



Introduction to Machine Learning

Introduction to Machine Learning

Objective



Objective

Define machine
learning



Objective

Illustrate key
elements of
machine learning

What is Machine Learning?



| Many different definitions for “machine learning”

- All involve *learning* by a machine (computer)

| Definition of *learning* in a typical dictionary: “the acquisition of knowledge or skills through experience, study, or by being taught”

- Can machines be enabled to learn, without being explicitly programmed?

| Learning and adaption

An Illustrative Example

| Given some example pictures, how a computer can learn to differentiate dogs from cats?

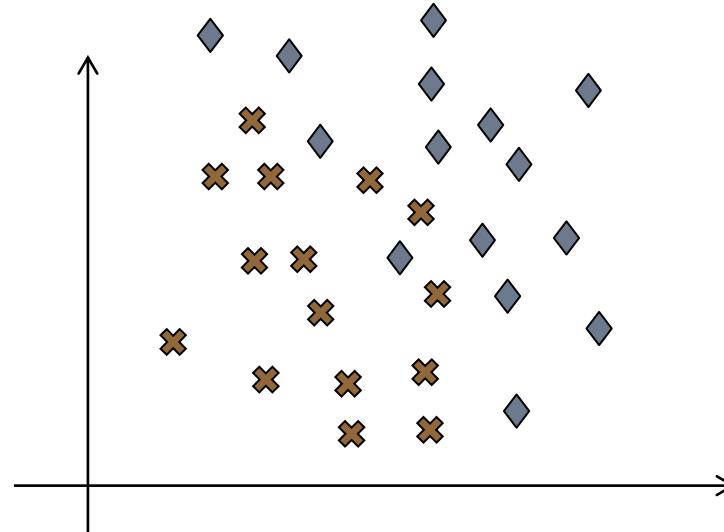


Data Representation – Feature Extraction

| Raw data: Images



| Features



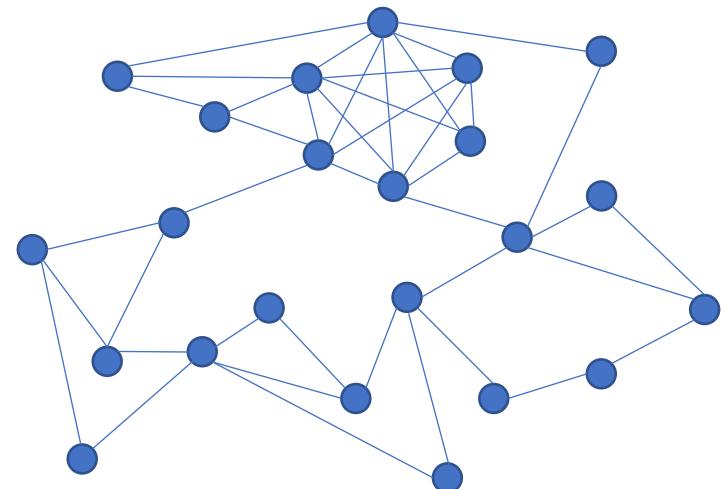
Different Types of Data Representation

| Numerical; Categorical; Ordinal

- Univariate or multivariate
 - All could be represented by numbers.

| Graphical representations in terms of nodes & edges

- E.g., Social network analysis



Preprocessing for Feature Extraction



| Segmentation

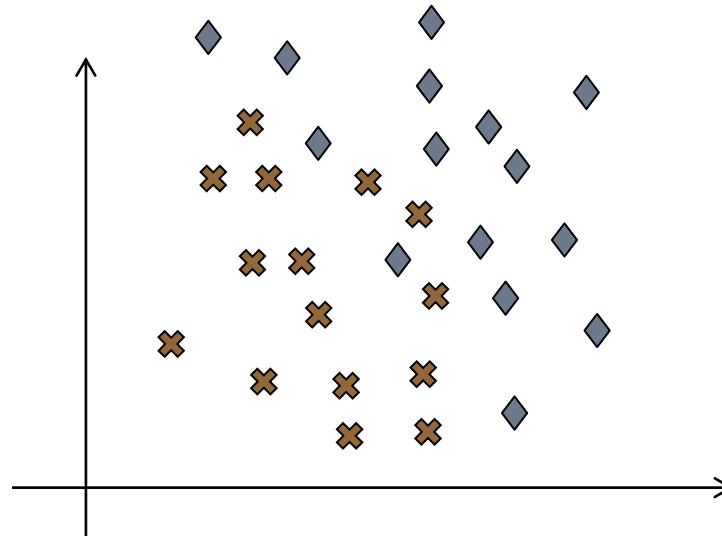
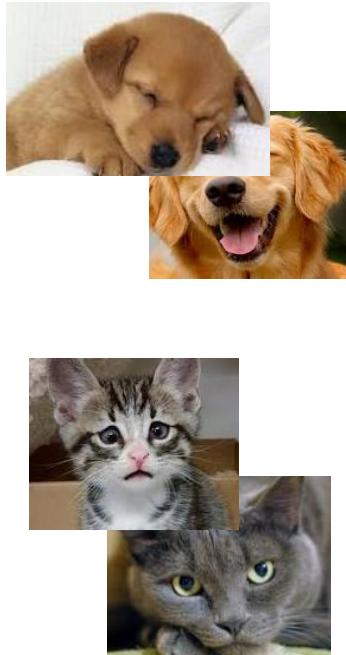
| Filtering

| Various transformations

→ All intended for facilitating feature extraction

| Good features should be *invariant* in some sense.

Mathematical Models for Classification



Importance of Statistical Modeling



- | Why we often rely on statistical methods in machine learning?
- | Data is noisy (measurement noise) → Features are often represented random variables/vectors.
- | Inaccuracy of the assumed model
- | Inherent ambiguity of many real-world problems

Basic Machine Learning Paradigms



| Supervised learning:

- the training samples have labels.

| Unsupervised learning:

- the training set is not labeled.

| Reinforcement learning:

- Learning to take actions to maximize some notion of *reward*.



Objective



Objective

Illustrate specific
machine learning
examples

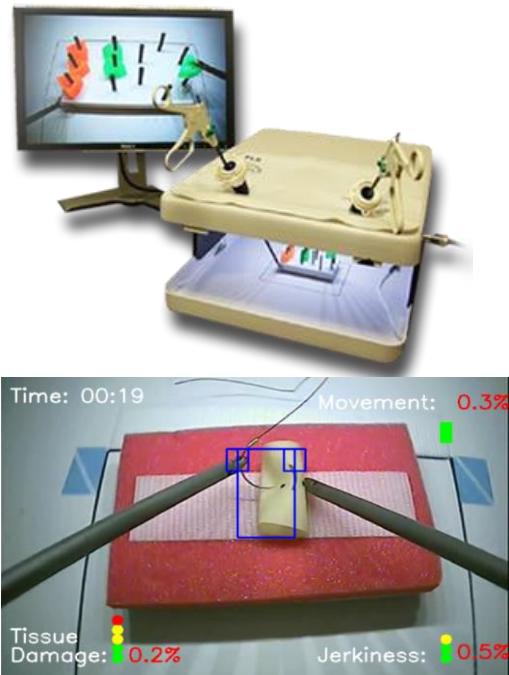


Objective

Describe broad
machine learning
applications

A Few Examples of Machine Learning (1/3)

| Learning to assess skills in simulation-based laparoscopic surgery training



| Learning to predict best answers in community Q & A

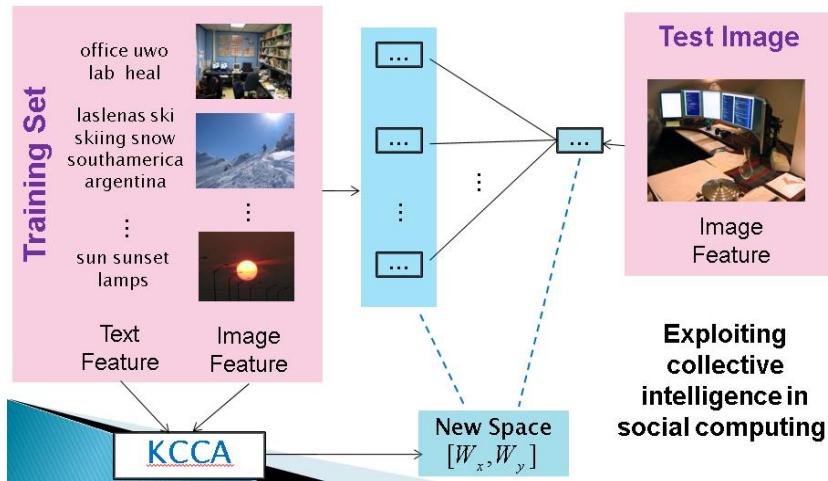


I have a hypothetical question

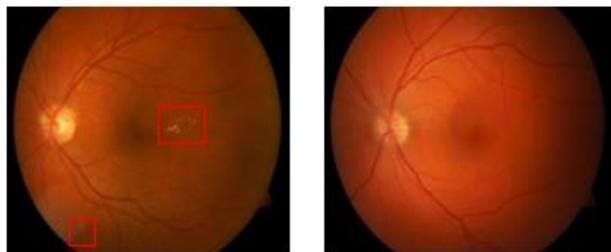
🏆 Best answer: Nope. He can say 'I

A Few Examples of Machine Learning (2/3)

| Tag prediction/recommendation

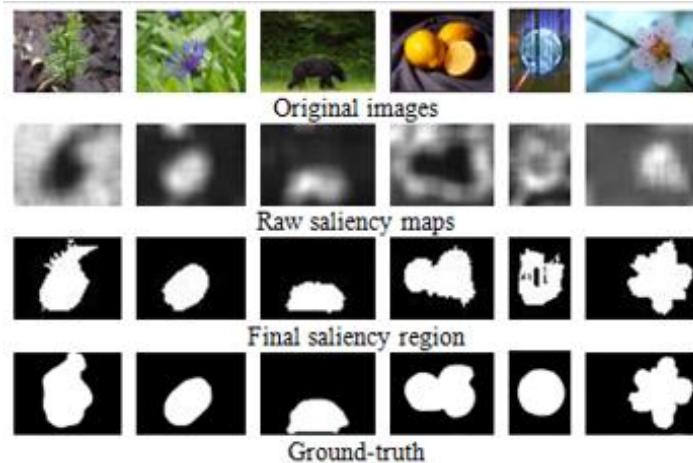


| Diabetic retinopathy detection



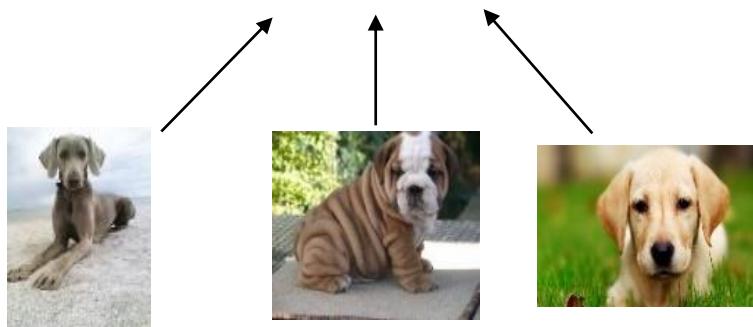
A Few Examples of Machine Learning (3/3)

| Visual saliency detection



| Computational visual aesthetics

Which one is more beautiful?



Broad Applications of Machine Learning



| Computer vision

| Speech recognition; natural language processing

| Medical informatics

| Robotics

| Computational biology

| Information technology

| Finance

Information Technology



| Spam detection

| Web image search

| Recommendation

| Information filtering

| Community detection

| Adaptive advertisement

| Sentiment analysis

Finance



| Credit risk assessment

| Fraud detection

| Stock market prediction

| Algorithmic trading

| Return forecasting





Review of Mathematical Foundations

Calculus, Set Theory, and Linear Algebra



Objective



Objective

Review basic
notations from
Calculus & Set
Theory



Objective

Review key
Linear Algebra
concepts and
operations

Basic Notations from Calculus (1/3)



| Derivative of $f(x)$ with respect to x

| Partial derivative of a function $f(x,y,\dots)$ with respect to x

– Note: the function may be scalar-valued or vector-valued

Basic Notations from Calculus (2/3)

| \mathbb{R}^d : d -dimensional Euclidean space.

| Gradient operator in \mathbb{R}^d : ∇

Basic Notations from Calculus (3/3)



| The integral of $f(x)$ between a and b

| The argmin or argmax notation

Basic Notations from Set Theory (1/2)



| A set S is a collection of objects.

- \emptyset : the empty set (a special set that contains no object)

| Some basic relations and operations

- $x \in A$: An object x is a member of a set A .
- $A \subseteq B$: Set A is a *subset* of B $\Leftrightarrow x \in A \Rightarrow x \in B$
- $B \subset C$: Set B is a *proper subset* of C .

Basic Notations from Set Theory (2/2)



| Some basic relations and operations

- $A \cup B$: The union of A and B .
- $A \cap B$: The intersection of A and B . (AB in shorthand)
- A^c or \bar{A} : The complement of A
- A and B are disjoint if $A \cap B = \emptyset$



Linear Algebra: Basic Notations (1/4)



| A d -dimensional column vector x and its transpose x^t

| n by d matrix M and its d by n transpose M^t

Linear Algebra: Basic Notations (2/4)



- | A square matrix M is symmetric if
- | Multiplying a vector by a matrix: $Mx = y$
- | Multiplying two matrices M_1 and M_2

Linear Algebra: Basic Notations (3/4)



- | The identity matrix I of d by d
- | Inner product of two vectors $x^t y$
- | Outer product of two vectors $x y^t$

Linear Algebra: Basic Notations (4/4)



- | The length or Euclidean norm of a vector x , denoted $\|x\|$
- | Normalized vector, $\|x\| = 1$

Matrix: Additional Definitions (1/2)



| Determinant of a matrix M: denoted $|M|$ or $\det(M)$

- Look at size 2×2
- What about size 3×3 and above?

| Trace of a matrix

Matrix: Additional Definitions (2/2)



| Matrix inversion M^{-1}

| Eigenvectors and eigenvalues of M

Derivatives Involving Matrices (1/3)

- | If the entries of a matrix M depend on a scalar parameter θ , we have $\frac{\partial M}{\partial \theta} =$
- | Derivative of a scalar-valued function $f(x)$ of d variables x_i , $i=1,\dots,d$, and $x=(x_1, \dots, x_d)^t$, or the gradient w.r.t. x is $\nabla f(x) = \frac{\partial f(x)}{\partial x} =$

Derivatives Involving Matrices (2/3)

| If $f(x)$ is an n -dimensional vector-valued function of d variables x_i , $i=1, \dots, d$, and $x=(x_1, \dots, x_d)^t$, we have the derivative as* $\frac{\partial f(x)}{\partial x} =$

* We could use the Jacobian form too; See “numerator layout” vs “denominator layout” in matrix calculus.

Derivatives Involving Matrices (3/3)

| Some useful results:

$$\frac{\partial}{\partial \mathbf{x}} [\mathbf{Mx}] =$$

$$\frac{\partial}{\partial \mathbf{x}} [\mathbf{y}^t \mathbf{x}] =$$

$$\frac{\partial}{\partial \mathbf{x}} [\mathbf{x}^t \mathbf{Mx}] =$$





Review of Mathematical Foundations

Basics in Probability Theory

Objective



Objective

Define
Probability
Space



Objective

Discuss
Conditional
Probability and
Bayes Rule

Probability Space (1/2)



A probability space is a triplet (Ω, \mathcal{B}, P) that is used to model a process or an experiment with random outcomes.

- The **sample space** Ω is the set of all possible outcomes of an experiment
 - Consider two different experiments
 - (1) Tossing a coin; (2) Tossing a die

Probability Space (2/2)



- | \mathcal{B} : a sigma algebra (or Borel field), or informally, a collection of subsets of Ω , subject to some constraints (like containing the empty set, being closed under complements and countable union)
- | P : a measure called probability defined on \mathcal{B} , that satisfies
 - $P(A) \geq 0$ for all $A \in \mathcal{B}$
 - $P(\Omega) = 1$
 - If $A_1, A_2, \dots \in \mathcal{B}$ are pairwise disjoint then $P(\bigcup A_i) = \sum P(A_i)$
(i.e., $A_j \cap A_k = \emptyset, \forall j \neq k$)

Conditional Probability

| Let (Ω, \mathcal{B}, P) be a probability space, and let $H \in \mathcal{B}$ with $P(H) > 0$. For any $B \in \mathcal{B}$, we define $P(B|H) = P(BH) / P(H)$ and call $P(B|H)$ the conditional probability of B , given H .

The Total Probability Rule

| Let (Ω, \mathcal{B}, P) be a probability space, and let $\{H_j\}$ be pairwise disjoint events in \mathcal{B} (i.e., $H_j H_k = \emptyset$, $\forall j \neq k$) and $\bigcup_{j=1, \dots, \infty} H_j = \Omega$. Suppose $P(H_j) > 0$, $\forall j$, then $P(B) = \sum_{j=1, \dots, \infty} P(H_j)P(B|H_j)$

- Such $\{H_j\}$ is called a partition of Ω .

The Bayes Rule

| Let (Ω, \mathcal{B}, P) be a probability space, and let $\{H_j\}$ be pairwise disjoint events in \mathcal{B} with $\bigcup_{j=1, \dots, \infty} H_j = \Omega$, and $P(H_j) > 0, \forall j$. We have, $\forall B \in \mathcal{B}$ and $P(B) > 0$,

$$P(H_j|B) = \frac{P(H_j) P(B|H_j)}{\sum_{i=1, \dots, \infty} P(H_i) P(B|H_i)}, \quad \forall j$$

Independence of Events



| Let (Ω, \mathcal{B}, P) be a probability space, $\forall A, B \in \mathcal{B}$, we say A and B are independent if $P(AB) = P(A)P(B)$.



Review of Mathematical Foundations

Random Variables and Common Distributions

Objective



Objective

Review random
variables & their
distributions

Discrete Random Variables



| Let x be a discrete random variable that can take any of the m different values in the set $V=\{v_1, v_2, \dots, v_m\}$ with respective probabilities $\{p_1, p_2, \dots, p_m\}$, i.e., $p_i = Prob[x=v_i]$.

$$- p_i \geq 0, \quad \sum_{j=1, \dots, m} p_j = 1$$

| Probability Mass Function $P(x)$ is used to represent the set of probabilities $\{p_1, p_2, \dots, p_m\}$

$$- P(x) \geq 0, \quad \sum_{x \text{ in } V} P(x) = 1$$

Expected Value (Means) & Variance



| The expected value (mean) of x , $E[x]$, often denoted μ

$$\mu = E[x] = \sum_{x \text{ in } V} xP(x)$$

| The expected value of a function $f(x)$, $E[f(x)]$,

$$E[f(x)] = \sum_{x \text{ in } V} f(x)P(x)$$

| $E[]$ is linear when viewed as an operator.

$$E[\alpha f(x) + \beta g(x)] =$$

| The variance of x , $Var[x]$, often denoted σ^2

$$\sigma^2 = Var(x) = E[(x-\mu)^2] = \sum_{x \text{ in } V} (x-\mu)^2 P(x)$$

Joint Distributions



| Consider a pair of discrete random variables, x and y , taking values in $V=\{v_1, v_2, \dots, v_m\}$ and $W=\{w_1, w_2, \dots, w_n\}$ respectively.

- (x, y) to take a pair of values (v_i, w_j) with probability p_{ij}
- Or, we consider the **joint probability mass function** $P(x, y)$

Marginal Distributions



| Knowing $P(x, y)$, can we figure out $P_x(x)$ or $P_y(y)$?

→ The concept of **marginal distribution** for x and y respectively.

Statistical Independence



| Random variables x and y are said to be statistically independent if and only if $P(x, y) = P_x(x) P_y(y)$

Covariance



| **Cov(x, y), often denoted σ_{xy}**

| **Covariance matrix Σ , $\Sigma = E[(x - \mu)(x - \mu)^t]$**

Conditional Density



| $P(x|y) =$

| Similarly, we may write the Bayes Rule in terms of densities.

How about continuous random variables?



- | Instead of $P(x)$, we have the probability density function (PDF) $p(x)$
- | Some properties of $p(x)$:
- | The cumulative distribution function (CDF) $F(x)$:

Continuous Random Variables



| Mean, variance, etc., are similarly defined, via integrals.

| Joint PDF $p(x,y)$ of two variables

- Marginal PDFs for x and y
- If $x \sim p_x(x)$ and $y \sim p_y(y)$ are independent $p(x,y) =$

Continuous Random Variables



| Conditional PDF $p(x|y)$

| Bayes rule for PDF:



Review of Mathematical Foundations

Common Densities

Objective



Objective

Discuss common
densities useful for
machine learning
application

Common Distributions



| Uniform Distribution

| Normal (Gaussian) Distribution

The Uniform Distribution, $U(a, b)$



| 1-D example, with PDF

$$p(x) = \begin{cases} \frac{1}{b-a} & \text{for } a \leq x \leq b \\ 0 & \text{o.w.} \end{cases}$$

The Uniform Distribution, $U(a, b)$



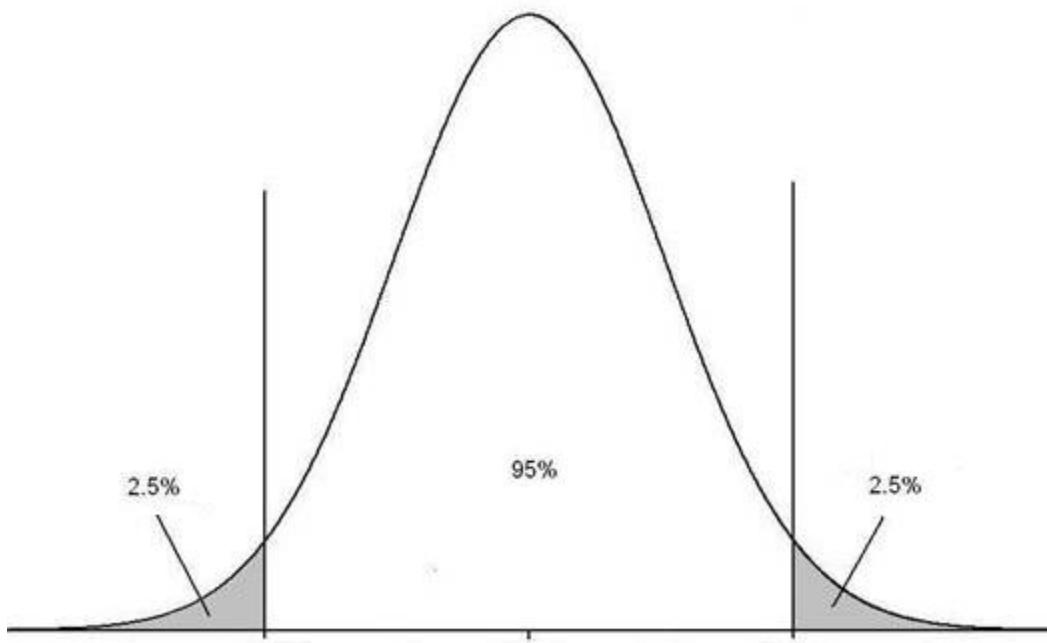
| What is the CDF of $p(x)$?

$$p(x) = \begin{cases} \frac{1}{b-a} & \text{for } a \leq x \leq b \\ 0 & \text{o.w.} \end{cases}$$

The Normal Distribution, $N(\mu, \sigma^2)$

| 1-D example, with PDF

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

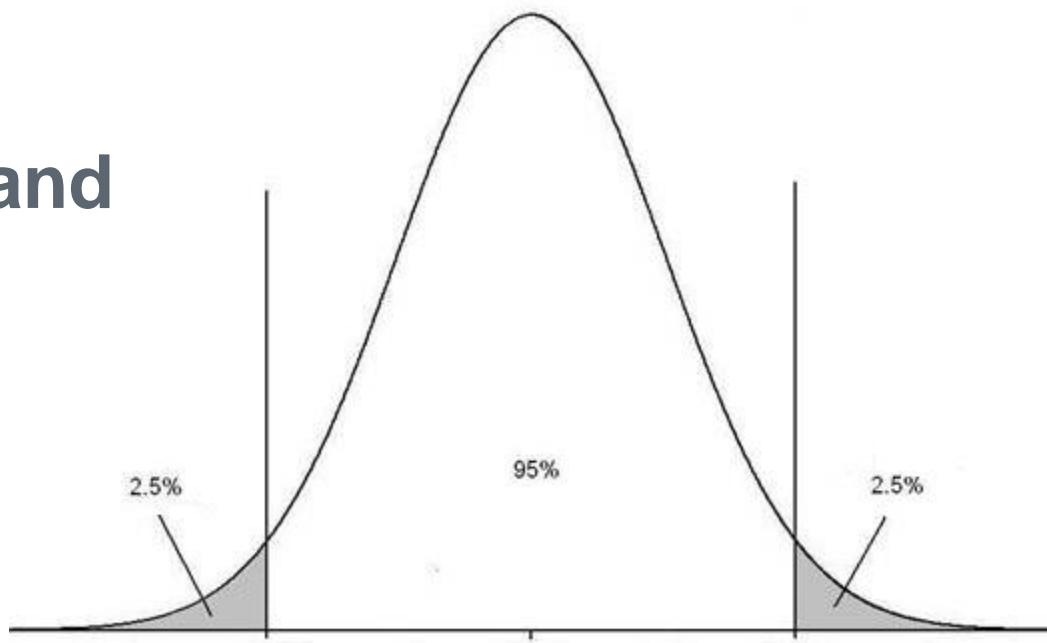


The Normal Distribution, $N(\mu, \sigma^2)$

| 1-D example, with PDF

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

| What is the mean and variance ?



Standardized Normal Distribution



| 1-D example, with PDF

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

| What is the CDF?

| The error function

$$\text{erf}(u) = \frac{2}{\sqrt{\pi}} \int_0^u e^{-x^2} dx$$

CDF for General Normal Distribution



| What is the CDF for $N(\mu, \sigma^2)$?

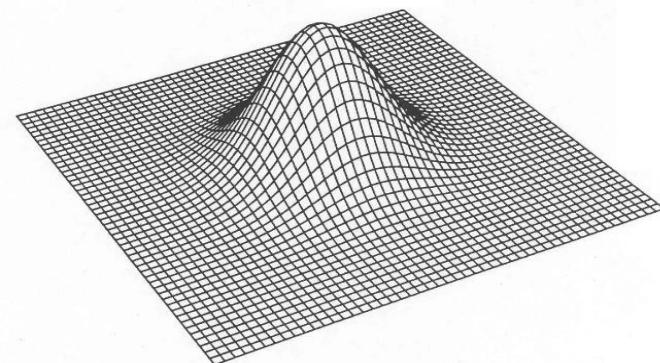
$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Multivariate Normal Distribution

| d -dimensional vector \mathbf{x} is said to be of multivariate normal distribution if its PDF is of the form

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^t \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right]$$

| Visualization of a 2-d example

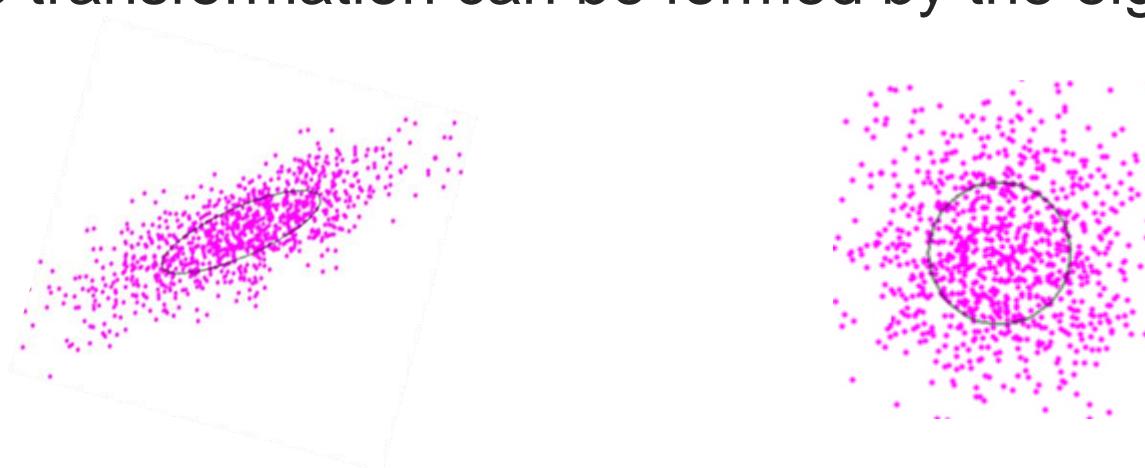


Whitening Transformation

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^t \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right]$$

| Given some data \mathbf{x} distributed according to the above density, we may apply some transformation to \mathbf{x} , so that the covariance matrix of the transformed data is diagonal.

- The transformation can be formed by the eigenvectors of Σ





Supervised Learning

Linear Regression

Objective



Objective

Define the set-up of Supervised Learning



Objective

Discuss basic regression models

Supervised Learning



- | The set-up: the given training data consist of **<sample, label>** pairs, or (x, y) ; the objective of learning is to figure out a way to predict label y for any new sample x .
- | Consider two types of problems:
 - **Regression:** y continuous
 - **Classification:** y is discrete, e.g., class labels.

The Task of Regression



- | Given: A training set of n samples $\langle \mathbf{x}^{(i)}, y^{(i)} \rangle$ where $y^{(i)}$ is a continuous “label” (or target value) for $\mathbf{x}^{(i)}$
- | To learn a model for predicting y for any new sample \mathbf{x} .
- | A simple model is linear regression: modeling the relation between y and \mathbf{x} via a linear function.

$$y \approx w_0 + w_1 x_1 + \dots + w_d x_d = \mathbf{w}^t \mathbf{x}$$

(Note: \mathbf{x} is *augmented* by adding a dimension of constant 1 to the original sample.)

Linear Regression

| We can introduce an error term to capture the residual $y = \mathbf{w}^t \mathbf{x} + e$

| Applying this to all n samples, we have: $y = \mathbf{X} \mathbf{w} + e$

$$\begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{pmatrix} \quad \left[\begin{array}{cccc} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \cdots & x_d^{(n)} \end{array} \right] \quad \begin{pmatrix} e^{(1)} \\ e^{(2)} \\ \vdots \\ e^{(n)} \end{pmatrix}$$

| Learning in this case is to figure out a good w .

