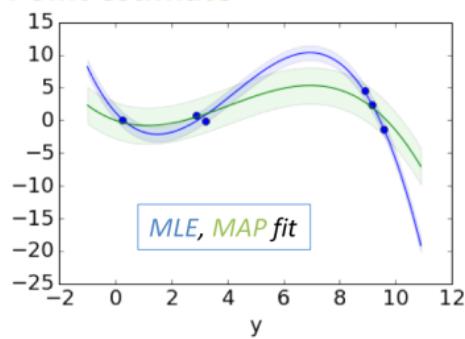
- \* additive variance based on  $\mathbf{x}_*$  match to training data
- \* cf. MLE/MAP estimate, where variance is a fixed constant

( $\mathbf{w}_N$  and  $\mathbf{V}_N$  defined in posterior when fitting Bayesian linear regression)

24

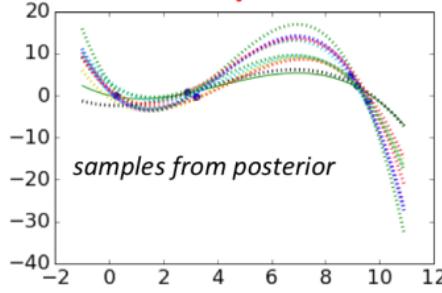
## Bayesian Prediction example

### Point estimate

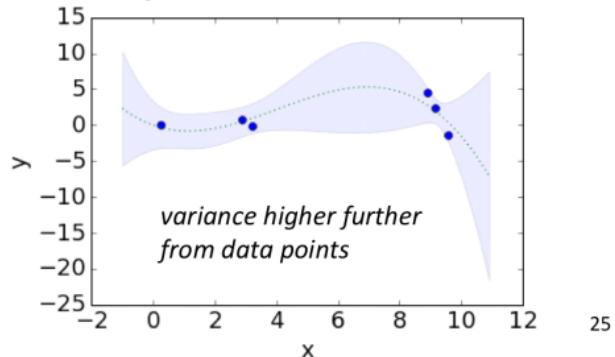


MLE (blue) and MAP (green)  
point estimates, with fixed  
variance

Data:  $y = x \sin(x)$ ; Model = cubic



### Bayesian inference



## Caveats

- Assumptions
  - \* known data noise parameter,  $\sigma^2$
  - \* data was drawn from the model distribution
- In real settings,  $\sigma^2$  is unknown
  - \* has its own conjugate prior  
*Normal* likelihood  $\times$  *InverseGamma* prior  
results in *InverseGamma* posterior
  - \* closed form predictive distribution, with student-T likelihood  
(see Murphy, 7.6.3)

## Mini Summary

- Uncertainty not captured by point estimates (MLE, MAP)
- Bayesian approach preserves uncertainty
  - \* care about predictions NOT parameters
  - \* choose prior over parameters, then model posterior
- New concepts:
  - \* sequential Bayesian updating
  - \* conjugate prior (Normal-Normal)
- Using posterior for Bayesian predictions on test

Next time: Bayesian classification, then PGMs

## Lecture 18

2020年10月2日 星期五 09:16



18\_bayesian\_classification...

# Lecture 18. Bayesian classification

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

COMP90051 Statistical Machine Learning

## This lecture

- Bayesian ideas in discrete settings
  - \* Beta-Binomial conjugacy
  - \* Uniqueness up to proportionality
  - \* Sunrise example
  - \* Common conjugate pairs
- Bayesian logistic regression
  - \* Non-conjugacy
  - \* Pointer: Laplace approximation

## How to apply Bayesian view to discrete data?

- First off consider models which *generate* the input
  - cf. discriminative models, which condition on the input
  - i.e.,  $p(y | x)$  vs  $p(x, y)$ , Logistic Regression vs Naïve Bayes
- For simplicity, start with most basic setting
  - $n$  coin tosses, of which  $k$  were heads
  - only have  $x$  (sequence of outcomes), but no ‘classes’  $y$
- Methods apply to *generative* models over discrete data
  - e.g., topic models, generative classifiers  
(Naïve Bayes, mixture of multinomials)

3

generative model  
 $p(x, y)$   
 phy/ax

## Discrete Conjugate prior: Beta-Binomial

- Conjugate priors also exist for discrete spaces
- Consider  $n$  coin tosses, of which  $k$  were heads
  - let  $p(\text{head}) = q$  from a single toss (*Bernoulli dist*)
  - Inference question is the coin biased, i.e., is  $q \approx 0.5$

- Several draws, use

*Binomial dist*

\* and its conjugate prior, *Beta dist*

$$p(k|n, q) = \binom{n}{k} q^k (1-q)^{n-k}$$

$$p(q) = \text{Beta}(q; \alpha, \beta)$$

$$= \frac{\gamma(\alpha + \beta)}{\gamma(\alpha)\gamma(\beta)} q^{\alpha-1} (1-q)^{\beta-1}$$

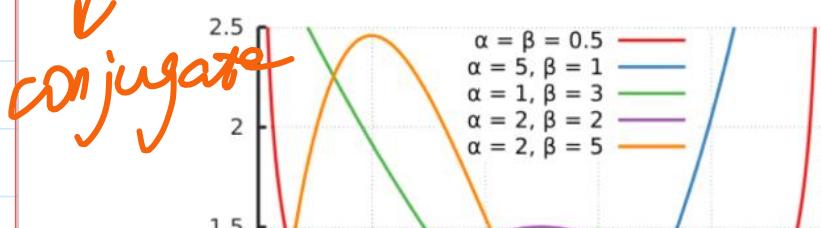
$$\int_0^1 p(q) dq = 1$$

Beta prior

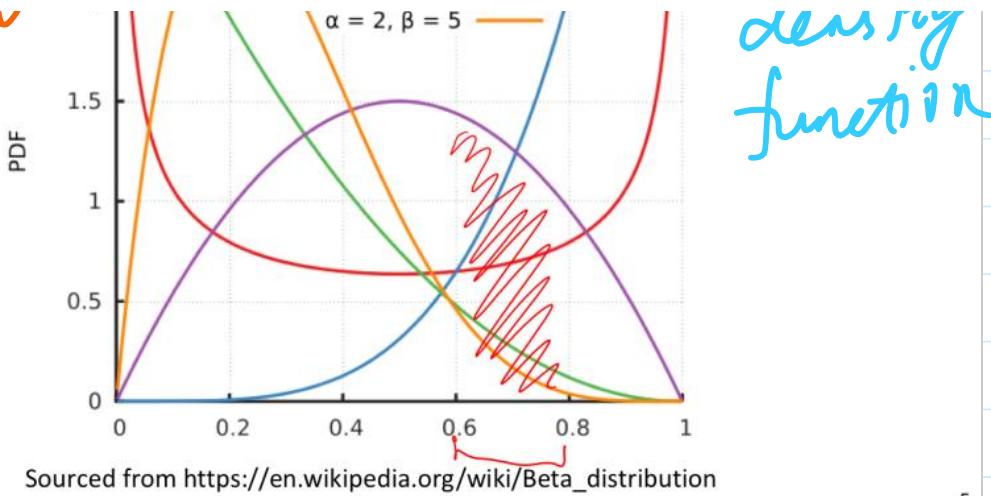
Bernoulli likelihood

posterior (Beta)

### Beta distribution



density function



5

COMP90051 Statistical Machine Learning

## Beta-Binomial conjugacy

$$p(k|n, q) = \binom{n}{k} q^k (1-q)^{n-k} \quad \text{likelihood}$$

$$p(q) = \text{Beta}(q; \alpha, \beta) \quad \text{prior}$$

$$= \frac{\gamma(\alpha + \beta)}{\gamma(\alpha)\gamma(\beta)} q^{\alpha-1} (1-q)^{\beta-1}$$

Sweet! We know the normaliser for Beta

Bayesian posterior

trick: ignore constant factors  
(normaliser)

$$\begin{aligned} p(q|k, n) &\propto p(k|n, q)p(q) \\ &\propto q^k (1-q)^{n-k} q^{\alpha-1} (1-q)^{\beta-1} \\ &= q^{k+\alpha-1} (1-q)^{n-k+\beta-1} \\ &\propto \text{Beta}(q; k + \alpha, n - k + \beta) \end{aligned}$$

6

## Uniqueness up to normalisation

- A trick we've used many times:

*When an unnormalized distribution is proportional to a recognised distribution, we say it must be that distribution*

- If  $f(\theta) \propto g(\theta)$  for  $g$  a distribution,  $\frac{\int_{\Theta} f(\theta)d\theta}{\int_{\Theta} g(\theta)d\theta} = 1$ .

- Proof:  $f(\theta) \propto g(\theta)$  means that

$$f(\theta) = C \cdot g(\theta)$$

$$\int_{\Theta} f(\theta)d\theta = C \int_{\Theta} g(\theta)d\theta = C$$

and the result follows from  $LHS1/LHS2 = RHS1/RHS2$

7

## Laplace's Sunrise Problem

*Every morning you observe the sun rising. Based solely on this fact, what's the probability that the sun will rise tomorrow?*

- Use Beta-Binomial, where  $q$  is the  $\text{Pr}(\text{sun rises in morning})$

- \* posterior  $p(q|k, n) = \text{Beta}(q; k + \alpha, n - k + \beta)$

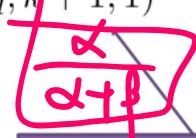
- \*  $n = k$  = observer's age in days

- \* let  $\alpha = \beta = 1$  (*uniform prior*)

- Under these assumptions

$$p(q|k) = \text{Beta}(q; k + 1, 1)$$

$$E_{p(q|k)}[q] = \frac{k+1}{k+2}$$



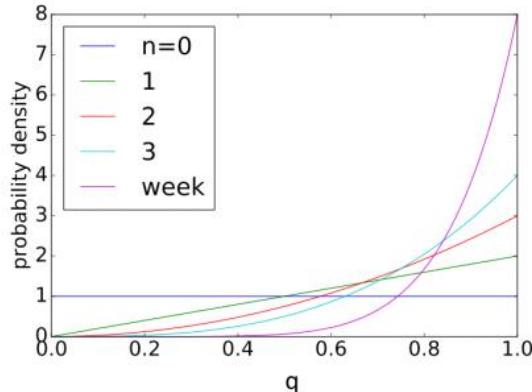
'smoothed' count of days  
where sun rose / did not

8

## Sunrise Problem (cont.)

Consider human-meaningful period

Day (n, k)	k+α	n-k+β	E[q]
0	1	1	0.5
1	2	1	0.667
2	3	1	0.75
...			
365	366	1	0.997
2920 (8 years)	2921	1	0.99997



Effect of prior diminishing with data, but never disappears completely.

9

## Suite of useful conjugate priors

	likelihood	conjugate prior
regression	Normal	Normal (for mean)
	Normal	Inverse Gamma (for variance) or Inverse Wishart (covariance)
classification	Binomial	Beta
	Multinomial	Dirichlet
counts	Poisson	Gamma

10

## Mini Summary

- Bayesian ideas in discrete settings
  - \* Beta-Binomial conjugacy
  - \* Uniqueness in proportionality
  - \* Sunrise example
  - \* Conjugate pairs

Next time: Bayesian logistic regression

11

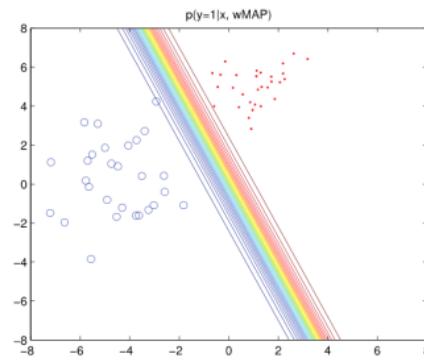
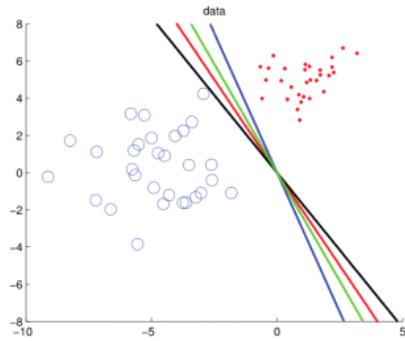
## Bayesian Logistic Regression

Discriminative classifier, which conditions on inputs. How can we do Bayesian inference in this setting?

12

## Now for Logistic Regression...

- Similar problems with parameter uncertainty compared to regression
  - \* although predictive uncertainty in-built to model outputs



Murphy Fig 8.5 &amp; 8.6 p257-8

13

## No conjugacy

- Can we use conjugate prior? E.g.,
  - \* Beta-Binomial for *generative* binary models
  - \* Dirichlet-Multinomial for multiclass (similar formulation)
- Model is *discriminative*, with parameters defined using logistic sigmoid\*
 
$$p(y|q, \mathbf{x}) = q^y (1-q)^{1-y}$$

$$q = \sigma(\mathbf{x}' \mathbf{w})$$
  - \* need prior over  $\mathbf{w}$ , not  $q$
  - \* no known conjugate prior (!), thus use a Gaussian prior
- Approach to inference: Monte Carlo sampling

\* Or softmax for multiclass; same problems arise and similar solution

14

# Approximation

- No known solution for the normalising constant

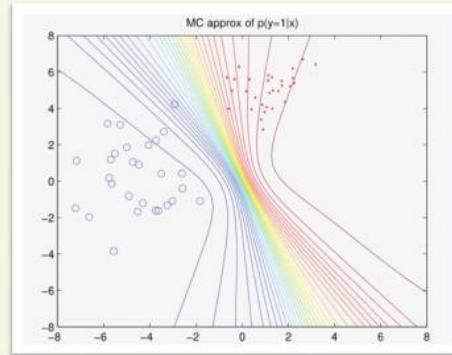
$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \propto p(\mathbf{w})p(\mathbf{y}|\mathbf{X}, \mathbf{w})$$

$$= \text{Normal}(\mathbf{0}, \sigma^2 \mathbf{I}) \prod_{i=1}^n \sigma(\mathbf{x}'_i \mathbf{w})^{y_i} (1 - \sigma(\mathbf{x}'_i \mathbf{w}))^{1-y_i}$$

- Resolve by *approximation*

*Laplace approx.:*

- assume posterior  $\simeq$  Normal about mode
- can compute normalisation constant, draw samples etc.
- Tractable MAP provides parameters for this (Normal) approximate posterior



Murphy Fig 8.6 p258

15

# Mini Summary

- Bayesian ideas in discrete settings
  - \* Beta-Binomial conjugacy
  - \* Conjugate pairs; Uniqueness in proportionality
- Bayesian classification (logistic regression)
  - \* Non-conjugacy necessitates approximation

Next time: probabilistic graphical models

16

## Lecture 19

2020年10月13日 星期二 10:14



19\_pgms

# Lecture 19. PGM Representation

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

## Next Lectures

- Representation of joint distributions
- Conditional/marginal independence
  - \* Directed vs undirected
- Probabilistic inference
  - \* Computing other distributions from joint
- Statistical inference
  - \* Learn parameters from (missing) data
- Examples



2

## This lecture

- (Directed) probabilistic graphical models
  - \* Motivations: applications, unifies algorithms
  - \* Motivation: ideal tool for Bayesians
  - \* Independence lowers computational/model complexity
    - Conditional independence
  - \* PGMs: compact representation of factorised joints
- Undirected PGMs and conversion from D-PGMs
- Example PGMs, applications

3

# Probabilistic Graphical Models

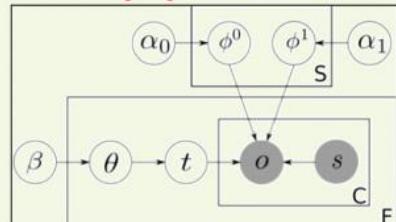
*Marriage of graph theory and probability theory.*

*Tool of choice for Bayesian statistical learning.*

*We'll stick with easier discrete case,  
ideas generalise to continuous.*

4

## Motivation by practical importance



- **Many applications**
  - \* Phylogenetic trees
  - \* Pedigrees, Linkage analysis
  - \* Error-control codes
  - \* Speech recognition
  - \* Document topic models
  - \* Probabilistic parsing
  - \* Image segmentation
  - \* ...
- **discovered algorithms**
  - \* HMMs
  - \* Kalman filters
  - \* Mixture models
  - \* LDA
  - \* MRFs
  - \* CRF
  - \* Logistic, linear regression
  - \* ...

5

## Motivation by way of comparison

### Bayesian statistical learning

- Model joint distribution of X's, Y and parameter r.v.'s
  - "Priors": marginals on parameters
- Training: update prior to posterior using observed data
- Prediction: output posterior, or some function of it (MAP)

### PGMs aka "Bayes Nets"

- Efficient joint representation
  - Independence made explicit
  - Trade-off between expressiveness and need for data, easy to make
  - Easy for practitioners to model
- Algorithms to fit parameters, compute marginals, posterior

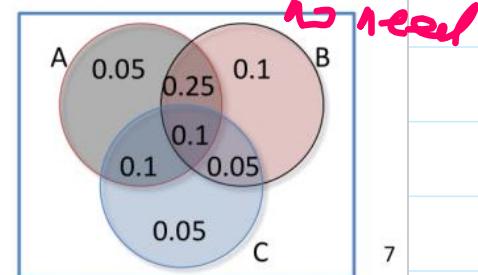
6

## Everything Starts at the Joint Distribution

- All joint distributions on discrete r.v.'s can be represented as tables
- #rows grows exponentially with #r.v.'s  
 $O(2^m)$
- Example: Truth Tables
  - $M$  Boolean r.v.'s require  $2^M - 1$  rows
  - Table assigns probability per row

Venn diagram

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	?



7

## The Good: What we can do with the joint

- Probabilistic inference from joint on r.v.'s
  - \* Computing any other distributions involving our r.v.'s
- Pattern: want a distribution, have joint; use:
  - Bayes rule + marginalisation
- Example: naïve Bayes classifier
  - \* Predict class  $y$  of instance  $x$  by maximising

$$\Pr(Y = y | X = x) = \frac{\Pr(Y=y, X=x)}{\Pr(X=x)} = \frac{\Pr(Y=y, X=x)}{\sum_y \Pr(X=x, Y=y)}$$

Recall: integration (over parameters) continuous equivalent of sum (both referred to as marginalisation)

8

## The Bad & Ugly: Tables waaaaay too large!!

- The Bad: Computational complexity
  - \* Tables have exponential number of rows in number of r.v.'s
  - \* Therefore → poor space & time to marginalise
- The Ugly: Model complexity
  - \* Way too flexible
  - \* Way too many parameters to fit  
→ need lots of data OR will overfit
- Antidote: assume independence!

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	?

9

## Example: You're late!

- Modeling a tardy lecturer. Boolean r.v's
  - $T$ : Ben teaches the class
  - $S$ : It is sunny (o.w. bad weather)
  - $L$ : The lecturer arrives late (o.w. on time)
- Assume: Ben sometimes delayed by bad weather, Ben more likely late than other lecturers
  - $\Pr(S|T) = \Pr(S)$ ,  $\Pr(S) = 0.3$   $\Pr(T) = 0.6$
- Lateness not independent on weather, lecturer
  - Need  $\Pr(L|T=t, S=s)$  for all combinations
- Need just 6 parameters
 
$$\Pr(L|T,S) \Pr(T|S) \Pr(S) \leftarrow \text{joint}$$



independent

		$T$	
		False	True
$S$	False	0.1	0.2
	True	0.05	0.1

10

## Independence: not a dirty word

Lazy Lecturer Model	Model details	# params
Our model with $S, T$ independence	$\Pr(S, T)$ factors to $\Pr(S) \Pr(T)$	2
	$\Pr(L T, S)$ modelled in full	4
Assumption-free model	$\Pr(L, T, S)$ modelled in full	7

3  
6

$2^3 - 1$

- Independence assumptions
  - Can be reasonable in light of domain expertise
  - Allow us to factor → Key to tractable models

11

## Factoring Joint Distributions

- Chain Rule: for any ordering of r.v.'s can always factor:

$$\Pr(X_1, X_2, \dots, X_k) = \prod_{i=1}^k \Pr(X_i | X_{i+1}, \dots, X_k)$$

- Model's independence assumptions correspond to

— Dropping conditioning r.v.'s in the factors!

— Example **unconditional indep.**:  $\Pr(X_1 | X_2) = \Pr(X_1)$

— Example **conditional indep.**:  $\Pr(X_1 | X_2, X_3) = \Pr(X_1 | X_2)$

$$\Pr(X_3 | X_1, X_2) = \Pr(X_3 | X_1)$$

$(X_1, X_3)$

- Example: independent r.v.'s  $\Pr(X_1, \dots, X_k) = \prod_{i=1}^k \Pr(X_i)$
- Simpler factors: speed up inference and avoid overfitting

12

## Mini Summary

- Joint distributions
- Probabilistic inference: Bayes rule & marginalisation
- Direct representation of joints
  - Probabilistic inference: Computationally costly
  - Statistical inference: Requires more data
- Factoring joints and conditional independence

Next: Directed probabilistic graphical models

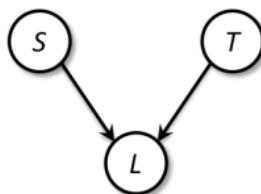
13

# Directed PGM

- Nodes
- Edges (acyclic)
- Random variables
- Conditional dependence
  - \* Node table:  $\Pr(\text{child}|\text{parents})$
  - \* Child directly depends on parents
- Joint factorisation

$$\Pr(X_1, X_2, \dots, X_k) = \prod_{i=1}^k \Pr(X_i | \text{parents}(X_i))$$

*Tardy Lecturer Example*

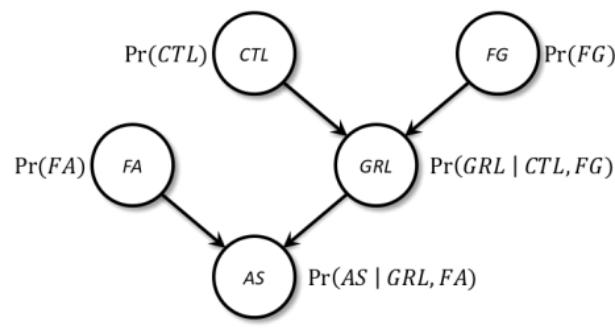


$\Pr(S)$   
 $\Pr(T)$   
 $\Pr(L|S, T)$

14

# Example: Nuclear power plant

- Core temperature  
→ Temperature Gauge  
→ Alarm
- Model uncertainty in monitoring failure
  - \* GRL: gauge reads low
  - \* CTL: core temperature low
  - \* FG: faulty gauge
  - \* FA: faulty alarm
  - \* AS: alarm sounds
- PGMs to the rescue!

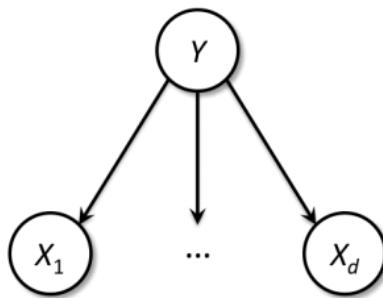


Joint  $\Pr(CTL, FG, FA, GRL, AS)$  given by

$$\Pr(AS|FA, GRL) \Pr(FA) \Pr(GRL|CTL, FG) \Pr(CTL) \Pr(FG)$$

15

# Naïve Bayes



$$Y \sim \text{Bernoulli}(\theta)$$

*Aside: Bernoulli is just Binomial with count=1*

$$X_j | Y \sim \text{Bernoulli}(\theta_{j,Y})$$

$$\begin{aligned} \Pr(Y, X_1, \dots, X_d) &= \Pr(X_1, \dots, X_d, Y) \\ &= \Pr(X_1|Y) \Pr(X_2|X_1, Y) \dots \Pr(X_d|X_1, \dots, X_{d-1}, Y) \Pr(Y) \\ &= \underbrace{\Pr(X_1|Y) \Pr(X_2|Y) \dots \Pr(X_d|Y)}_{\text{independence}} \Pr(Y) \end{aligned}$$

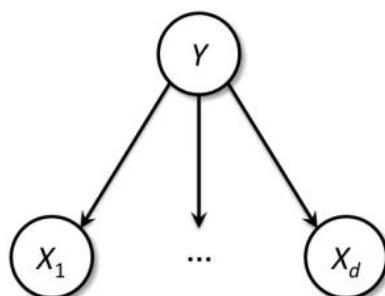
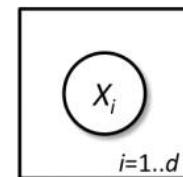
Prediction: predict label maximising  $\Pr(Y|X_1, \dots, X_d)$

16

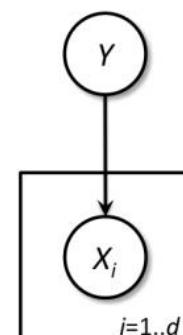
# Short-hand for repeats: Plate notation



=



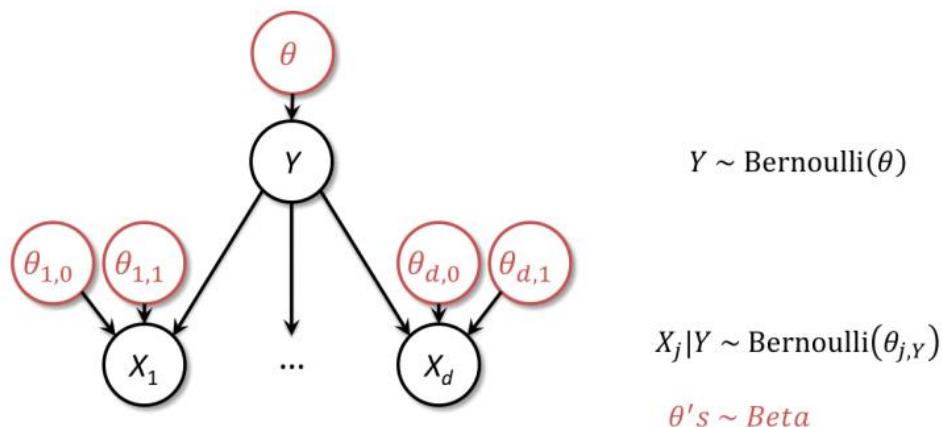
=



17

## PGMs: frequentist OR Bayesian...

- PGMs represent joints, which are central to Bayes
- Catch is that Bayesians add: node per parameters, with table being the parameter's prior



## Mini Summary

Directed probabilistic graphical models (D-PGMs)

- Definition as graph and conditionals
- Definition as joint distribution factorisation
- Plate notation
- Bayesian D-PGMs

Next: Undirected probabilistic graphical models

# Undirected PGMs

*Undirected variant of PGM, parameterised by arbitrary positive valued functions of the variables, and global normalisation.*

*A.k.a. Markov Random Field.*

20

## Undirected vs directed

### Undirected PGM

- Graph
  - \* Edges undirected
- Probability
  - \* Each node a r.v.
  - \* Each clique  $C$  has “factor”  
 $\psi_C(X_j : j \in C) \geq 0$
  - \* Joint  $\propto$  product of factors

*potential function*

### Directed PGM

- Graph
  - \* Edges directed
- Probability
  - \* Each node a r.v.
  - \* Each node has conditional  
 $p(X_i | X_j \in \text{parents}(X_i))$
  - \* Joint = product of cond'l's

**Key difference = normalisation**

21

## Undirected PGM formulation

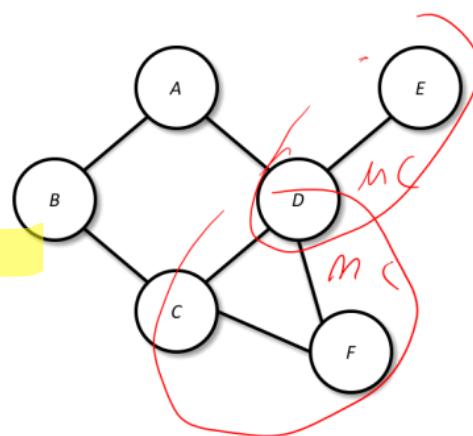
- Based on notion of
  - Clique:** a set of fully connected nodes (e.g., A-D, C-D, C-D-F)
  - Maximal clique:** largest cliques in graph (not C-D, due to C-D-F)
- Joint probability defined as

$$P(a, b, c, d, e, f) = \frac{1}{Z} \psi_1(a, b) \psi_2(b, c) \psi_3(a, d) \psi_4(d, c, f) \psi_5(d, e)$$

- where each  $\psi$  is a positive function and Z is the normalising 'partition' function

$$Z = \sum_{a,b,c,d,e,f} \psi_1(a, b) \psi_2(b, c) \psi_3(a, d) \psi_4(d, c, f) \psi_5(d, e)$$

22



## Directed to undirected

- Directed PGM formulated as

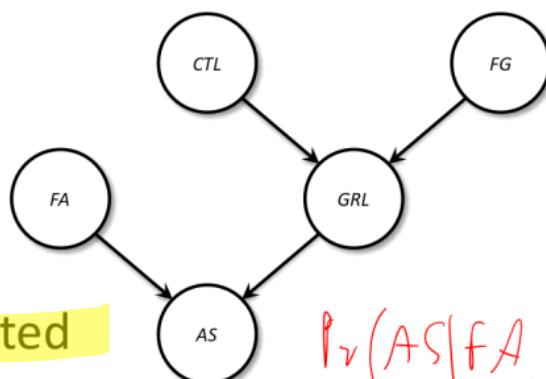
$$P(X_1, X_2, \dots, X_k) = \prod_{i=1}^k Pr(X_i | X_{\pi_i})$$

where  $\pi$  indexes parents.

- Equivalent to U-PGM with
  - each conditional probability term is included in one factor function,  $\psi_c$
  - clique structure links *groups of variables*, i.e.,  $\{\{X_i\} \cup X_{\pi_i}, \forall i\}$
  - normalisation term trivial,  $Z = 1$

23

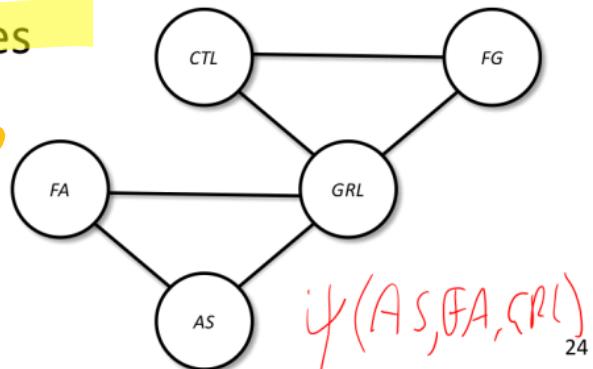
1. copy nodes



$$\Pr(AS|FA, GRL)$$

2. copy edges, undirected

*connect parents*



$$\Pr(AS, FA, GRL)$$

24

## Why U-PGM?

- Pros

- \* generalisation of D-PGM
- \* simpler means of modelling without the need for per-factor normalisation
- \* general inference algorithms use U-PGM representation (supporting both types of PGM)

- Cons

- \* (slightly) weaker independence
- \* calculating global normalisation term ( $Z$ ) intractable in general (but tractable for chains/trees, e.g., CRFs)

## Mini Summary

Undirected probabilistic graphical models (U-PGMs)

- Definition
- Conversion to D-PGMs
- Pros/Cons over D-PGMs

Next: Examples and applications of PGMs

26

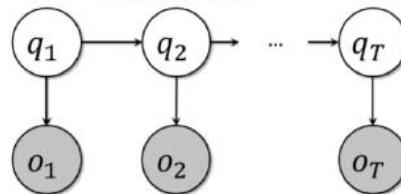
## Example PGMs

*The hidden Markov model (HMM);  
lattice Markov random field (MRF);  
Conditional random field (CRF)*

27

# The HMM (and Kalman Filter)

- Sequential observed **outputs** from hidden **state**



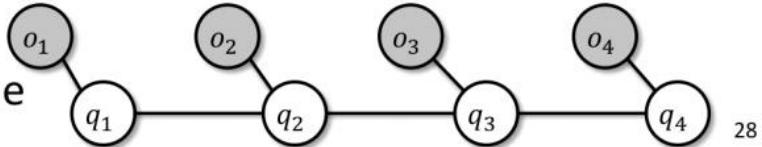
$A = \{a_{ij}\}$  transition probability matrix;  $\forall i : \sum_j a_{ij} = 1$

$B = \{b_i(o_k)\}$  output probability matrix;  $\forall i : \sum_k b_i(o_k) = 1$

$\Pi = \{\pi_i\}$  the initial state distribution;  $\sum_i \pi_i = 1$

- The **Kalman filter** same with continuous Gaussian r.v.'s

- A **CRF** is the undirected analogue



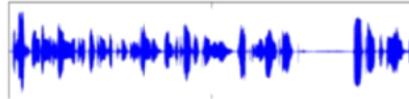
28

## HMM Applications

- NLP – part of speech tagging: given words in sentence, infer hidden parts of speech

“I love Machine Learning” → noun, verb, noun, noun

- Speech recognition: given waveform, determine phonemes



- Biological sequences: classification, search, alignment

- Computer vision: identify who's walking in video, tracking

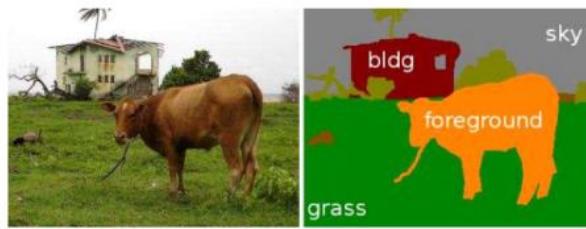
29

## Fundamental HMM Tasks

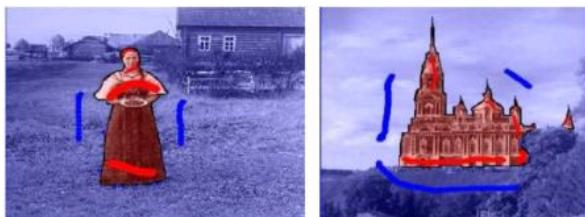
HMM Task	PGM Task
<b>Evaluation.</b> Given an HMM $\mu$ and observation sequence $O$ , determine likelihood $\Pr(O \mu)$	Probabilistic inference
<b>Decoding.</b> Given an HMM $\mu$ and observation sequence $O$ , determine most probable hidden state sequence $Q$	MAP point estimate
<b>Learning.</b> Given an observation sequence $O$ and set of states, learn parameters $A, B, \Pi$	Statistical inference

30

## Pixel labelling tasks in Computer Vision



Semantic labelling (Gould et al. 09)



Interactive figure-ground segmentation (Boykov &amp; Jolly 2011)

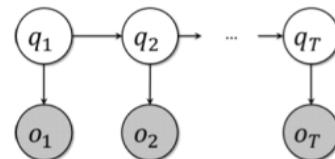


Denoising (Felzenszwalb &amp; Huttenlocher 04)

31

# What these tasks have in common

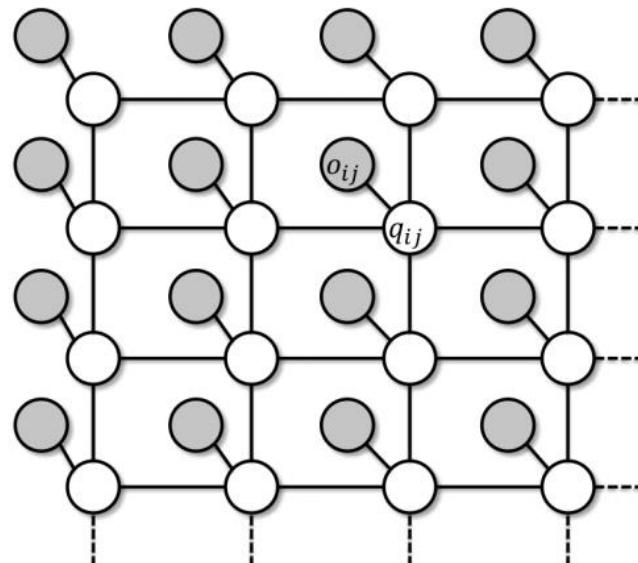
- **Hidden state representing semantics of image**
  - \* Semantic labelling: Cow vs. tree vs. grass vs. sky vs. house
  - \* Fore-back segment: Figure vs. ground
  - \* Denoising: Clean pixels
- Pixels of image
  - \* What we observe of hidden state
- Remind you of HMMs?