Linear Regression (cont'd)

| Find an optimal w by minimizing the squared error

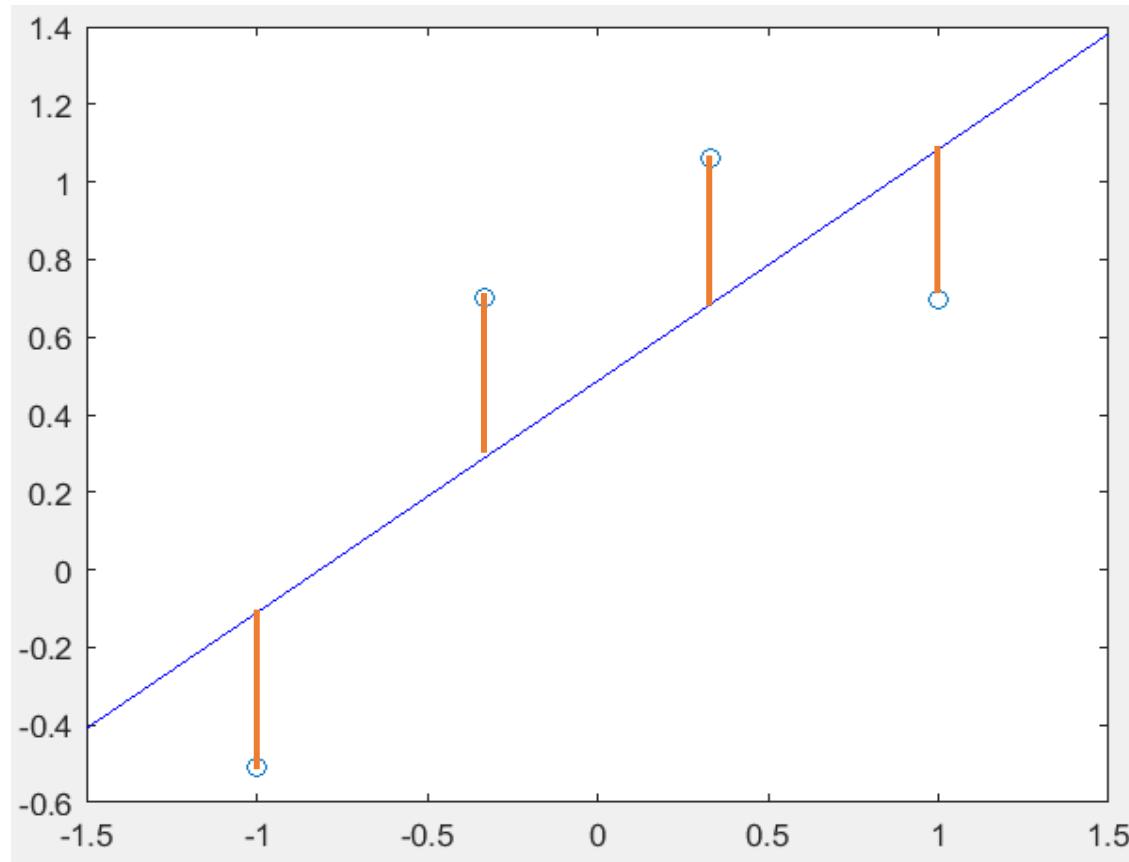
$$\|e\|^2 = \|y - X w\|^2$$

| The solution can be found to be:

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

| In practice, some iterative approaches may be used (e.g., gradient descent search).

A simple example



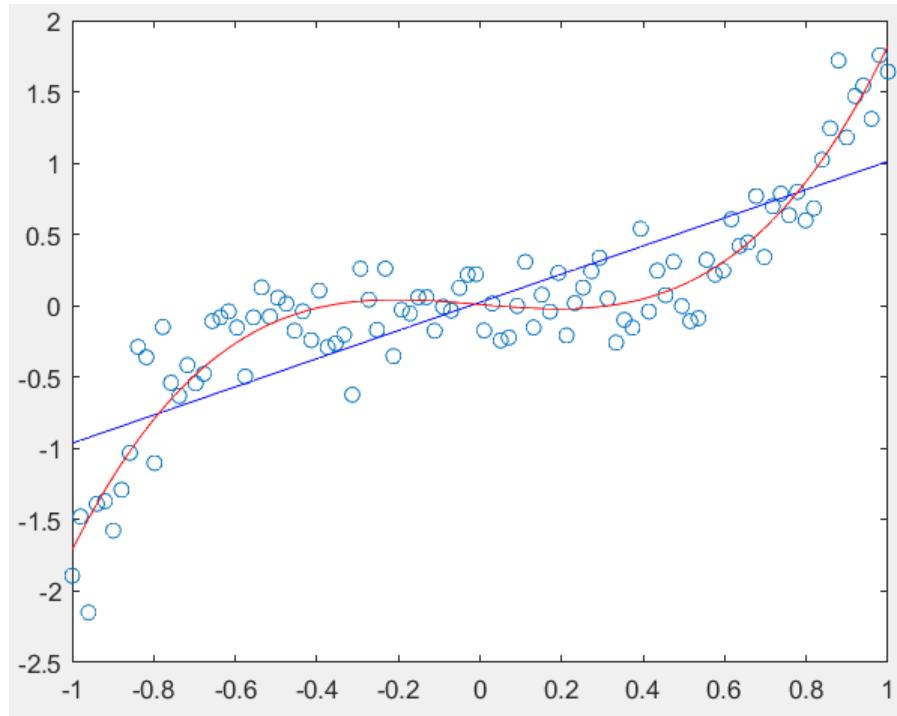
Generalizing Linear Regression

| Introducing some basis functions $\phi_j(x)$:

$$y = w_0 + w_1\phi_1(x) + \dots + w_{M-1}\phi_{M-1}(x)$$

| Compare:

- Blue: Linear Regression
- Red: With $\phi_j(x) = x^j$



Regularized Least Squares



| E.g., use a new error function: $E_D(w) + \lambda E_W(w)$

- λ is the regularization coefficient
- $E_D(w)$ is the data-dependent error
- $E_W(w)$ is the *regularization term*, e.g., $E_W(w) = \|w\|^q$

| Help to alleviate overfitting.





Supervised Learning

Density Estimation in Supervised Learning

Objective



Objective

Illustrate
classification in
Supervised
Learning



Objective

Discuss basic
density estimation
techniques

Supervised Learning



- | The set-up: the given training data consist of **<sample, label>** pairs, or (x, y) ; the objective of learning is to figure out a way to predict label y for any new sample x .
- | Consider two types of problems:
 - **Regression:** y continuous
 - **Classification:** y is discrete, e.g., class labels.

Examples of Image Classification

| The MNIST training images of hand-written digits



| The Extended Yale B Face Images



How do we model the training images?



| **Parametric:** each class of images (the feature vectors) may be modeled by a density function $p_{\theta}(x)$ with parameter θ .

- To emphasize the density is for images from class/label y , we may write $p_{\theta}(x|y)$.
 - We may also use the notation $p(x|\theta)$, if the discussion is true for any y .
- How to estimate θ from the training images?

| **Note:** We may also consider non-parametric approaches.

MLE for Density Estimation (1/3)

| Given some training data;
Assuming a parametric model
 $p(x|\theta)$; What specific θ will
fit/explain the data best?

– E.g., Consider a simple 1-D normal
density with only a parameter μ
(assuming the variance is known)



| Given a sample x_i , $p(x_i | \mu)$
gives an indication of how
likely x_i is from $p(x_i | \mu)$

→ the concept of the likelihood
function.

MLE for Density Estimation (2/3)

| The likelihood function: the density function $p(x|\theta)$ evaluated at the given data sample x_i , and viewed as a function of the parameter θ .

- Assessing how likely the parameter θ (defining the corresponding $p(x|\theta)$) gives arise to the sample x_i .
- We often use $L(\theta)$ to denote the likelihood function, and $l(\theta) = \log(L(\theta))$ is called the log-likelihood.

| Maximum Likelihood Estimation (MLE): Finding the parameter that maximizes the likelihood function

$$\hat{\theta} = \operatorname{argmax}_{\theta} p(x|\theta)$$

MLE for Density Estimation (3/3)



- | How to consider *all* the given samples $D=\{x_i, i=1,\dots,n\}$?
- | The concept of i.i.d. samples: the samples are assumed to be *independent* and *identically distributed*
- | So, the data likelihood is given by

$$L(\theta) = P(D|\theta) =$$

MLE Example 1



| Tossing a coin for n times, observing n_1 times for head.

- Estimate the probability θ for head

| The likelihood function is:

$$L(\theta) = P(D|\theta) = \theta^{n_1}(1 - \theta)^{n-n_1}$$

MLE Example 1 (cont'd)

| We want to find what θ maximizes this likelihood, or equivalently, the log-likelihood

$$l(\theta) = \log P(D|\theta) = \log(\theta^{n_1}(1-\theta)^{n-n_1}) \\ = \dots$$

| Take the derivative and set to 0:

$$\frac{d}{d\theta} l(\theta) = 0$$

| This will give us:

$$\hat{\theta} = \frac{n_1}{n}$$

MLE Example 2

| Given n i.i.d. samples $\{x_i\}$ from the 1-D normal distribution $N(\mu, \sigma^2)$, find the MLE for μ and σ^2

| The likelihood function is:

$$L(\mu, \sigma) = p(D|\mu, \sigma) = \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \prod_{i=1}^n e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

| The log-likelihood is:

$$\begin{aligned} l(\mu, \sigma) &= \log P(D|\mu, \sigma) \\ &= \log \left(\left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \prod_{i=1}^n e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \right) \\ &= -n \log(\sigma\sqrt{2\pi}) - \sum_{i=1}^n \frac{(x_i-\mu)^2}{2\sigma^2} \end{aligned}$$

MLE Example 2 (cont'd)

| The MLE solution for μ

$$\begin{aligned}\hat{\mu} &= \operatorname{argmax}_{\mu} l(\mu, \sigma) \\ &= \operatorname{argmax}_{\mu} \left\{ -n \log(\sigma \sqrt{2\pi}) - \sum_{i=1}^n \frac{(x_i - \mu)^2}{2\sigma^2} \right\}\end{aligned}$$

| Set the derivative to 0:

$$\frac{\partial}{\partial \mu} l(\mu, \sigma) = 0$$

| The solution is:

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

MLE Example 2 (cont'd)

| The MLE solution for μ

$$\begin{aligned}\hat{\sigma} &= \operatorname{argmax}_{\sigma} l(\mu, \sigma) \\ &= \operatorname{argmax}_{\sigma} \left\{ -n \log(\sigma \sqrt{2\pi}) - \sum_{i=1}^n \frac{(x_i - \mu)^2}{2\sigma^2} \right\}\end{aligned}$$

| Set the derivative to 0:

$$\frac{\partial}{\partial \sigma} l(\mu, \sigma) = 0$$

| The solution is:

$$\widehat{\sigma^2} = \frac{\sum_{i=1}^n (x_i - \mu)^2}{n}$$





Supervised Learning

Generative vs Discriminative Models in Supervised Learning

Objective



Objective
Differentiate
between generative
and discriminative
models of
supervised learning



Objective
Discuss challenges
in Bayesian
learning

Supervised Learning



| The set-up: the given training data consist of **<sample, label>** pairs, or (x, y) ; the objective of learning is to figure out a way to predict label y for any new sample x .

– E.g., Given n pairs $\langle \mathbf{x}^{(i)}, y^{(i)} \rangle$, $i=1, \dots, n$; $\mathbf{x}^{(i)}$: i -th sample represented as d -dimensional vectors; $y^{(i)}$: corresponding labels.

| Equivalently, to find $P(y|x)$

Two Types of Models



| Generative Model

- $P(y|x) \propto P(y) p(x|y)$ '
- → To learn $P(y)$ and $p(x|y)$.

| Discriminative Model

- Directly learn $P(y|x)$
- No assumption made on $p(x|y)$

Two Types of Models



| Generative Model

- $P(y|x) \propto P(y) p(x|y)$ '
- → To learn $P(y)$ and $p(x|y)$.
- → Bayesian learning,
Bayes classifiers.
- Example: Naïve Bayes
Classifier

| Discriminative Model

- Directly learn $P(y|x)$
- No assumption made on $p(x|y)$
- Example: Logistic Regression

Practical Difficulty of Bayesian Learning



| Consider doing Bayesian learning without making simplifying assumptions.

- Given n training pairs $\langle \mathbf{x}^{(i)}, y^{(i)} \rangle$, $i=1, \dots, n$. Each $\mathbf{x}^{(i)}$ is d -dimensional.
- We need to learn $P(y)$ and $p(\mathbf{x}|y)$

→ $p(\mathbf{x}|y)$ can be very difficult to estimate:

- Consider a very simple case: binary features, and y is also binary. How many probabilities do we need to estimate?



Supervised Learning

Naïve Bayes Classifier

Objective



Objective
Implement the
fundamental
learning algorithm
Naive Bayes

Naïve Bayesian Classifier



| The "naive" *conditional independence* assumption: each feature is (conditionally) independent of every other feature, given the label, i.e., $p(x_i | \{x_j \text{ for any } j \neq i\}, y) = p(x_i | y)$

| How does this assumption simplify the problem?

- Consider the previous example again: d-dimensional binary features, and y is also binary.
- How many probabilities do we need to estimate now?

$$p(\mathbf{x} | y) = p(x_1, x_2, \dots, x_d | y) = \dots$$

Naïve Bayesian Classifier (cont'd)

| The naïve Bayes classifier: the predicted label is given by

$$\hat{y} = \operatorname{argmax}_y P(y) \prod_{i=1}^d p(x_i|y)$$

| “Parameters” of the classifier:

- $P(y)$
- $p(x_i|y)$ for all i, y

Naïve Bayesian Classifier (cont'd)

E.g., estimating the “parameters” of the classifier

- $P(y)$ & $p(x_i | y)$ for all i, y -

for the following familiar example



Discrete Feature Example



| $x = \langle x_1, x_2, \dots, x_d \rangle$ where each x_i can take only a finite number of values from $\{v_1, v_2, \dots, v_m\}$:

| In this case, the “parameters” of the classifier are

- $P(y)$
- $P(x_i=v_k|y)$, for all i, k , and y

| Given: A training set of n labelled samples $\langle x^{(i)}, y^{(i)} \rangle$, $i=1, \dots, n$

→ How to estimate the model parameters?

Discrete Feature Example (cont'd)

| Given: A training set of n labelled samples $\langle \mathbf{x}^{(i)}, \mathbf{y}^{(i)} \rangle, i=1, \dots, n$

→ How to estimate the model parameters?

$$P(\mathbf{y}) =$$

$$P(x_i=v_k | \mathbf{y}) =$$

| These are in fact the MLE solutions for the corresponding parameters.



Supervised Learning

Logistic Regression

Objective



Objective

Implement the fundamental
learning algorithm Logistic
Regression

Discriminative Model: Example

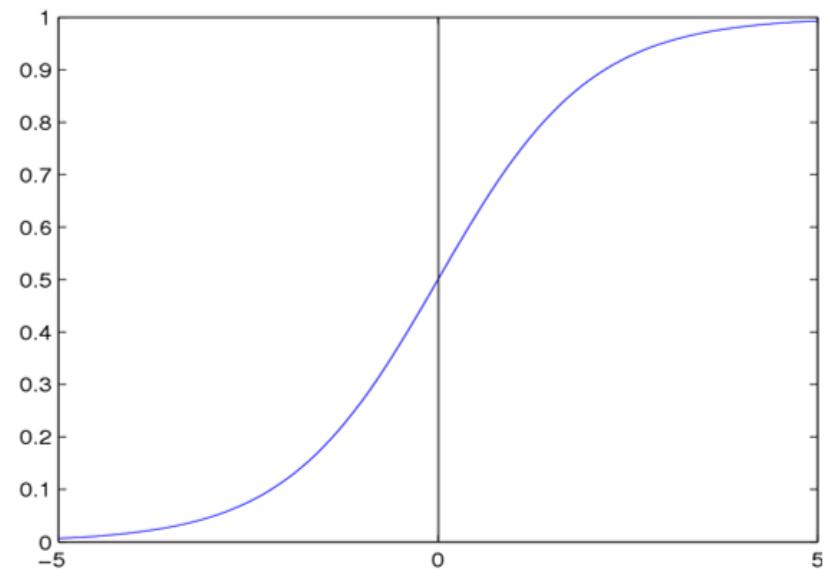
| Again, we are given a training set of n labelled samples $\langle \mathbf{x}^{(i)}, y^{(i)} \rangle$

| Why not directly model/learn $P(y|x)$?

- Discriminative model

| Further assume $P(y|x)$ takes the form of a logistic sigmoid function

→ Logistic Regression



Logistic Regression



| Logistic regression: use the logistic function for modeling $P(y|x)$, considering only the case of $y \in \{0, 1\}$

$$P(y = 0|\mathbf{x}) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^d w_i x_i)}$$

$$P(y = 1|\mathbf{x}) = \frac{\exp(w_0 + \sum_{i=1}^d w_i x_i)}{1 + \exp(w_0 + \sum_{i=1}^d w_i x_i)}$$

| The *logistic function*

$$\sigma(t) = \frac{1}{1+e^{-t}} = \frac{e^t}{1+e^t}$$

Logistic Regression → Linear Classifier



| Given a sample x , we classify it as 0 (i.e., predicting $y=0$) if

$$P(y=0|x) \geq P(y=1|x)$$

→ This is a linear classifier.

The Parameters of the Model



- | What are the model parameters in logistic regression?
- | Given a parameter w , we have $P(y|x) =$

$$[\sigma(w^T x)]^y [1 - \sigma(w^T x)]^{1-y}$$

- | Suppose we have two different sets of parameters, $w^{(1)}$ and $w^{(2)}$, whichever giving a larger $P(y|x)$ should be a better parameter.

The Conditional Likelihood

| Given n training samples, $\langle \mathbf{x}^{(i)}, y^{(i)} \rangle$, $i=1,\dots,n$, how can we use them to estimate the parameters?

→ For a given w , the probability of getting all those $y^{(1)}, y^{(2)}, \dots, y^{(n)}$ from the corresponding data $\mathbf{x}^{(i)}$, $i=1,\dots,n$, is

$$\begin{aligned} P\left[y^{(1)}, y^{(2)}, \dots, y^{(n)} \mid \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}, w\right] &= \prod_{i=1}^n P(y^{(i)} \mid \mathbf{x}^{(i)}, w) \\ &= \prod_{i=1}^n \left[\sigma(w^T \mathbf{x}^{(i)})\right]^{y^{(i)}} \left[1 - \sigma(w^T \mathbf{x}^{(i)})\right]^{1-y^{(i)}} \end{aligned}$$

→ Call this $L(w)$, the (conditional) likelihood.

The Conditional Log Likelihood

$$\begin{aligned} l(w) &= \log L(w) = \log \prod_{i=1}^n (\dots) \\ &= \sum_{i=1}^n \log \left[\sigma(w^T x^{(i)})^{y^{(i)}} (1 - \sigma(w^T x^{(i)}))^{1-y^{(i)}} \right] \\ &= \sum_{i=1}^n \left[\log(\sigma(w^T x^{(i)}))^{y^{(i)}} + \log((1 - \sigma(w^T x^{(i)}))^{1-y^{(i)}}) \right] \end{aligned}$$

Maximizing Conditional Log Likelihood



| Optimal parameters

$$\begin{aligned}\mathbf{w}^* &= \operatorname{argmax}_{\mathbf{w}} l(\mathbf{w}) \\ &= \operatorname{argmax}_{\mathbf{w}} \sum_{i=1}^n [y^{(i)} \mathbf{w}^t \mathbf{x}^{(i)} - \log(1 + \exp(\mathbf{w}^t \mathbf{x}^{(i)}))]\end{aligned}$$

| We cannot really solve for \mathbf{w}^* analytically (no closed-form solution)

- We can use a commonly-used optimization technique, gradient descent/ascent, to find a solution.

Finding the Gradient of $l(w)$

$$\nabla_w l(w) = \nabla_w \left[\sum_{i=1}^n \left(y^{(i)} w^T x^{(i)} - \log(1 + e^{w^T x^{(i)}}) \right) \right],$$

Recall: $\frac{\partial(w^T x)}{\partial w} = x$, $\frac{\partial \log f(x)}{\partial x} = \frac{1}{f(x)} \frac{\partial f(x)}{\partial x}$
 $\frac{\partial e^x}{\partial x} = e^x$

$$= \sum_{i=1}^n \left[y^{(i)} x^{(i)} - \frac{e^{w^T x^{(i)}} \cdot x^{(i)}}{1 + e^{w^T x^{(i)}}} \right]$$

↑
Setting this to 0 cannot really give us a closed-form solution for w .
So we will do gradient ascent.)

Gradient Ascent Algorithm



The algorithm

Iterate until converge

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \eta \nabla_{\mathbf{w}^{(k)}} l(\mathbf{w})$$

$\eta > 0$ is a constant called the learning rate.





Linear Machines & SVM

Linear Machines

Objective



Objective

Define general
linear classifiers

Revisiting Logistic Regression

| In Logistic Regression: given a training set of n labelled samples $\langle \mathbf{x}^{(i)}, y^{(i)} \rangle$, we learn $P(y|\mathbf{x})$ by assuming a logistic sigmoid function.

- We end up with a *linear classifier*.
- $g(\mathbf{x}) = \mathbf{w}^t \mathbf{x}$ is called the *discriminant function*.

Linear Discriminant Functions



- | In general, taking a discriminative approach, we can *assume* some form for the discriminant function that defines the classifier.
 - The learning task is to use the training samples to estimate the parameters of the classifier.

Linear Decision Boundaries



| Linear discriminant functions give arise to liner decision boundaries

→ *linear classifiers* or *linear machines*

| We will use both notations:

$$g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} \quad \text{or} \quad g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0$$

Linear Machine for $C > 2$ Classes



| We can define C linear discriminant functions:

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x}, \quad i = 1, 2, \dots, C$$

| What is the decision rule for the classifier?

The Learning Task



| Finding $w_i, i = 1, 2, \dots, C$

| Let's use the 2-class case as an example

- For n samples $\mathbf{x}_1, \dots, \mathbf{x}_n$, of 2 classes ω_1 and ω_2 , if there exists a vector \mathbf{w} such that $g(\mathbf{x}) = \mathbf{w}^t \mathbf{x}$ classifies them all correctly → Finding \mathbf{w}

| i.e., finding \mathbf{w} such that

$\mathbf{w}^t \mathbf{x}_i \geq 0$ for samples of ω_1 and

$\mathbf{w}^t \mathbf{x}_i < 0$ for samples of ω_2 ,

Linear Separability



| If we can find at least one vector w such that $g(x) = w^t x$ classifies all samples

→ We say the samples are linearly separable.

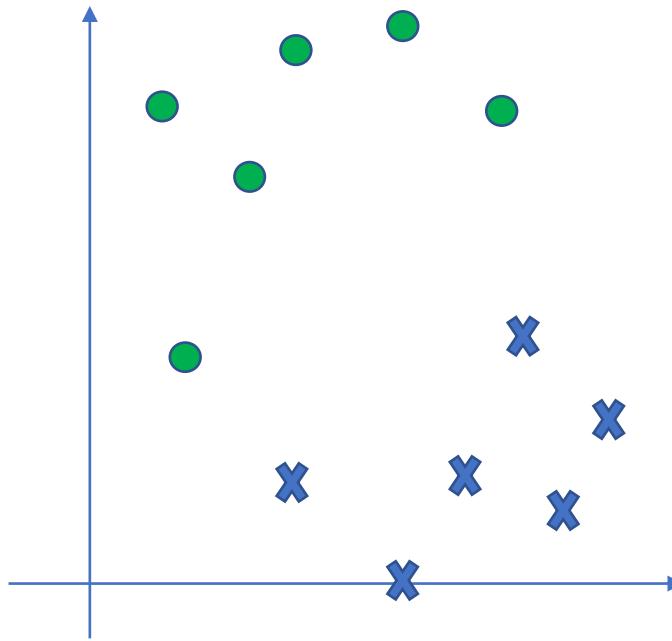
| An example of not linearly separable in 2-D:

The Solution Region

| There may be many different weight vectors that can all be valid solutions for a given training set

→ The solution regions

| If the solution vector is not unique, *Which one is the best?*



Solving for the Weight Vector



| Consider the following approach: finding a solution vector which optimizes some objective function.

- We may introduce additional constraints for a “good” solution”
- Solving a constrained optimization problem.

| Theoretical: Lagrange or Karush-Kuhn-Tucker.

| In practice: e.g., gradient-descent-based search

Gradient Descent Procedure



| Basic idea:

- Define a cost function $J(\mathbf{w})$
- Starting from an initial weight vector $\mathbf{w}(0)$
- Update \mathbf{w} by

$$\mathbf{w}(k + 1) = \mathbf{w}(k) - \eta(k) \nabla J(\mathbf{w}(k)),$$

| $\eta > 0$ is the *learning rate*.





Linear Machines & SVM

The Concept of Margins

Objective



Objective
Illustrate Margins
in Classifier

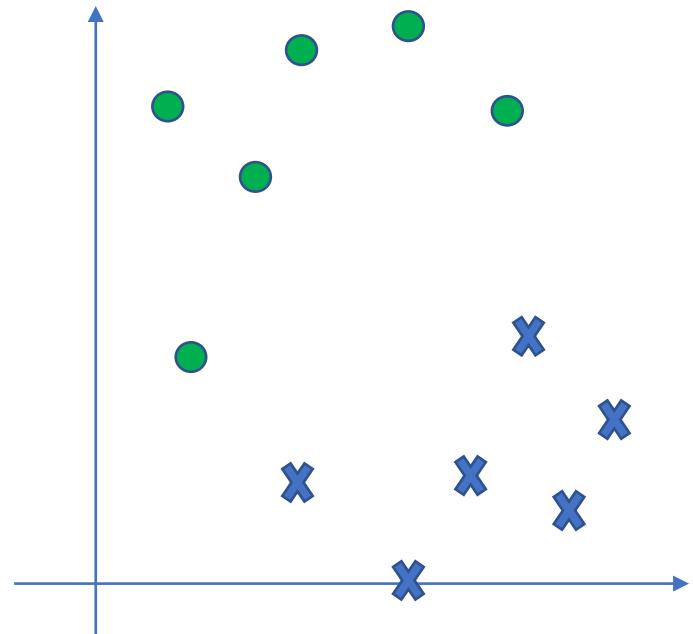
Illustrating Linear Boundaries

| The decision boundaries is given by the line $g(\mathbf{x}) = 0$.