32

# A hidden square-lattice Markov random field

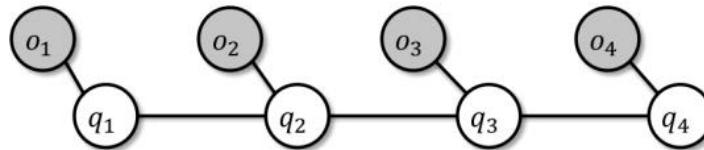
- **Hidden states:**  
square-lattice model
  - \* Boolean for two-class states
  - \* Discrete for multi-class
  - \* Continuous for denoising
- **Pixels: observed outputs**
  - \* Continuous e.g. Normal



33

## Application to sequences: CRFs

- Conditional Random Field: Same model applied to sequences
  - \* observed outputs are words, speech, amino acids etc
  - \* states are tags: part-of-speech, phone, alignment...
- CRFs are discriminative, model  $P(Q|O)$ 
  - \* versus HMM's which are generative,  $P(O, Q)$
  - \* undirected PGM more general and expressive



34

## Summary

- Probabilistic graphical models
  - \* Motivation: applications, unifies algorithms
  - \* Motivation: ideal tool for Bayesians
  - \* Independence lowers computational/model complexity
  - \* PGMs: compact representation of factorised joints
  - \* U-PGMs
- Example PGMs and applications

**Next time:** elimination for probabilistic inference

35

## Lecture 20

2020年10月16日 星期五 08:13



20\_pgm\_in  
ference

# Lecture 20. Inference on PGMs

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

## This lecture

- Probabilistic inference: computing (conditional) marginals from joint distributions
  - \* Needed to learn (posterior update) in Bayesian ML
  - \* Exact inference: Elimination algorithm
  - \* Approximate inference: Sampling
- Statistical inference: Parameter estimation
  - \* Fully observed case: Factors decompose under MLE
  - \* Latent variables: Motivates the EM algorithm

2

## Probabilistic inference on PGMs

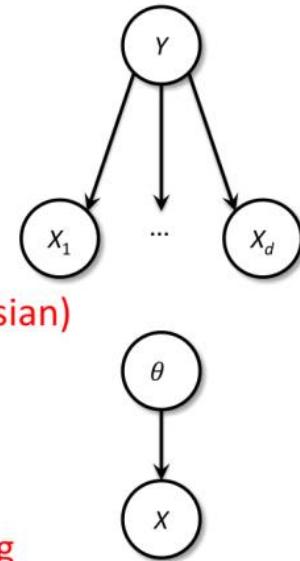
*Computing marginal and conditional distributions from the joint of a PGM using Bayes rule and marginalisation.*

*This deck: how to do it efficiently.*

3

## Two familiar examples

- Naïve Bayes (frequentist/Bayesian)
  - \* Chooses most likely class given data
  - \*  $\Pr(Y|X_1, \dots, X_d) = \frac{\Pr(Y, X_1, \dots, X_d)}{\Pr(X_1, \dots, X_d)} = \frac{\Pr(Y, X_1, \dots, X_d)}{\sum_y \Pr(Y=y, X_1, \dots, X_d)}$
- Data  $X|\theta \sim N(\theta, 1)$  with prior  $\theta \sim N(0, 1)$  (Bayesian)
  - \* Given observation  $X = x$  update posterior
  - \*  $\Pr(\theta|X) = \frac{\Pr(\theta, X)}{\Pr(X)} = \frac{\Pr(\theta, X)}{\sum_\theta \Pr(\theta, X)}$
- Joint + Bayes rule + marginalisation → anything



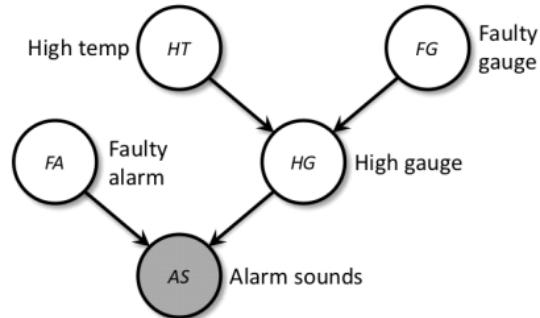
4

## Nuclear power plant

- Alarm sounds; meltdown?!

$$\Pr(HT|AS=t) = \frac{\Pr(HT, AS=t)}{\Pr(AS=t)}$$

$$= \frac{\sum_{FG, HG, FA} \Pr(AS=t, FA, HG, FG, HT)}{\sum_{FG, HG, FA, HT'} \Pr(AS=t, FA, HR, FG, HT')}$$



- Numerator (denominator similar)

expanding out sums, joint *summing once over 2<sup>5</sup> table*

$$= \sum_{FG} \sum_{HG} \sum_{FA} \Pr(HT) \Pr(HG|HT, FG) \Pr(FG) \Pr(AS=t|FA, HG) \Pr(FA)$$

distributing the sums as far down as possible *summing over several smaller tables*

$$= \Pr(HT) \sum_{FG} \Pr(FG) \sum_{HG} \Pr(HG|HT, FG) \sum_{FA} \Pr(FA) \Pr(AS=t|FA, HG)$$

*dynamic programming*

5

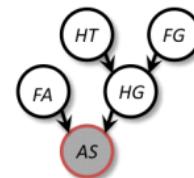
$$\phi(X \geq a) = \sum_x \phi(X=x) \delta(x=a)$$

## Nuclear power plant (cont.)

$$= \Pr(HT) \sum_{FG} \Pr(FG) \sum_{HG} \Pr(HG|HT, FG) \sum_{FA} \Pr(FA) \Pr(AS = t|FA, HG)$$

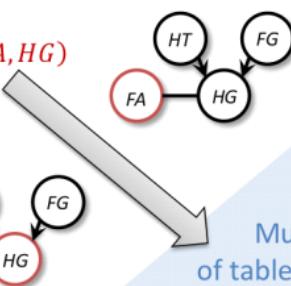
eliminate AS: since AS observed, really a no-op

*intermediate table*



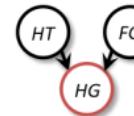
$$= \Pr(HT) \sum_{FG} \Pr(FG) \sum_{HG} \Pr(HG|HT, FG) \sum_{FA} \Pr(FA) m_{AS}(FA, HG)$$

eliminate FA: multiplying 1x2 by 2x2



$$= \Pr(HT) \sum_{FG} \Pr(FG) \sum_{HG} \Pr(HG|HT, FG) m_{FA}(HG)$$

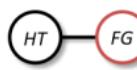
eliminate HG: multiplying 2x2x2 by 2x1



Multiplication of tables, followed by summing, is actually matrix multiplication

$$= \Pr(HT) \sum_{FG} \Pr(FG) m_{HG}(HT, FG)$$

eliminate FG: multiplying 1x2 by 2x2



$$= \Pr(HT) m_{HG}(HT)$$



$$m_{FA}(HG) = \begin{matrix} & \text{FA} \\ & \begin{matrix} f & t \end{matrix} \\ \begin{matrix} f & t \end{matrix} & \begin{bmatrix} 0.6 & 0.4 \end{bmatrix} \times \begin{matrix} & \text{HG} \\ & \begin{matrix} f & t \end{matrix} \\ \begin{matrix} f & t \end{matrix} & \begin{bmatrix} 1.0 & 0 \\ 0.8 & 0.2 \end{bmatrix} \end{matrix} \end{matrix}$$

6

## Elimination algorithm

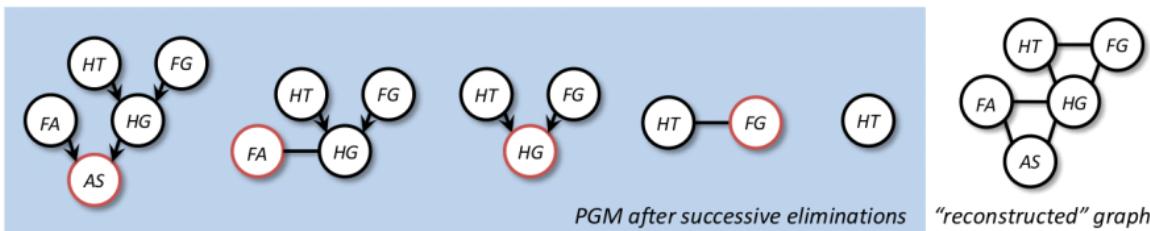
**Eliminate** (Graph  $G$ , Evidence nodes  $E$ , Query nodes  $Q$ )

1. Choose node ordering  $I$  such that  $Q$  appears last
2. Initialise empty list  $active$
3. For each node  $X_i$  in  $G$ 
  - a) Append  $\Pr(X_i | parents(X_i))$  to  $active$
4. For each node  $X_i$  in  $E$ 
  - a) Append  $\delta(X_i, x_i)$  to  $active$
5. For each  $i$  in  $I$ 
  - a) potentials = Remove tables referencing  $X_i$  from  $active$
  - b)  $N_i$  = nodes other than  $X_i$  referenced by tables
  - c) Table  $\phi_i(X_i, X_{N_i}) = \text{product of tables}$
  - d) Table  $m_i(X_{N_i}) = \sum_{X_i} \phi_i(X_i, X_{N_i})$
  - e) Append  $m_i(X_{N_i})$  to  $active$
6. Return  $\Pr(X_Q | X_E = x_E) = \phi_Q(X_Q) / \sum_{X_Q} \phi_Q(X_Q)$

initialise  
evidence  
marginalise  
normalise

7

## Runtime of elimination algorithm



- Each step of elimination
  - \* Removes a node
  - \* Connects node's remaining neighbours  
→ forms a clique in the "reconstructed" graph  
(cliques are exactly r.v.'s involved in each sum)
- Time complexity exponential in largest clique
- Different elimination orderings produce different cliques
  - \* Treewidth: minimum over orderings of the largest clique
  - \* Best possible time complexity is exponential in the treewidth e.g.  $O(2^{\text{tw}})$

8

## Mini Summary

### (Exact) probabilistic inference on PGMs

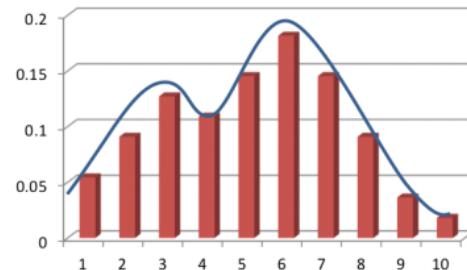
- What? Marginalise out variables, Condition
- Why? Example: Bayesian posterior updates!
- How? The elimination algorithm
- How long? Time exponential in treewidth

Next time: Approximate PGM probabilistic inference

9

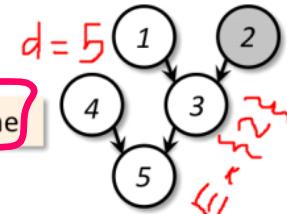
# Probabilistic inference by simulation

- Exact probabilistic inference can be expensive/impossible
  - \* Integration may not have analytical solution!
- Can we approximate numerically?
- Idea: sampling methods
  - \* Approximate distribution by histogram of a sample
  - \* We can't trivially sample: (1) only know desired distribution up to a (normalising) constant (2) naïve sampling approaches are inefficient in high dimensions.



10

## Gibbs sampling



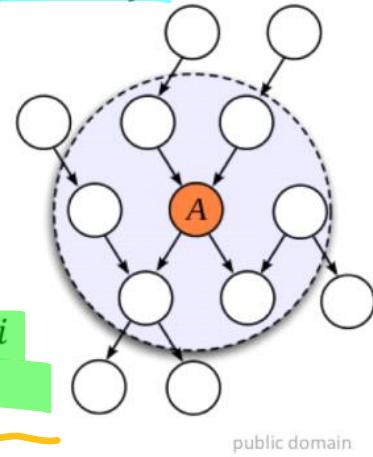
Divide and conquer: Sampling single variable at a time

- Given: D-PGM on  $d$  random variables  
Given: evidence values  $\mathbf{x}_E$  over variables  $E \subset \{1, \dots, d\}$   
Goal: many approximately independent samples from joint conditioned on  $\mathbf{x}_E$
- 1. Initialise with a starting  $\mathbf{X}^{(0)} = (X_1^{(0)}, \dots, X_d^{(0)})$  with  $\mathbf{X}_E^{(0)} = \mathbf{x}_E$
- 2. Repeat many times
  - Pick non-evidence node  $X_j$  uniformly at random
  - Sample single node  $X'_j \sim p(X_j | X_1^{(i-1)}, \dots, X_{j-1}^{(i-1)}, X_{j+1}^{(i-1)}, \dots, X_d^{(i-1)})$
  - Save entire joint sample  $\mathbf{X}^{(i)} = (X_1^{(i-1)}, \dots, X_{j-1}^{(i-1)}, X'_j, X_{j+1}^{(i-1)}, \dots, X_d^{(i-1)})$
- Exercise: Why always  $\mathbf{X}_E^{(i)} = \mathbf{x}_E$ ?
- Need not update nodes in random order, e.g. parents first order  
But do need to be able to sample from conditionals (e.g. conjugacy)

11

## Markov blanket

- Intuition: all the nodes that you directly depend on.  
*Not just your parents/children!*
- Consider node  $X_i$  in D-PGM on nodes  $N = \{1, \dots, d\}$
- Markov blanket  $\text{MB}(i)$  of  $X_i$ :
  - Nodes  $B \subseteq N \setminus \{i\}$  such that...
  - $X_i$  independent of  $\mathbf{X}_{\bar{B} \setminus \{i\}}$  given  $\mathbf{X}_B$
  - $p(X_i | X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_d) = p(X_i | \text{MB}(X_i))$
- In D-PGM Markov blanket is:
  - Parents of  $i$ , children of  $i$ , parents of children of  $i$
  - $p(X_i | \text{MB}(X_i)) \propto p(X_i | X_{\pi_i}) \prod_{k:i \in \pi_k} p(X_k | X_{\pi_k})$



12

## Markov Chain Monte Carlo (MCMC)

- Gibbs sampling produces a chain of samples  $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \dots$  approximating draws from  $p(\mathbf{X}_{\bar{E}} | \mathbf{X}_E = \mathbf{x}_E)$
- How good an approximation? Independent draws possible?
- Samples form a Markov chain: Each  $\mathbf{X}^{(i)}$  depends only  $\mathbf{X}^{(i-1)}$ 
  - States are all possible values taken by joint samples
  - Initial distribution  $\mathbf{p}_0$  of state  $\mathbf{X}^{(0)}$  given by initialisation process
  - Transition probability matrix  $\mathbf{T}$  given by PGM conditional probabilities
  - Combines to: distribution  $\mathbf{p}_i = (\mathbf{T})^i \mathbf{p}_0$  of state  $\mathbf{X}^{(i)}$ .
- Burn in:** Run Gibbs long enough and  $\mathbf{X}^{(i)} \sim p(\mathbf{X}_{\bar{E}} | \mathbf{X}_E = \mathbf{x}_E)$ 
  - "Limiting distribution"  $\lim_{i \rightarrow \infty} \mathbf{p}_i$  is  $p(\mathbf{X}_{\bar{E}} | \mathbf{X}_E = \mathbf{x}_E)$  under condition that no entry of  $\mathbf{T}$  is zero ("ergodicity" – may not always hold)
  - Solution: throw away first few thousand samples
- Thinning:** Want saved full samples to be independent
  - Neighbouring  $\mathbf{X}^{(i)}, \mathbf{X}^{(i+1)}$  are highly correlated. **Intuition why?**
  - Solution: only keep every 100 or so samples



13

## Initialising Gibbs: Forward Sampling

- Set all evidence nodes to observed values
- Remaining nodes, parent-first order
  - \* Node has no parents? Sample from its D-PGM marginal
  - \* Sample node given previously sampled parents
- However Markov chain theory tells us MCMC converges irrespective of initial sample's distribution
  - \* The limiting distribution – the “equilibrium distribution” – is a property of the transition matrix (the PGM's joint) not the initial distribution

14

## Now what??



- With our  $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(T)}$  in hand after running Gibbs for a while with burn-in and thinning...
- These form “i.i.d.” sample of  $p(\mathbf{X}_{\bar{E}} | \mathbf{X}_E = \mathbf{x}_E)$
- We can do heaps!
  - a) Can approximate the distribution via a histogram of these samples (make bins, form counts).
  - b) Marginalising out variables == Dropping components from samples
  - c) Expectations: Estimating by sample mean of samples
- Posterior  $p(\mathbf{w} | \mathbf{X}_{tr}, \mathbf{y}_{tr})$  combine (a) and (b)  
Mean posterior point estimate, combine with (c)

15

## Mini Summary

Approximate probabilistic inference on PGMs

- Why? Summation/integration may be costly
- Why? Integration may be impossible analytically
- Briefly: Gibbs sampling

Next time: Statistical inference on PGMs

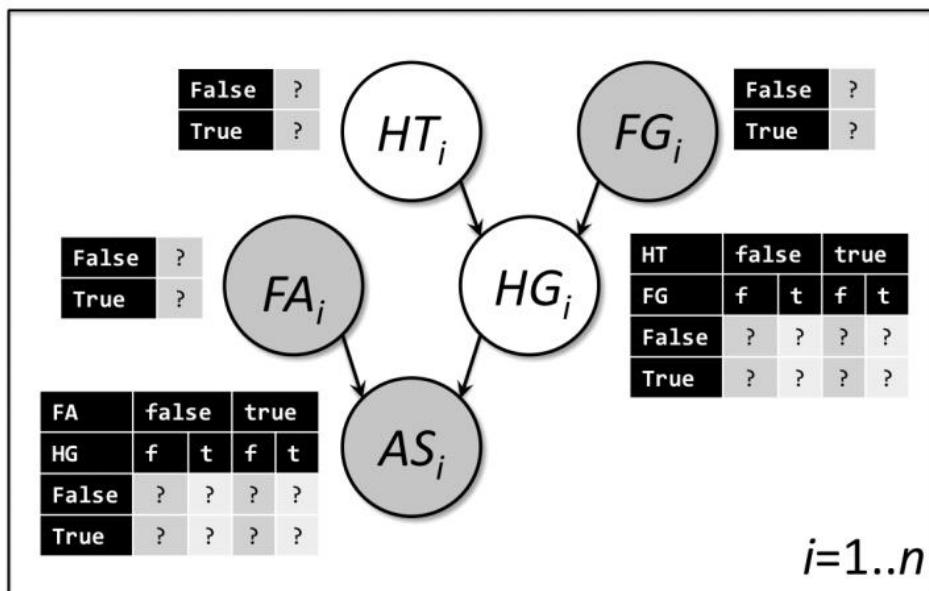
16

## Statistical inference on PGMs

*Learning from data – fitting probability tables to observations (eg as a frequentist; a Bayesian would just use probabilistic inference to update prior to posterior)*

17

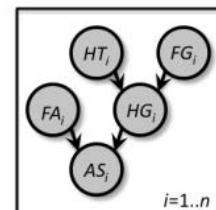
## Have PGM, Some observations, No tables...



18

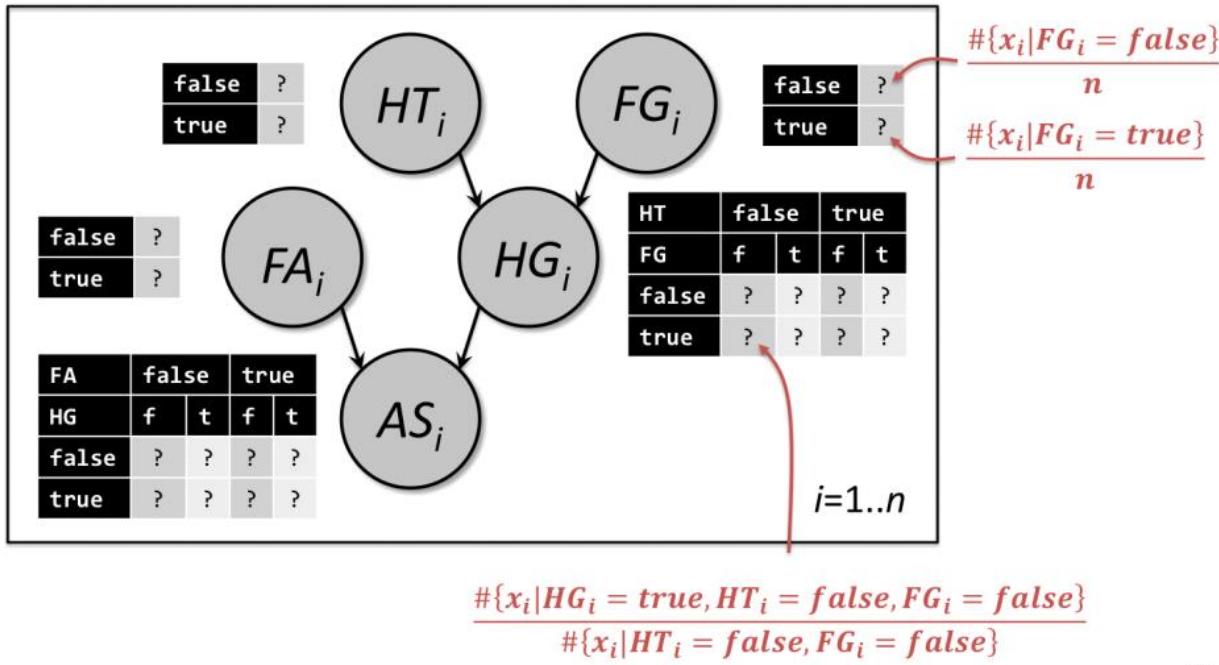
## Fully-observed case is “easy”

- Max-Likelihood Estimator (MLE) says
  - \* If we observe *all* r.v.’s  $X$  in a PGM independently  $n$  times  $x_i$
  - \* Then maximise the *full* joint
 
$$\arg \max_{\theta \in \Theta} \prod_{i=1}^n \prod_j p(X^j = x_i^j | X^{\text{parents}(j)} = x_i^{\text{parents}(j)})$$
- Decomposes easily, leads to counts-based estimates
  - \* Maximise log-likelihood instead; becomes sum of logs
 
$$\arg \max_{\theta \in \Theta} \sum_{i=1}^n \sum_j \log p(X^j = x_i^j | X^{\text{parents}(j)} = x_i^{\text{parents}(j)})$$
  - \* Big maximisation of all parameters together, decouples into small independent problems
- Example is training a naïve Bayes classifier



19

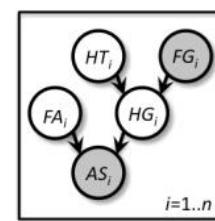
## Example: Fully-observed case



20

## Presence of unobserved variables trickier

- But most PGMs you'll encounter will have latent, or unobserved, variables
  - What happens to the MLE?
    - \* Maximise likelihood of observed data only
    - \* Marginalise full joint to get to desired "partial" joint
    - \*  $\arg \max_{\theta \in \Theta} \prod_{i=1}^n \sum_{\text{latent } j} \prod_j p(X^j = x_i^j | X^{\text{parents}(j)} = x_i^{\text{parents}(j)})$
    - \* This won't decouple – oh-no's!!
- Use EM algorithm!



21

# Summary

- Probabilistic inference on PGMs
  - \* What is it and why do we care?
  - \* Elimination algorithm; complexity via cliques
  - \* Monte Carlo approaches as alternate to exact integration
- Statistical inference on PGMs
  - \* What is it and why do we care?
  - \* Straight MLE for fully-observed data
  - \* EM algorithm for mixed latent/observed data

Next time: deeper dive into HMMs and more

## Lecture 21

2020年10月20日 星期二 10:37



21\_hmm

# Lecture 21. HMMs and Message Passing

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



THE UNIVERSITY OF  
MELBOURNE

Copyright: University of Melbourne

COMP90051 Statistical Machine Learning

## This lecture

- Hidden Markov models – detailed PGM case study
  - \* Brief recap of model
  - \* “Evaluation”: Forward-Background Algorithm = elimination
  - \* “Learning”: Baum Welch = MLE
  - \* “Decoding”: Viterbi = elimination variant with sum → max
- Message passing
  - \* Sum-product generalises elimination algorithm
  - \* Variants for ring operators, max-product for Viterbi
  - \* Factor graphs

# Hidden Markov Models

**Model of choice for sequential data.** A form of clustering (or dimensionality reduction) for discrete time series.

3

## HMM Formulation

- Formulated as directed PGM
  - \* therefore joint expressed as

$$P(\mathbf{o}, \mathbf{q}) = P(q_1)P(o_1|q_1) \prod_{i=2}^T P(q_i|q_{i-1})P(o_i|q_i)$$

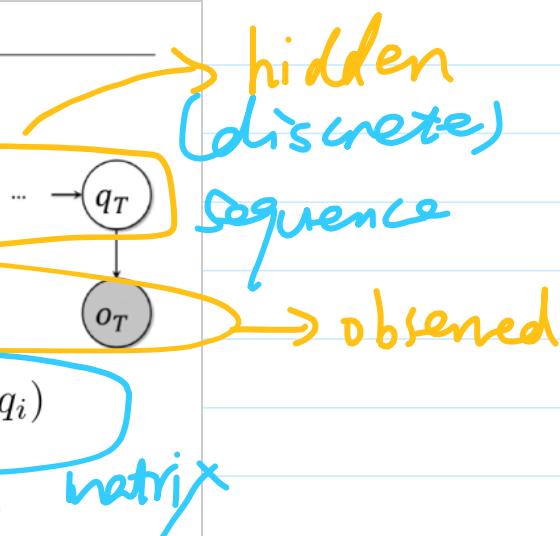
\* **bold** variables are shorthand for vector of  $T$  values

- Parameters (for *homogenous* HMM)

$A = \{a_{ij}\}$  transition probability matrix;  $\forall i : \sum_j a_{ij} = 1$

$B = \{b_i(o_k)\}$  output probability matrix;  $\forall i : \sum_k b_i(o_k) = 1$

$\Pi = \{\pi_i\}$  the initial state distribution;  $\sum_i \pi_i = 1$



4

## Fundamental HMM Tasks

HMM Task	PGM Task
<b>Evaluation.</b> Given an HMM $\mu$ and observation sequence $\mathbf{o}$ , determine likelihood $\Pr(\mathbf{o} \mu)$	Probabilistic inference
<b>Decoding.</b> Given an HMM $\mu$ and observation sequence $\mathbf{o}$ , determine most probable hidden state sequence $\mathbf{q}$	MAP point estimate
<b>Learning.</b> Given an observation sequence $\mathbf{o}$ and set of states, learn parameters $A, B, \Pi$	Statistical inference

5

## “Evaluation” a.k.a. marginalisation

- Compute prob. of observations  $\mathbf{o}$  by summing out  $\mathbf{q}$

$$\begin{aligned} P(\mathbf{o}|\mu) &= \sum_{\mathbf{q}} P(\mathbf{o}, \mathbf{q}|\mu) \quad \text{marginalisation} \\ &= \sum_{q_1} \sum_{q_2} \dots \sum_{q_T} P(q_1)P(o_1|q_1)P(q_2|q_1)P(o_2|q_2) \dots P(q_T|q_{T-1})P(o_T|q_T) \end{aligned}$$

$\underbrace{\qquad}_{\mathbf{q} \rightarrow \text{vector}}$

- Make this more efficient by moving the sums

$$P(\mathbf{o}|\mu) = \sum_{q_1} P(q_1)P(o_1|q_1) \sum_{q_2} P(q_2|q_1)P(o_2|q_2) \dots \sum_{q_T} P(q_T|q_{T-1})P(o_T|q_T)$$

- Déjà vu? Maybe we could do var. elimination...

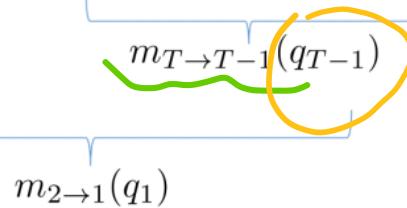
6

## Elimination = Backward Algorithm

$$P(\mathbf{o}|\mu) = \sum_{q_1} P(q_1) P(o_1|q_1) \sum_{q_2} P(q_2|q_1) P(o_2|q_2) \dots \sum_{q_T} P(q_T|q_{T-1}) P(o_T|q_T)$$

Eliminate  $q_T$ 

...

Eliminate  $q_2$ “Eliminate”  $q_1$ 

$$P(\mathbf{o}|\mu) = \sum_{q_1} P(q_1) P(o_1|q_1) m_{2\rightarrow 1}(q_1)$$



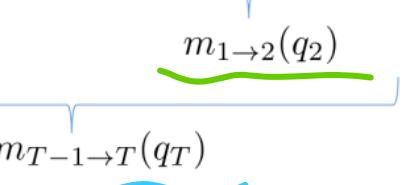
7

## Elimination = Forward Algorithm

$$P(\mathbf{o}|\mu) = \sum_{q_T} P(o_T|q_T) \sum_{q_{T-1}} P(q_T|q_{T-1}) P(o_T|q_T) \dots \sum_{q_1} P(q_2|q_1) P(q_1) P(o_1|q_1)$$

Eliminate  $q_1$ 

...

Eliminate  $q_{T-1}$ “Eliminate”  $q_T$ 

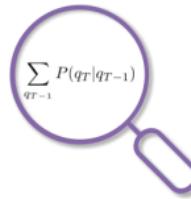
$$P(\mathbf{o}|\mu) = \sum_{q_1} P(o_T|q_T) m_{T-1\rightarrow T}(q_T)$$



8

## Variable elimination perspective

- Both algorithms are just *variable elimination* using different orderings
  - $q_T \dots q_1 \rightarrow$  backward algorithm
  - $q_1 \dots q_T \rightarrow$  forward algorithm
  - both have time complexity  $O(TL^2)$  for  $L$  the label set size
- Can use either to compute  $P(\mathbf{o})$
- Even though these are just instances of elimination, they pre-date general PGM inference.
  - E.g. called the “forward-background algorithm”
  - Both directions useful in statistical inference (next)



9

## Mini Summary

- HMM
  - Powerful and versatile model
  - “Algorithms” for HMM just instances of PGM machinery
- Evaluation by Forward / Backward
  - Just elimination by two different orderings

Next time: Statistical inference (learning) example of EM

10

# Statistical Inference (Learning)

- Learn parameters  $\mu = (A, B, \pi)$ , given observation sequence  $\mathbf{o}$  MLE
- Called "Baum Welch" algorithm which uses  $\text{EM}^*$  to approximate MLE,  $\operatorname{argmax}_{\mu} P(\mathbf{o} | \mu)$ :
  - initialise  $\mu^1$ , let  $j=1$
  - compute expected marginal distributions  $P(q_t | \mathbf{o}, \mu^j)$  for all  $t$ ; and  $P(q_{t-1}, q_t | \mathbf{o}, \mu^j)$  for  $t=2..T$
  - fit model  $\mu^{j+1}$  based on expectations
  - repeat from step 2, with  $j=j+1$E step  
M step
- Expectations (2) computed using forward-backward

\* Expectation-Maximisation (EM) is coming up

11

# Forward-Backward for $P(q_i | \mathbf{o})$

- Forward-Backward gives: messages,  $P(\mathbf{o})$
  - Bayes rule:  $P(q_i | \mathbf{o}) = \frac{P(q_i, \mathbf{o})}{P(\mathbf{o})}$
  - Marginalisation:  $P(q_i | \mathbf{o}) = \sum_{q_1, \dots, q_{i-1}, q_{i+1}, \dots, q_T} P(\mathbf{q}, \mathbf{o})$   
 $= \left( \sum_{q_1, \dots, q_{i-1}} P(o_1, \dots, o_{i-1}, q_1, \dots, q_i) \right) P(o_i | q_i) \left( \sum_{q_{i+1}, \dots, q_T} P(o_{i+1}, \dots, o_T, q_{i+1}, \dots, q_T | q_i) \right)$   
 $= m_{i-1 \rightarrow i}(q_i) P(o_i | q_i) m_{i \rightarrow i-1}(q_i)$
- $$P(q_i | \mathbf{o}) = \frac{1}{P(\mathbf{o})} m_{i-1 \rightarrow i}(q_i) P(o_i | q_i) m_{i \rightarrow i-1}(q_i)$$

forward
backward

12

## Forward-Backward for $P(q_{i-1}, q_i | \mathbf{o})$

- Similar pattern:  $P(q_{i-1}, q_i | \mathbf{o}) = \frac{P(q_{i-1}, q_i, \mathbf{o})}{P(\mathbf{o})}$
- Marginalisation:  $P(q_{i-1}, q_i | \mathbf{o}) = \sum_{q_1, \dots, q_{i-2}, q_{i+1}, \dots, q_T} P(\mathbf{q}, \mathbf{o})$   
 $= \left( \sum_{q_1, \dots, q_{i-2}} P(o_1, \dots, o_{i-2}, q_1, \dots, q_{i-1}) \right) P(o_{i-1} | q_{i-1}) P(q_i | q_{i-1}) P(o_i | q_i) \left( \sum_{q_{i+1}, \dots, q_T} P(o_{i+1}, \dots, o_T, q_{i+1}, \dots, q_T | q_i) \right)$   
 $= m_{i-2 \rightarrow i-1}(q_{i-1}) P(o_{i-1} | q_{i-1}) P(q_i | q_{i-1}) P(o_i | q_i) m_{i \rightarrow i-1}(q_i)$

$$\frac{1}{P(\mathbf{o})} m_{i-2 \rightarrow i-1}(q_{i-1}) P(o_{i-1} | q_{i-1}) P(q_i | q_{i-1}) P(o_i | q_i) m_{i \rightarrow i-1}(q_i)$$

**forward** **backward**

13

## Mini Summary

- Statistical inference for HMMs
  - \* “Just” learning or MLE as we’re frequentist here
  - \* Unobserved random variables means: EM (more later on)
  - \* Maximisation step: looks like MLE – nothing new
  - \* Expectation step: achieved by forward-backward messages
- “Baum-Welch” is the original name of this algorithm