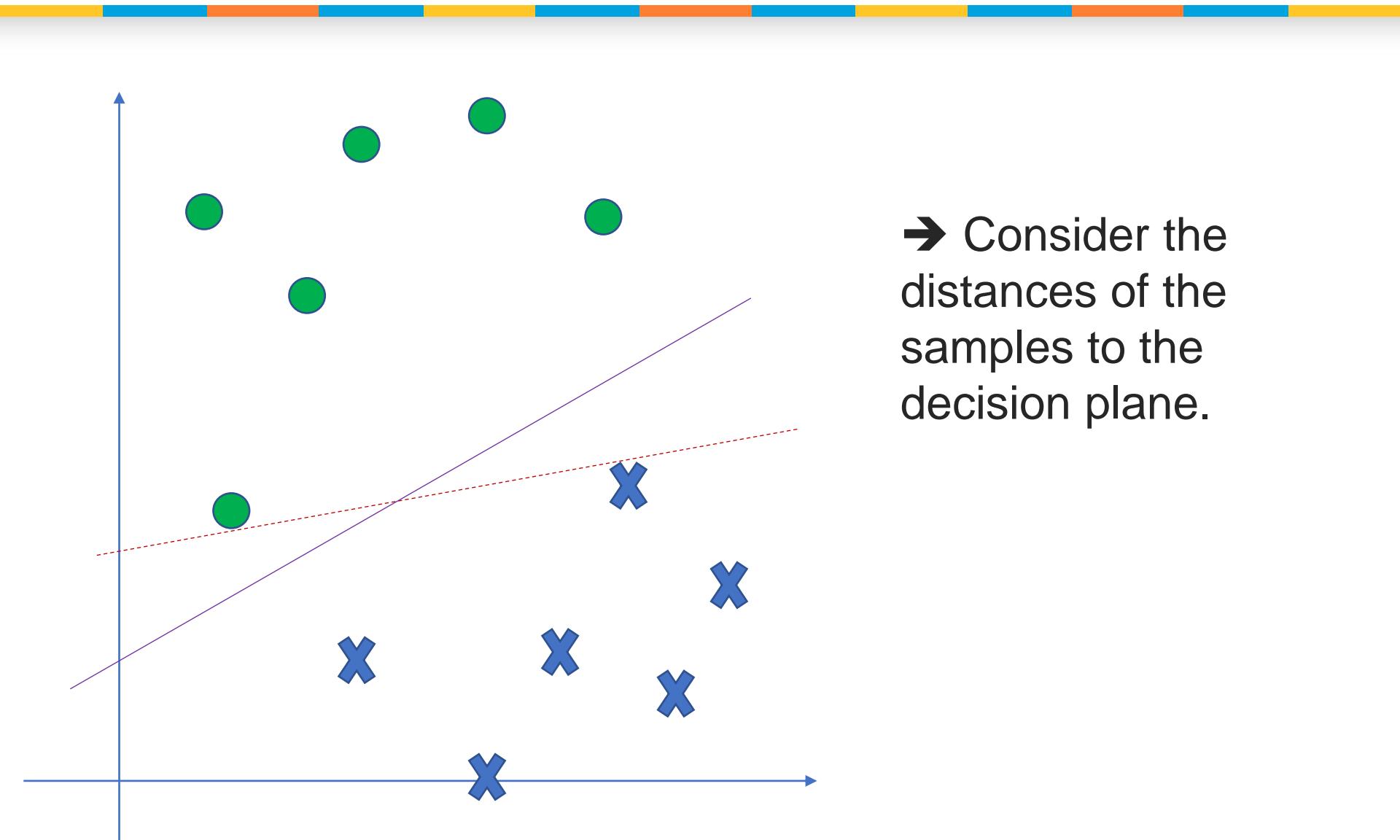
- For appreciating a geometric interpretation, we will write w_0 explicitly, i.e., we have

$$g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + w_0$$

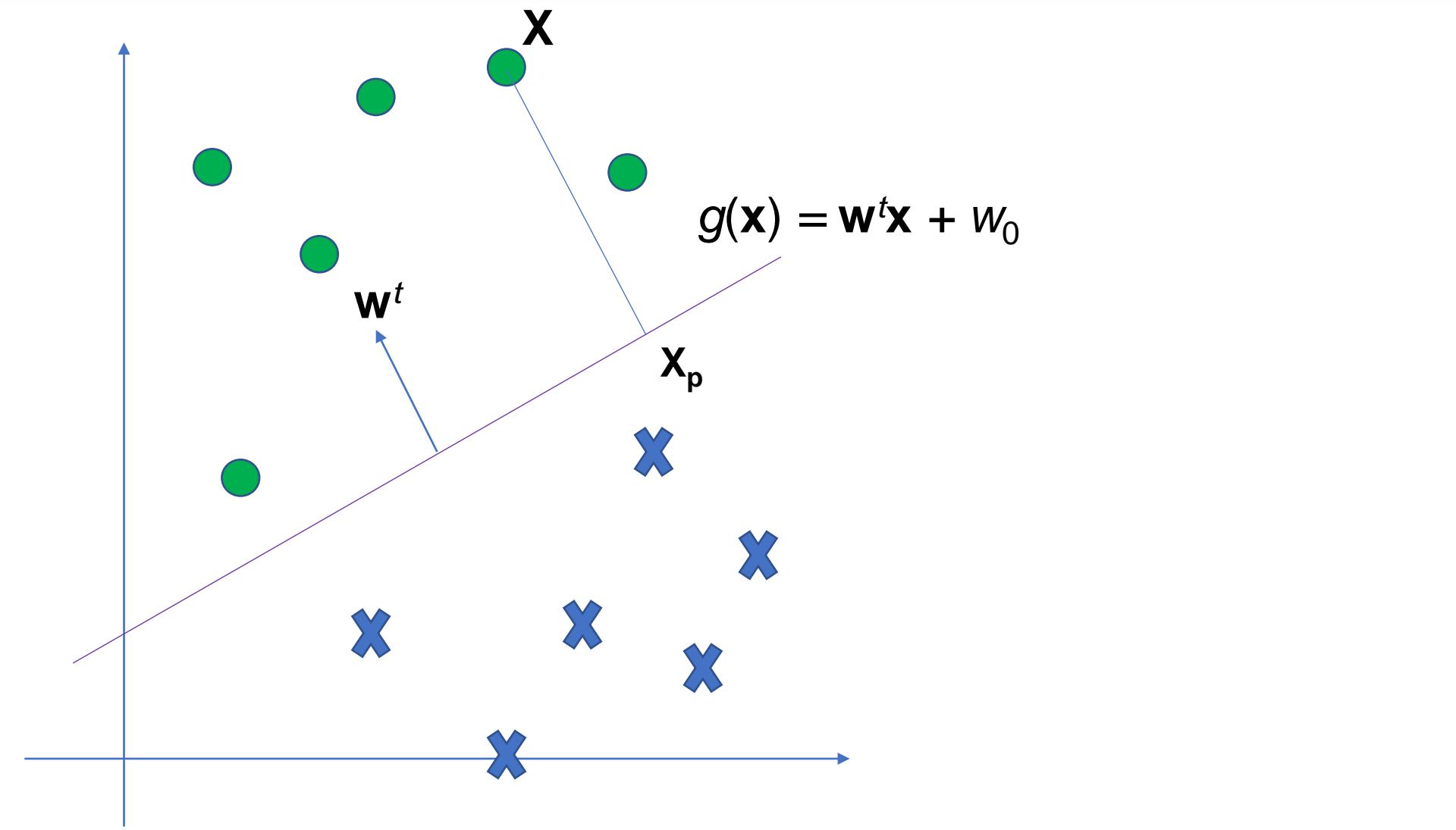
| The normal vector of the decision line/plane is



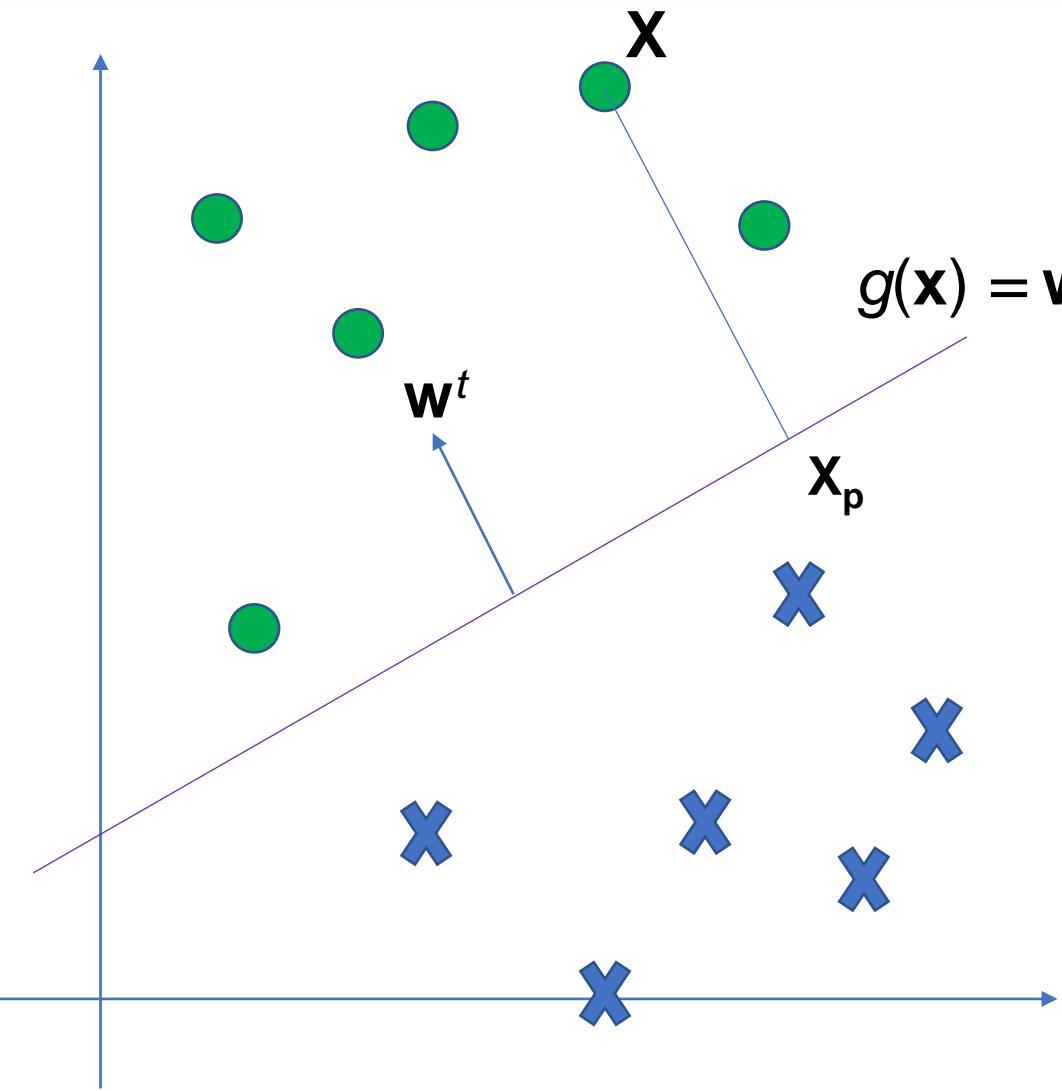
Which one is better?



Distance to the Decision Plane



Distance to the Decision Plane



| $g(x)$ gives an algebraic measure of the distance from x to the decision plane.

The Concept of Margins

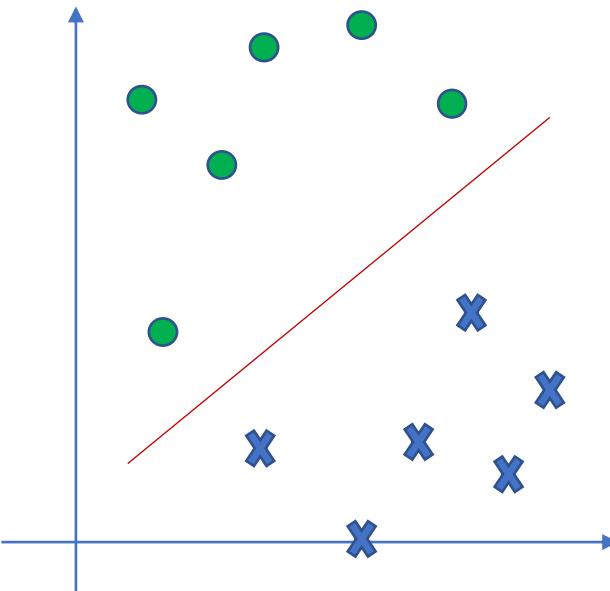
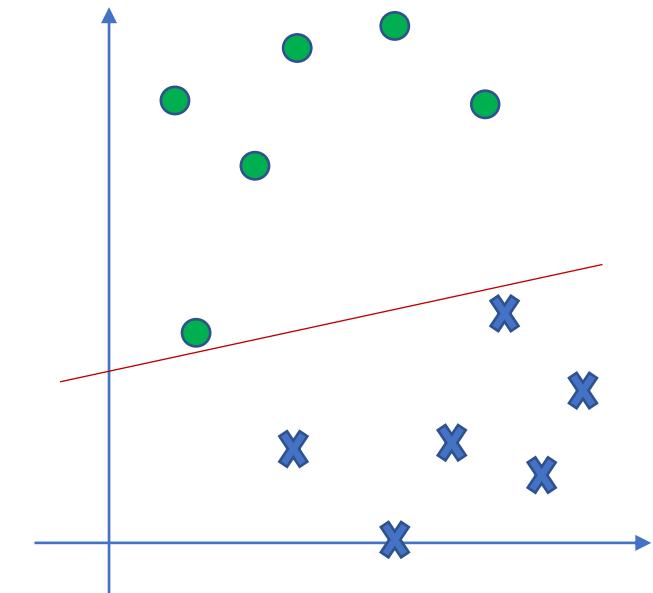
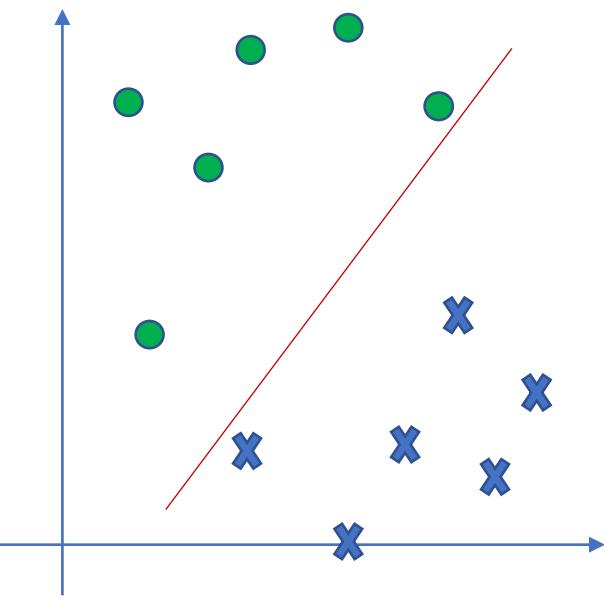


| Let $g(x) = 0$ be a decision plane

- The **margin** of a sample x (w.r.t. the decision plane) is the distance from x to the plane.
- For a given set of samples S , the margin (w.r.t a decision plane) is the smallest margin over all x in S .

| For a given set, a classifier that gives rise to a larger margin will be better.

Use Margins to Compare Solutions



→ Max margin
→ SVM



Linear Machines & SVM

Linear SVM: Linearly Separable Case

Objective



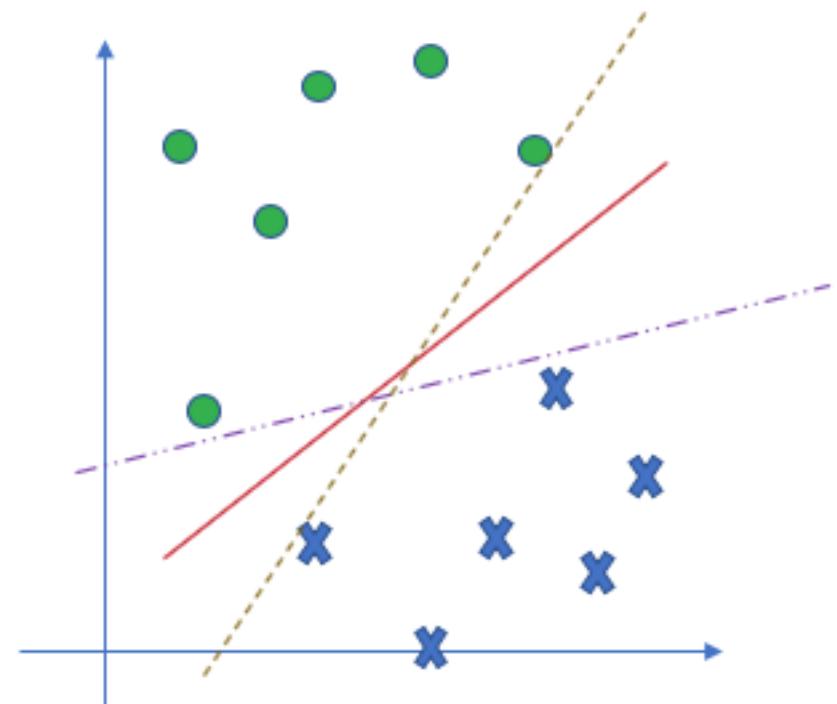
Objective

Construct SVM for
Linearly Separable
Data

Key Idea of Support Vector Machines

| For a given set, a classifier that gives rise to a larger margin will be better.

| SVM: To find the decision boundary such that the margin is maximized.



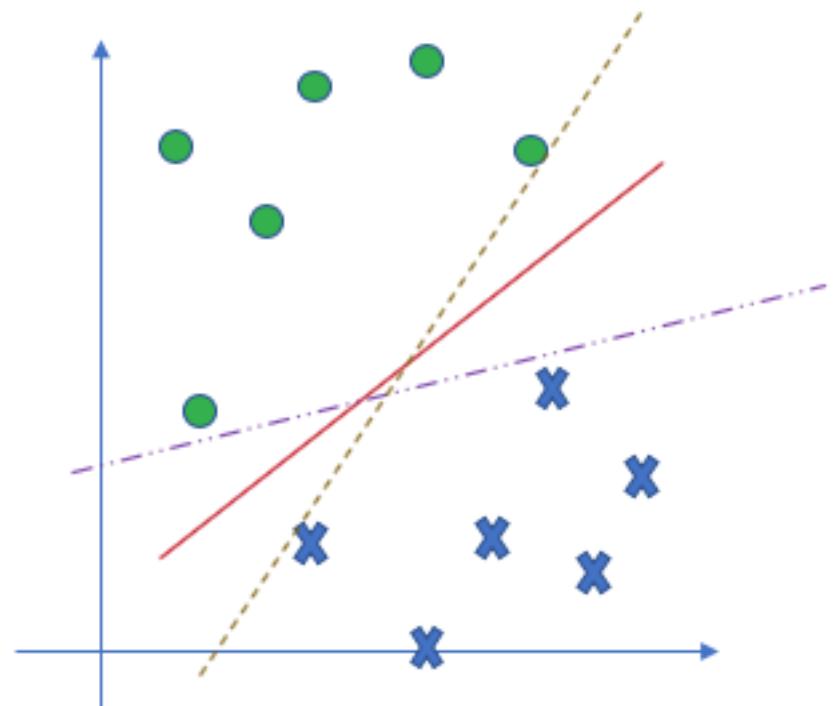
Formulating the Problem

| Given labeled training data:

$$\langle \mathbf{x}^{(i)}, y^{(i)} \rangle, y^{(i)} \in \{-1, 1\}, \mathbf{x}^{(i)} \in \mathbb{R}^d, i=1, \dots, n,$$

| Assuming the points are linearly separable, let's write a separating hyperplane as:

$$H: \mathbf{w}^t \mathbf{x} + b = 0$$

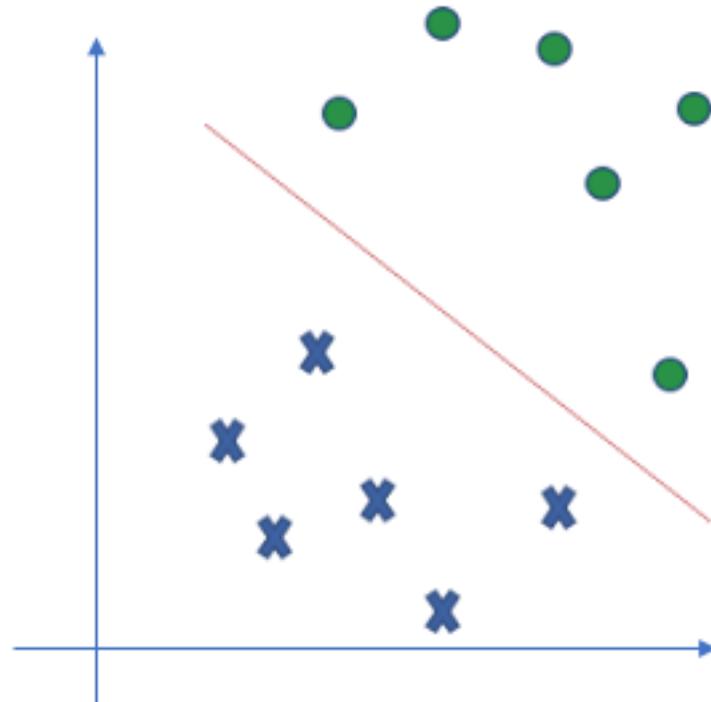


Formulating the Problem (cont'd)

| Let d_+ (d_-) be the shortest distance from the separating hyperplane to the *closest* positive (negative) examples.

| These defines planes H_1 and H_2 .

| We can let $d_+ = d_- = d$
→ Find a solution maximizing $2d$.

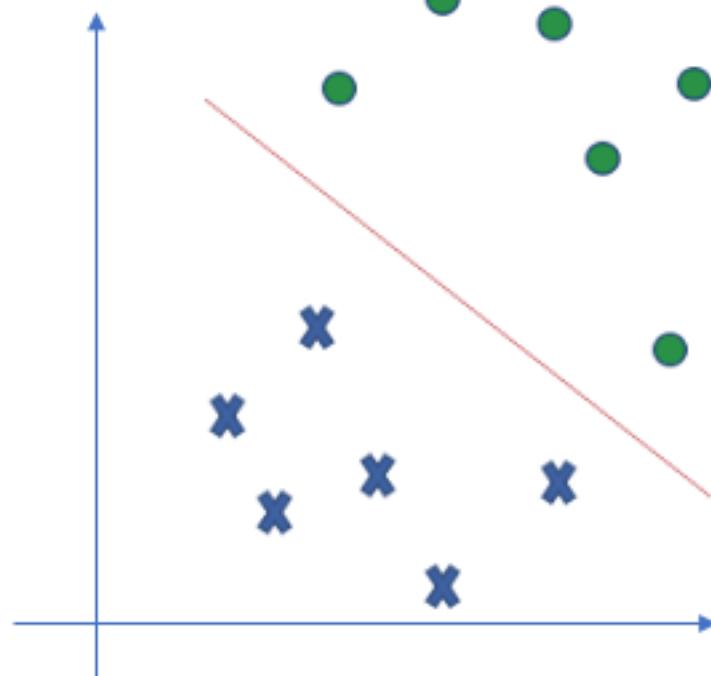


Formulating the Margin

| Given separating plane H :
 $w^t x + b = 0$ and distance d ,
what are the equations for
 H_1 and H_2 ?

| Consider the plane H^*
given by $w^t x + b = \|w\|d$

- Check its orientation
- Check its distance to H



Formulating the Margin (cont'd)

| H_1 is given by $w^t x + b = \|w\|/d$

| Similarly, H_2 is given by $w^t x + b = -\|w\|/d$

| Note: for any plane equation, $w^t x + b = 0$, $\{w, b\}$ is defined only up to an unknown scale:

- $\{sw, sb\}$ is also a valid solution to the equation, for any constant s .

Formulating the Margin (cont'd)

→ We can have the canonical formulation for all the planes as

$$H: \mathbf{w}^t \mathbf{x} + b = 0$$

$$H_1: \mathbf{w}^t \mathbf{x} + b = 1$$

$$H_2: \mathbf{w}^t \mathbf{x} + b = -1$$

→ The region between H_1 and H_2 is also called the margin, and its width is $\frac{2}{\|\mathbf{w}\|}$

Formulating SVM



$$\{\mathbf{w}^*, b^*\} = \underset{\mathbf{w}, b}{\operatorname{argmin}} \|\mathbf{w}\| \text{ or } \{\mathbf{w}^*, b^*\} = \underset{\mathbf{w}, b}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{w}\|^2$$

Subject to

$$\mathbf{w}^t \mathbf{x}^{(i)} + b \geq 1 \quad \text{for } y^{(i)} = +1$$

$$\mathbf{w}^t \mathbf{x}^{(i)} + b \leq -1 \quad \text{for } y^{(i)} = -1$$

The constraints can be combined into:

$$y^{(i)}(\mathbf{w}^t \mathbf{x}^{(i)} + b) - 1 \geq 0 \quad \forall i$$

→ A nonlinear (quadratic) optimization problem
with linear inequality constraints.

How to solve SVM? (Outline)



| Reformulate the problem using Lagrange multipliers

α

- Lagrangian Primal Problem
- Lagrangian Dual Problem

| The Karush-Kuhn-Tucker Conditions

- *Necessary and sufficient* for \mathbf{w} , b , α .
- Solving the SVM problem → finding a solution to the KKT conditions.

SVM: Lagrangian Primal Formulation

| Define

$$L_P(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_i \alpha_i [y^{(i)} (\mathbf{w}^t \mathbf{x}^{(i)} + b) - 1]$$

| then the SVM solution should satisfy

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0, \quad \frac{\partial L_P}{\partial b} = 0,$$

$$\alpha_i \geq 0,$$

$$\alpha_i [y^{(i)} (\mathbf{w}^t \mathbf{x}^{(i)} + b) - 1] = 0$$



The final \mathbf{w} is given by

$$\mathbf{w} = \sum_i \alpha_i y^{(i)} \mathbf{x}^{(i)}$$

and b is given by

$$y^{(k)} - \mathbf{w}^t \mathbf{x}^{(k)}$$

for any k such that $\alpha_k > 0$

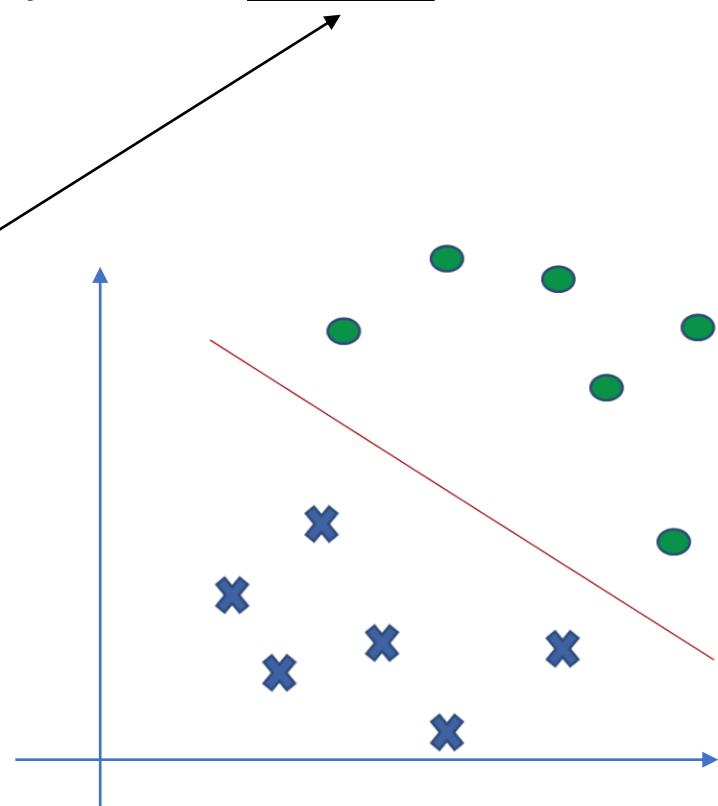
SVM: Lagrangian Dual Formulation

| The objective function is

$$L_D(\mathbf{w}, b, \alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \overrightarrow{\mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}}$$

| The solution is the same as before. But there is an important observation.

| Points for which $\alpha_i > 0$ are called support vectors





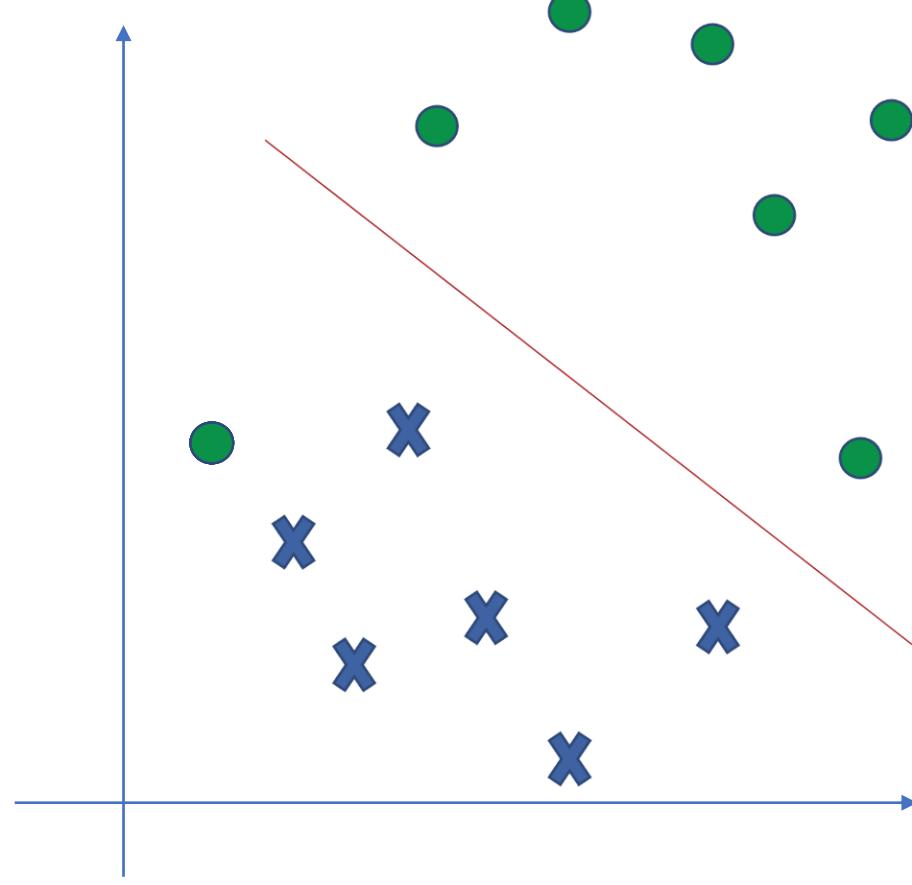


Linear Machines & SVM

SVM for Non-linearly-separable Case

Linear Separability Violated

| Some samples will always be misclassified no matter what $\{w, b\}$ is used.

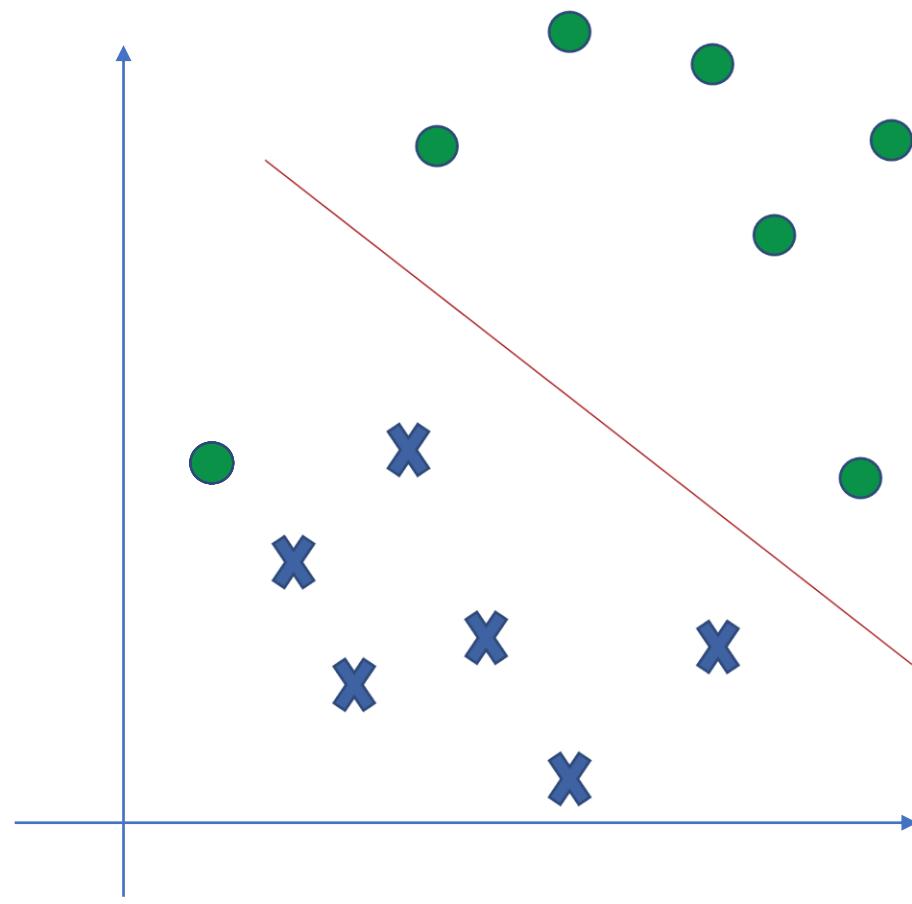


Examining Misclassified Samples

| They will violate the constraints:

$$\mathbf{w}^t \mathbf{x}^{(i)} + b \geq 1 \quad \text{for } y^{(i)} = +1$$

$$\mathbf{w}^t \mathbf{x}^{(i)} + b \leq -1 \quad \text{for } y^{(i)} = -1$$



Relaxing the Constraints

| Introducing *non-negative* slack variables ξ_i

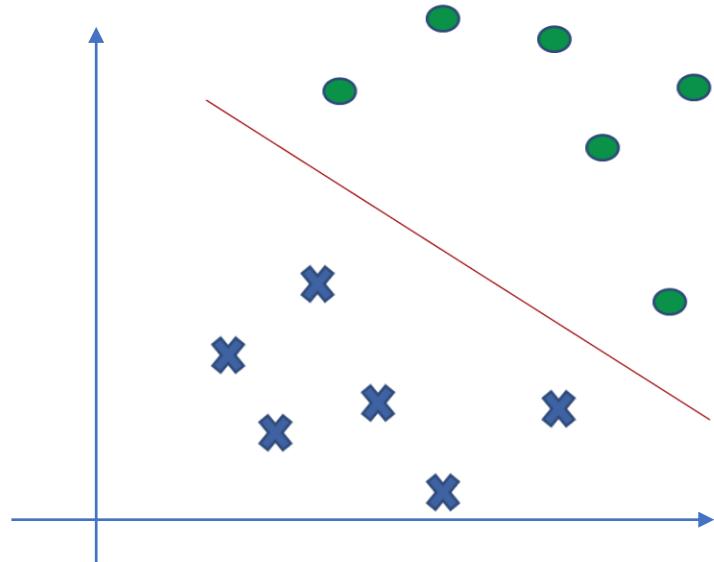
$$\mathbf{w}^t \mathbf{x}^{(i)} + b \geq 1 - \xi_i \text{ for } y^{(i)} = +1$$

$$\mathbf{w}^t \mathbf{x}^{(i)} + b \leq -1 + \xi_i \text{ for } y^{(i)} = -1$$

| For an error to occur, the corresponding ξ_i must exceed unity.