**Next time:** Message passing a little more generally

14

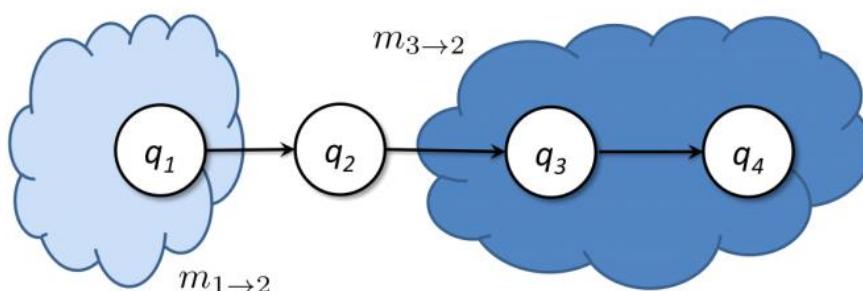
# Message Passing

*Sum-product algorithm for efficiently computing marginal distributions over trees. An extension of variable elimination algorithm.*

15

## Inference as message passing

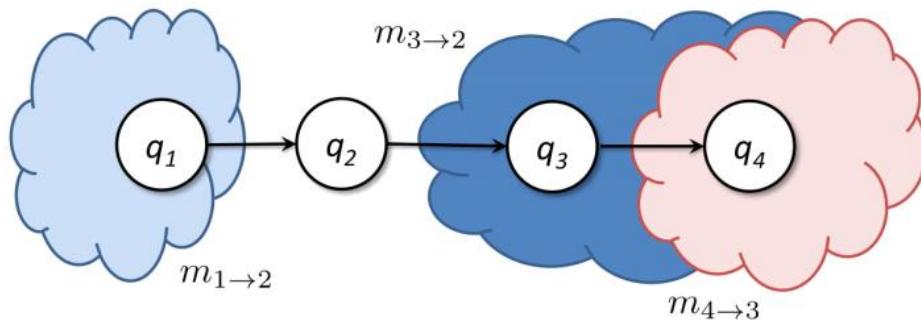
- Each  $m$  can be considered as a **message** which summarises the effect of the rest of the graph on the current node marginal.
  - \* Inference = *passing messages between all nodes*



16

## Inference as message passing

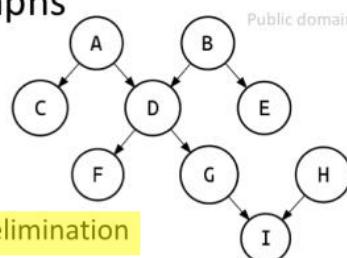
- Messages vector valued, i.e., function of target label
- Messages defined recursively: left to right, or right to left for the HMM



17

## Sum-product algorithm

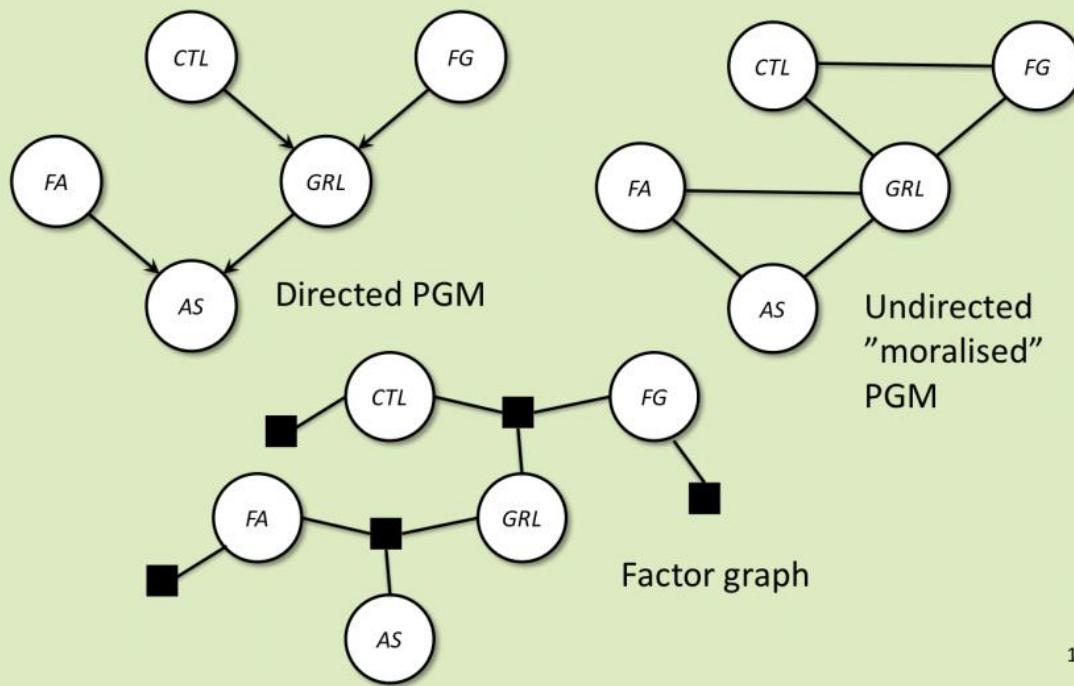
- Message passing in more general graphs
  - \* applies to chains, trees and poly-trees (D-PGMs with >1 parent)
  - \* 'sum-product' derives from:
    - product = product of incoming messages
    - sum = summing out the effect of rv(s) aka elimination
- Algorithm supports other operations (semi-rings\*)
  - \* e.g., max-product, swapping sum for max
  - \* Viterbi algorithm is the max-product variant of forward algorithm, solves the  $\text{argmax}_q P(q|o)$



\* A ring is an algebraic structure generalizing addition/multiplication on reals.  
Semi-ring relaxes requirement of additive inverse.

18

## Application to Directed PGMS

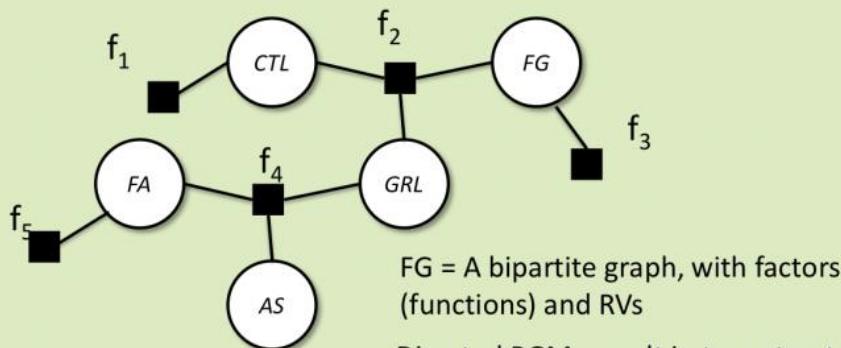


19

## Factor graphs

$$f_1(CTL) = P(CTL)$$

$$f_2(CTL, GRL, FG) = P(GRL|CTL, FG)$$

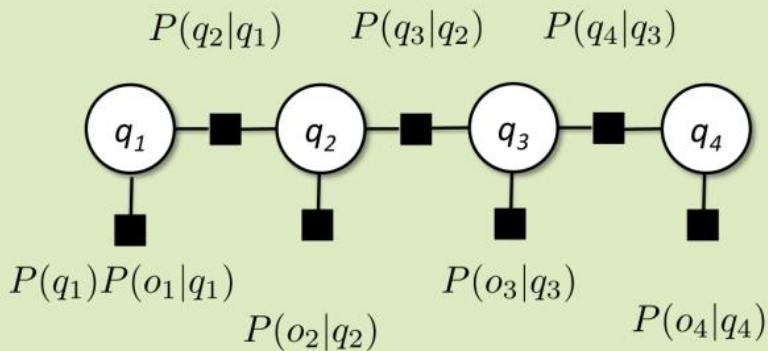


FG = A bipartite graph, with factors (functions) and RVs

Directed PGMs result in tree-structured FG

20

## Factor graph for the HMM

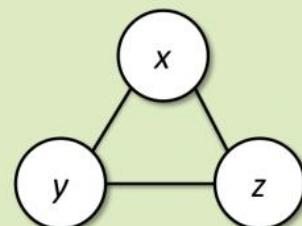


Effect of observed nodes incorporated into unary factors

21

## Advantage of Factor Graphs

- Factorisation is a central idea
- D-PGMs and U-PGMs not able to fully represent arbitrary factorisations of joints



$$\begin{aligned} p(x, y, z) &\propto \varphi(x, y)\varphi(y, z)\varphi(z, x) \\ p(x, y, z) &\propto \varphi(x, y, z) \end{aligned}$$

- Better representation of factorisations has advantages; factor graphs are general.

22

## Sum-Product over Factor Graphs

- Two types of messages :
  - \* between factors and RVs; and between RVs and factors
  - \* they summarise a complete sub-graph
- E.g.,
 
$$m_{f_2 \rightarrow GRL}(GRL) = \sum_{CTL} \sum_{FG} f_2(GRL, CTL, FG) m_{CTL \rightarrow f_2}(CTL) m_{FG \rightarrow f_2}(FG)$$
- Structure inference as “gather-and-distribute”
  - \* gather messages from leaves of tree towards root
  - \* then propagate message back down from root to leaves

23

## Summary

- HMMs as example PGMs
  - \* formulation as PGM
  - \* independence assumptions
  - \* probabilistic inference using forward-backward
  - \* statistical inference using expectation-maximization
  - \* decoding as max-product
- Message passing: general inference method for U-PGMs
  - \* sum-product & max-product
  - \* factor graphs

**Next time:** Gaussian mixture models and EM

24

## Lecture 22

2020年10月21日 星期三 14:56



22\_gmm

# Lecture 22. Gaussian Mixture Models.

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



THE UNIVERSITY OF  
MELBOURNE

Copyright: University of Melbourne

# This lecture

- Unsupervised learning
  - \* Diversity of problems
  - \*  $k$ -means refresher
- Gaussian mixture model (GMM)
  - \* A probabilistic approach to clustering
  - \* The GMM model
  - \* GMM clustering as an optimisation problem
- Starting Expectation-Maximisation (EM) algorithm

# Unsupervised Learning

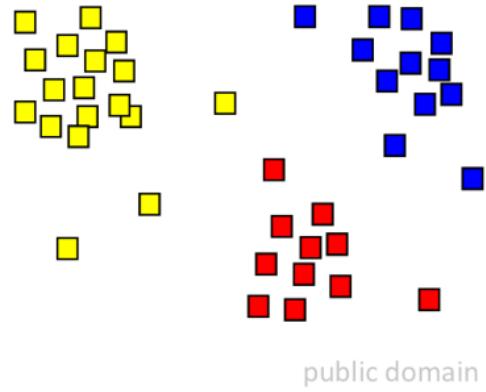
A large branch of ML that concerns  
with learning the structure of the  
data in the absence of labels

## Main learning paradigms so far

- Supervised learning: Overarching aim is making predictions from data
- We studied methods in the context of this aim: e.g. linear/logistic regression, DNN, SVM
- We had instances  $x_i \in \mathbb{R}^m$ ,  $i = 1, \dots, n$  and corresponding labels  $y_i$  for model fitting, aiming to predict labels for new instances
- Can be viewed as a function approximation problem, but with a big caveat: ability to generalise is critical
- Bandits: a setting of partial supervision where subroutine in contextual bandits requires supervised learning

## Now: Unsupervised learning

- In unsupervised learning, there is no dedicated variable called a “label”
- Instead, we just have a set of points  $x_i \in \mathbb{R}^m, i = 1, \dots, n$
- Aim of unsupervised learning is to explore the structure (patterns, regularities) of data
- The aim of “exploring the structure” is vague



# Unsupervised learning tasks

- Diversity of tasks fall into unsupervised learning category
  - \* Clustering (now)
  - \* Dimensionality reduction (autoencoders)
  - \* Learning parameters of probabilistic models (before/now)
- Applications and related tasks are numerous :
  - \* Market basket analysis. E.g., use supermarket transaction logs to find items that are frequently purchased together
  - \* Outlier detection. E.g., find potentially fraudulent credit card transactions
  - \* Often unsupervised tasks in (supervised) ML pipelines

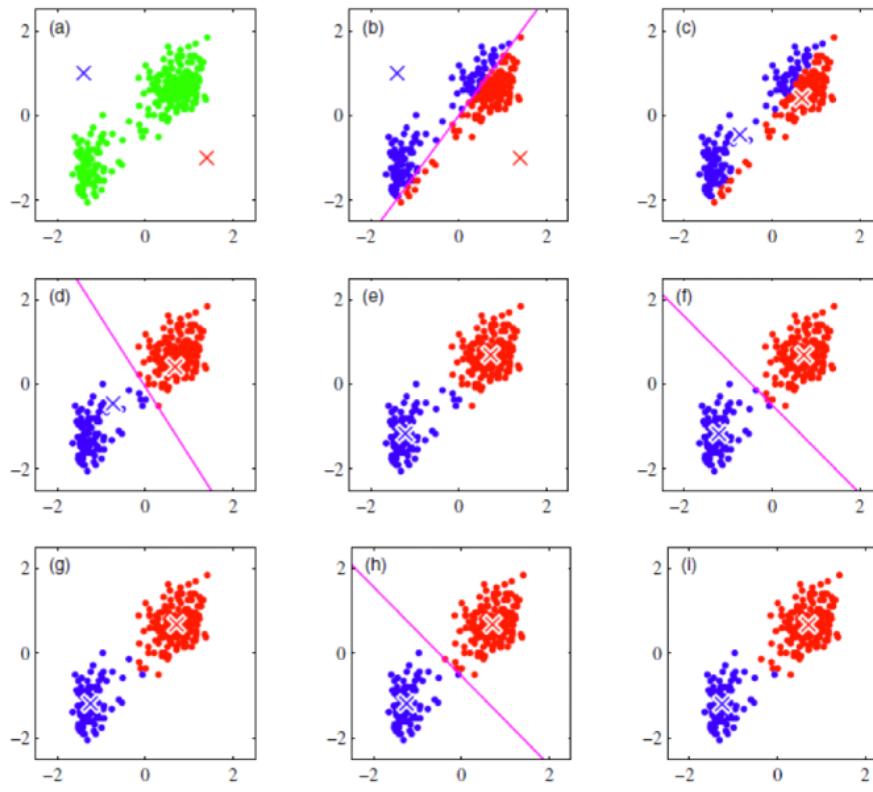
## Refresher: k-means clustering

1. Initialisation: choose  $k$  cluster **centroids** randomly
2. Update:
  - a) Assign points to the nearest\* centroid
  - b) Compute centroids under the current assignment
3. Termination: if no change then stop
4. Go to Step 2

\*Distance represented by choice of metric typically  $L_2$

Still one of the most popular data mining algorithms.

# Refresher: $k$ -means clustering



Requires specifying  
the number of  
clusters in advance

Measures  
“dissimilarity” using  
Euclidean distance

Finds “spherical”  
clusters

An iterative  
optimization  
procedure

Data: Old Faithful  
Geyser Data: waiting  
time between  
eruptions and the  
duration of eruptions

Figure: Bishop, Section 9.1

## Mini Summary

- Unsupervised learning
  - \* Face value: drop labels from training. That's it
  - \* Actually: catch-all for many many ML tasks, even as steps in supervised learning pipelines
- Refresher: *k*-means
  - \* Import next as we introduce GMMs

Next time: The Gaussian mixture model

# Gaussian Mixture Model

*A probabilistic view of clustering.  
Simple example of a latent variable model.*

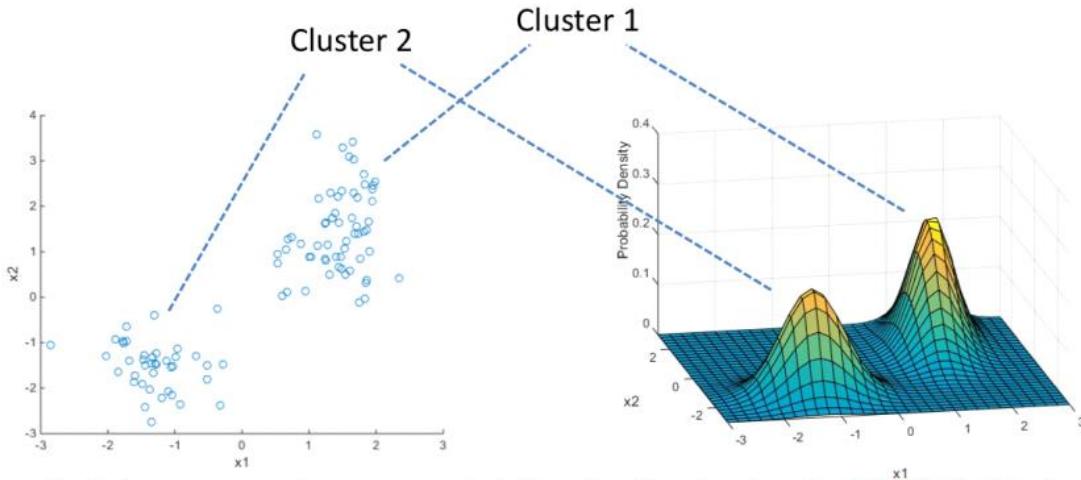
# Modelling uncertainty in data clustering

- ***k*-means clustering assigns each point to exactly one cluster**
  - \* Does this make sense for points that are between two clusters?
  - \* Clustering is often not well defined to begin with!
- Like *k*-means, a probabilistic mixture model requires the user to choose the number of clusters in advance
- Unlike *k*-means, the probabilistic model gives us a power to express uncertainty about the origin of each point
  - \* Each point originates from cluster  $c$  with probability  $w_c$ ,  $c = 1, \dots, k$
- That is, each point still originates from one particular cluster (aka component), but we are not sure from which one
- Next
  - \* Clustering becomes model fitting in probabilistic sense. Philosophically satisfying.
  - \* Individual components modelled as Gaussians
  - \* Fitting illustrates general Expectation Maximization (EM) algorithm

# Clustering: probabilistic model

Data points  $x_i$  are independent and identically distributed (i.i.d.) samples from a mixture of  $K$  distributions (components)

Each component in the mixture is what we call a cluster



In principle, we can adopt any probability distribution for the **components**, however, the normal distribution is a common modelling choice → Gaussian Mixture Model

# Normal (aka Gaussian) distribution

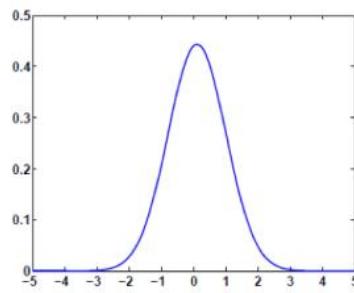
- Recall that a 1D Gaussian is

$$\mathcal{N}(x|\mu, \sigma) \equiv \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

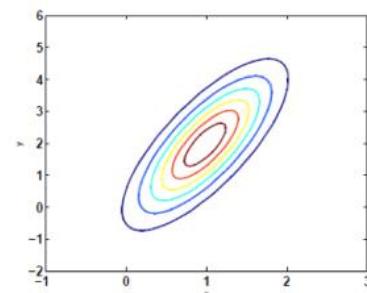
- And a  $d$ -dimensional Gaussian is

$$\mathcal{N}(x|\mu, \Sigma) \equiv (2\pi)^{-\frac{d}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

- \*  $\Sigma$  is a PSD symmetric  $d \times d$  matrix, the **covariance matrix**
- \*  $|\Sigma|$  denotes determinant
- \* No need to memorize the full formula.



(a) 1-Dim

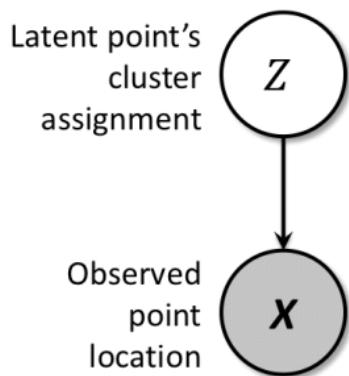


(b) 2-Dim

Figure: Bishop 13

## Gaussian mixture model (GMM): One point

- Cluster assignment of point
  - \* Multinomial distribution on  $k$  outcomes
  - \*  $P(Z = j)$  described by  $P(C_j) = w_j \geq 0$  with  $\sum_{j=1}^k w_j = 1$
- Location of point
  - \* Each cluster has its own Gaussian distribution
  - \* Location of point governed by its cluster assignment
  - \*  $P(X|Z = j) = \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$  class conditional density
- Model's parameters:  $w_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, j = 1, \dots, k$



## From marginalisation to mixture distribution

- When fitting the model to observations, we'll be maximising likelihood of observed portions of the data (the  $\mathbf{X}$ 's) not the latent parts (the  $\mathbf{Z}$ 's)

- Marginalising out the  $\mathbf{Z}$ 's derives the "familiar" mixture distribution

- Gaussian mixture distribution:

$$\begin{aligned} P(\mathbf{x}) &\equiv \sum_{j=1}^k w_j \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \\ &\equiv \sum_{j=1}^k P(C_j) P(\mathbf{x} | C_j) \end{aligned}$$

P(z)  $P(\mathbf{x}|z_j)$

- A convex combination of Gaussians
- Simply marginalization at work

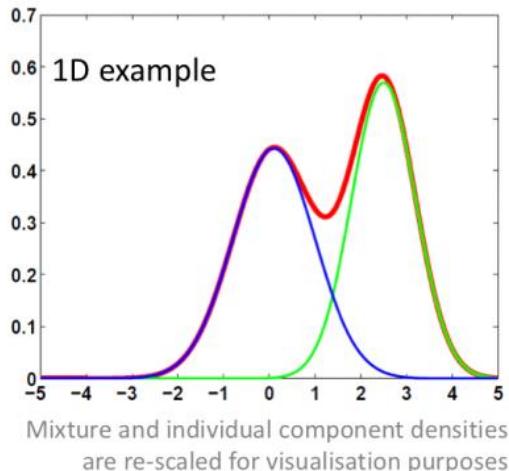


Figure: Bishop

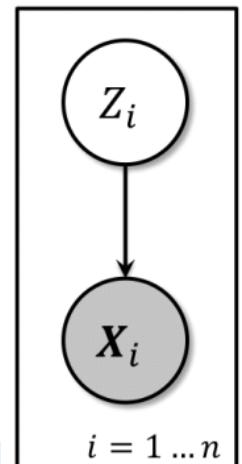
# Clustering as model estimation

- Given a set of data points, we assume that data points are generated by a GMM

- Each point in our dataset originates from our mixture distribution
  - Shared parameters between points:  
w00t independence assumption

- Clustering now amounts to finding parameters of the GMM that “best explains” observed data

- Call upon old friend MLE principle to find parameter values that maximise  $p(x_1, \dots, x_n)$



## Mini Summary

- GMM is just another D-PGM
- Some variables are observed some latent
- Convenient to model location as generated by cluster assignment
- Shared clusters arise from independence b/w points
- Mixture distribution arises algebraically from marginalisation

**Next:** MLE to fit the model, again motivating EM algorithm

# Motivating (again) Expectation-Maximisation Algorithm

*We want to implement MLE but we have unobserved r.v.'s that prevent clean decomposition as happens in fully observed settings*

# Fitting the GMM

- Modelling the data points as independent, aim is to find  $P(C_j), \mu_j, \Sigma_j, j = 1, \dots, k$  that maximise

$$P(x_1, \dots, x_n) = \prod_{i=1}^n \sum_{j=1}^k P(C_j) P(x_i | C_j)$$

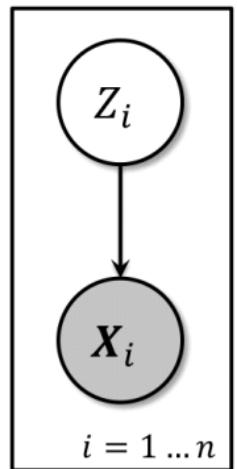
where  $P(x|C_j) \equiv \mathcal{N}(x|\mu_j, \Sigma_j)$

Can be solved analytically?

- Taking the derivative of this expression is pretty awkward, try the usual log trick

$$\log P(x_1, \dots, x_n) = \sum_{i=1}^n \log \left( \sum_{j=1}^k P(C_j) P(x_i | C_j) \right)$$

→ Expectation-Maximisation (EM)



# Motivation of EM

- Consider a parametric probabilistic model  $p(\mathbf{X}|\boldsymbol{\theta})$ , where  $\mathbf{X}$  denotes data and  $\boldsymbol{\theta}$  denotes a vector of parameters
- According to MLE, we need to maximise  $p(\mathbf{X}|\boldsymbol{\theta})$  as a function of  $\boldsymbol{\theta}$ 
  - \* equivalently maximise  $\log p(\mathbf{X}|\boldsymbol{\theta})$
- There can be a couple of issues with this task
  1. Sometimes we don't observe some of the variables needed to compute the log likelihood
    - \* Example: GMM cluster membership  $Z$  is not known in advance
  2. Sometimes the form of the log likelihood is inconvenient to work with
    - \* Example: taking a derivative of GMM log likelihood results in a cumbersome equation



art: Ebrahim at Wikimedia Commons (CC4) 20

# Expectation-Maximisation (EM) Algorithm

- Initialisation Step:

- \* Initialize  $K$  clusters:  $C_1, \dots, C_K$

$(\mu_j, \Sigma_j)$  and  $P(C_j)$  for each cluster  $j$ .

- Iteration Step:

- \* Estimate the cluster of each datum

$$p(C_j | x_i)$$

soft assign → **Expectation**  
points to cluster

- \* Re-estimate the cluster parameters

$$(\mu_j, \Sigma_j), p(C_j) \text{ for each cluster } j$$

→ **Maximisation**

# Summary

- Unsupervised learning
  - \* Diversity of problems
- Gaussian mixture model (GMM)
  - \* A probabilistic approach to clustering
  - \* The GMM model
  - \* GMM clustering as an optimisation problem
- MLE: Motivating Expectation Maximization (EM)

**Next lecture:** Getting to the bottom of EM

## Lecture 23

2020年10月27日 星期二 14:27



23\_em

# Lecture 23. Expectation Maximization.

COMP90051 Statistical Machine Learning

Semester 2, 2020  
Lecturer: Ben Rubinstein



Copyright: University of Melbourne

# This lecture

- EM intuition by GMM with recap
- Lower-bound  $\log p(\mathbf{X}|\boldsymbol{\theta})$  by  $\mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})] - \mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{Z})]$ 
  - \* Holds for any  $\boldsymbol{\theta}, p(\mathbf{Z})$
  - \* Uses Jensen's inequality (concavity of log)
- Maximise not  $\log p(\mathbf{X}|\boldsymbol{\theta})$  but lower bound, alternating:
  - \* E-Step: choose  $p(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^*)$  raises lower bound up to log-likelihood, for any  $\boldsymbol{\theta}^*$
  - \* M-Step:  $\boldsymbol{\theta}^*$  by max'ing "completed" log-likelihood; ideally, easy MLE
- Proving the E-step
- Applying EM to GMM MLE-based training
  - \* Coming full circle back to comparison to  $k$ -means

2

## Recap: EM for fitting the GMM

- Initialisation Step:
  - \* Initialize  $K$  clusters:  $C_1, \dots, C_K$   
 $(\mu_j, \Sigma_j)$  and  $P(C_j)$  for each cluster  $j$ .
- Iteration Step:
  - \* Estimate the cluster of each datum  
 $p(C_j | x_i)$   **Expectation**
  - \* Re-estimate the cluster parameters  
 $(\mu_j, \Sigma_j), P(C_j)$  for each cluster  $j$   **Maximisation**

3

## MLE vs EM

- MLE is a frequentist principle that suggests that given a dataset, the “best” parameters to use are the ones that maximise the probability of the data
  - \* MLE is a way to *formally pose* the problem
- EM is an *algorithm*
  - \* EM is a way to *solve* the problem posed by MLE
  - \* Especially convenient under unobserved latent variables
- MLE can be found by other methods such as gradient descent (but gradient descent is not always the most convenient method)

4

## EM for GMM and generally

- EM is a general approach, goes beyond GMMs
  - \* Purpose: Implement MLE under latent variables  $Z$  ('latent' is fancy for 'missing')
- What are variables, parameters in GMMs?
  - \* Variables: Point locations  $X$  and cluster assignments  $Z$ 
    - let  $z_i$  denote true cluster membership for each point  $x_i$ , computing the likelihood with known values  $z$  is simplified (see next section)
  - \* Parameters:  $\theta$  are cluster locations and scales
- What is EM really doing?
  - \* Coordinate ascent on a lower bound on the log-likelihood
    - M-step: ascent in modeled parameters  $\theta$
    - E-step: ascent in the marginal likelihood  $P(Z)$
  - \* Each step moves towards a *local* optimum
  - \* Can get stuck, can need random restarts

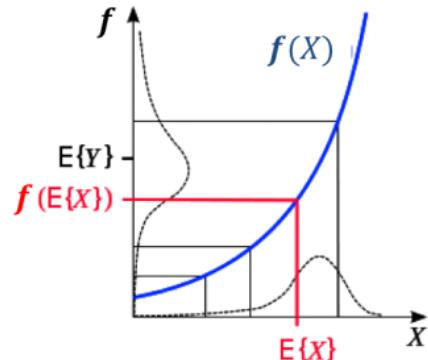
5

## Using convexity: Jensen's inequality

- Compares effect of averaging before and after applying a **convex function**:  
 $f(\text{Average}(x)) \leq \text{Average}(f(x))$
- Example:
  - \* Let  $f$  be some convex function, such as  $f(x) = x^2$
  - \* Consider  $x = [1,2,3,4,5]'$ , then  $f(x) = [1,4,9,16,25]'$
  - \* Average of input  $\text{Average}(x) = 3$
  - \*  $f(\text{Average}(x)) = 9$
  - \* Average of output  $\text{Average}(f(x)) = 12.4$
- Proof follows from the definition of convexity
  - \* Proof by induction

General statement:

- \* If  $X$  random variable,  $f$  is a convex function
- \*  $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$



plot: MHz/as at Wikimedia Commons (public domain)

6

## Putting the latent variables to use

We want to maximise  $\log p(X|\theta)$ . We don't observe  $Z$  (here discrete), but can (re)introduce it nonetheless.

$$\begin{aligned}
 \log p(X|\theta) &= \log \sum_Z p(X, Z|\theta) && \leftarrow \text{Marginalisation (here } \sum_Z \dots \text{ iterates over all possible values of } Z\right) \\
 &= \log \sum_Z \left( p(X, Z|\theta) \frac{p(Z)}{p(Z)} \right) && \leftarrow \text{Need } Z \text{ to have non-zero marginal} \\
 &= \log \sum_Z \left( p(Z) \underbrace{\frac{p(X, Z|\theta)}{p(Z)}}_{\text{expectation}} \right) \\
 &= \log \mathbb{E}_Z \left[ \frac{p(X, Z|\theta)}{p(Z)} \right] \\
 &\geq \mathbb{E}_Z \left[ \log \frac{p(X, Z|\theta)}{p(Z)} \right] && \leftarrow \text{Jensen's inequality holds in this direction since } \log(\dots) \text{ is a concave function} \\
 &= \mathbb{E}_Z [\log p(X, Z|\theta)] - \mathbb{E}_Z [\log p(Z)]
 \end{aligned}$$

7

## Maximising the lower bound (1/2)

- $\log p(\mathbf{X}|\boldsymbol{\theta}) \geq \mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})] - \mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{Z})]$
- The right hand side (RHS) is a **lower bound on the original log likelihood**
  - \* This holds for any  $\boldsymbol{\theta}$  and any non zero  $p(\mathbf{Z})$
- Intuitively, we want to push the lower bound up
- This lower bound is a function of **two “variables”  $\boldsymbol{\theta}$  and  $p(\mathbf{Z})$** . We want to maximise the RHS as a function of these two “variables”
- It is hard to optimise with respect to both at the same time, so EM resorts to an iterative procedure

8

## Maximising the lower bound (2/2)

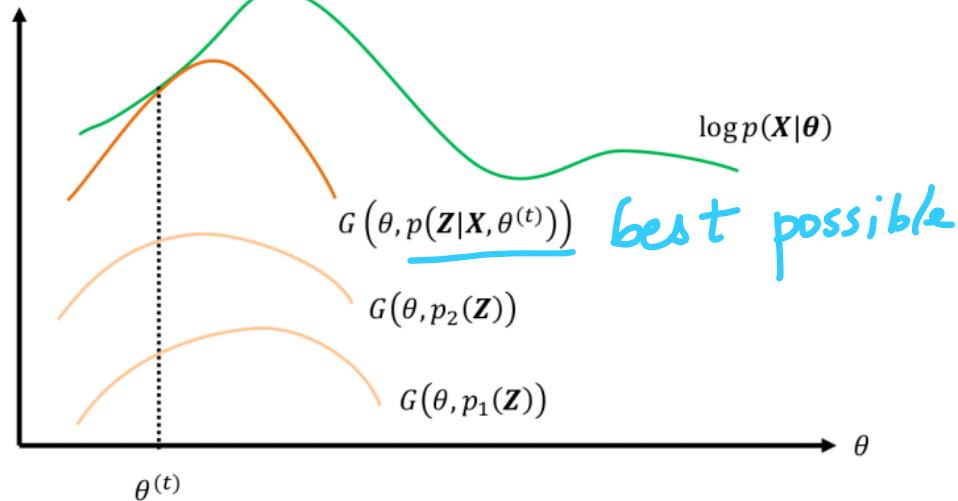
- $\log p(\mathbf{X}|\boldsymbol{\theta}) \geq \mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})] - \mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{Z})]$
- EM is essentially **coordinate ascent**:
  - \* Fix  $\boldsymbol{\theta}$  and optimise the lower bound for  $p(\mathbf{Z})$
  - \* Fix  $p(\mathbf{Z})$  and optimise for  $\boldsymbol{\theta}$
- The convenience of EM comes from the following
  - For any point  $\boldsymbol{\theta}^*$ , it can be shown that **setting  $p(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^*)$  makes the lower bound tight**
  - For any  $p(\mathbf{Z})$ , the second term does not depend on  $\boldsymbol{\theta}$
  - When  $p(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^*)$ , the first term can usually be maximised as a function of  $\boldsymbol{\theta}$  in a closed-form
    - \* If not, then probably don't use EM

we will  
prove this  
shortly

9

## Example (1/3)

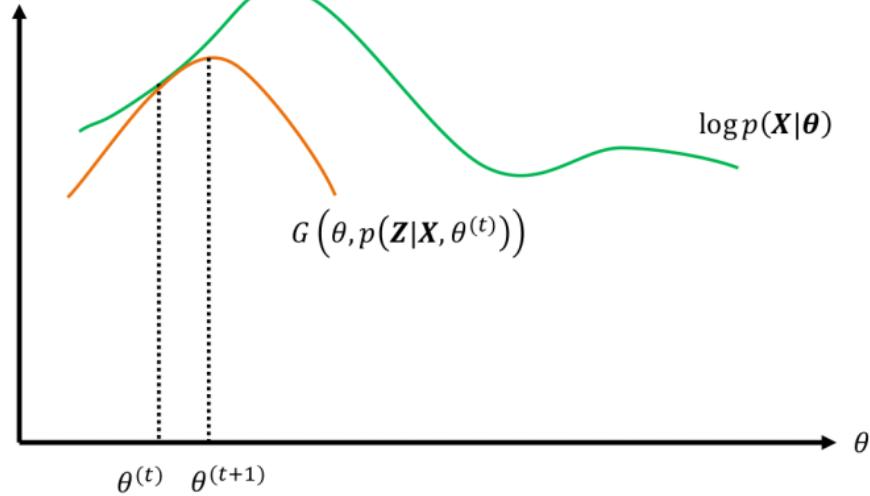
$$\log p(\mathbf{X}|\boldsymbol{\theta}) \geq \underbrace{\mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})] - \mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{Z})]}_{\equiv G(\boldsymbol{\theta}, p(\mathbf{Z}))}$$