– *Hinge loss or soft margin.*

→ $\sum_i \xi_i$ provides an upper bound on the number of training errors.



Updating the Formulation



| C is a parameter to control how much penalty is assigned to errors.

$$\{\mathbf{w}^*, b^*\} = \operatorname{argmin}_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C (\sum_i \xi_i)$$

Subject to

$$\mathbf{w}^t \mathbf{x}^{(i)} + b \geq 1 - \xi_i \text{ for } y^{(i)} = +1$$

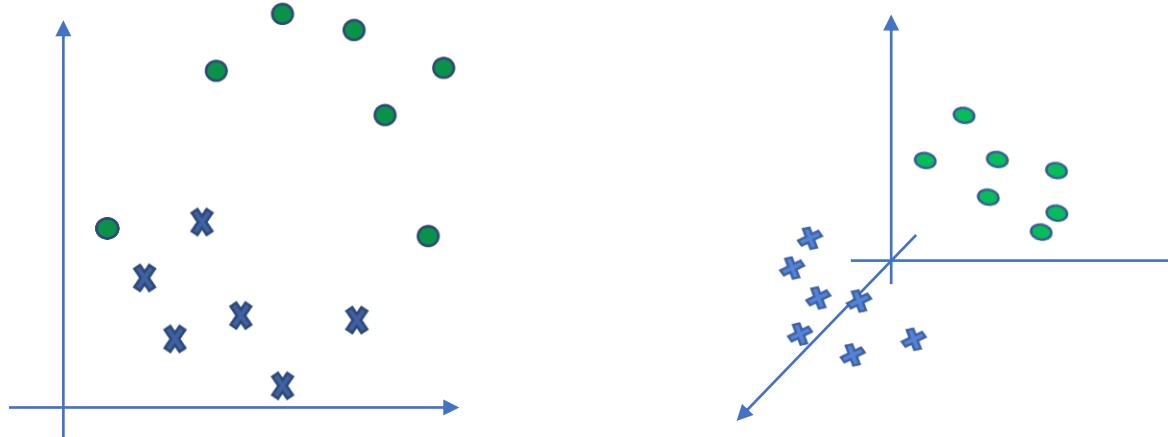
$$\mathbf{w}^t \mathbf{x}^{(i)} + b \leq -1 + \xi_i \text{ for } y^{(i)} = -1$$

$$\xi_i \geq 0, \forall i$$

Are Non-linear Decision Boundaries Possible?

| Transform data to higher dimensions using a mapping

- More freedom to position the samples
- May make the samples linearly separable
- Run linear SVM in the new space → may be equivalent to non-linear boundaries in the original space



| What mapping to use?

The Kernel Trick



| Revisit the Lagrange Dual Formulation for SVM

$$L_D(\mathbf{w}, b, \alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}$$

| Introduce a kernel function

$$L_D(\mathbf{w}, b, \alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y^{(i)} y^{(j)} K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

The Kernel Trick (cont'd)

| Mercer's Theorem: for a symmetric, non-negative definite kernel function satisfying some minor conditions, there exists a mapping $\Phi(x)$ such that

$$K(x^{(i)}, x^{(j)}) = \Phi(x^{(i)}) \cdot \Phi(x^{(j)})$$

- Using a kernel function in L_D can effectively defines an implicit mapping to a higher-dimensional space, where linear SVM was run.
- The decision boundaries in the original space can be highly non-linear.

Common Kernel Functions



| Polynomials of
degree d

$$K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle^d$$

| Polynomials of
degree up to d

$$K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = (\langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle + 1)^d$$

| Gaussian kernels

$$K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2}{2\sigma^2}\right)$$

| Sigmoid kernel

$$\begin{aligned} K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \\ = \tanh(\eta \langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \rangle + \nu) \end{aligned}$$





Graphical Models

Bayesian Networks

Objectives



Objective

Describe Bayesian
Networks

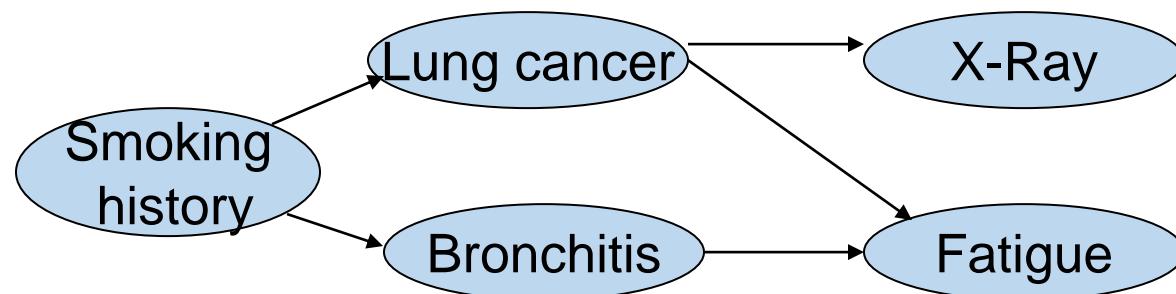


Objective

Illustrate key tasks in
implementing
Bayesian Networks

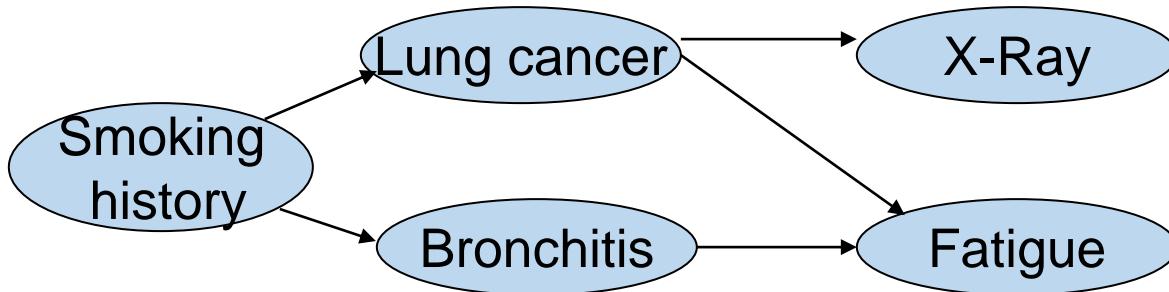
Why do we use graphical models?

- | In machine learning, we are often concerned with joint distributions of many random variables.
- | A graph may provide an intuitive way of representing or visualizing the relationships of the variables.
 - Making it easier for domain experts to build a model



Graphical Models for Casual Relations

| Graphical models arise naturally from, often causal, independency relations of physical events.



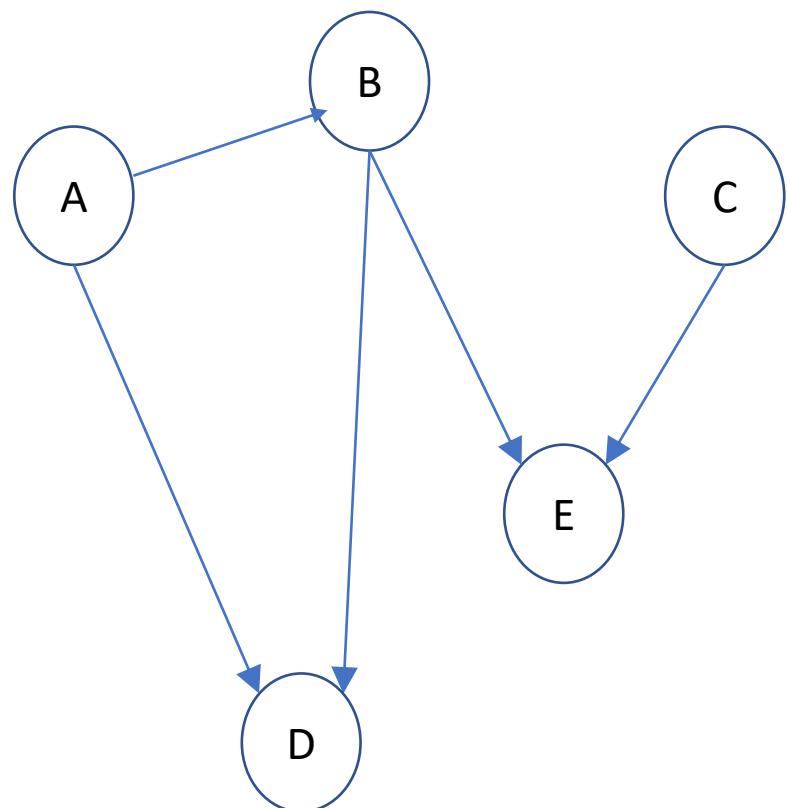
| Caveat: probabilistic relationship does not imply causality.

Bayesian Networks

| A BN is directed acyclic graph (DAG), where

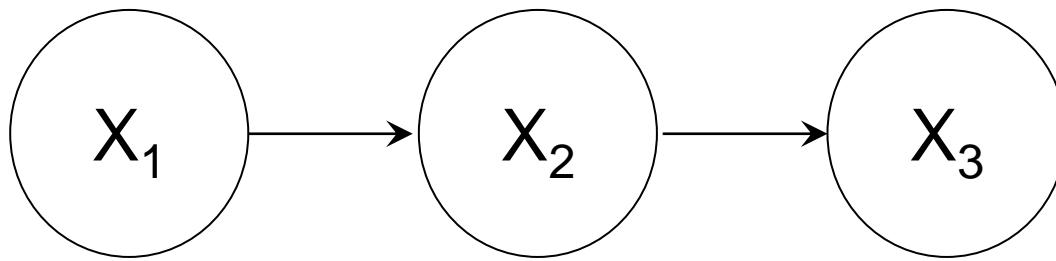
- Nodes (vertices) represent random variables.
- Directed edges represent immediate dependence of nodes.

| Other names: Belief networks, Bayes nets, etc.



Conditional Independence

| E.g., given the following graph, check the relationship between X_3 and X_1



- X_3 is dependent of X_2 , and X_2 is dependent of X_1
- Thus X_3 is dependent of X_1
- But given X_2 , X_3 is independent of X_1

→ Conditional Independence

BN for General Conditional Dependency

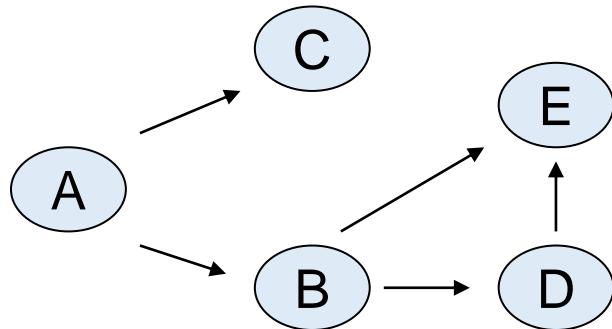
| A BN can be used to model given conditional dependencies

- For example, using the *chain rule of probability*, we have

$$P(A,B,C,D,E) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)P(E|A,B,C,D)$$

| If we know that, given A, C won't rely on B, and so forth, we may have

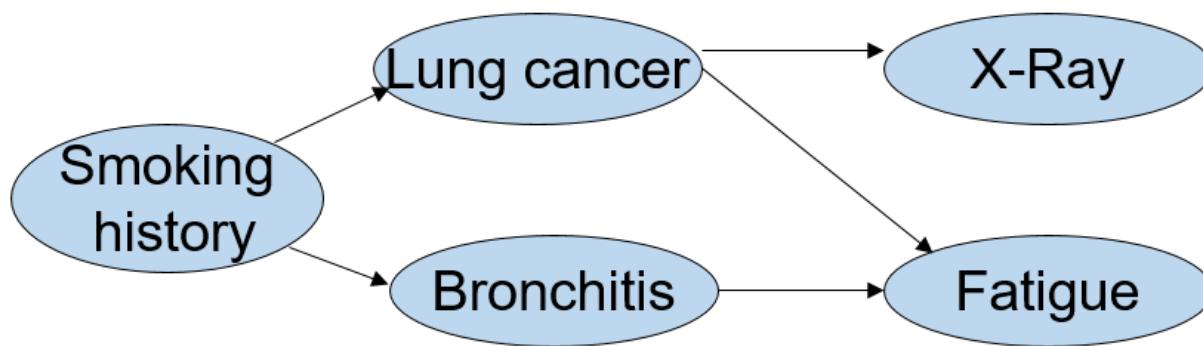
$$P(A,B,C,D,E) = P(A)P(B|A)P(C|A)P(D|B)P(E|B,D)$$



➤ We could represent joint distributions more compactly in BN → Efficient computation

Inference in Bayesian Networks

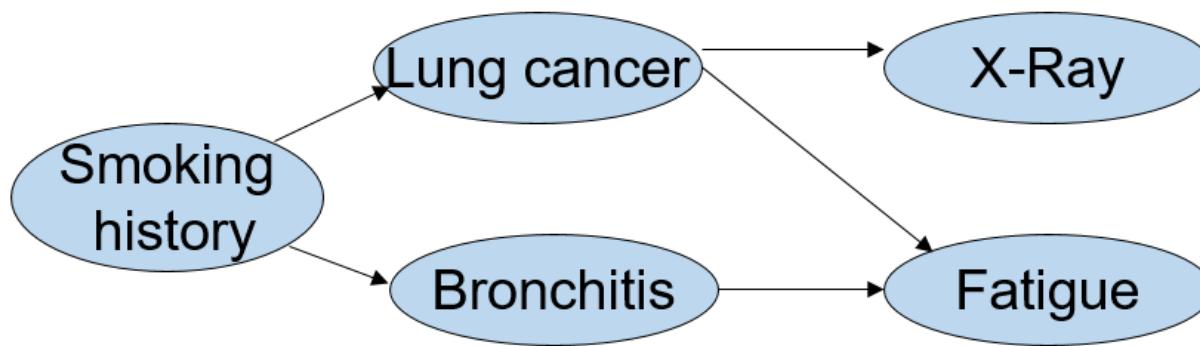
| Given a model and some data (“evidence”), how to update our belief?



| What are the model parameters?

Inference in Bayesian Networks (cont'd)

| Given a model and some data (“evidence”), how to update our belief?

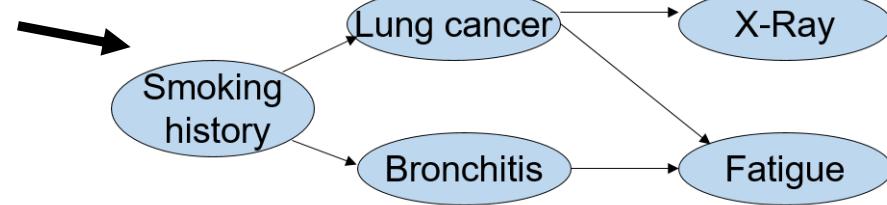


| E.g., for a patient with certain smoking history (non-smoker), whose X-ray result is positive, and who does not experience fatigue:

- What is probability of having lung cancer?

Inference in Bayesian Networks (cont'd)

| In a simple BN like this,
we can compute the
exact probabilities.



| In general, for a tree-structured BN, we may use belief propagation for the inference problem.

| For general structures, sometimes it is possible to generalize the above method (e.g., the *junction tree algorithm*). More often, we must resort to approximation methods

- E.g. Variational methods, Sampling (Monte Carlo) methods.

Learning in Bayesian Networks



| Learning parameters (probabilities) for a given BN (the graph is given).

- Estimate the (conditional) probabilities from past data.

| Learning both the structure and the parameters for a BN

- A more challenging task beyond the scope of this discussion.

Learning the Probabilities



| Basic ideas

- Use relative frequency for estimating probability.
- A prior distribution is typically assumed.
- The prior is then updated by the data into posterior.
- Using the MLE principle

| The so-called “Expectation-Maximization (EM) Algorithm” is often used.

- Iteratively update our guess for the parameter and each step attempts to apply the MLE principle.





Graphical Models

Hidden Markov Formulation

Objectives



Objective

Introduce Hidden Markov
Models



Objective

Illustrate HMM with
intuitive examples

Hidden Markov Models



| **Hidden Markov Models (HMMs) are a type of dynamic Bayesian Network**

- Modeling a process indexed by time

| **“Hidden”: the observations are due to some underlying (hidden) states not directly observable.**

| **“Markov”: the state transitions are governed by a Markov process.**

Discrete Markov Process



| Consider a system which may be described at any time as being in one of a set of N distinct states, S_1, \dots, S_N .

| At time instances $t=1,2,3, \dots$, the system changes its state according to certain probability. The full description requires us to know $P(s^t=S_j | s^{t-1}=S_i, s^{t-2}=S_k, \dots, s^1=S_m)$ for all t, i, k, \dots, m , where s^t stands for the state of the system at time t .

- For a first-order Markov chain, we need to consider only

$$P(s^t=S_j | s^{t-1}=S_i)$$

- Further assume P s are “stationary”:

$$a_{ij} = P(s^t=S_j | s^{t-1}=S_i), \quad 1 \leq i, j \leq N, \text{ for any } t.$$

A Simple Example



| Assume one of the three states for each day:

S_1 -rainy, S_2 -cloudy, S_3 -sunny

| Assume the transition probability matrix

$$A = \{a_{ij}\} = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.3 & 0.4 \\ 0.1 & 0.2 & 0.7 \end{bmatrix}$$

| Many questions we may ask, based on this model.

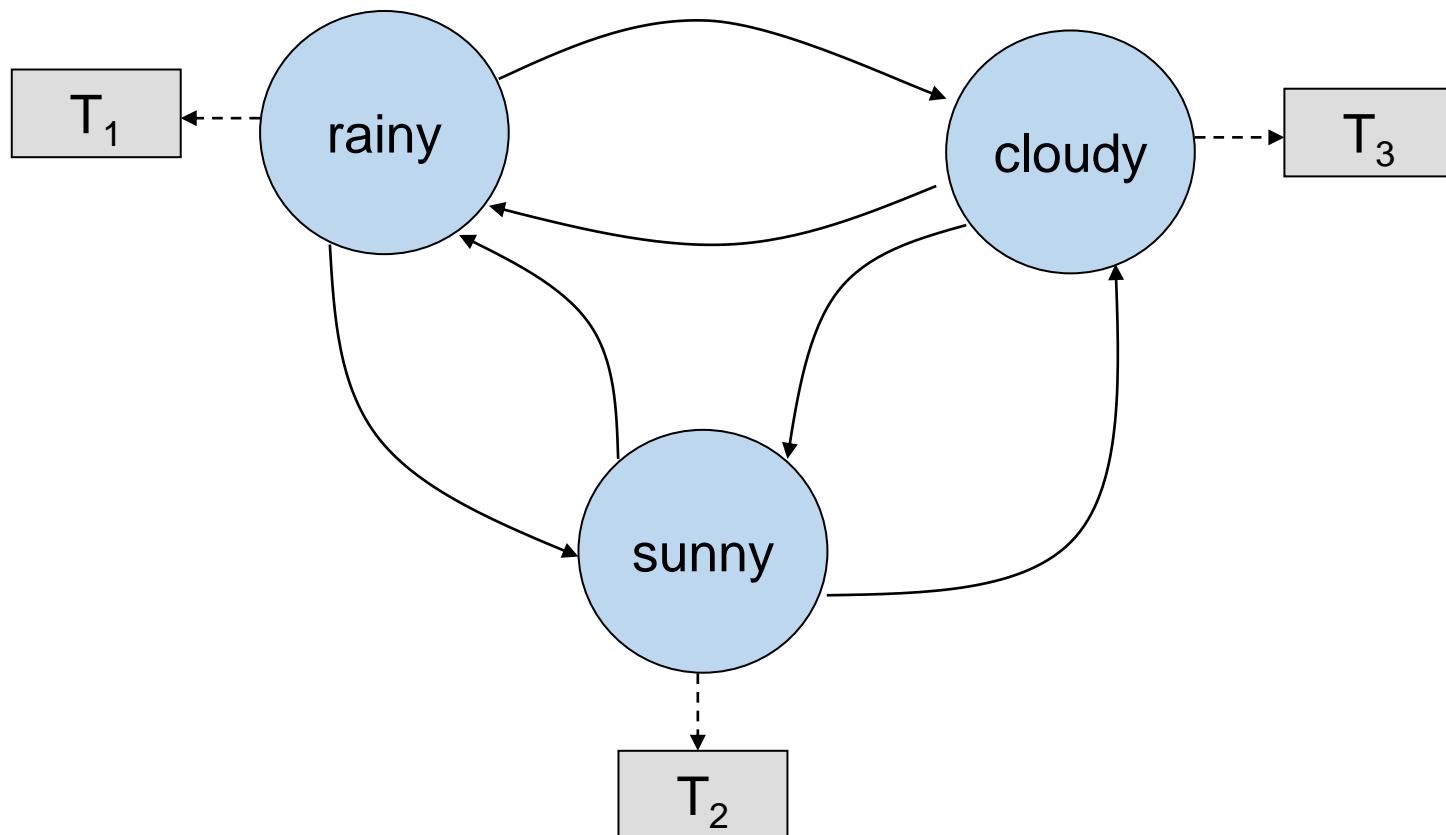
- E.g., Given today is cloudy, what is the probability it remains to be cloudy for next 5 days?

Extending to “Hidden” States

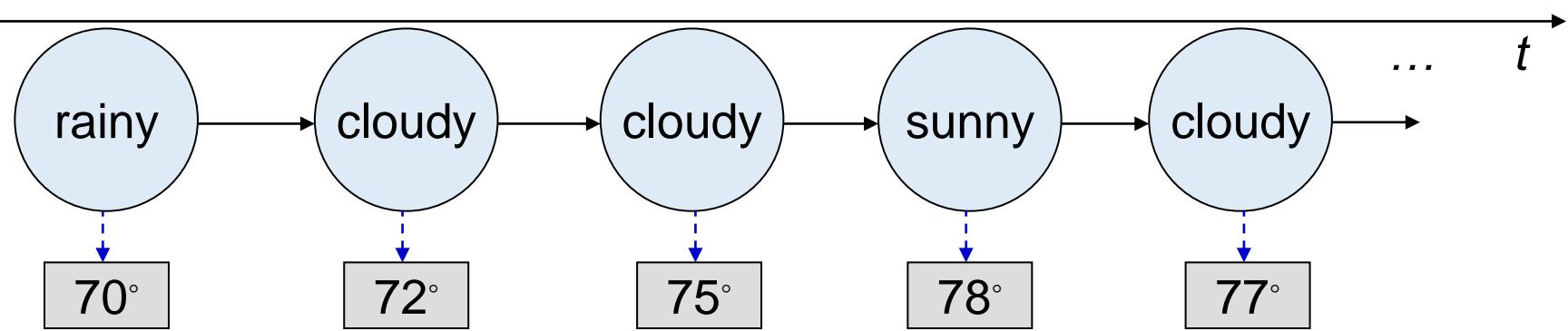


- | The previous example is an “observable” Markov model: the output of the system/process is the states of interest.
- | Now assume that we can only measure the (average) temperature of a day
 - Further assume this measurement is useful for predicting the weather states (rainy, cloudy, sunny).
 - We can view the temperature values as being produced by the *hidden states* of interest, i.e., the weather.

A Simple HMM



A Specific Process from the Model



Specifying an HMM



| **Θ : the set of hidden states.**

| **The state transition probabilities** $a_{ij} = P(s^t=S_j \mid s^{t-1}=S_i)$, $1 \leq i, j \leq N$

- Let $A=\{a_{ij}\}$ be the transition probability matrix

| **Ω : the set of outputs (observations).**

Specifying an HMM (cont'd)

| The observation probabilities: $P(o^t|s^t)$, where o^t stands for the observation at time t , given the state s^t . This is also called the emission probability.

- For discrete observation space, we can define $B=\{b_{jk}\}=P(o^t=v_k \text{ at } t|s^t=S_j)$ as the emission probability matrix, where v_k is the k^{th} symbol in Ω

| The initial state distribution $\pi = \{\pi_i\}$, $\pi_i=P(s^1=S_i)$

- Sometimes we are given an initial state, i.e., $P(s^1=S_i)=1$ for certain i .

Basic Problems in HMM



| For a given HMM $\Lambda=\{\Theta, \Omega, A, B, \pi\}$

- Problem 1: Given an observation (sequence) $O=\{o^1, o^2, \dots, o^k\}$, what is the most likely state sequence $S=\{s^1, s^2, \dots, s^k\}$ that has produced O ?
- Problem 2: How likely is an observation O (i.e., what is $P(O)$) ?
- Problem 3: How to estimate the model parameters (A, B, π) ?





Graphical Models

Hidden Markov Models: Learning & Inference

Objective



Objective

Implement HMM
learning & inference
algorithms

Basic Problems in HMM



| For a given HMM $\Lambda = \{\Theta, \Omega, A, B, \pi\}$

- Problem 1: Given an observation (sequence) $O = \{o^1, o^2, \dots, o^k\}$, what is the most likely state sequence $S = \{s^1, s^2, \dots, s^k\}$ that has produced O ?
- Problem 2: How likely is an observation O (i.e., what is $P(O)$) ?
- Problem 3: How to estimate the model parameters (A, B, π) ?

Problem 1: State Estimation



| Given an observation (sequence) $O=\{o^1, o^2, \dots, o^k\}$, what is the most likely state sequence $S=\{s^1, s^2, \dots, s^k\}$ that has produced O ?

| Formally, we need to solve

$$\operatorname{argmax}_S P(S|O)$$

| Or, equivalently,

$$\operatorname{argmax}_S \frac{P(S, O)}{P(O)} = \operatorname{argmax}_S P(S, O)$$

Problem 1: State Estimation (cont'd)

| For a given HMM, we may simplify $P(S, O)$ as

$$P(S, O) = P(O|S)P(S)$$

$$= P(o^1 \dots o^k | s^1 \dots s^k) \prod_{j=1}^k P(s^j | s^1 \dots s^{j-1})$$

$$\simeq P(o^1 \dots o^k | s^1 \dots s^k) \prod_{j=1}^k P(s^j | s^{j-1})$$

$$= \prod_{i=1}^k P(o^i | o^1 \dots o^{i-1}, s^1 \dots s^i) \prod_{j=1}^k P(s^j | s^{j-1})$$

$$\simeq \prod_{i=1}^k P(o^i | s^i) \prod_{j=1}^k P(s^j | s^{j-1}) = \prod_{i=1}^k P(o^i | s^i) P(s^i | s^{i-1})$$

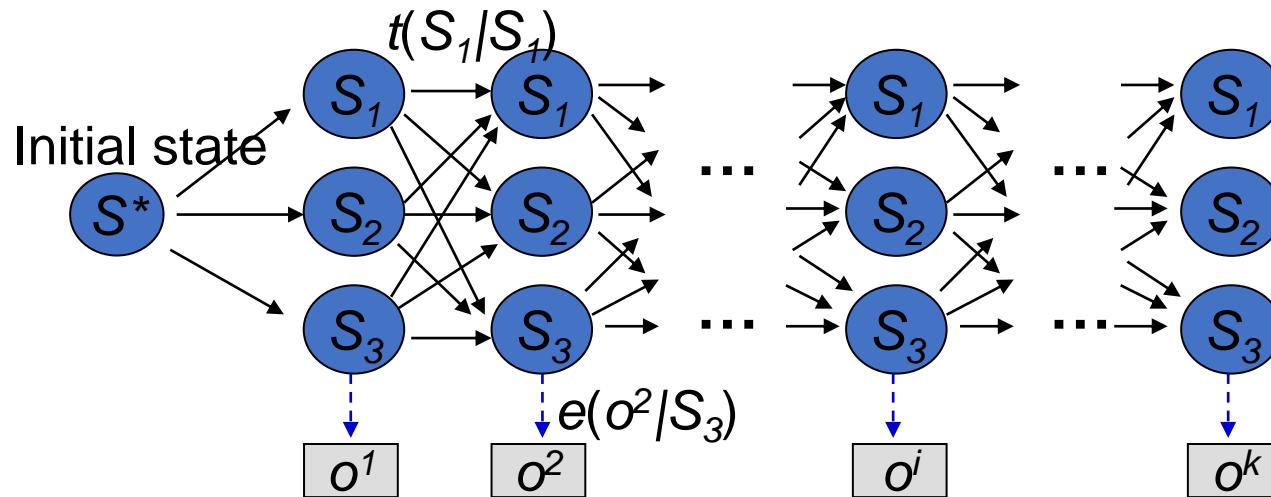
The "Weather" Example

| Let's expand the state space as a trellis, for the earlier example:

S_1 -rain, S_2 -cloudy, S_3 -sunny

-- $t(\cdot|\cdot)$ is the transition probability and $e(\cdot|\cdot)$ the emission probability.

→ To identify a path for which the product of t 's and the e 's is maximized.



Viterbi Algorithm for Problem 1



| A dynamic programming solution

- For each state in the trellis, we record:
 1. $\delta_{s_i}(t)$ is the probability of taking the maximal path up to time $t-1$ ending at state S_i at time t and while generating $o^1 \dots o^t$
 2. $\psi_{s_i}(t)$ is the state sequence that resulted in the maximal probability up to state S_i at time t .

Viterbi Algorithm (cont'd)

| Initialization

$$\delta_{S_i}(1) = t(S_i|s^*)e(o^1|S_i), \quad \forall S_i \in \Theta$$

| Induction:

for $2 \leq t \leq k$, do

$$\delta_{S_i}(t) = \max_{S_j} t(S_i|S_j)e(o^t|S_i)\delta_{S_j}(t-1)$$

$$\psi_{S_i}(t) = \operatorname{argmax}_{S_j} t(S_i|S_j)e(o^t|S_i)\delta_{S_j}(t-1)$$

| Termination:

- The probability of the best state sequence: $\max_{S_j} \delta_{S_j}(k)$
- The best last state: $\hat{s}^k = \operatorname{argmax}_{S_j} \delta_{S_j}(k)$
- Back trace to get other states:

$$\hat{s}^t = \psi_{\hat{s}^{t+1}}(t), \text{ for } t = k-1, \dots, 1.$$

Problem 2: Evaluate $P(O)$



- | To evaluate $P(O)$, we can do $P(O) = \sum_s P(s, o)$
- | From the trellis, a solution can be found by summing the probabilities of all paths generating the given observation sequence.
- | A dynamic programming solution: the forward algorithm or the backward algorithm.

The Forward Algorithm



| Define the forward probability $\alpha_{S_i}(t)$, which is the probability for all paths up to time $t-1$ ending at state S_i at time t and generating $o^1 \dots o^t$.

1. Initialization: $\alpha_{S_i}(1) = t(S_i|s^*)e(o^1|S_i), \quad \forall S_i \in \Theta$

2. Induction:

for $2 \leq t \leq k$, do $\alpha_{S_i}(t) = \sum_{S_j} t(S_i|S_j)e(o^t|S_i)\alpha_{S_j}(t-1)$

3. Termination:

$$P(\mathbf{o}) = \sum_{S_j} \alpha_{S_j}(k)$$

Problem 3: Parameter Learning



| Case 1: we have a set of labeled data – sequences in which we have the `<state, observation>` information

- Use relative frequency for estimating the probabilities
→ the MLE solution

$$t(S_i|S_j) = \frac{\text{number of } (s^t = S_i, s^{t-1} = S_j)}{\text{number of } S_j} \quad e(o_r|S_j) = \frac{\text{number of } (o^t = o_r, s^t = S_j)}{\text{number of } S_j}$$

| Case 2: we have only the observation sequence

- The Forward-Backward Algorithm (a.k.a. Baum-Welch Algorithm): An EM approach.





Unsupervised Learning & Data Clustering

Problem Set-up

Objective



Objective

Define the set-up
of unsupervised
learning

Learning from Unlabeled Data



| Given a training set of n unlabeled samples $\{x^{(i)}\}$

| What can we learn from the samples?

- We could estimate the overall distribution of the data without knowing their label.
- We could figure out the groupings of the samples (if any).
- We could identify some features that may be more important than others.

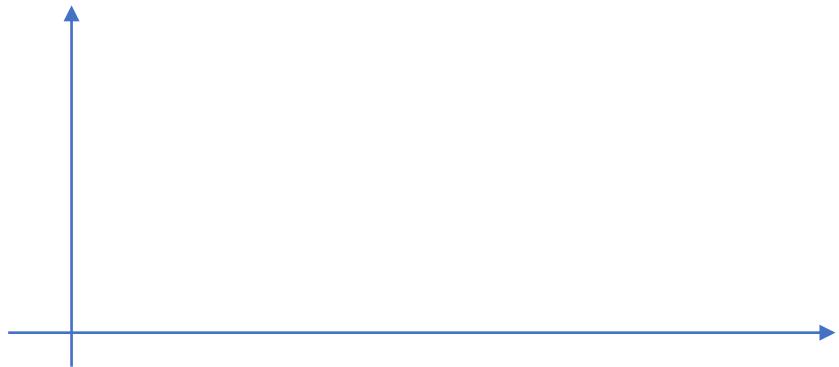
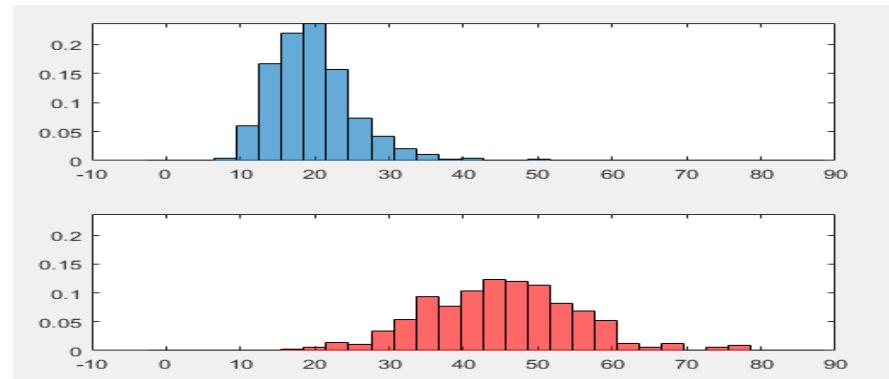
An Example

Illustrating structures/groupings of unlabeled samples may relate to the (unknown) labels of the samples

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

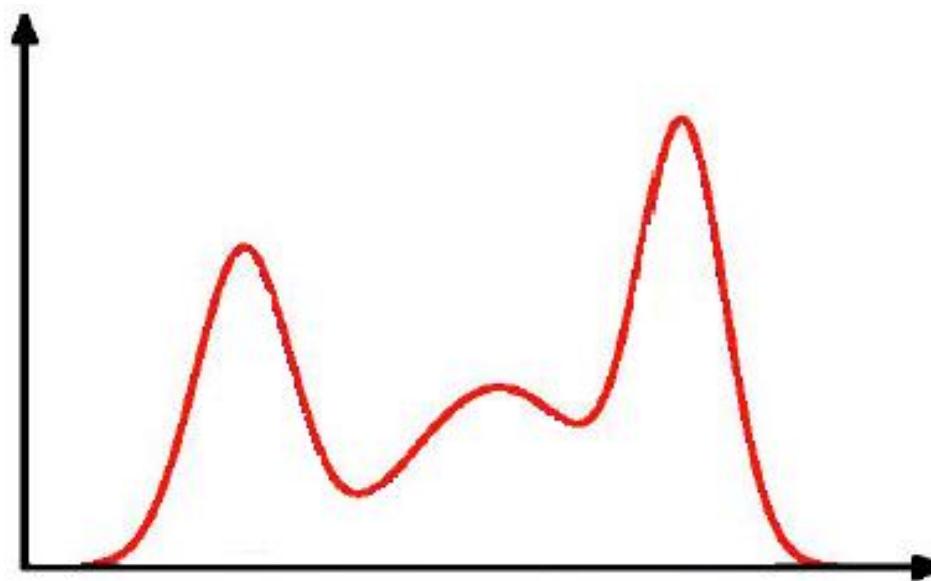
→ If we know the labels, we may find the densities of the classes →

→ What may we see if we have no label for the data samples?



Another Example

- | A density estimated from unlabeled samples may help us to identify densities of different classes
- | If we know there are three classes in the data, each having a normal distribution ...



A Mixture-Density Model



| Assume a parametric model like this:

- The samples come from C classes.
- The prior probabilities $P(\omega_j)$ for each class are known, for $j = 1, \dots, C$.
- The form of $p(\mathbf{x} | \omega_j, \theta_j)$ ($j = 1, \dots, C$) are known.
- The C parameter vectors $\theta_1, \theta_2, \dots, \theta_C$ are unknown.

| Samples from this distribution are given, but the labels of the training samples are *unknown*.

A Mixture-Density Model (cont'd)

| What is the PDF of the unlabeled samples?

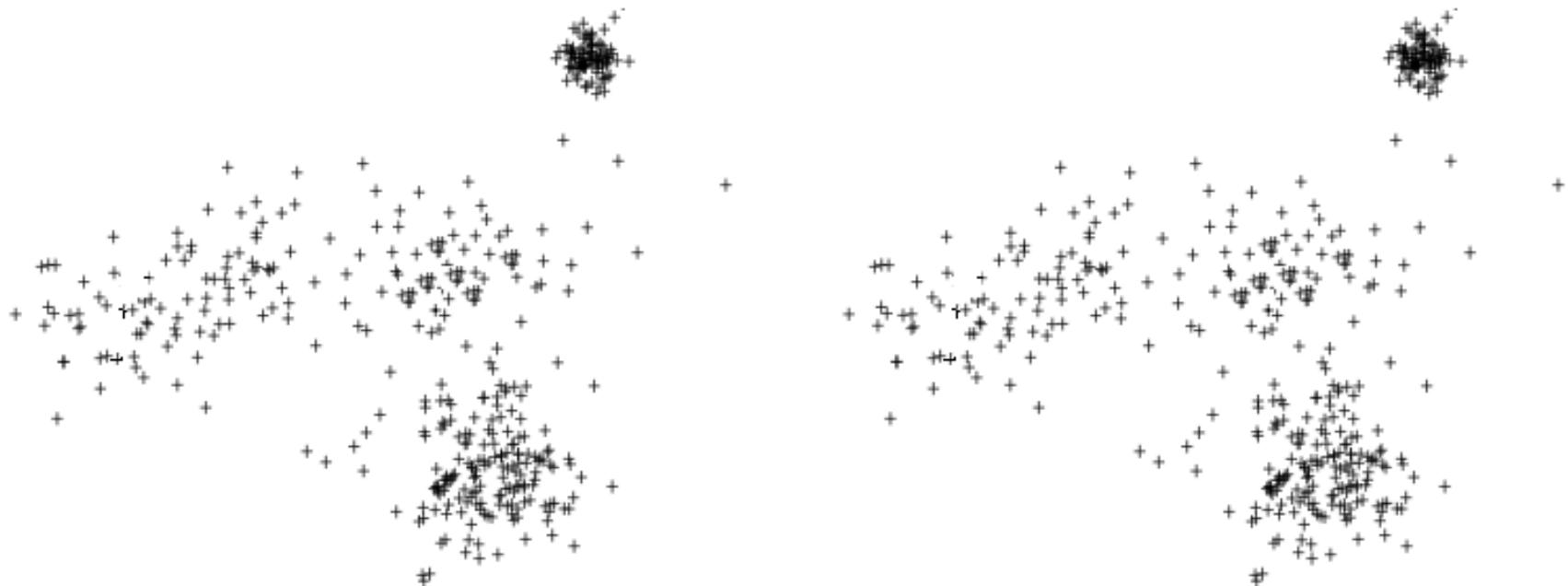
$$p(\mathbf{x} | \boldsymbol{\theta}) = \sum_{j=1}^C p(\mathbf{x} | \omega_j, \boldsymbol{\theta}_j) P(\omega_j)$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots, \theta_C)$

| Can we learn $\boldsymbol{\theta}$ from unlabeled samples from this mixture density?

Illustrating Mixture-Density Model

| An example: with the assumption of 4 classes



Illustrating Mixture-Density Model

| An example: with the assumption of 2 classes



The Question of Identifiability



| Can we learn a unique θ from a set of unlabeled samples from a mixture density?

- For continuous features (with PDFs), the answer is often “Yes”.

| An example in discrete case (with PMF).

- Two coins with $P(\text{head})$ being p & q respectively.
- Randomly pick one and toss it; Record the outcome.
- With only the outcomes of N tosses, but not knowing which coin was used each time (\rightarrow unsupervised), can we figure out p and q ?





Unsupervised Learning

Gaussian Mixture Models and the EM Algorithm

Objective



Objective

Define the Gaussian
Mixture Model



Objective

Illustrate the Expectation-
Maximization Algorithm

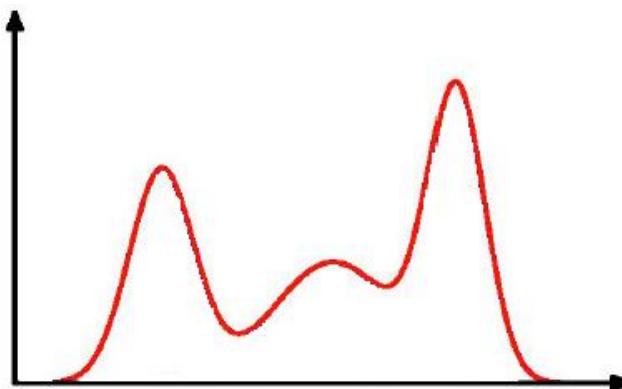
The Gaussian Mixture Models

| The mixture model:

$$p(\mathbf{x} | \boldsymbol{\theta}) = \sum_{j=1}^C p(\mathbf{x} | \omega_j, \boldsymbol{\theta}_j) P(\omega_j)$$

| GMM: each component density is a Gaussian distribution.

- Can be a good approximation to many real data distributions.



If We Do Have Labels...



$$p(\mathbf{x} | \boldsymbol{\theta}) = \sum_{j=1}^C p(\mathbf{x} | \omega_j, \boldsymbol{\theta}_j) P(\omega_j)$$

- | This becomes supervised learning for each component (class).
- | It is more difficult without labels.

Unsupervised Case

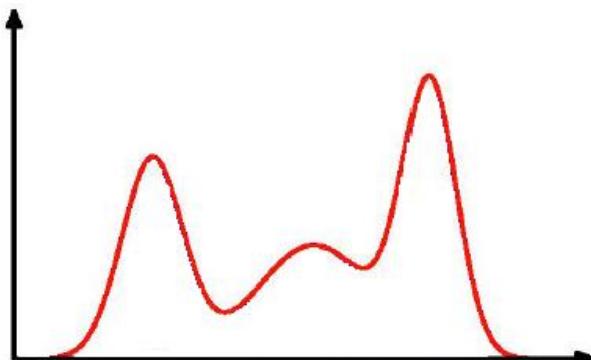
| Consider an iterative method using the *maximum likelihood estimation* concept.

| Consider a 3-component 1-d example.

| What are the parameters in this case?

| We might have some initial (imprecise) guesses for the parameter, e.g., vs the *k-means algorithm*.

- *How to improve the initial guesses?*



Unsupervised Case (cont'd)

Iterate on t

Given parameter estimates at iteration $t-1$

An example of
Expectation-
Maximization
Algorithm.

Step 1. For a sample j , compute its probability of being from class k

$$P[y_j=k|x_j, \theta^{(t-1)}] \propto P_k^{(t-1)} P(x_j | \mu_k^{(t-1)}, \sigma_k^{(t-1)}), \forall k=1, 2, 3$$

Step 2. Update the estimates of the parameters

$$\mu_k^{(t)} = \frac{\sum_j x_j P[y_j=k|x_j, \theta^{(t-1)}]}{\sum_j P[y_j=k|x_j, \theta^{(t-1)}]}$$

$$P_k^{(t)} = \frac{\sum_j P[y_j=k|x_j, \theta^{(t-1)}]}{\text{total # of samples}}$$





Unsupervised Learning

The k-Means Algorithm

Objective



Objective
Discuss the basics of
data clustering

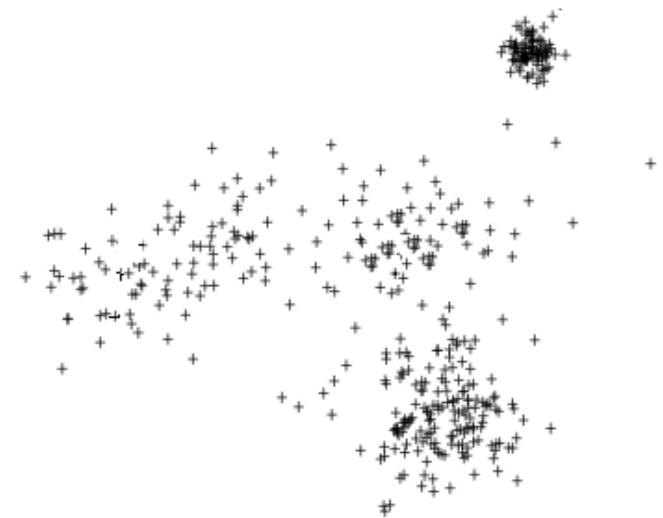


Objective
Illustrate the k-
Means Algorithm

Finding the Clusters/Groupings of the Samples

| A few basic questions to answer

- How to represent the clusters?
 - ➔ We will use the centroid to represent a cluster.
- Which cluster a sample should be assigned to (e.g., membership)?
 - ➔ We will use the similarity to the centroid to determine the membership.
- What similarity measure to use?
 - E.g., Euclidean distance



More on Similarity Measures



| **If we use Euclidean distance as the measure:**

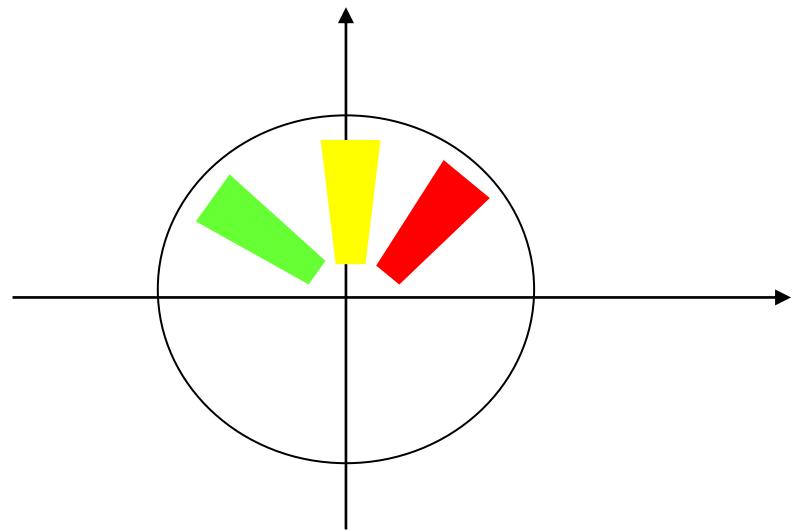
- It is invariant to translations & rotations of the feature space.
- But not to more general transformations.

| **E.g., if one feature is scaled.**

More on Similarity Measures (cont'd)

- | Other types of similarity measures
- | E.g., cosine similarity
- | E.g., distance on a graph, like shortest path.

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^t \mathbf{x}'}{\|\mathbf{x}\| \|\mathbf{x}'\|}$$



Clustering as Optimization

| The sum-of-squared-error criterion/cost

- Let D_i be the subset of samples from class i.
- Let n_i be the number of samples in D_i , and \mathbf{m}_i the mean of those samples

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{x}$$

- The sum of squared error is:

$$J_e = \sum_{i=1}^C \sum_{\mathbf{x} \in D_i} \|\mathbf{x} - \mathbf{m}_i\|^2$$

→ Well-separated, compact data “clouds” tend to give small errors when the clusters coincide with the clouds.



Clustering as Optimization (cont'd)



$$J_e = \sum_{i=1}^C \sum_{\mathbf{x} \in D_i} \|\mathbf{x} - \mathbf{m}_i\|^2$$

- An optimization problem to solve for finding a “good” clustering: to find the partition of the data that minimizes J_e
- If the membership of a sample is determined by the distance to the means \mathbf{m}_i ;
- Then the task is to find the optimal set of $\{\mathbf{m}_i\}$
- The problem is NP-hard.

k-Means Clustering



| Input: Given n data samples

| Goal: Partition them into k clusters/sets D_i , with respective center/mean vectors $\mu_1, \mu_2, \dots, \mu_k$, so as to minimize

$$\sum_{i=1}^k \sum_{x \in D_i} \|x - \mu_i\|^2$$

| Comparing with the mixture models:

- Here we do “hard” assignment of the membership to a sample (simply based on its distance to the cluster center).

The Basic k-Means Algorithm



Given: n samples, a number k.

Begin

 initialize $\mu_1, \mu_2, \dots, \mu_k$ (randomly selected)

do classify n samples according to
 nearest μ_i

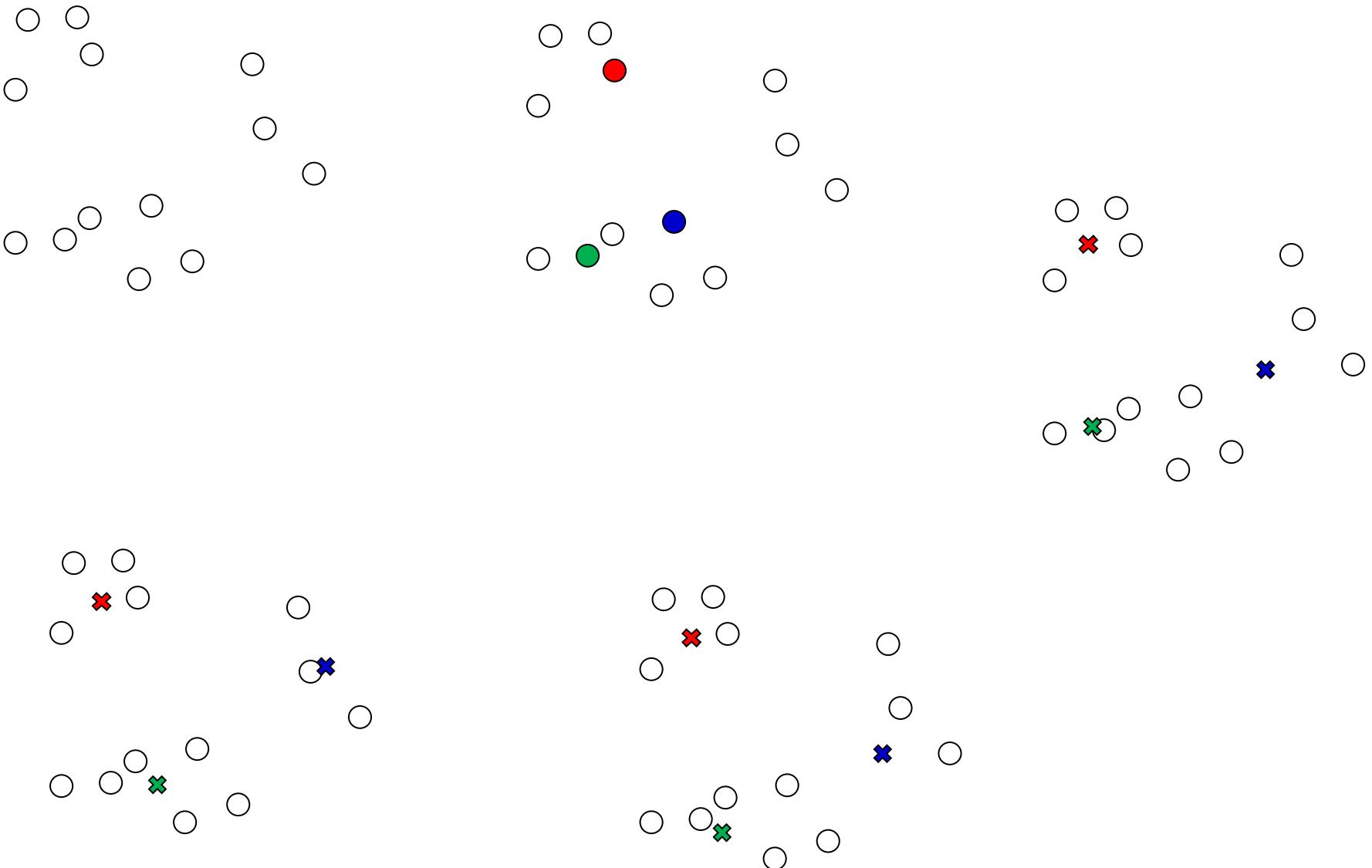
 recompute μ_i

until no change in μ_i

return $\mu_1, \mu_2, \dots, \mu_k$

End

Illustrating the Algorithm







Unsupervised Learning

Analyzing the k-Means Algorithm

Objective



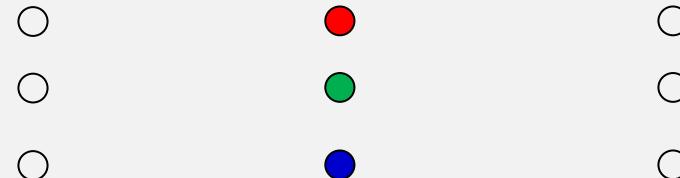
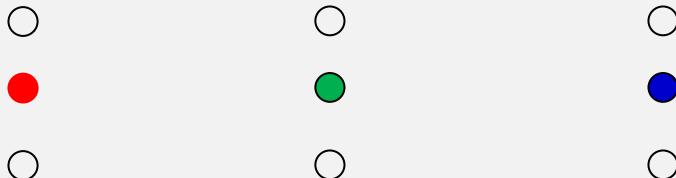
Objective
Discuss the
weaknesses of the k-
means algorithm



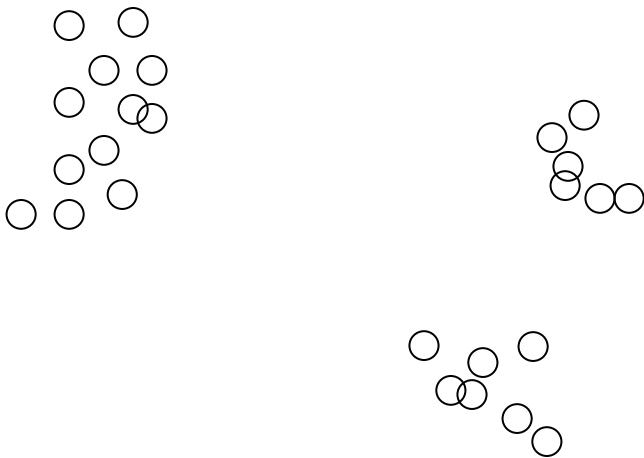
Objective
Discuss a few common
techniques for potential
improvement

Properties of the k-Means Algorithm

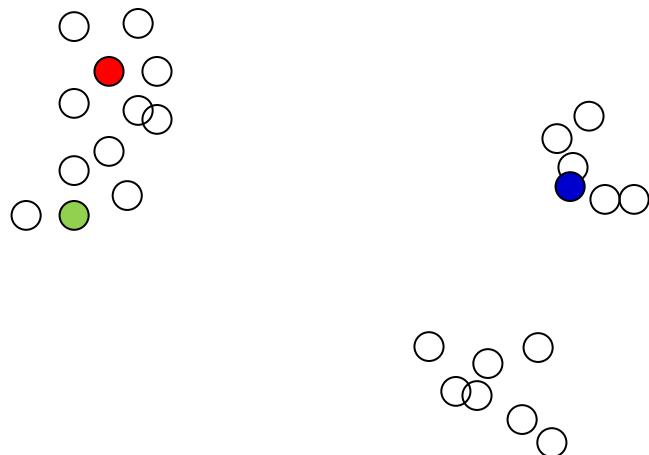
- | The algorithm will converge when the cluster centers no longer change.
- | But the results may not be an optimal solution.
→ Sensitivity to initialization



Another Example



- The natural grouping seems to be so well defined.
- For $k=3$, what will be the clusters?

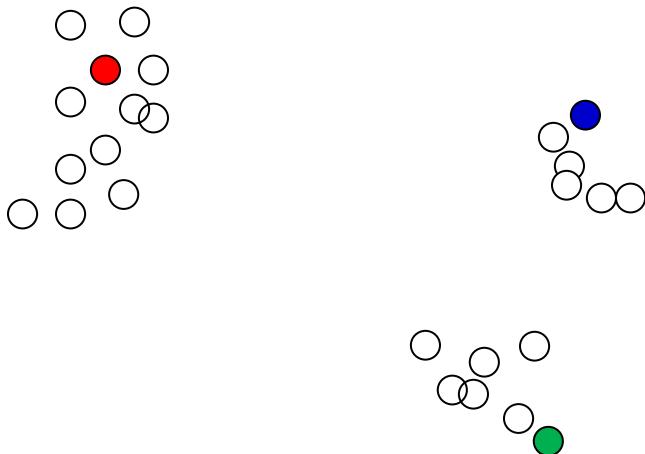


| What can we do to improve?

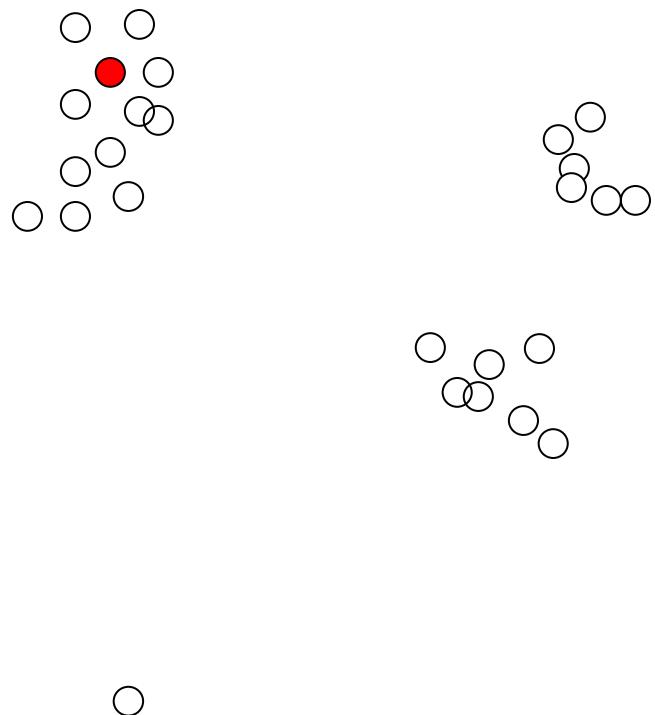
A Few Common “Tricks”

| Choosing the point furthest from the previous centers.

- Drawback: might be sensitive to “outliers”.



| Multiple runs with different initial centers.



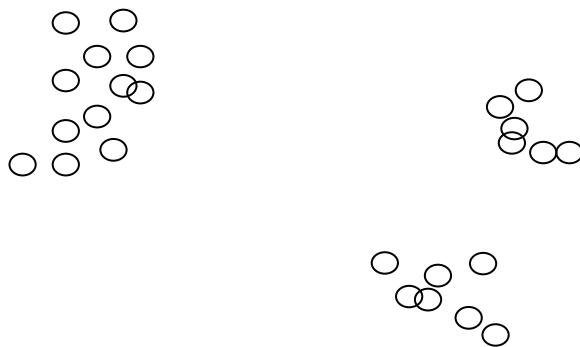
Other Variants of Basic k-Means

| k-Means++:

- New centers are chosen with probabilities (as a function of distance to closest prior centers).
- Kind of between “random” and “furthest point” techniques.

| Hierarchical approaches

- Agglomerative vs divisive.