10

## Example (2/3)

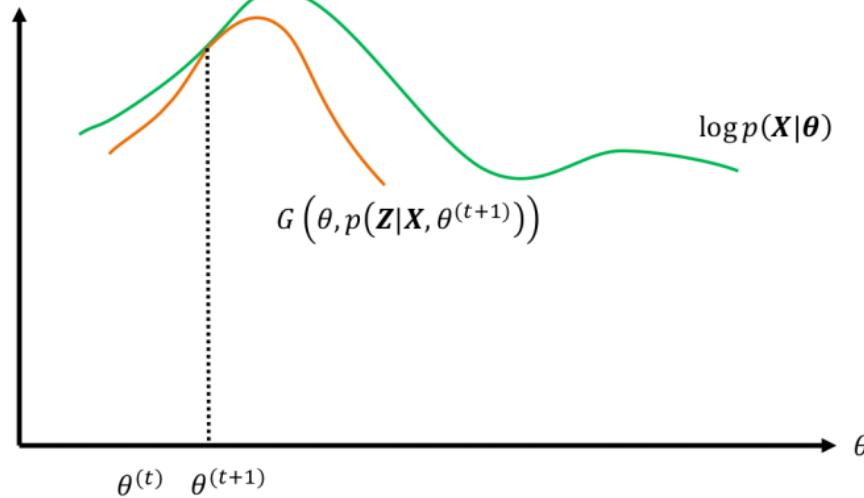
$$\log p(\mathbf{X}|\boldsymbol{\theta}) \geq \underbrace{\mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})] - \mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{Z})]}_{\equiv G(\boldsymbol{\theta}, p(\mathbf{Z}))}$$



11

## Example (3/3)

$$\log p(\mathbf{X}|\boldsymbol{\theta}) \geq \underbrace{\mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})] - \mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{Z})]}_{\equiv G(\boldsymbol{\theta}, p(\mathbf{Z}))}$$

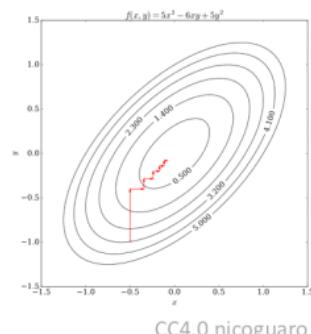


12

## Mini Summary

- EM intuition by GMM with recap
- Lower-bound  $\log p(\mathbf{X}|\boldsymbol{\theta})$  by  $\mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})] - \mathbb{E}_{\mathbf{Z}}[\log p(\mathbf{Z})]$ 
  - \* Holds for any  $\boldsymbol{\theta}, p(\mathbf{Z})$
  - \* Uses Jensen's inequality (concavity of  $\log$ )
- Maximise not  $\log p(\mathbf{X}|\boldsymbol{\theta})$  but lower bound, alternating:
  - \* E-Step: choose  $p(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^*)$  raises lower bound up to log-likelihood, for any  $\boldsymbol{\theta}^*$
  - \* M-Step:  $\boldsymbol{\theta}^*$  by max'ing “completed” log-likelihood; ideally, easy MLE
- The E- and M-steps implement **coordinate ascent**

**Next:** Proving the E-step



CC4.0 nicoguaro

13

## EM as iterative (coordinate) ascent

1. Initialisation: choose (random) initial values of  $\theta^{(1)}$
2. Update:  
  - \* E-step: compute  $Q(\theta, \theta^{(t)}) \equiv \mathbb{E}_{Z|X, \theta^{(t)}}[\log p(X, Z|\theta)]$
  - \* M-step:  $\theta^{(t+1)} = \underset{\theta}{\operatorname{argmax}} Q(\theta, \theta^{(t)})$
3. Termination: if no change then stop
4. Go to Step 2

This algorithm will eventually stop (converge), but the resulting estimate can be only a local maximum

14

## Maximising the lower bound (2/2)

- $\log p(X|\theta) \geq \mathbb{E}_Z[\log p(X, Z|\theta)] - \mathbb{E}_Z[\log p(Z)]$
- EM is essentially coordinate descent:
  - \* Fix  $\theta$  and optimise the lower bound for  $p(Z)$
  - \* Fix  $p(Z)$  and optimise for  $\theta$
- The convenience of EM follows from the following
- For any point  $\theta^*$ , it can be shown that setting  $p(Z) = p(Z|X, \theta^*)$  makes the lower bound tight
- For any  $p(Z)$ , the second term does not depend on  $\theta$
- When  $p(Z) = p(Z|X, \theta^*)$ , the first term can usually be maximised as a function of  $\theta$  in a closed-form
  - \* If not, then probably don't use EM

we will  
prove this  
now

15

## Putting the latent variables in use

We want to maximise  $\log p(\mathbf{X}|\boldsymbol{\theta})$ . We don't know  $\mathbf{Z}$ , but consider an arbitrary non-zero distribution  $p(\mathbf{Z})$

$$\begin{aligned}
 \log p(\mathbf{X}|\boldsymbol{\theta}) &= \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) && \leftarrow \text{Rule of marginal distribution} \\
 &= \log \sum_{\mathbf{Z}} \left( p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) \frac{p(\mathbf{Z})}{p(\mathbf{Z})} \right) && (\text{here } \sum_{\mathbf{Z}} \dots \text{ iterates over all possible values of } \mathbf{Z}) \\
 &= \log \sum_{\mathbf{Z}} \left( p(\mathbf{Z}) \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{p(\mathbf{Z})} \right) \\
 &= \log \mathbb{E}_{\mathbf{Z}} \left[ \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{p(\mathbf{Z})} \right] && \leftarrow \text{Jensen's inequality holds since } \log(\dots) \text{ is a concave function} \\
 &\geq \mathbb{E}_{\mathbf{Z}} \left[ \log \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{p(\mathbf{Z})} \right] \\
 &= \mathbb{E}_{\mathbf{Z}} [\log p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})] - \mathbb{E}_{\mathbf{Z}} [\log p(\mathbf{Z})]
 \end{aligned}$$

16

## Setting a tight lower bound (1/2)

- $$\begin{aligned}
 \log p(\mathbf{X}|\boldsymbol{\theta}) &\geq \mathbb{E}_{\mathbf{Z}} \left[ \log \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{p(\mathbf{Z})} \right] \\
 &= \mathbb{E}_{\mathbf{Z}} \left[ \log \frac{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})p(\mathbf{X}|\boldsymbol{\theta})}{p(\mathbf{Z})} \right] && \leftarrow \text{Chain rule of probability} \\
 &= \mathbb{E}_{\mathbf{Z}} \left[ \log \frac{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})}{p(\mathbf{Z})} + \log p(\mathbf{X}|\boldsymbol{\theta}) \right] \\
 &= \mathbb{E}_{\mathbf{Z}} \left[ \log \frac{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})}{p(\mathbf{Z})} \right] + \mathbb{E}_{\mathbf{Z}} [\log p(\mathbf{X}|\boldsymbol{\theta})] && \leftarrow \text{Linearity of } \mathbb{E}[\cdot] \\
 &= \mathbb{E}_{\mathbf{Z}} \left[ \log \frac{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})}{p(\mathbf{Z})} \right] + \log p(\mathbf{X}|\boldsymbol{\theta}) && \leftarrow \mathbb{E}[\cdot] \text{ of a constant}
 \end{aligned}$$
- $$\log p(\mathbf{X}|\boldsymbol{\theta}) \geq \mathbb{E}_{\mathbf{Z}} \left[ \log \frac{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})}{p(\mathbf{Z})} \right] + \log p(\mathbf{X}|\boldsymbol{\theta})$$

17

## Setting a tight lower bound (2/2)

Ultimate aim:  
maximise this

Lower bound of what  
we want to maximise

$$\log p(X|\theta) \geq \mathbb{E}_Z \left[ \log \frac{p(Z|X, \theta)}{p(Z)} \right] + \boxed{\log p(X|\theta)}$$

First, note that this term\*  $\leq 0$

Second, note that if  $p(Z) = p(Z|X, \theta)$ , then

$$\mathbb{E}_{p(Z|X, \theta)} \left[ \log \frac{p(Z|X, \theta)}{p(Z|X, \theta)} \right] = \mathbb{E}_{p(Z|X, \theta)} [\log 1] = 0 \quad \text{max}$$

For any  $\theta^*$ , setting  $p(Z) = p(Z|X, \theta^*)$  maximises the  
lower bound on  $\log p(X|\theta^*)$  and makes it tight

\*Negative Kullback-Leibler divergence between  $p(Z)$  and  $p(Z|X, \theta)$

## Mini Summary

- We're wanting to maximise the lower bound

$$\log p(X|\theta) \geq \mathbb{E}_Z \left[ \log \frac{p(X, Z|\theta)}{p(Z)} \right]$$

- We've shown RHS is  $\mathbb{E}_Z \left[ \log \frac{p(Z|X, \theta)}{p(Z)} \right] + \log p(X|\theta)$

- And that setting  $p(Z) = p(Z|X, \theta)$ 
  - Makes this RHS as big as possible
  - Makes this RHS equal to  $\log p(X|\theta)$
  - maximises the lower bound as desired!

**Next:** Application of EM to GMM learning

# Estimating Parameters of Gaussian Mixture Model

*Classical application of the Expectation-Maximisation algorithm.  
Completes previous lecture.*

20

## Latent variables of GMM

- Let  $z_1, \dots, z_n$  denote **true origins** of the corresponding points  $\mathbf{x}_1, \dots, \mathbf{x}_n$ . Each  $z_i$  is a discrete variable that takes values in  $1, \dots, k$ , where  $k$  is a number of clusters
- Now compare the original log likelihood

$$\log p(\mathbf{x}_1, \dots, \mathbf{x}_n) = \sum_{i=1}^n \log \left( \sum_{c=1}^k w_c \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \right)$$

marginalise  $\mathbf{z}$

- With **complete log likelihood** (if we knew  $\mathbf{z}$ )

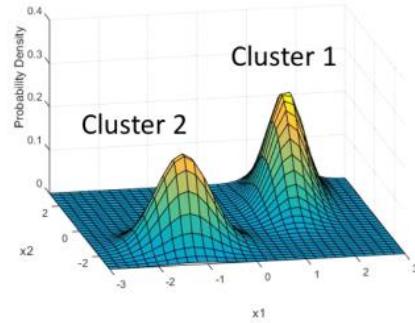
$$\log p(\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{z}) = \sum_{i=1}^n \log (w_{z_i} \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_{z_i}, \boldsymbol{\Sigma}_{z_i}))$$

- Recall that taking a log of a normal density function results in a **tractable** expression

21

## Handling uncertainty about $Z$

- We cannot compute complete log likelihood because we don't know  $Z$
- EM algorithm handles this uncertainty replacing  $\log p(X, z|\theta)$  with expectation  $\mathbb{E}_{Z|X, \theta^{(t)}}[\log p(X, Z|\theta)]$
- This in turn requires the distribution of  $p(Z|X, \theta^{(t)})$  given current parameter estimates
- Assuming that  $Z_i$  are pairwise independent, we need  $P(Z_i = c|x_i, \theta^{(t)})$
- E.g., suppose  $x_i = (-2, -2)$ . What is the probability that this point originated from Cluster 1



22

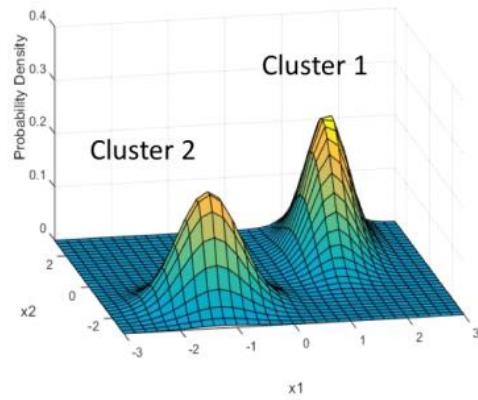
## E-step: Cluster responsibilities

- Setting latent  $Z$  as originating cluster, yields (via Bayes rule)

$$P(z_i = c|x_i, \theta^{(t)}) = \frac{w_c \mathcal{N}(x_i|\mu_c, \Sigma_c)}{\sum_{l=1}^k w_l \mathcal{N}(x_i|\mu_l, \Sigma_l)}$$

- This probability is called **responsibility** that cluster  $c$  takes for data point  $i$

$$r_{ic} \equiv P(z_i = c|x_i, \theta^{(t)})$$



23

## Expectation step for GMM

To simplify notation, we denote  $x_1, \dots, x_n$  as  $X$

$$\begin{aligned}
 Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) &\equiv \mathbb{E}_{z|X, \boldsymbol{\theta}^{(t)}} [\log p(X, z|\boldsymbol{\theta})] \\
 &= \sum_z p(z|X, \boldsymbol{\theta}^{(t)}) \log p(X, z|\boldsymbol{\theta}) \\
 &= \sum_z p(z|X, \boldsymbol{\theta}^{(t)}) \sum_{i=1}^n \log w_{zi} \mathcal{N}(x_i | \boldsymbol{\mu}_{zi}, \boldsymbol{\Sigma}_{zi}) \\
 &= \sum_{i=1}^n \sum_z p(z|X, \boldsymbol{\theta}^{(t)}) \log w_{zi} \mathcal{N}(x_i | \boldsymbol{\mu}_{zi}, \boldsymbol{\Sigma}_{zi}) \\
 &= \sum_{i=1}^n \sum_{c=1}^k r_{ic} \log w_{zi} \mathcal{N}(x_i | \boldsymbol{\mu}_{zi}, \boldsymbol{\Sigma}_{zi}) \\
 &= \sum_{i=1}^n \sum_{c=1}^k r_{ic} \log w_{zi} \\
 &\quad + \sum_{i=1}^n \sum_{c=1}^k r_{ic} \log \mathcal{N}(x_i | \boldsymbol{\mu}_{zi}, \boldsymbol{\Sigma}_{zi})
 \end{aligned}$$

*definition*

$\frac{\partial}{\partial \theta} \approx$

*parameters separately*

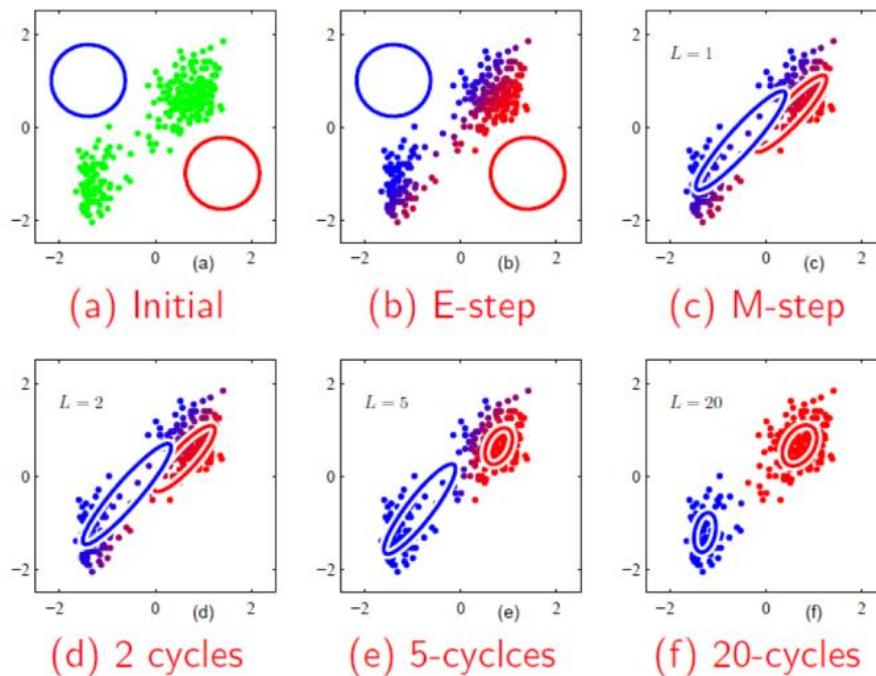
24

## Maximisation step for GMM

- In the maximisation step, take partial derivatives of  $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$  with respect to each of the parameters and set the derivatives to zero to obtain new parameter estimates
- $w_c^{(t+1)} = \frac{1}{n} \sum_{i=1}^n r_{ic}$
- $\boldsymbol{\mu}_c^{(t+1)} = \frac{\sum_{i=1}^n r_{ic} \mathbf{x}_i}{r_c}$   
Here  $r_c \equiv \sum_{i=1}^n r_{ic}$
- $\boldsymbol{\Sigma}_c^{(t+1)} = \frac{\sum_{i=1}^n r_{ic} \mathbf{x}_i \mathbf{x}_i'}{r_c} - \boldsymbol{\mu}_c^{(t)} (\boldsymbol{\mu}_c^{(t)})'$
- Note that these are the estimates for step  $(t + 1)$

25

## Example of fitting Gaussian Mixture model



26

## k-means as a EM for a restricted GMM

- Consider a GMM model in which all components have the same fixed probability  $w_c = 1/k$ , and each Gaussian has the same fixed covariance matrix  $\Sigma_c = \sigma^2 I$ , where  $I$  is the identity matrix
- In such a model, only component centroids  $\mu_c$  need to be estimated
- Next approximate a probabilistic cluster responsibility  $r_{ic} = P(z_i = c | \mathbf{x}_i, \boldsymbol{\mu}_c^{(t)})$  with a deterministic assignment  $r_{ic} = 1$  if centroid  $\boldsymbol{\mu}_c^{(t)}$  is closest to point  $\mathbf{x}_i$ , and  $r_{ic} = 0$  otherwise (E-step)
- Such a formulation results in a M-step where  $\boldsymbol{\mu}_c$  should be set as a centroid of points assigned to cluster  $c$
- In other words, k-means algorithm is a EM algorithm for the restricted GMM model described above!!!