The Question of Choosing k



| Two trivial extremes

- If $k=1$, the error is the variance of the samples.
- If $k=n$, the error can become 0.

| What is a proper $1 < k < n$ for capturing the structure of the samples?

| Some tricks

- Trick 1: Will the cost function drop dramatically at some point?
- Trick 2: Cross-validation (on, e.g., a classification task)





Spectral Clustering

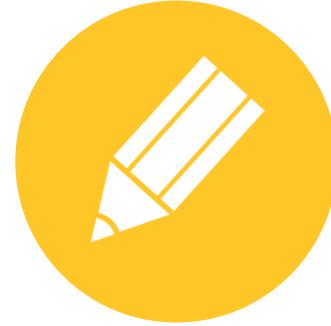
Introduction

Objective



Objective

Illustrate the key idea of
spectral clustering



Objective

Define basic graph
notations useful for spectral
clustering

Revisiting k-means & Mixture Models



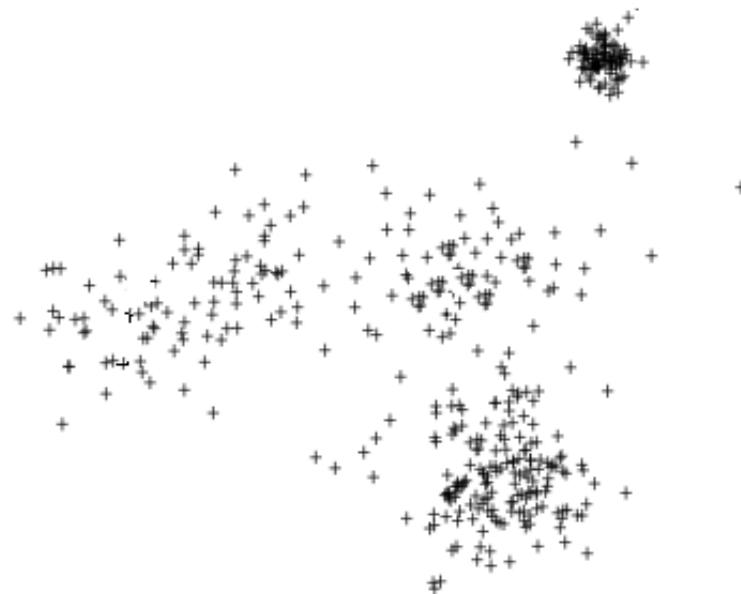
| K-means use “hard” membership while mixture models allow “soft” membership

| Both use feature/vector representation of the data as input → E.g., Euclidean distance is one natural (dis)similarity measure.

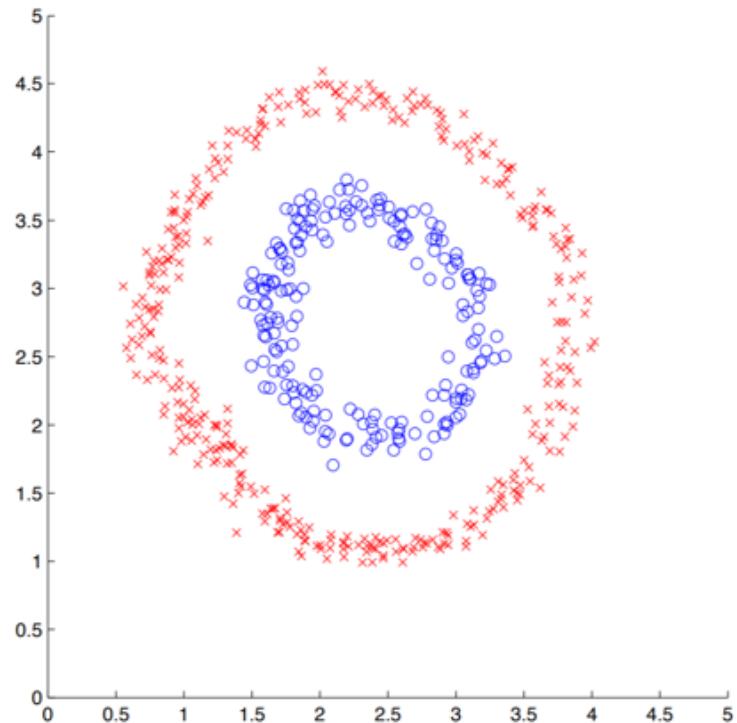
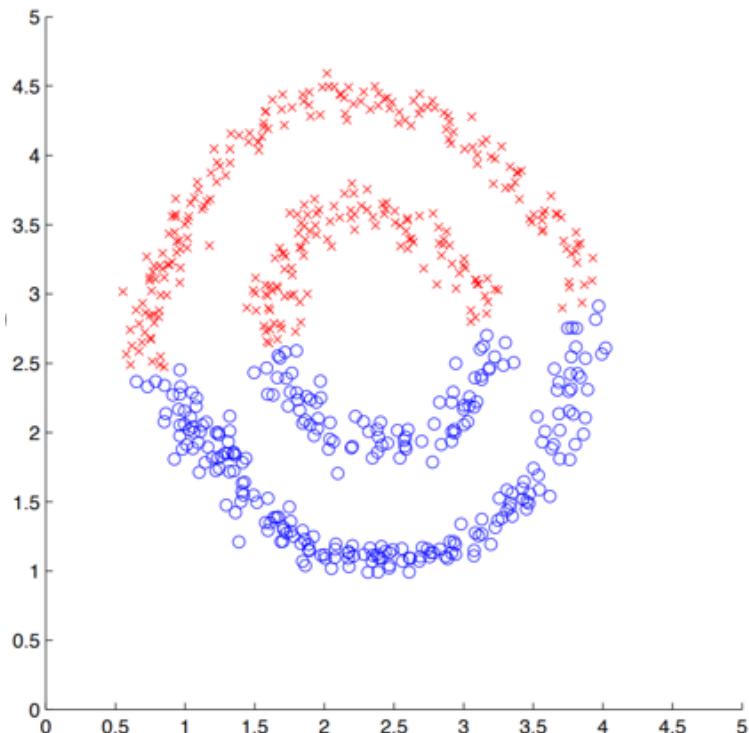
- What if the input data is NOT represented in feature/vector, format?
 - E.g., graph data.
 - E.g., objects with only pair-wise similarities (like individuals on a social network → community detection)

Revisiting k-means & Mixture Models

- | In both k-means and mixture models, we look for compact clustering structures.
- | In some cases, connected-component structures may be more desirable.



Example



Source: Ng, A.Y., Michael I.J., and Yair, W. "On spectral clustering: Analysis and an algorithm." *Advances in neural information processing systems*. 2002.

Spectral Clustering



| A family of methods for finding such similarity-based clusters

- “Spectral”: for using the eigenvalues (spectrum) of the *similarity matrix* of the data.
- Graph clustering, similarity-based clustering

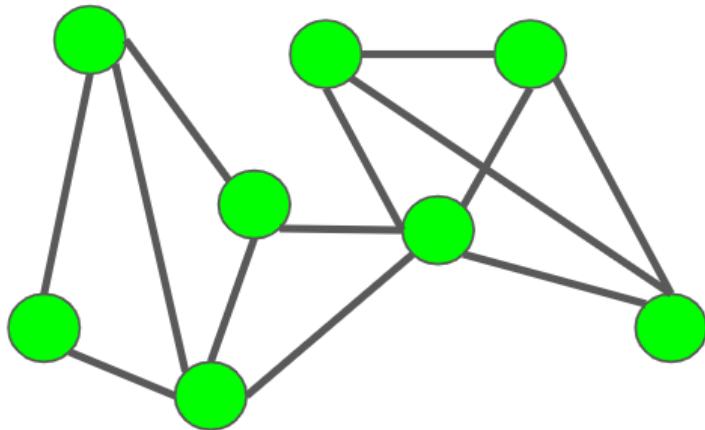
| The objects to be clustered are not in a vector space.

- The primary feature is the similarity between objects.
- For any pair of objects i and j , we have a value $s(i,j)$ measuring their similarity; all such values form the similarity matrix.

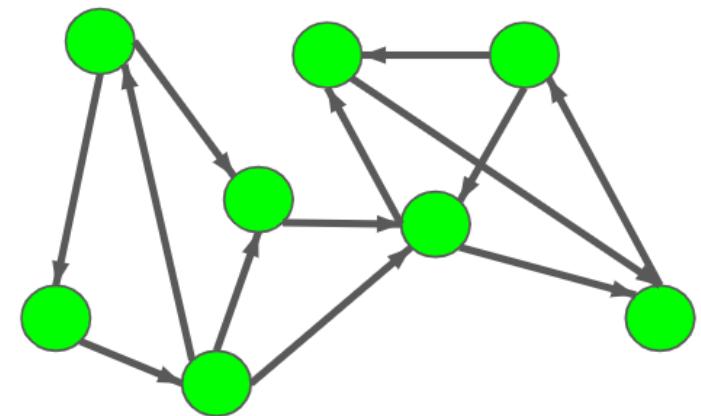
→ **Graphs** are intuitive for representing/visualizing such data.

Graph Representation

| Definition: A graph $G = (V, E)$ is defined by V , a set of N vertices, and E , a set of edges.



Undirected graph



Directed graph

| In spectral clustering, we consider undirected graphs.

Graph Representation (1/4)



| Adjacency matrix W of undirected graph

- $N \times N$ symmetric binary matrix
- The row and columns are indexed by the vertices and the entries represent the edges of the graph

$$\begin{cases} w_{i,j} = 0 & \text{if vertices } i, j \text{ are not connected} \\ w_{i,j} = 1 & \text{if vertices } i, j \text{ are connected} \end{cases}$$

- Simple graph = zero diagonal

Graph Representation (2/4)



| Weighted adjacency matrix (sometimes called affinity matrix)

- Allow values other than 0 or 1
- Each edge is weighted by pairwise similarity

$$\begin{cases} w_{i,j} = 0 & \text{if } i, j \text{ are not connected} \\ w_{i,j} = s(i, j) & \text{if } i, j \text{ are connected} \end{cases}$$

| $w_{i,j}$ may be defined through some kernel functions.

Graph Representation (3/4)

| Degree matrix D of undirected graph

- $N \times N$ diagonal matrix that contains information about the degree of each vertex.
- Degree $d(v_i)$ of a vertex v_i : # of edges incident to the vertex.
 - Extended to sum of weights from edges incident to the vertex.
- So, we have:

$$\mathbf{D} = \begin{bmatrix} d(v_1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & d(v_N) \end{bmatrix}$$

Graph Representation (4/4)



| Laplacian matrix L of undirected graph

- $L = D - W$ (Degree-Affinity) (Unnormalized)
- L is symmetric and positive semi-definite
- N non-negative real-valued eigenvalues
- The smallest eigen-value is 0, the corresponding eigenvector is the 1-vector (all elements being 1).
- The smallest non-zero eigenvalue of L is called the spectral gap.





Spectral Clustering

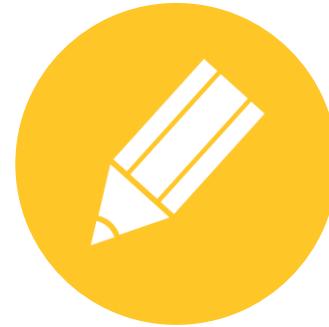
A Graph Cut Formulation

Objective



Objective

Define the graph partition
formulation



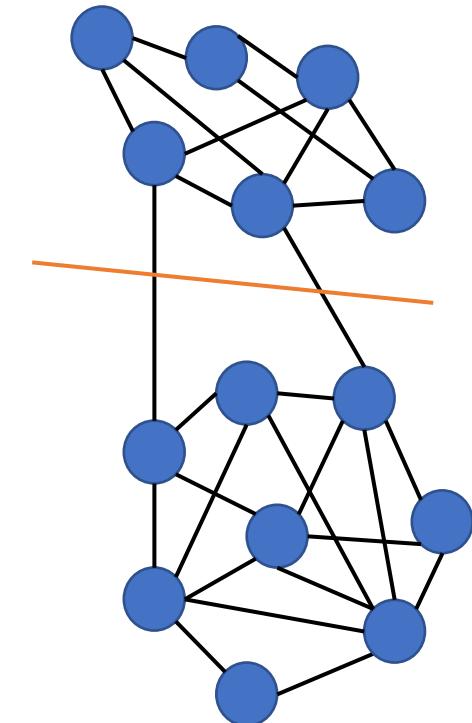
Objective

Learn how to solve simple
graph partition

Clustering as Graph Partition/Cut

- | Find a partition of a graph such that the edges between different groups have a very low weight while the edges within a group have high weight.
- | E.g., minimum cut
- | More general, consider weighted edges.

CutSize = 2



2-way Spectral Graph Partitioning

| Weighted adjacency matrix \mathbf{W}

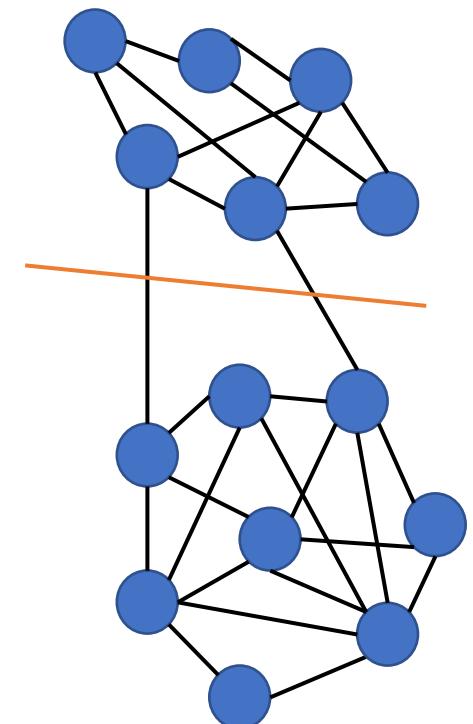
– $w_{i,j}$: the weight between two vertices i and j

| (Cluster) Membership vector \mathbf{q}

$$q_i = \begin{cases} 1 & i \in \text{Cluster } A \\ -1 & i \in \text{Cluster } B \end{cases}$$

$$\mathbf{q} = \underset{\mathbf{q} \in [-1, 1]^n}{\operatorname{argmin}} \text{CutSize},$$

$$\text{CutSize} = J = \frac{1}{4} \sum_{i,j} (q_i - q_j)^2 w_{i,j}$$



Solving the Optimization Problem



$$\mathbf{q} = \underset{\mathbf{q} \in [-1, 1]^n}{\operatorname{argmin}} \frac{1}{4} \sum_{i,j} (q_i - q_j)^2 w_{i,j},$$

- | Directly solving the above problem requires combinatorial search → exponential complexity
- | How to reduce the computational complexity?

Relaxation Approach



| Key difficulty: q_i has to be either -1,1.

- Relax q_i to be any real number.
- Impose Constraint: $\sum_{i=1}^n q_i^2 = n$

$$J = \frac{1}{4} \sum_{i,j} (q_i - q_j)^2 w_{i,j} = \frac{1}{4} \sum_{i,j} (q_i^2 - 2q_i q_j + q_j^2) w_{i,j}$$

$$= \frac{1}{4} \sum_i 2q_i^2 (\sum_j w_{i,j}) - \frac{1}{4} \sum_{i,j} 2q_i q_j w_{i,j}$$

$$= \frac{1}{2} \sum_i q_i^2 d_i - \frac{1}{2} \sum_{i,j} q_i (d_i \delta_{i,j} - w_{i,j}) q_j$$

where $d_i = \sum_j w_{i,j}$ and $D \equiv [d_i \delta_{i,j}]$

$$\rightarrow J = \frac{1}{2} \mathbf{q}^T (\mathbf{D} - \mathbf{W}) \mathbf{q}$$

Relaxation Approach (cont'd)

| The final problem formulation:

$$\mathbf{q} = \underset{\mathbf{q}}{\operatorname{argmin}} J = \underset{\mathbf{q}}{\operatorname{argmin}} \mathbf{q}^T (\mathbf{D} - \mathbf{W})\mathbf{q} ,$$

$$\text{subject to } \sum_{i=1}^n q_i^2 = n$$

| Solution: the second minimum eigenvector for
 $\mathbf{D} - \mathbf{W}$

$$(\mathbf{D} - \mathbf{W})\mathbf{q} = \lambda_2 \mathbf{q}$$

Graph Laplacian



| $L = D - W$

| L is semi-positive definite matrix.

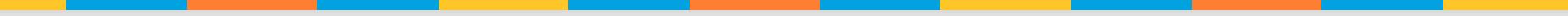
- For any x , we have $x^T L x \geq 0$. (Why?)

| Minimum eigenvalue $\lambda_1 = 0$ (what is the eigenvector?)

$$0 = \lambda_1 < \lambda_2 < \lambda_3 \dots < \lambda_k$$

| The eigenvector that corresponds to the second minimum eigenvalue λ_2 gives the best bipartite graph partition.

Recovering the Partitions

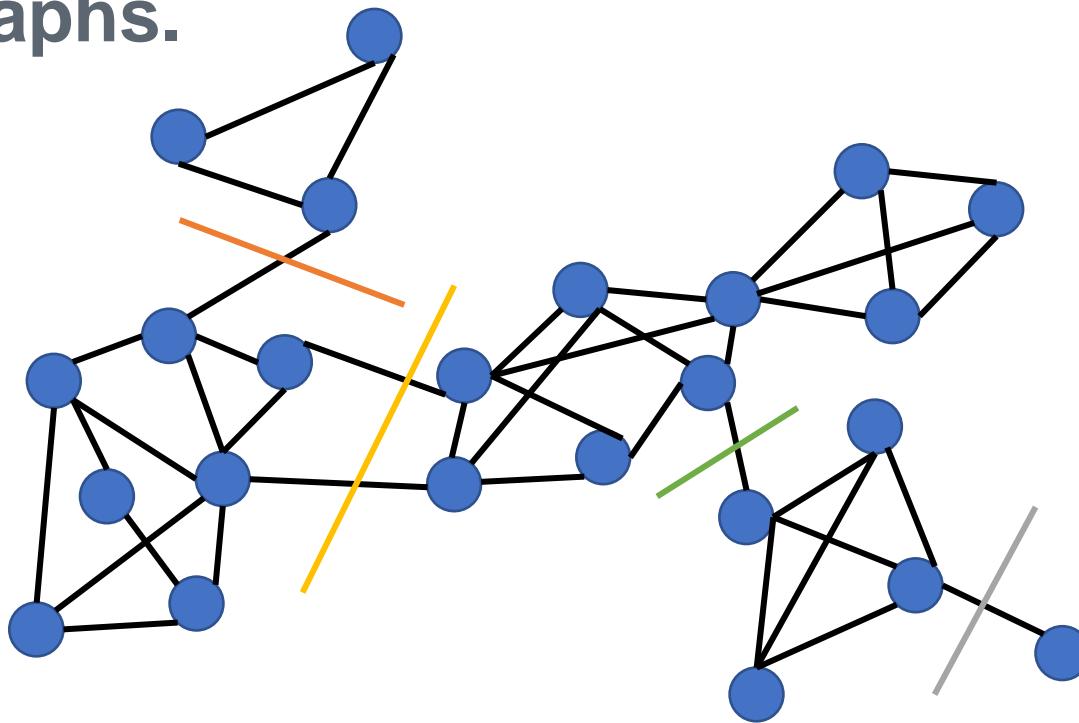


- | Due to the relaxation, q can be any number (not just -1 and 1)
- | How to construct the partition based on the eigenvector?
- | A simple strategy :

$$A = \{i | q_i < 0\}, \quad B = \{i | q_i \geq 0\}$$

One Obvious Drawback

| Minimum cut does not balance the size of bipartite graphs.



| How should we consider other factors like the sizes of the partitions?



Spectral Clustering

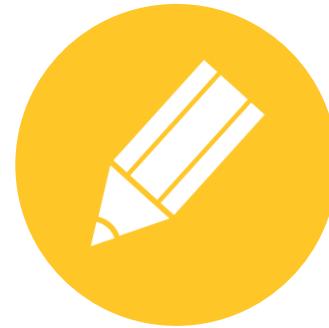
Going Beyond MinCut

Objective



Objective

Discuss several graph cut
approaches



Objective

Illustrate the algorithm
through an example

MinCut



| In MinCut, we used the following objective function:

$$J_{MinCut} = Cut(A, B)$$

| We noted one drawback of MinCut: the sizes of the partitions are not considered.

| A few extensions exist.

Characterizing Graph Cut

| $Cut(A, B) = \sum_{i \in A, j \in B} w_{ij}$ e.g., $Cut(A, B) = 0.3$

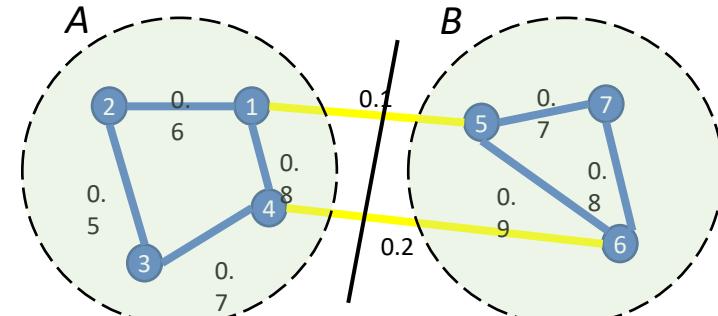
| $Cut(A, A) = \sum_{i \in A, j \in A} w_{ij}$ e.g., $Cut(A, A) = 2.6$

| $Cut(B, B) = \sum_{i \in B, j \in B} w_{ij}$ e.g., $Cut(B, B) = 2.4$

| $Vol(A) = \sum_{i \in A} \sum_{j=1}^n w_{ij}$ e.g., $Vol(A) = 5.5$

| $Vol(B) = \sum_{i \in B} \sum_{j=1}^n w_{ij}$ e.g., $Vol(B) = 5.1$

| $|A| = 4, |B| = 3$



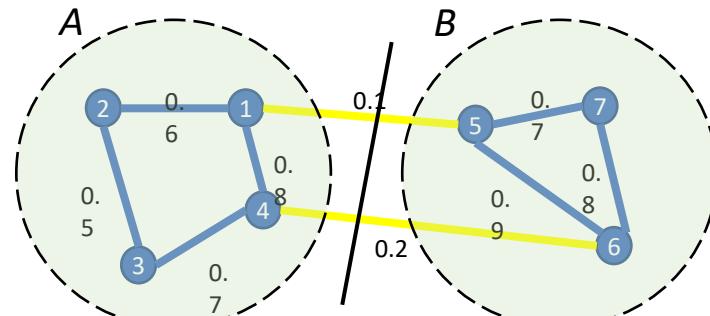
The Ratio Cut Method

| The Objective function:

$$J_{RatioCut}(A, B) = Cut(A, B) \left(\frac{1}{|A|} + \frac{1}{|B|} \right)$$

| Attempts to produce balanced clusters.

Example: $J_{RatioCut}(A, B) = \frac{7}{40}$



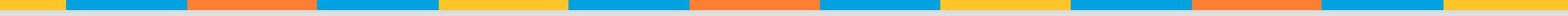
The Ratio Cut Method (cont'd)

| Similar to MinCut, the solution can be found by the following generalized eigenvalue problem:

$$(\mathbf{D} - \mathbf{W})\mathbf{q} = \lambda \mathbf{D}\mathbf{q}$$

$$\mathbf{L}\mathbf{q} = \lambda \mathbf{D}\mathbf{q}$$

Normalized Cut (NCut)

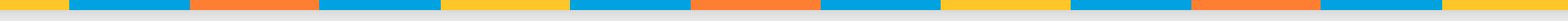


- | In Ratio Cut, the balance of the partitions is defined based on the number of vertices.
- | We may consider the “size” of a set based on weights of its edges → Ncut
- | The objective function is:

$$J_{NCut}(A, B) = Cut(A, B) \left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)} \right)$$

Example: $J_{NCut}(A, B) = 0.1134$

Additional Considerations



| In clustering, we should also consider within-cluster connections.

| A good partition should consider

- Inter-cluster connections, and
- Intra-cluster connections.

MinMaxCut



- | 1st constraint: inter-connection should be minimized: $MinCut(A, B)$
- | 2nd constraint: intra-connection should be maximized : $MaxCut(A, A)$ and $MaxCut(B, B)$
- | These requirements may be simultaneously satisfied by minimizing the objective function:

$$J_{MinMaxCut}(A, B) = Cut(A, B) \left(\frac{1}{Cut(A, A)} + \frac{1}{Cut(B, B)} \right)$$

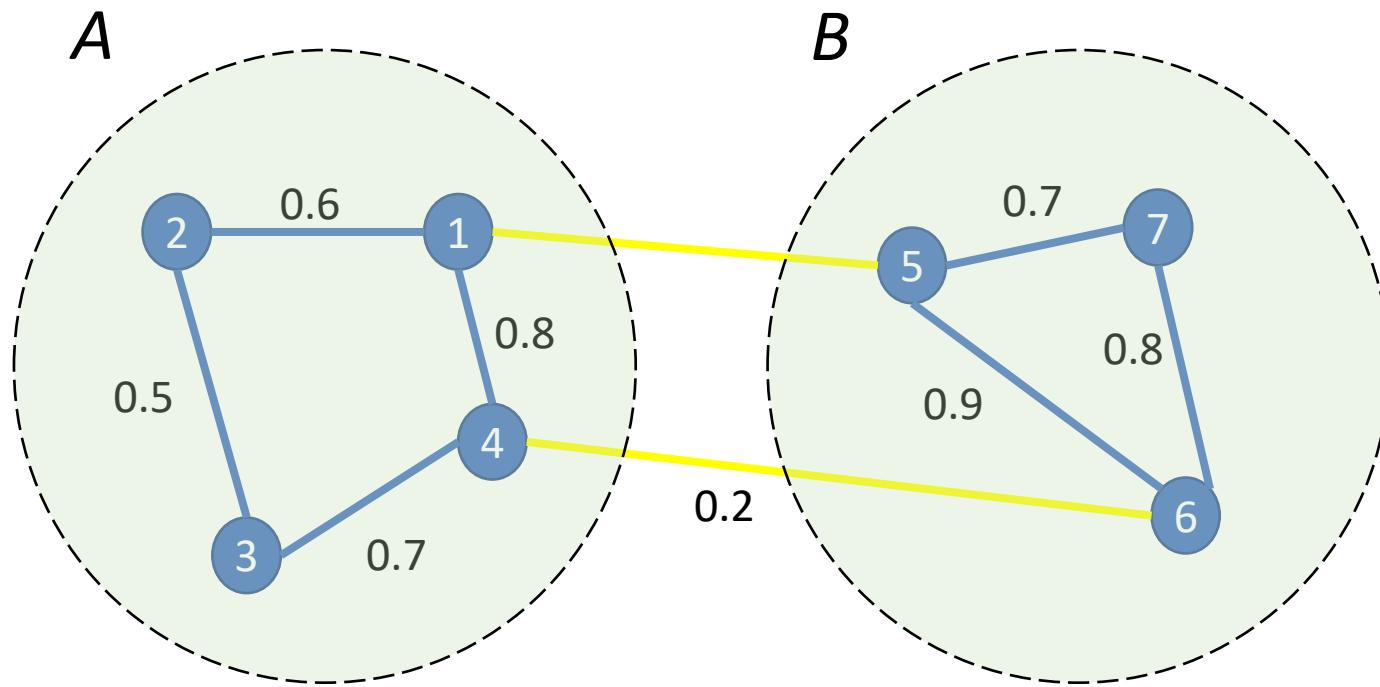
Example: $J_{MinMaxCut}(A, B) = 0.240$

Normalized and MinMaxCut Methods

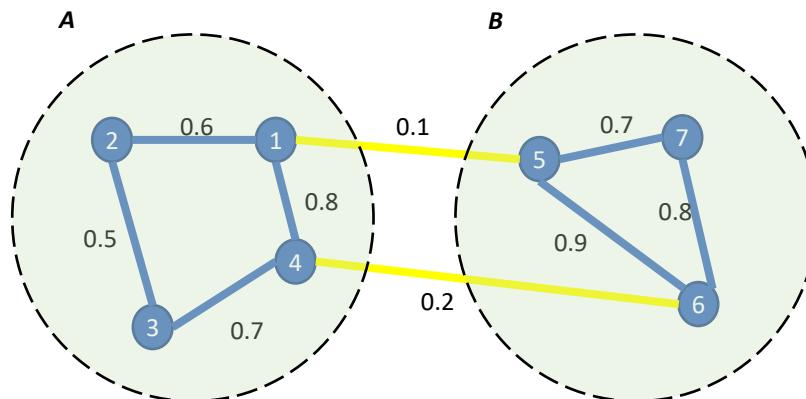


- | Similar to before, we may relax the indicator vector q to real values.
- | For both NCut and MinMaxCut, the solution may be found by solving generalized eigenvalue problems.

An Illustrative Example

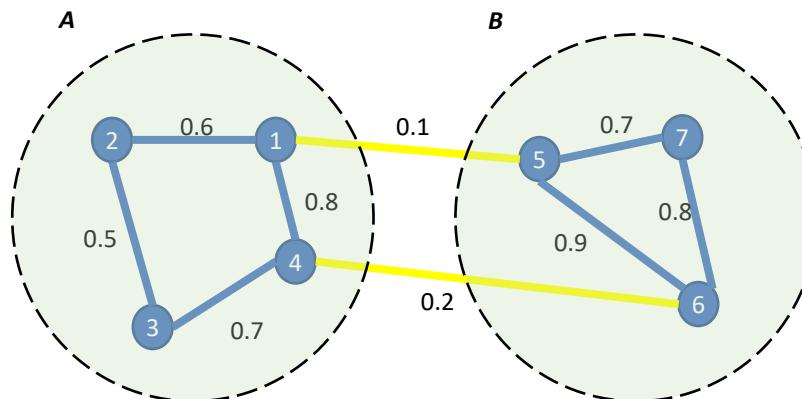


Graph and Similarity Matrix



	x1	x2	x3	x4	x5	x6	x7
x1	0	0.6	0	0.8	0.1	0	0
x2	0.6	0	0.5	0	0	0	0
x3	0	0.5	0	0.7	0	0	0
x4	0.8	0	0.7	0	0	0.2	0
x5	0.1	0	0	0	0	0.9	0.7
x6	0	0	0	0.2	0.9	0	0.8
x7	0	0	0	0	0.7	0.8	0

Graph and Laplacian Matrix

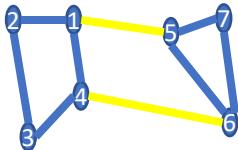


	x1	x2	x3	x4	x5	x6	x7
x1	1.5	-0.6	0	-0.8	-0.1	0	0
x2	-0.6	1.1	-0.5	0	0	0	0
x3	0	-0.5	1.2	-0.7	0	0	0
x4	-0.8	0	-0.7	1.7	0	-0.2	0
x5	-0.1	0	0	0	1.7	-0.9	-0.7
x6	0	0	0	-0.2	-0.9	1.9	-0.8
x7	0	0	0	0	-0.7	-0.8	1.5

Solve Eigen Problem

Pre-processing

- Build Laplacian matrix L of the graph.



$\Lambda =$

0
0.1588
1.2705
1.3692
2.2751
2.6238
2.9027

0.378	-0.2962	0.3027	-0.6041	0.0429	0.3638	-0.4226
0.378	-0.3805	0.6392	0.4487	0.0125	-0.233	0.2192
0.378	-0.3608	-0.5812	0.4834	0.0221	0.2736	-0.2832
0.378	-0.2649	-0.398	-0.4373	0.0429	-0.3899	0.5323
0.378	0.4298	0.0443	0.0159	0.6004	0.4291	0.3544
0.378	0.406	-0.0317	0.0012	0.2174	-0.6116	-0.5196
0.378	0.4665	0.0247	0.0923	0.7667	0.1681	0.1195

Find

- Eigenvalues Λ and eigenvectors x of matrix L .
- Map vertices to the corresponding components of the 2nd eigenvector.

x1	-0.2962
x2	-0.3805
x3	-0.3608
x4	-0.2649
x5	0.4298
x6	0.406
x7	0.4665



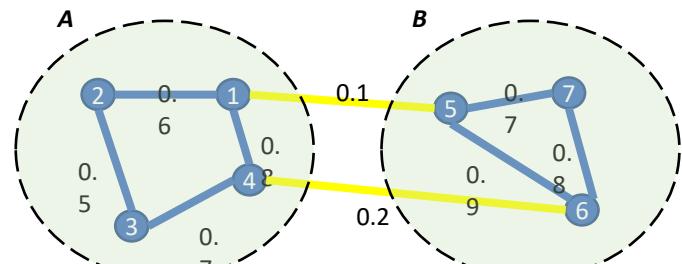
Spectral Clustering

x1	-0.2962
x2	-0.3805
x3	-0.3608
x4	-0.2649
x5	0.4298
x6	0.406
x7	0.4665

Split at value 0
Cluster A: Negative points
Cluster B: Positive Points



x1	-0.2962	X5	0.4298
x2	-0.3805	X6	0.406
x3	-0.3608	X7	0.4665
x4	-0.2649		

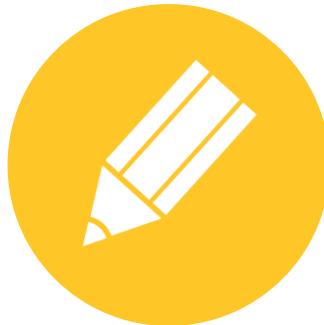




Spectral Clustering

Practical Considerations in Implementation

Objective

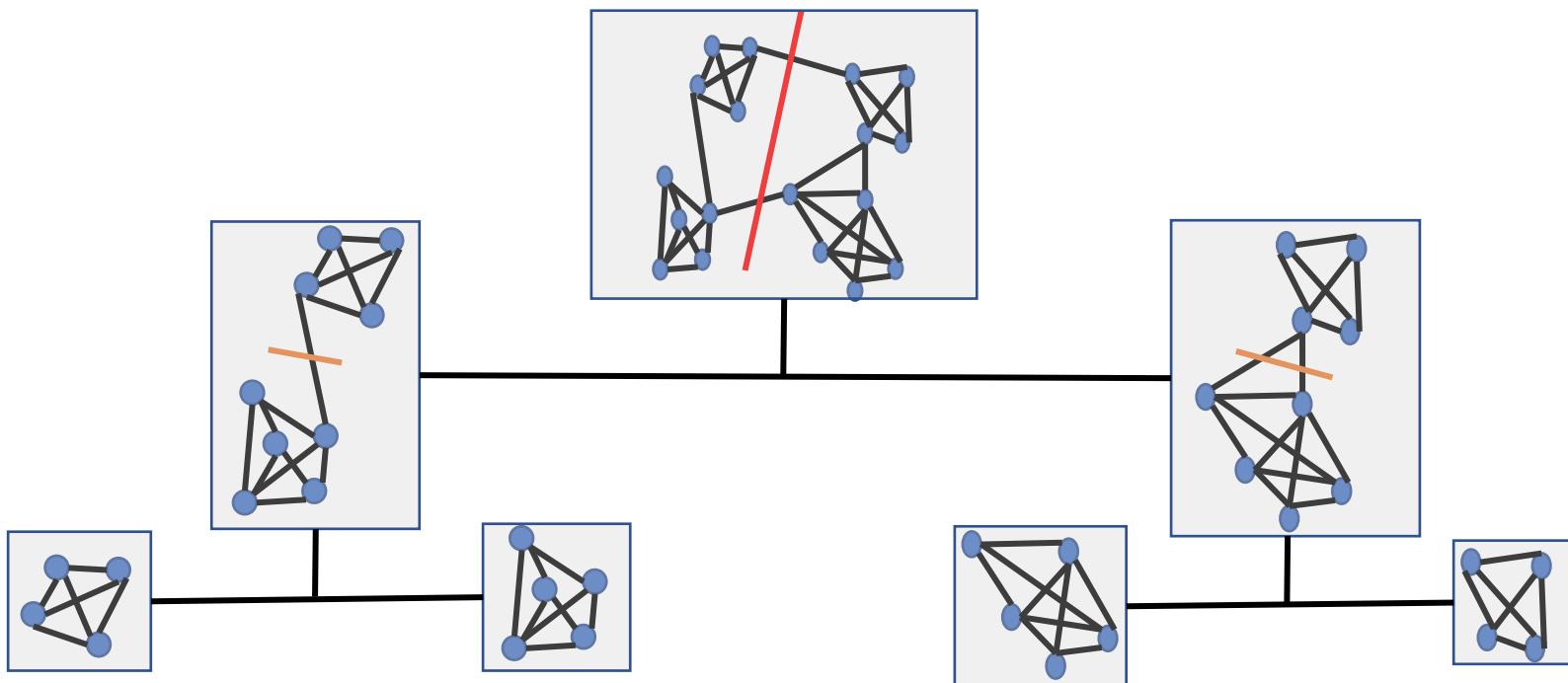


Objective

Discuss several
practical
implementation issues

Recursive Bi-partitioning

- | Recursively apply bi-partitioning algorithm in a hierarchical divisive manner.
- | Disadvantages: inefficient, stability issues.



K-way Graph Cuts



| Generalizing the 2-way objective functions :

$$J_{RatioCut}(A_1, \dots, A_k) = \sum_{i=1}^k \frac{Cut(A_i, \overline{A_i})}{|A_i|}$$

$$J_{NCut}(A_1, \dots, A_k) = \sum_{i=1}^k \frac{Cut(A_i, \overline{A_i})}{Vol(A_i)}$$

$$J_{MinMaxCut}(A_1, \dots, A_k) = \sum_{i=1}^k \frac{Cut(A_i, \overline{A_i})}{Cut(A_i, A_i)}$$

Implementation Considerations (1/4)



| **Preprocessing:** spectral clustering methods can be interpreted as tools for analysis of the block structure of the similarity matrix.

- Building such matrices may certainly ameliorate the results.

| **When building graphs from real data**

- Calculation of the similarity matrix is not evident.
- Choosing the similarity function can highly affect the results of the following steps.
- A Gaussian kernel is often chosen, but other similarities like cosine similarity might be proper for specific applications.

Implementation Considerations (2/4)

| Graph and similarity matrix construction:
Laplacian matrices are generally chosen to be
positive and semi-definite thus their eigenvalues
will be non-negatives.

– A few variants

Unnormalized	$L = D - W$
symmetric	$L_{Sy} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$
Asymmetric	$L_{As} = D^{-1} L = I - D^{-1} W$

Implementation Considerations (3/4)



| Computing the eigenvectors.

- Efficient methods exist for sparse matrices.

| Different ways of building the similarity graphs

- ε -neighborhood graph.
- k-nearest neighbor graph.
- fully connected graph.

Implementation Considerations (4/4)



| Choosing k :

- Similar to k-means, there are many heuristics to use.
- The eigengap heuristic: to choose a k such that first k eigenvalues are very small but the $(k+1)$ th one is relatively large.

| Clustering: simple algorithms other than k-means can be used in the last stage, such as simple linkage, k-lines, elongated k-means, mixture model, etc.

Recap: Pros and Cons of Spectral Clustering



Advantages:

- Does not make strong assumptions on the forms of the clusters.
- Easy to implement, and can be implemented efficiently even for large data sets as long as the similarity graph is sparse.
- Good clustering results.
- Reasonably fast for sparse data sets of several thousand elements.

Disadvantages:

- May be sensitive to choice of parameters for neighborhood graph.
- Computationally expensive for large datasets.





Dimensionality Reduction

Introduction

Objective



Objective

Illustrate the need for
dimensionality
reduction

What is Dimensionality Reduction?



| **We have N data points in a high-dimensional space,**

- e.g., in the order of tens of thousands of dimensions.

| **We want to project them into some low-dimensional space,**

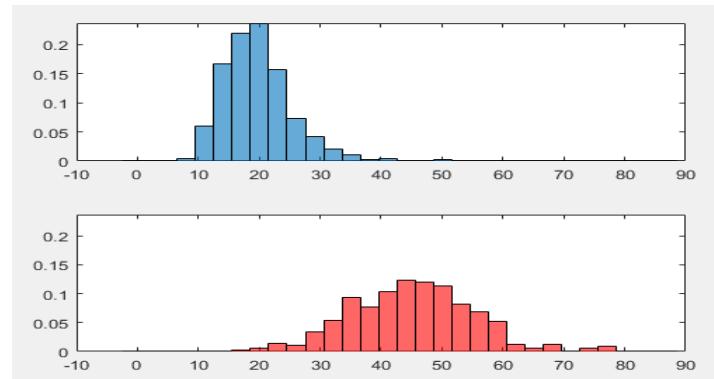
- e.g., in the order of tens of dimensions.

| **Why dimensionality reduction?**

- A key technique to mitigate *curse of dimensionality*

The Curse of Dimensionality

| Consider histogram as a density estimator.



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

| Exponentially more samples would be needed in higher-dimensional spaces for the same “resolution”.

Many Techniques for Dimensionality Reduction



| **Many ways for going from a higher-dimensional space to a lower-dimensional space.**

- Feature Selection achieves this by keeping only a subset of the original features/dimension.

| **There are many other techniques, employing a feature mapping/projection approach.**

- New features are generated (instead of selecting only from the original features).
- The underlying assumptions and/or goals of the techniques are often different.

Examples of Feature Mapping



- | Linear discriminant analysis (LDA)
- | Independent component analysis (ICA)
- | Non-negative matrix factorization (NMF)
- | Auto-encoder
- | Self-organizing maps
- | Principal component analysis (and its variants)





Dimensionality Reduction

Principal Component Analysis: Basic Idea

Objective

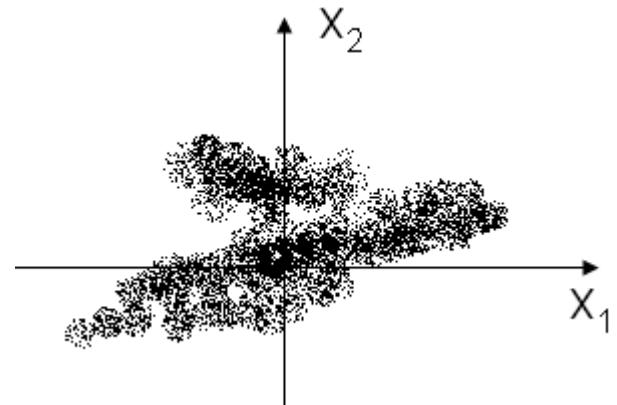


Objective

Illustrate the basic idea
of Principal Component
Analysis

Principal Component Analysis: Basic Idea

| Look at a simple 2-D to 1-D example: we want to use a single feature to describe the 2-D samples



| Consider these possibilities

- Naïve: randomly discard one dimension
- Better: discard the less-descriptive one (x_2 in the figure)
- Much better: project the data to a most-descriptive direction and use the projections.

How to Formulate this Idea?



| “Most descriptive” \approx Largest “variance”

| So the problem is to find the direction of the largest variance.

Problem

Given n samples $D=\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ in d -dimensional space, find a direction \mathbf{e}_1 , such that the projection of D onto \mathbf{e}_1 gives the largest variance (compared with any other direction).

\mathbf{e}_1 is a d -dimensional vector with unit norm.

Find \mathbf{e}_1



| Let's compute the variance of the projected data on a given direction \mathbf{e} .

- The n projected samples are given as, for $i = 1, \dots, n$,

$$y_i = \mathbf{x}_i \cdot \mathbf{e}$$

- The mean of the projections:

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \cdot \mathbf{e} = \bar{\mathbf{x}} \cdot \mathbf{e}$$

- Thus the (sample) variance of the projections:

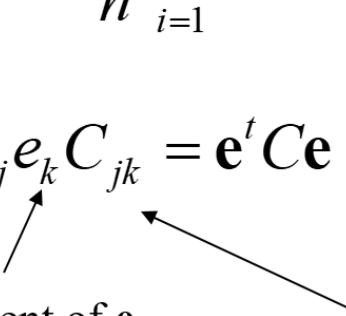
$$\sigma^2(\mathbf{e}) = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2 = \frac{1}{n} \sum_{i=1}^n [(\mathbf{x}_i - \bar{\mathbf{x}}) \cdot \mathbf{e}]^2$$

n vs $n-1$

Find e_1 (cont'd)

| Expand the previous expression

$$\begin{aligned}\sigma^2(\mathbf{e}) &= \sum_{j=1}^d \sum_{k=1}^d e_j e_k \left[\frac{1}{n} \sum_{i=1}^n (x_{i,j} - \bar{x}_{i,j})(x_{i,k} - \bar{x}_{i,k}) \right] \\ &= \sum_{j=1}^d \sum_{k=1}^d e_j e_k C_{jk} = \mathbf{e}^t C \mathbf{e}\end{aligned}$$



k-th component of \mathbf{x}_i

(j,k)-th element of the matrix C

k-th component of \mathbf{e}

| C is the sample covariance matrix.

Find \mathbf{e}_1 (cont'd)

| To find \mathbf{e}_1 , we can do

$$\mathbf{e}_1 = \arg \max_{\mathbf{e}} \sigma^2(\mathbf{e}) \quad \text{subject to } \|\mathbf{e}\|=1$$



what if without this constraint?

| Constrained maximization: use Lagrange multiplier method.

$$\text{maximize } F(\mathbf{e}) = \mathbf{e}^t C \mathbf{e} - \lambda (\mathbf{e}^t \mathbf{e} - 1)$$



Lagrange multiplier

Find e_1 (cont'd)

| Set the partial derivative to 0, we have

$$\frac{\partial F}{\partial e} = 2Ce - 2\lambda e = 0$$

$$\rightarrow Ce = \lambda e$$

| The solution is an eigenvector of C , with eigenvalue λ , which is also the variance under e :

$$\sigma^2(e) = e^t Ce = \lambda$$

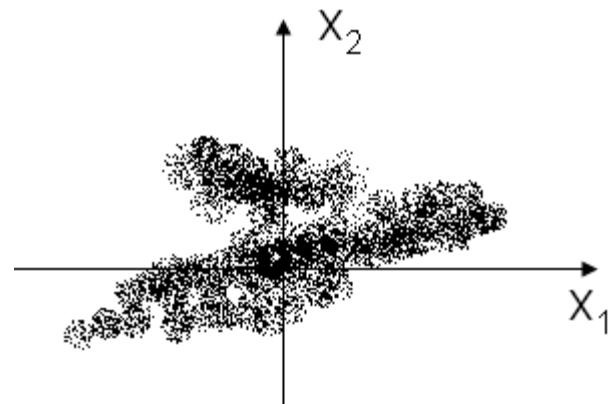
| We should set e_1 to be the eigenvector corresponding to the largest eigenvalue λ_1 .

Recap of the Key Idea

| We want to project the given data samples to certain direction so that the variance is maximized, compared with any other direction.

| We figured out what this optimal direction e_1 should be:

- It should be the eigenvector of corresponding to the largest eigenvalue λ_1 , of the covariance matrix.







Dimensionality Reduction

Principal Component Analysis: The Algorithm & Important Properties

Objective



Objective

Implement the PCA
algorithm



Objective

Discuss some important
properties of PCA

Principal Components



| We found e_1 , which gives the direction of the largest variance after projection

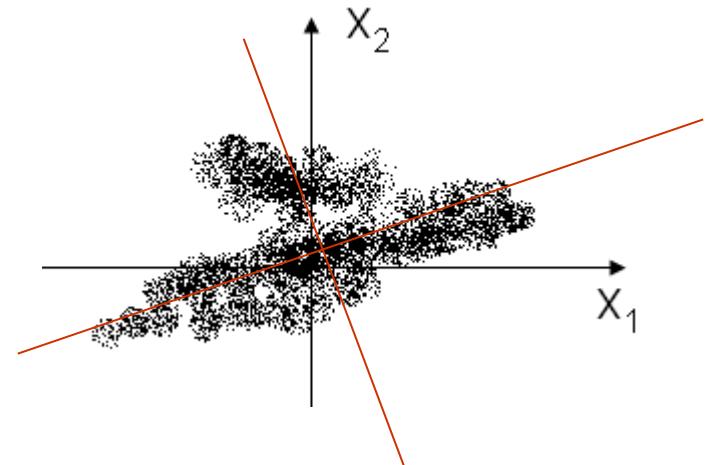
- The first principal component.

| The process can be continued in the subspace orthogonal to e_1 , and so on and so forth.

- Obtaining other principal components: e_2, e_3 , etc., corresponding to other eigenvectors of C , ordered by the corresponding eigenvalues λ_i

Principal Components (cont'd)

| The principal components are orthogonal to each other → $\{e_i\}$ forms an orthonormal basis in the d -dimensional space.



| The total variance is given by the sum of the variances of the projections.

$$\sigma^2 = \sum_{j=1}^d \lambda_j$$

How Many Principal Components to Keep?



- | To reduce dimensions, we will need to keep only $d' \ll d$ projections.
- | We can measure how much of the total variance a d' -dimensional subspace captures, by the ratio
- | Variance may be related to the “energy” of a signal: how accurately we want to represent the data.
 - The ratio can be used to guide in choosing a proper d' for desired accuracy.

$$\sum_{j=1}^{d'} \lambda_j \Bigg/ \sum_{j=1}^d \lambda_j$$

The PCA Algorithm



1. Compute the $d \times d$ sample covariance matrix C
2. Find the eigenvalues and corresponding eigenvectors of C
3. Project the original data onto the space spanned by the eigenvectors
 - The projection may be done onto a d' -dimensional subspace spanned by the first d' eigenvectors (ordered by the eigenvalue in descending order)
 - d' is determined by the desired accuracy

Important Properties of PCA



| PCA represents the data in a new space, in which the components of the data is ordered by their “significance”.

- Dimension reduction can be done by simply discarding less significant dimensions.

| Linearity assumption → extensions exist

| “Variance ≈ Importance” is meaningful only under large *signal-to-noise ratio*

PCA as Feature Mapping



| When we use only d' dimensions from PCA (with original dimension $d > d'$), this may look like feature selection.

- But in general they are different approaches.

| PCA

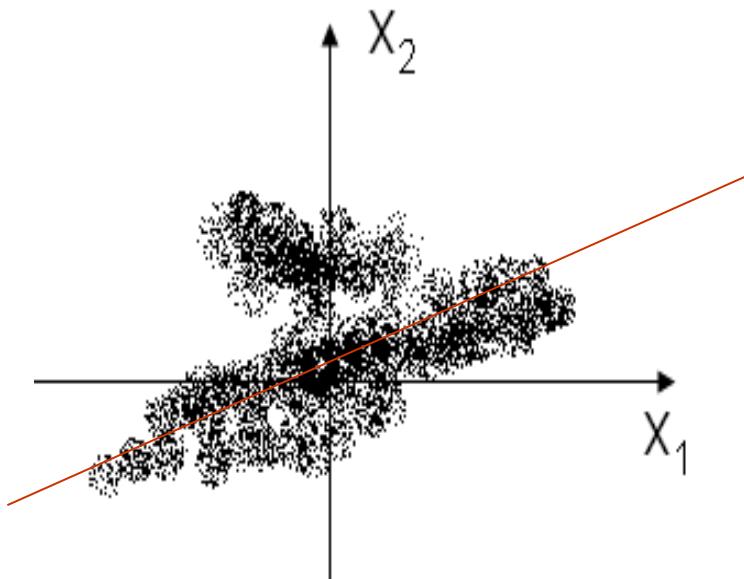
- Unsupervised (in general)
- Generates new features (linear combination of original ones)

| Feature Selection

- Supervised (in general)
- Selects a few original features (e.g., for better classification)

Can PCA Help Classification?

- | Can we do better classification in a lower-dimensional space from d' principal components given by PCA?
 - Not necessarily.
- | LDA may be better posed for such a task.







Introduction to Deep Learning

Neural Networks and Deep Learning



Objective



Objective

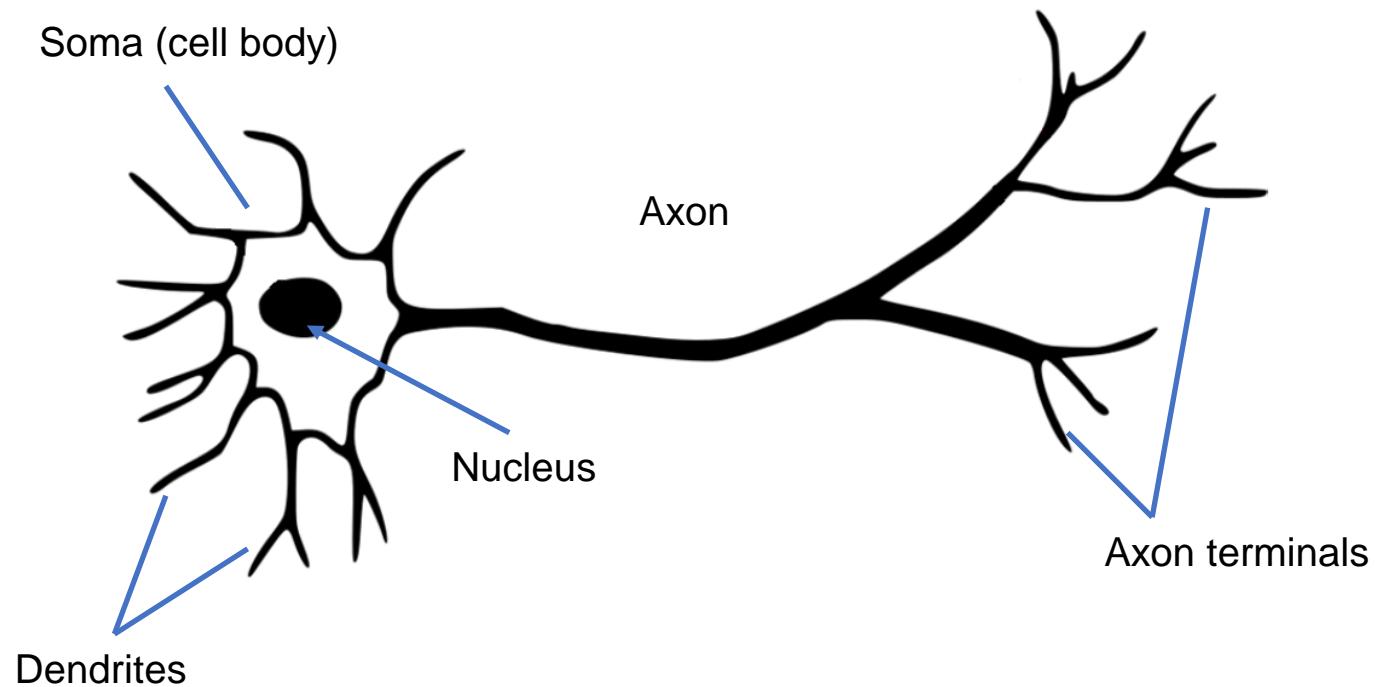
Describe the big-picture view of how neural networks work



Objective

Identify the basic building blocks and notations of deep neural networks

Illustrating A Biological Neuron



Neural Network

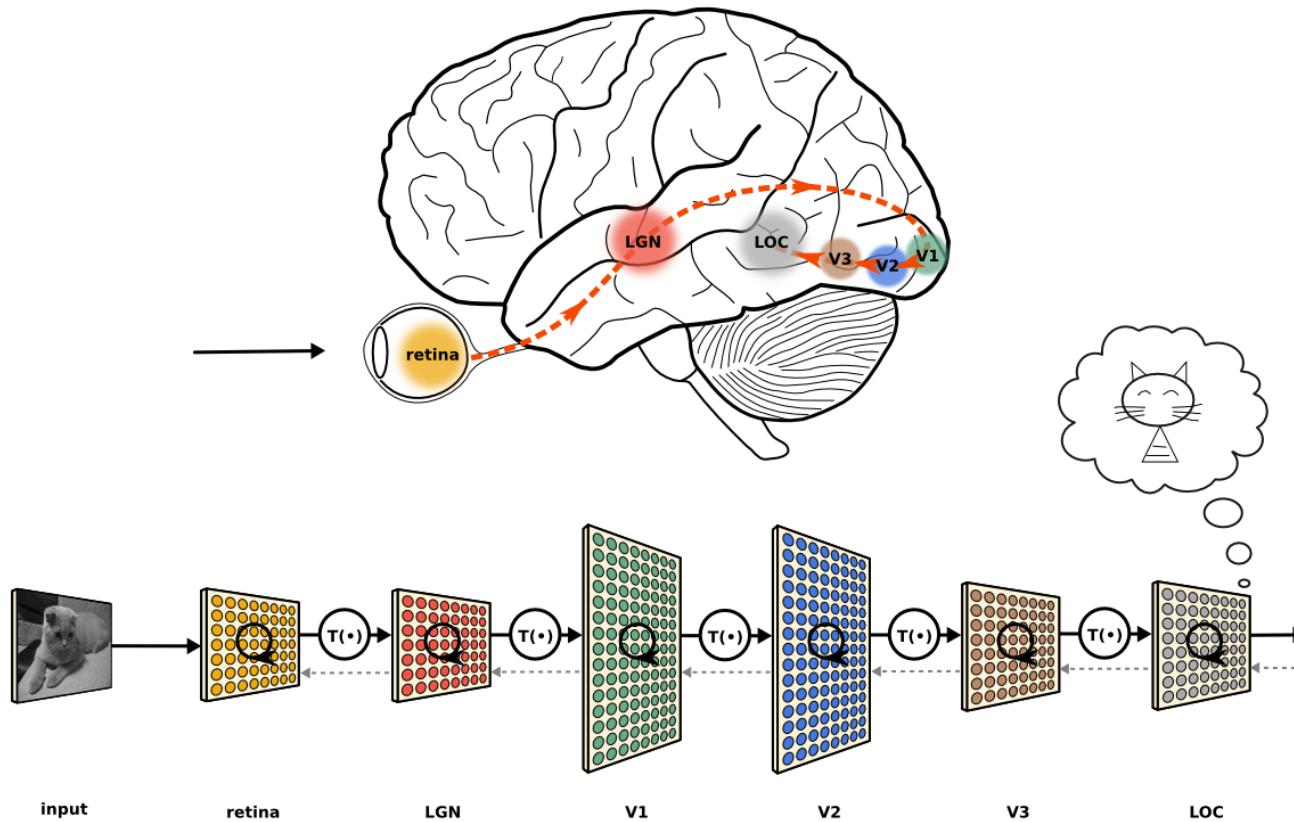
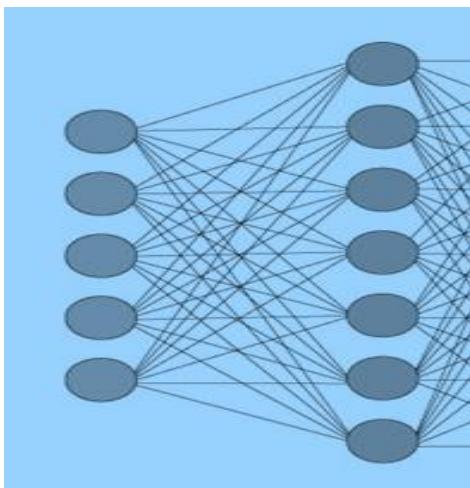


Figure source: <https://neuwritesd.org/2015/10/22/deep-neural-networks-help-us-read-your-mind/>

Artificial Neural Networks



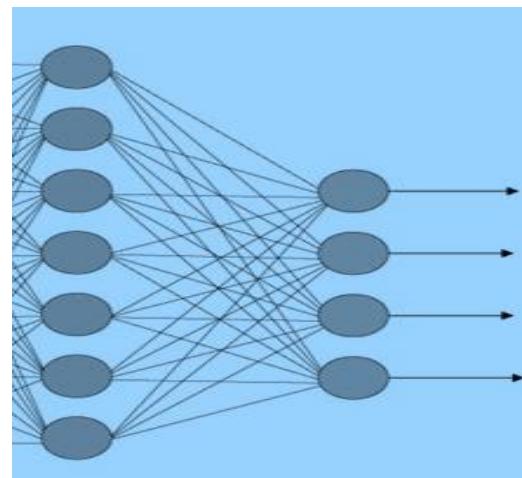
...

...