27

# Summary

- EM as MLE algorithm under latent variables
- Maximise not  $\log p(\mathbf{X}|\boldsymbol{\theta})$  but lower bound, alternating:
  - \* E-Step: choose  $p(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^*)$  raises lower bound up to log-likelihood, for any  $\boldsymbol{\theta}^*$
  - \* M-Step:  $\boldsymbol{\theta}^*$  by max'ing “completed” log-likelihood; ideally, easy MLE
- Applying EM to GMM MLE-based training
  - \* “New” interpretation of  $k$ -means as MLE with constraints

## Lecture 24

2020年10月29日 星期四 16:30



24\_bayesian\_rl



COMP90051 Statistical Machine Learning

# Lecture 24: Bayesian Record Linkage

Semester 2, 2020

Guest Lecturer: Neil Marchant

Copyright © 2020 University of Melbourne

## This lecture

- Application of PGMs to perform *record linkage*
- Opportunity to hear about recent research
- We'll cover:
  - Brief background on record linkage
  - A solution based on Bayesian models (D-PGM)
  - How to perform inference
  - Research challenges

# Record linkage

*Identifying records that refer to the same entity*

3

## Motivation: integrating data from different sources

- Scenario: public health researchers want to investigate risk factors associated with COVID-19-related deaths
- Information not available from a single source
  - Risk factors: primary care (GP) records
  - COVID-19 deaths: health department
- Need to merge data, but it's non-trivial without a shared identifier (e.g. Medicare numbers)
- Problem known as record linkage

Name	...	Health conditions
Steven Butler	...	Diabetes, Heart disease
Alanna Thompson	...	
Antonio Ortiz	...	
Evelyn Zhang	...	
Abigail Williams	...	Hypertension

Name	DOD	Postcode
Phoebe Welch	03/03/20	3032
Vanessa Lowry	10/02/20	3130
Stephen Butler	05/07/20	3042
Mary Merritt	13/07/20	
Antonia Ortiz	29/08/20	3150

4

# The record linkage (RL) problem

## Definition

Consider a set of data sources providing a set of records  $\mathcal{R}$ . Let  $P$  be a (coreference) relation on  $\mathcal{R}$  such that:

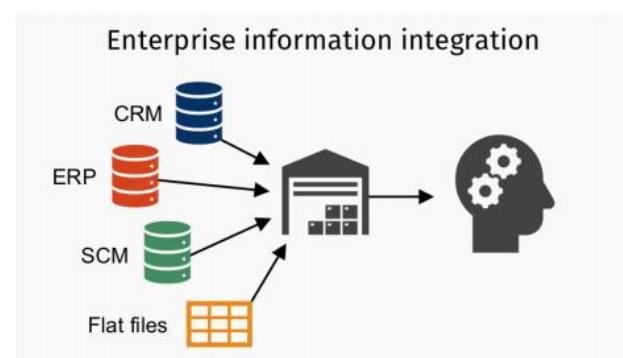
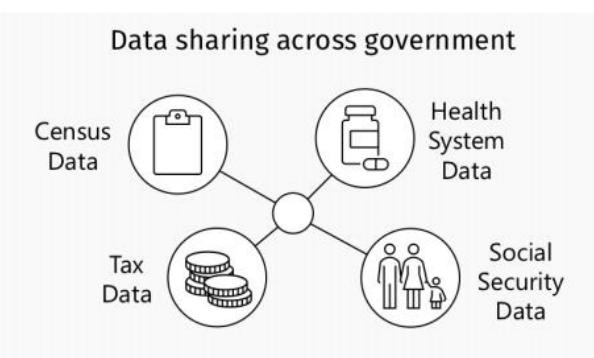
- $(r, r') \in P$  for any pair of records  $r, r' \in \mathcal{R}$  that refer to the same entity,
- $(r, r') \notin P$  for any pair of records  $r, r' \in \mathcal{R}$  that refer to distinct entities.

The record linkage (RL) problem is to approximate the true relation  $P$  by a predicted relation  $\hat{P}$ .

- Also known as entity resolution, data matching, deduplication, merge/purge
- Can formulate as a classification problem on pairs of records, although may get conflicting predictions
- Practical issue: ground truth labels are often unavailable

5

## RL: a ubiquitous problem



And many others: linking product listings across the web to build an e-commerce aggregator, linking accounts across social networks, linking records to produce credit ratings, building knowledge graphs using web sources, ...

6

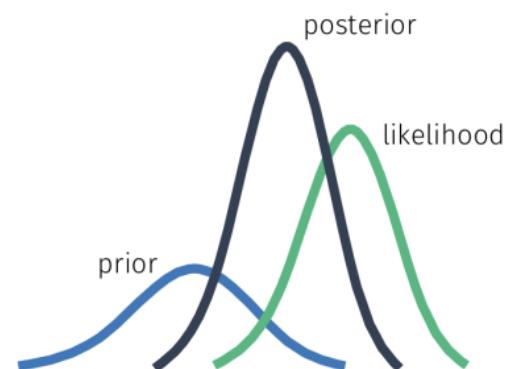
# Bayesian record linkage

An effective class of methods for solving RL under uncertainty

7

## Why Bayesian models?

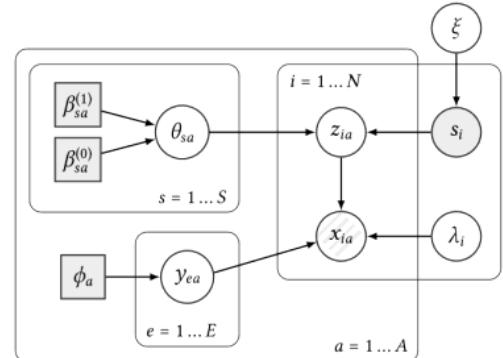
- They tend to be **data-efficient**—important since we often have no ground truth for RL
- Model encodes **constraints** and **prior beliefs** about the generative process
- Apply Bayes' rule to **update beliefs** about **unknown parameters** (i.e. coreference relation), conditional on observed data
- Distributions represent **uncertainty in beliefs**—can propagate to analyses on linked data



8

# blink model for RL

- A Bayesian model for record linkage of structured data from multiple sources
- The model incorporates a population of latent entities with “true” attributes
- Records are generated from the entities by copying their attributes subject to distortion
- More sophisticated distortion model than previous methods—e.g. allows for typos
- Proposed by Steorts (2015)

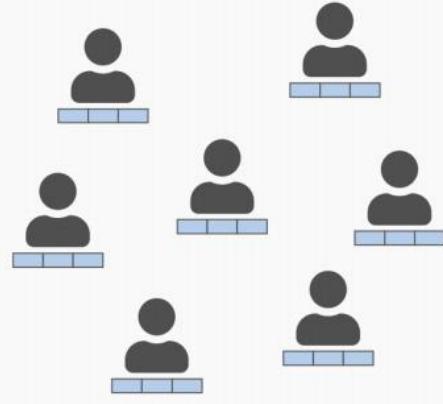


9

# blink model for RL

## Entity model

- Fixed population of entities indexed by  $e \in \{1, \dots, E\}$
- Each entity  $e$  described by a tuple of true attributes  $\mathbf{y}_e = (y_{e1}, \dots, y_{eA})$
- Value of attribute  $a$  for entity  $e$  is generated according to  $y_{ea} \sim \text{Categorical}(\boldsymbol{\phi}_a)$   
where  $\boldsymbol{\phi}_a$  is a distribution over attribute domain  $\mathcal{V}_a$  (set empirically)

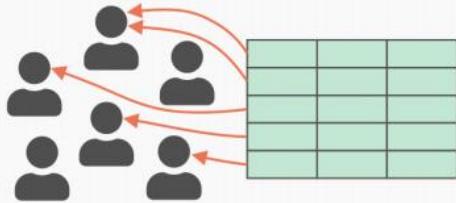


10

# blink model for RL

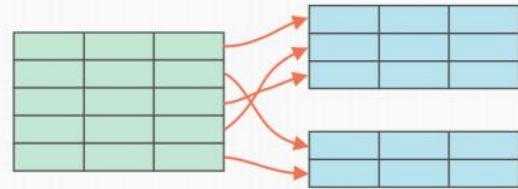
## Linkage model

- Record  $i$  is generated by linking to an entity uniformly at random  
 $\lambda_i \sim \text{DiscreteUniform}(1, \dots, E)$



## Source model

- Record  $i$  is associated with source  
 $s_i \sim \text{Categorical}(\xi)$   
 where  $\xi$  is an unknown distribution over sources  $s \in \{1, \dots, S\}$



11

# blink model for RL

## Distortion model

- A distortion probability is associated with each attribute  $a$  and source  $s$ :  
 $\theta_{sa} \sim \text{Beta}(\alpha_a, \beta_a)$
- The value of attribute  $a$  for record  $i$  follows a hit-miss model:

$$z_{ia} | \theta_{sa} \sim \text{Bernoulli}(\theta_{sa})$$

$$x_{ia} | z_{ia}, y_{\lambda_{ia}} \sim (1 - z_{ia})\delta(y_{\lambda_{ia}}) + z_{ia} \text{Discrete}(\psi_a(y_{\lambda_{ia}}))$$

Binary distortion indicator

$0 \rightarrow$  not distorted  
 $1 \rightarrow$  distorted

perfect probability (0,1)

Kimberley	Brown	12/30/1974
Kim	Brown	12/30/1974

Distortion distribution over domain of attribute

12

## Joint distribution for blink

- Can write down the joint distribution over all variables:

$$p(\Lambda, \mathbf{Y}, \mathbf{X}, \mathbf{Z}, \boldsymbol{\Theta}) = \prod_{e,a} p(y_{ea} | \phi_a) \times \prod_i p(\lambda_i) \times \prod_{s,a} p(\theta_{sa} | \alpha_a, \beta_a)$$

## Joint distribution for blink

- Can write down the joint distribution over all variables:

$$p(\Lambda, \mathbf{Y}, \mathbf{X}, \mathbf{Z}, \Theta) = \prod_{e,a} p(y_{ea} | \phi_a) \times \prod_i p(\lambda_i) \times \prod_{s,a} p(\theta_{sa} | \alpha_a, \beta_a) \\ \prod_{i,a} p(z_{ia} | \theta_{sia}) p(x_{ia} | z_{ia}, \lambda_i, y_{\lambda_i a}, \psi_a)$$

- Conditionals specified on previous slides
- To do record linkage, we infer  $\Lambda = (\lambda_1, \dots, \lambda_N)$  (the linkage structure)  
conditional on  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$  (the records)
- Talk about inference next

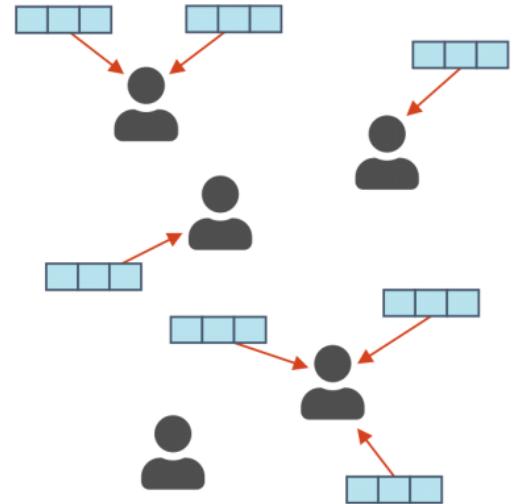
13

## Inference for blink

14

# How to make predictions?

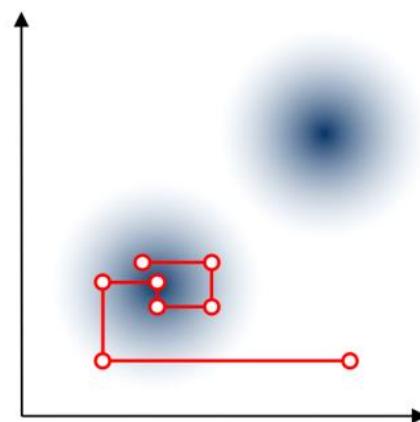
- Want to compute  $p(\Lambda|X) = \frac{p(\Lambda, X)}{p(X)}$  for record linkage
- Although we can write down the joint  $p(\Lambda, Y, X, Z, \Theta)$ , marginalising out latent variables is infeasible
- Must resort to approximate inference
- Standard approach is to approximate  $p(\Lambda|X)$  using samples obtained via Markov chain Monte Carlo (MCMC)
- Gibbs sampling is one of the simplest MCMC methods



15

## Refresher: Gibbs sampling

- Method for obtaining samples from a (high-dimensional) joint distribution—in our case  $p(\Lambda, Y, Z, \Theta | X)$
- Only need to know the joint distribution up to a constant factor
- Sample one variable at a time, holding all others fixed
- Caveat: conditional distributions must be known and easy to sample from → they are for blink



16

## Gibbs sampler for blink

Need to derive conditional distributions for each unobserved variable and ensure we can sample from them. Let's look at an example.

Conditional for  $\lambda_i$

$$\begin{aligned} p(\lambda_i | \Lambda_{-i}, \mathbf{Y}, \mathbf{X}, \mathbf{Z}, \boldsymbol{\Theta}) &\propto p(\lambda_i) \prod_a p(x_{ia} | z_{ia}, \lambda_i, y_{\lambda_i a}, \boldsymbol{\psi}_a) \\ &\propto \prod_a \{(1 - z_{ia}) \mathbb{I}(x_{ia} = y_{\lambda_i a}) + z_{ia} \boldsymbol{\psi}_a(x_{ia} | y_{\lambda_i a})\} \end{aligned}$$

- A discrete distribution over the entities  $1, \dots, E$ , although some entities may have zero weight if the entity attributes are a poor match for the record
- Notice: sampling naively takes  $O(E)$  time—inefficient for large  $E$

17

## Gibbs sampler for blink

- Relatively straightforward to derive conditional distributions for the other variables  $\theta_{sa}, y_{ea}, z_{ia}$  [exercise: try it yourself]
- Gibbs sampler is implemented in an R package released with the blink paper

18

# Research directions

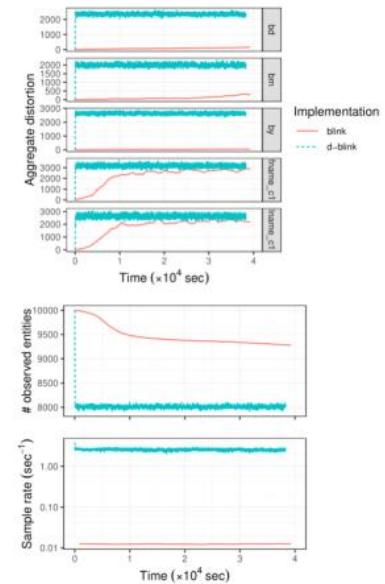
19

## Improving MCMC efficiency

Gibbs sampling: Markov chain converges slowly and exhibits high autocorrelation

Research directions:

- Marginalising out latent variables can help (teal curve on right demonstrates improvement)
- Designing proposals that make more “global” updates under a Metropolis-Hastings framework—e.g. proposing to split/merge entities
- Bear in mind: parameter space is discrete  
→ challenging for gradient-based methods



20

# Scaling to large databases

A single Gibbs update for  $\Lambda$  (linkage structure) takes  $O(N \cdot E)$  time. Since  $E \approx N$ , inference scales roughly **quadratically** in the number of records  $N$ .

Research directions:

- Can speed up Gibbs update for blink using an inverted index
- Parallel/distributed MCMC
- More generally, can exploit the fact that many links are extremely unlikely, so it's wasteful to consider them
  - Blocking
  - Locality sensitive hashing (Indyk & Motwani, 1998)
  - Canopy clustering (McCallum et al., 2000)

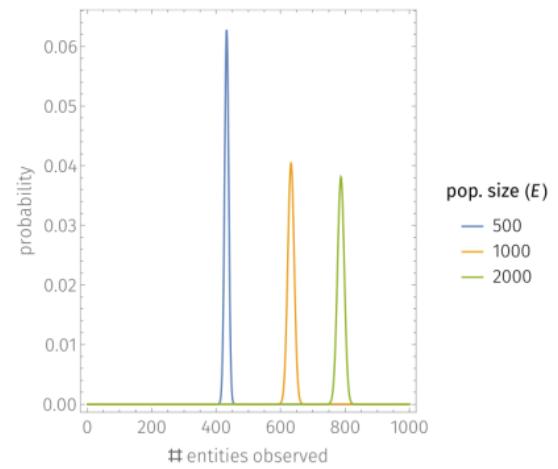
21

# Modelling improvements

Prior on  $\Lambda$  used in blink is too informative—have no control over spread (see right plot). Furthermore, several parameters are assumed known and are set empirically.

Research directions:

- Appropriate priors on  $\Lambda$ —surprisingly challenging to ensure appropriate behaviour asymptotically
- Bayesian nonparametrics—scaling “number” of parameters based on data
- More sophisticated distortion models
- Fewer independence assumptions



22

# Summary

- Introduced record linkage (RL) → an important task for integrating and cleaning data that can be solved using ML methods
- blink Bayesian model for RL
  - Suited to linking/deduplicating structured databases
  - Unsupervised
  - Relatively simple to implement → leverage concepts covered in this subject
- Inference using Gibbs sampling
- Active areas of research