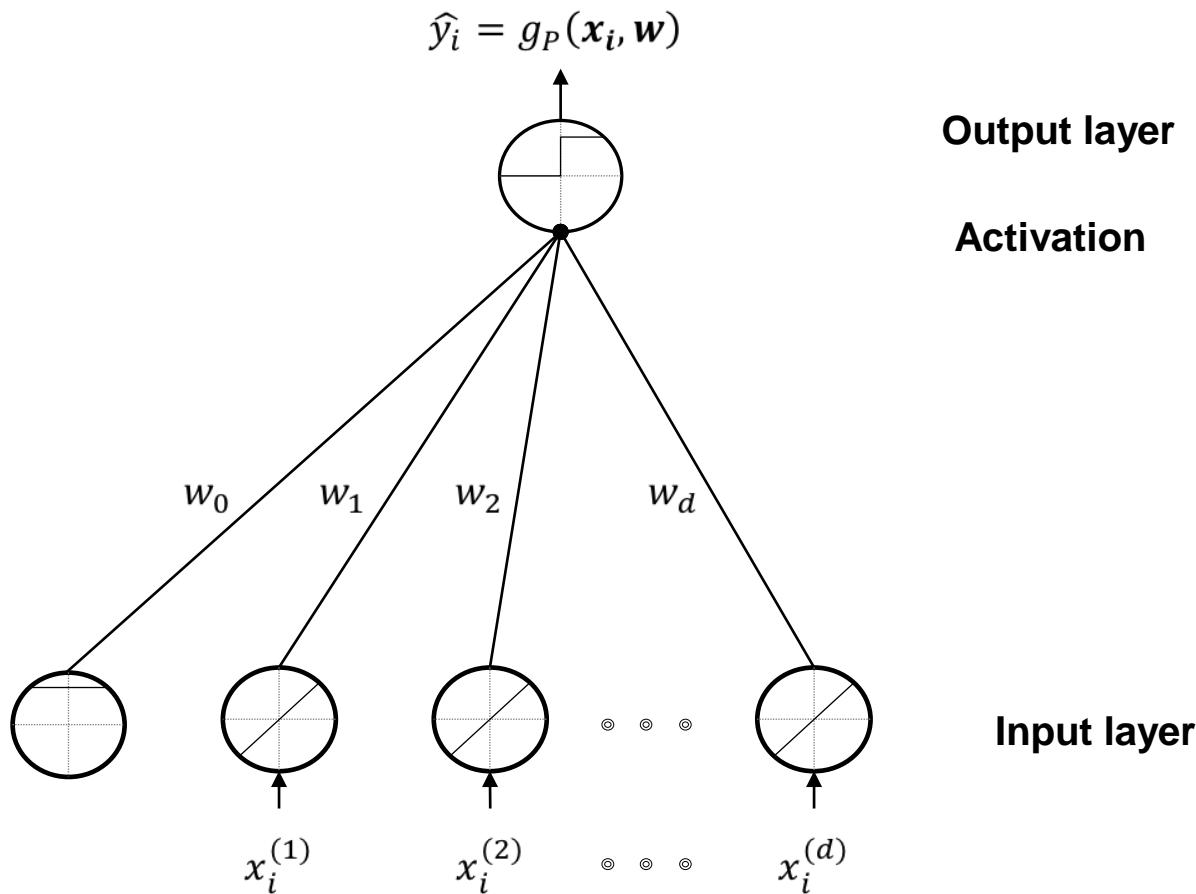
...

...

...



Building Artificial Neural Networks

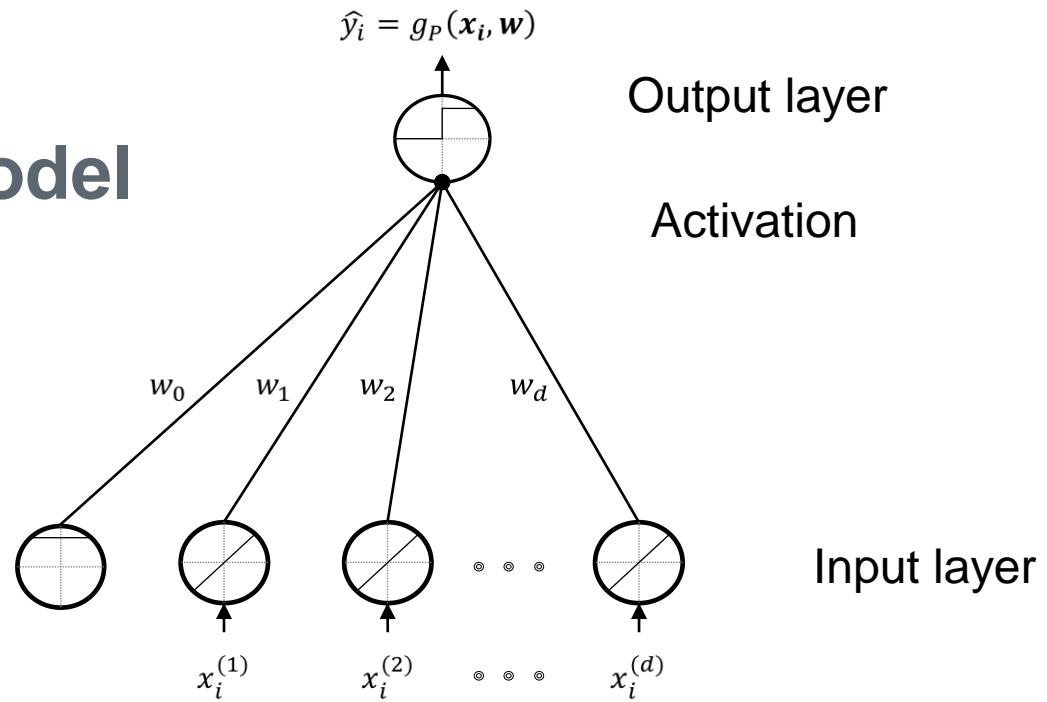


Building Artificial Neural Networks (cont'd)

| What does this “neuron” do?

$$g_P(\mathbf{x}_i, \mathbf{w}) = \begin{cases} 1, & \text{if } \mathbf{w}^T \mathbf{x}_i > 0 \\ 0, & \text{otherwise} \end{cases}$$

| The Perceptron model



Logistic Neuron

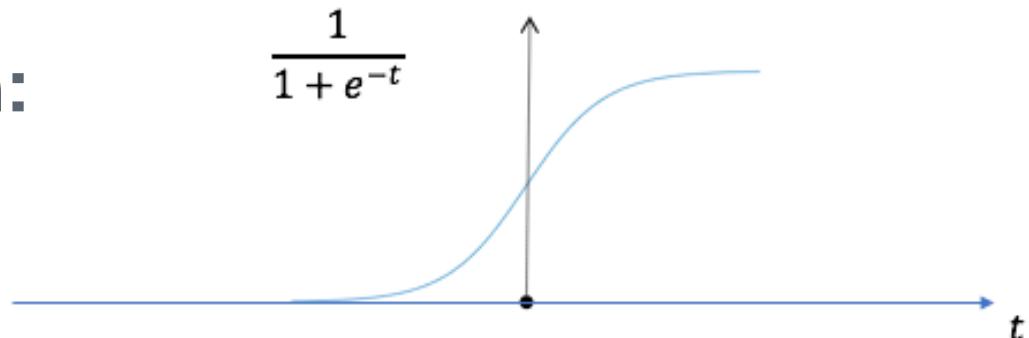
| In Perceptron:

$$g_P(x_i, w) = \begin{cases} 1, & \text{if } w^T x_i > 0 \\ 0, & \text{otherwise} \end{cases}$$

| If we let:

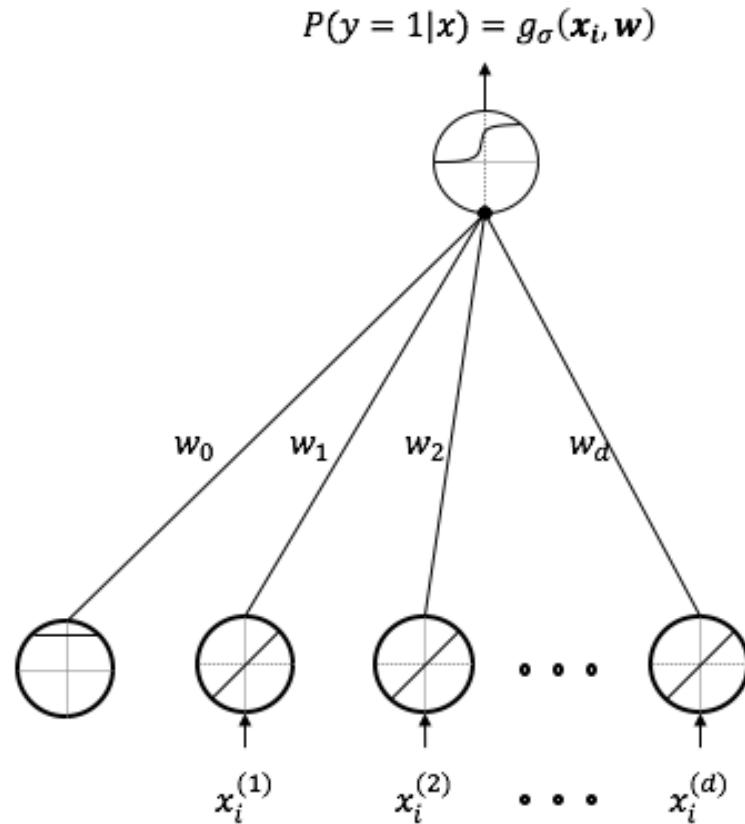
$$g_\sigma(x_i, w) = \frac{1}{1 + e^{-w^T x_i}}.$$

| The logistic function:



Logistic Neuron (cont'd)

We have:



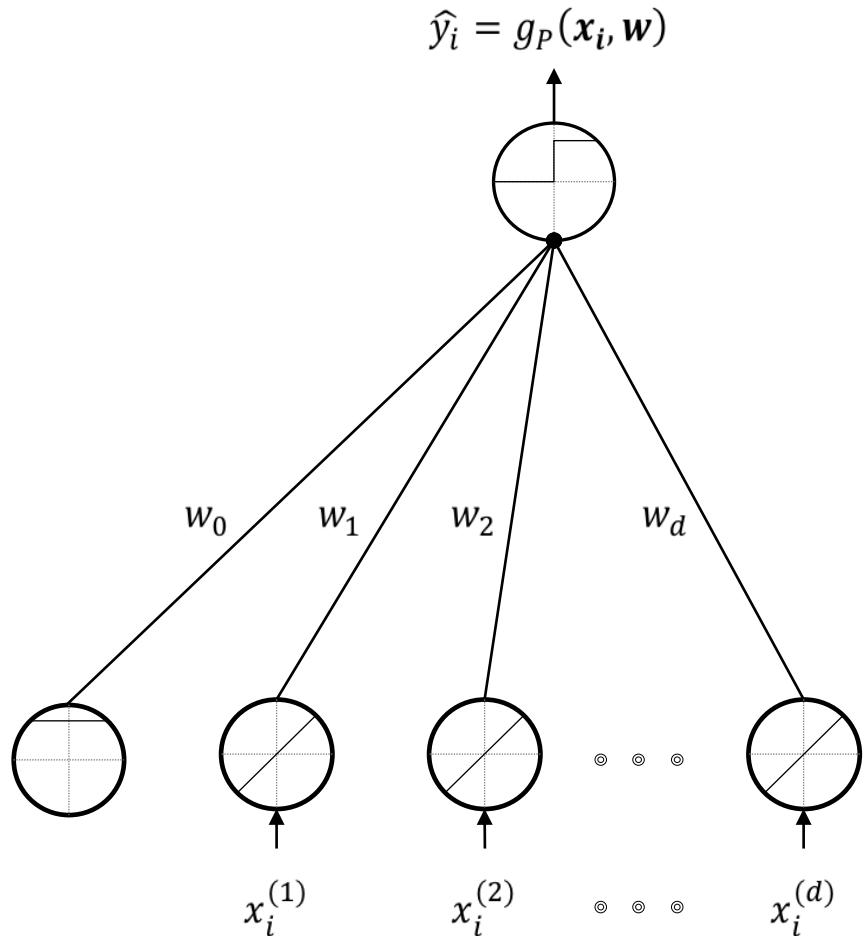
A probability prediction

Activation

Input layer

Learning in the Perceptron

“Learning”: how does the neuron adapt its weights in response to the inputs?



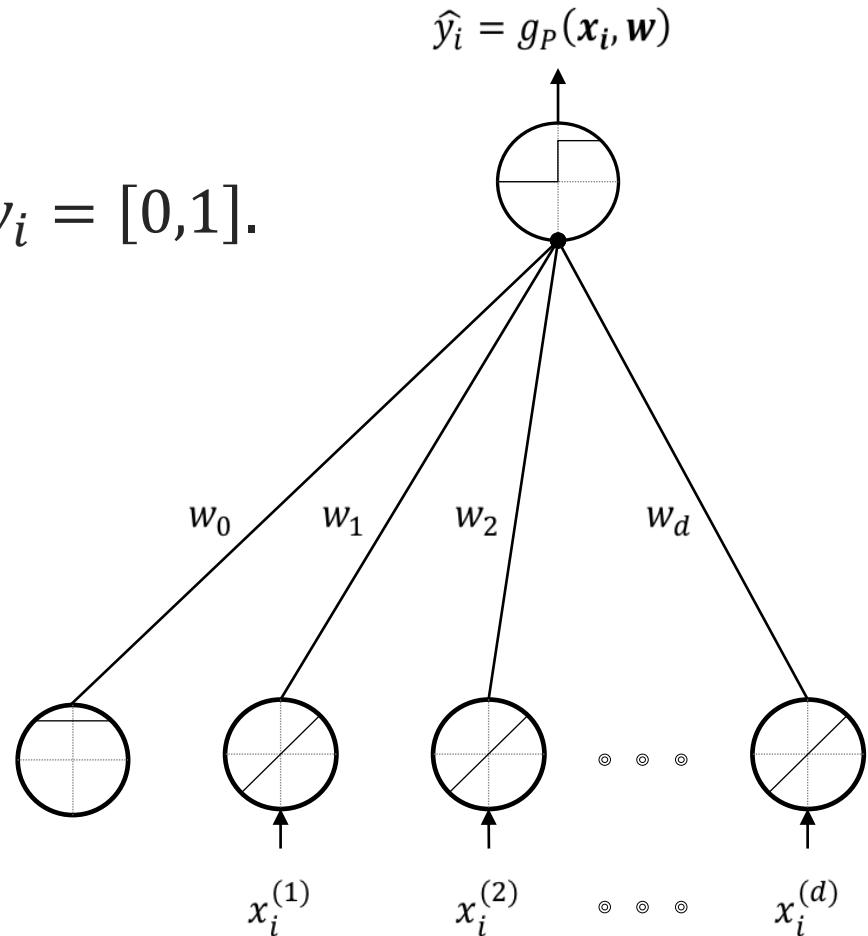
The Perceptron Learning Algorithm

Input

- Training set
 $D = \{(x_i, y_i), i \in [1, 2, \dots, n]\}, y_i = [0, 1].$

Initialization

- Initialize the weights $w(0)$ (and some thresholds)
- Weights may be set to 0 or small random values



The Perceptron Learning Algorithm (cont'd)

| Iterate for t until a stop criterion is met

{

for each sample x_i with label y_i :

{

compute the output \tilde{y}_i of the network

estimate the error of the network $e(w(t)) = y_i - \tilde{y}_i$

update the weight $w(t + 1) = w(t) + e(w(t))x_i$

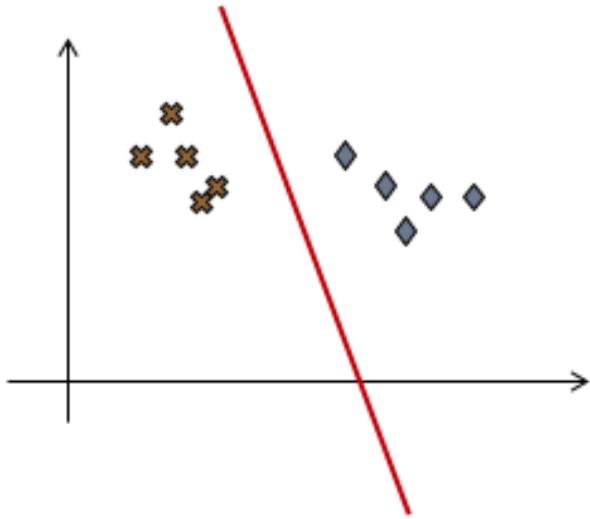
}

$t++$

}

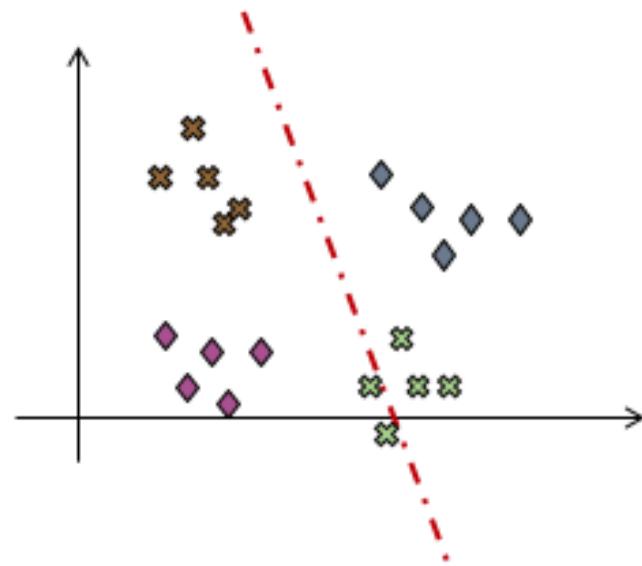
The Need for Multiple Layers

| This is easy and can
be learned by the
Perceptron Algorithm



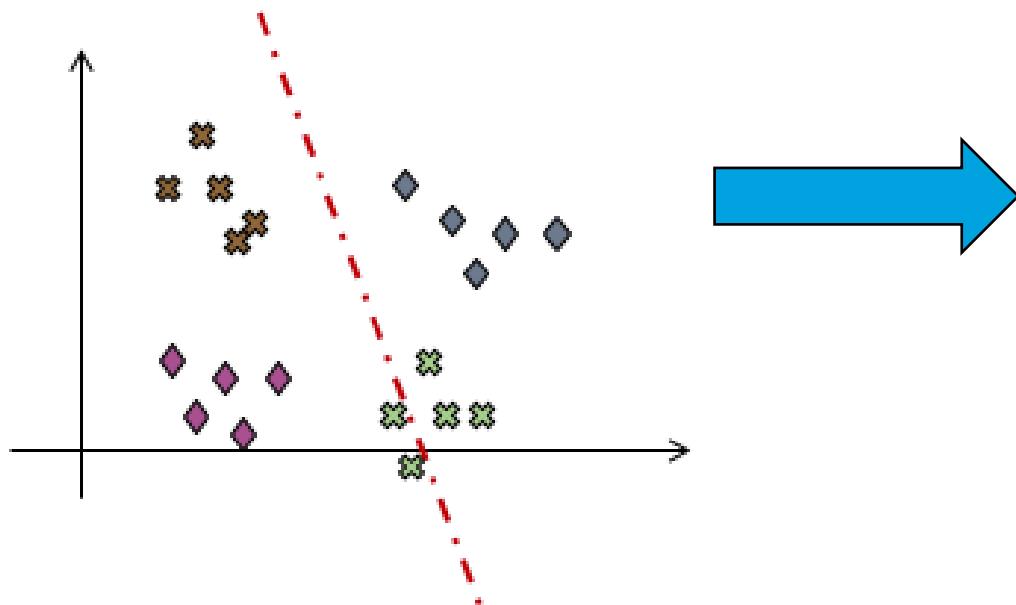
$$w_1x_1 + w_2x_2 + w_0 = 0$$

| But how about this?



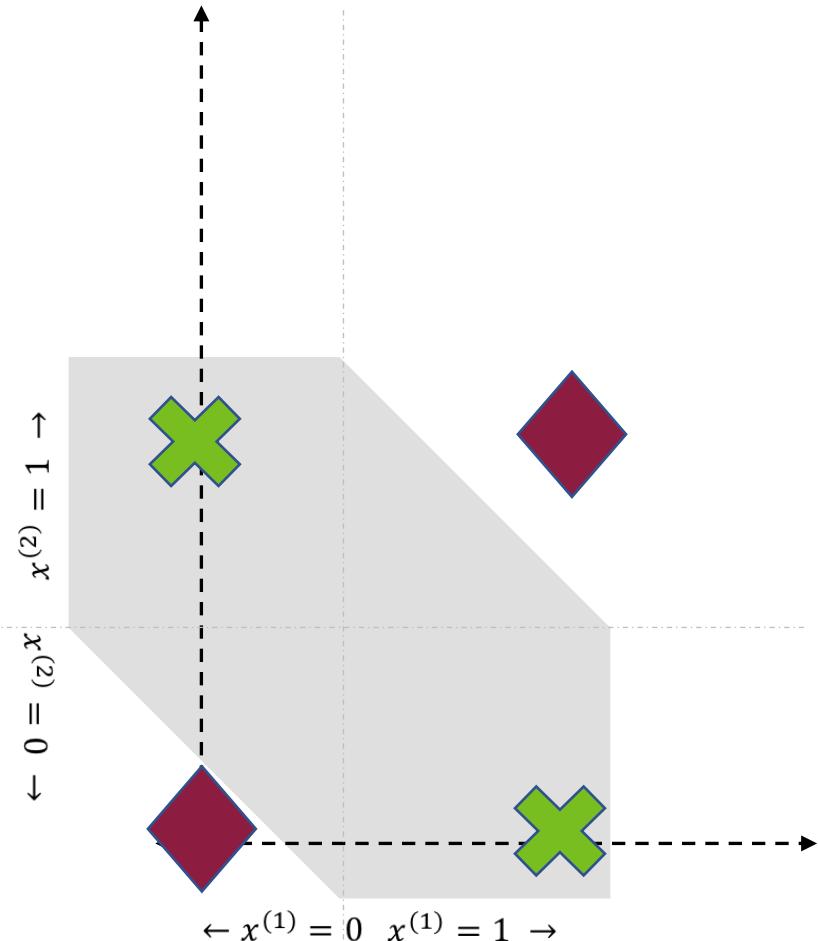
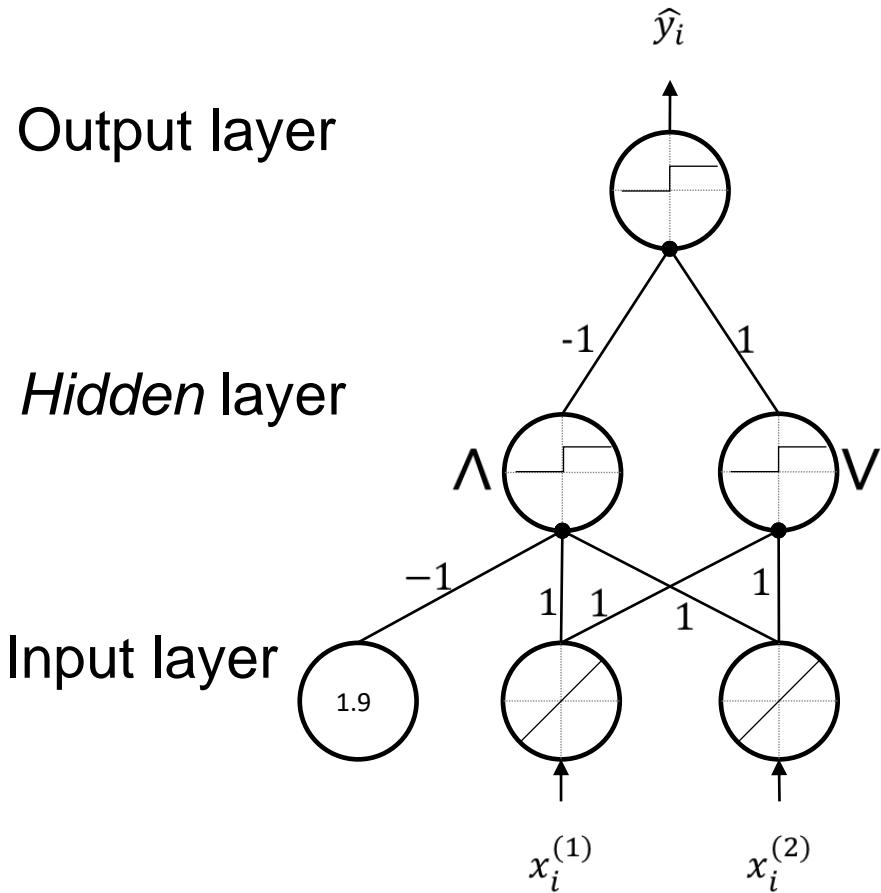
Extending to Multi-layer Neural Networks

| Question: Can a multi-layer version of the Perceptron (MLP) help solving the XOR problem?



The XOR Problem

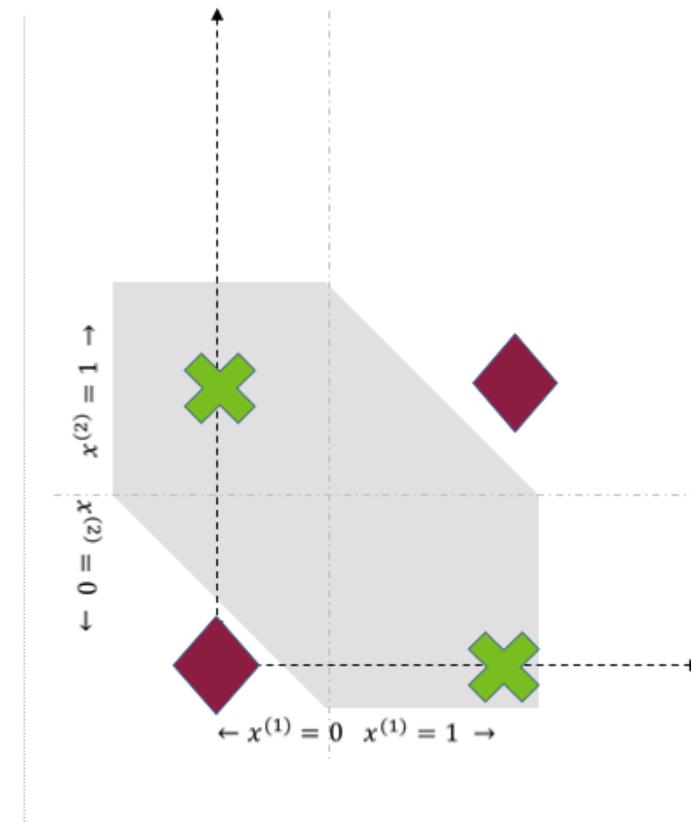
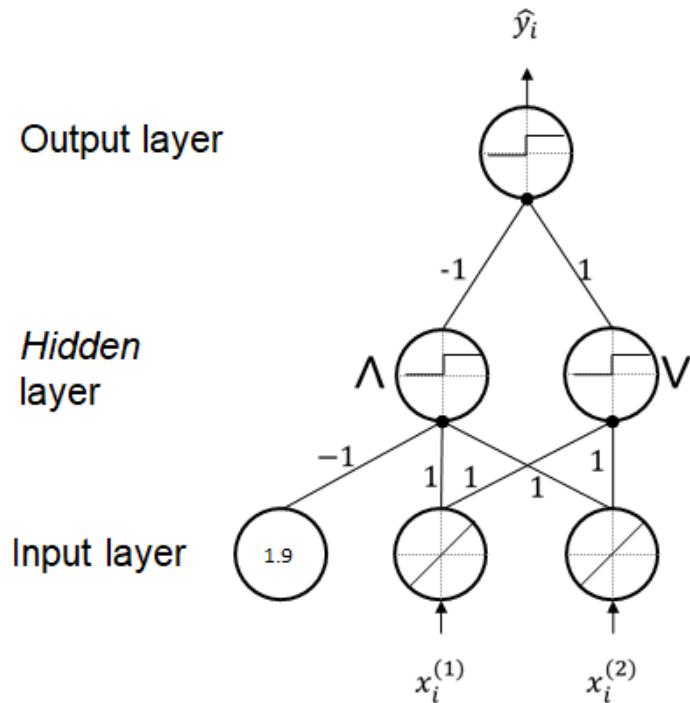
An MLP Solving the XOR Problem



The Question of Learning

| How can the network learn proper parameters from the given samples?

- Can the Perceptron algorithm be used?



Difficulty in Learning for MLP



Perceptron Learning Algorithm

- The weight update of the neuron is proportional to the “error” computed as $y_i - \hat{y}_i$.
 - This requires us to know the target output y_i .

Multi-layer Perceptron

- Except for the neurons on the output layer, other neurons (on the *hidden* layers) do not really have a target output given.

Back-propagation (BP) Learning for MLP



| **The key: Properly distribute error computed from output layer back to earlier layers to allow their weights to be updated in a way that reduce the error**

- The basic philosophy of the BP algorithm

| **Differentiable activation functions**

- We can use – e.g.,
 - the logistic neurons
 - neurons with sigmoid activation
 - or its variants

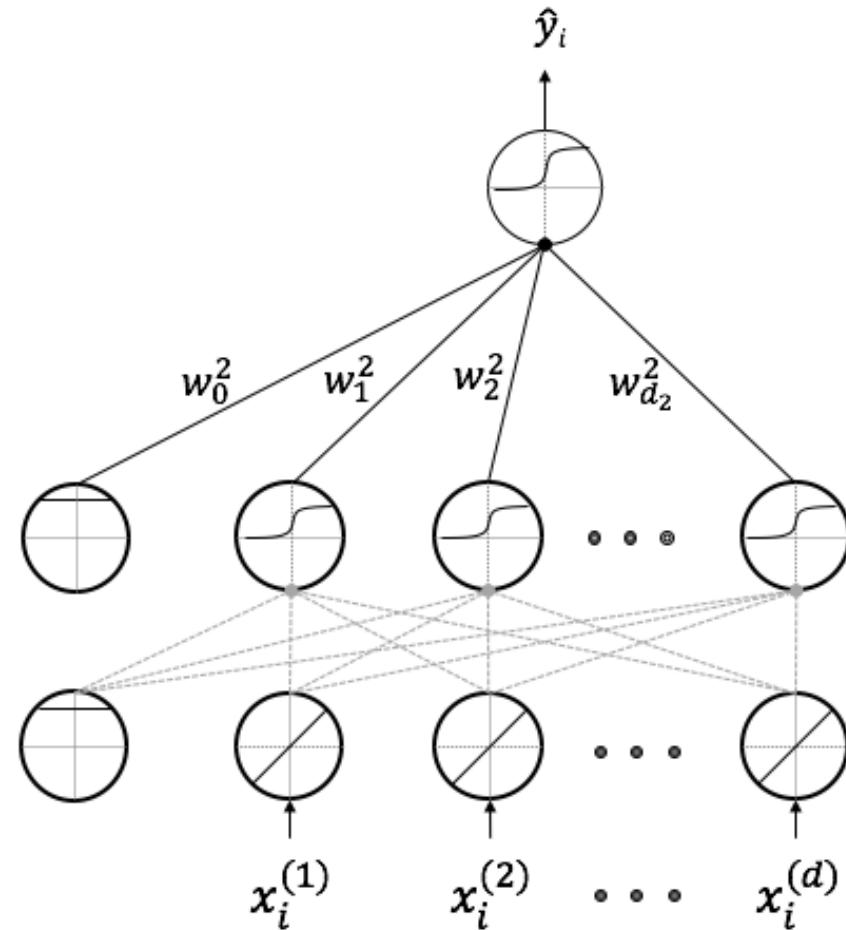
A Multi-Layer Neural Network

| Using Logistic Neurons

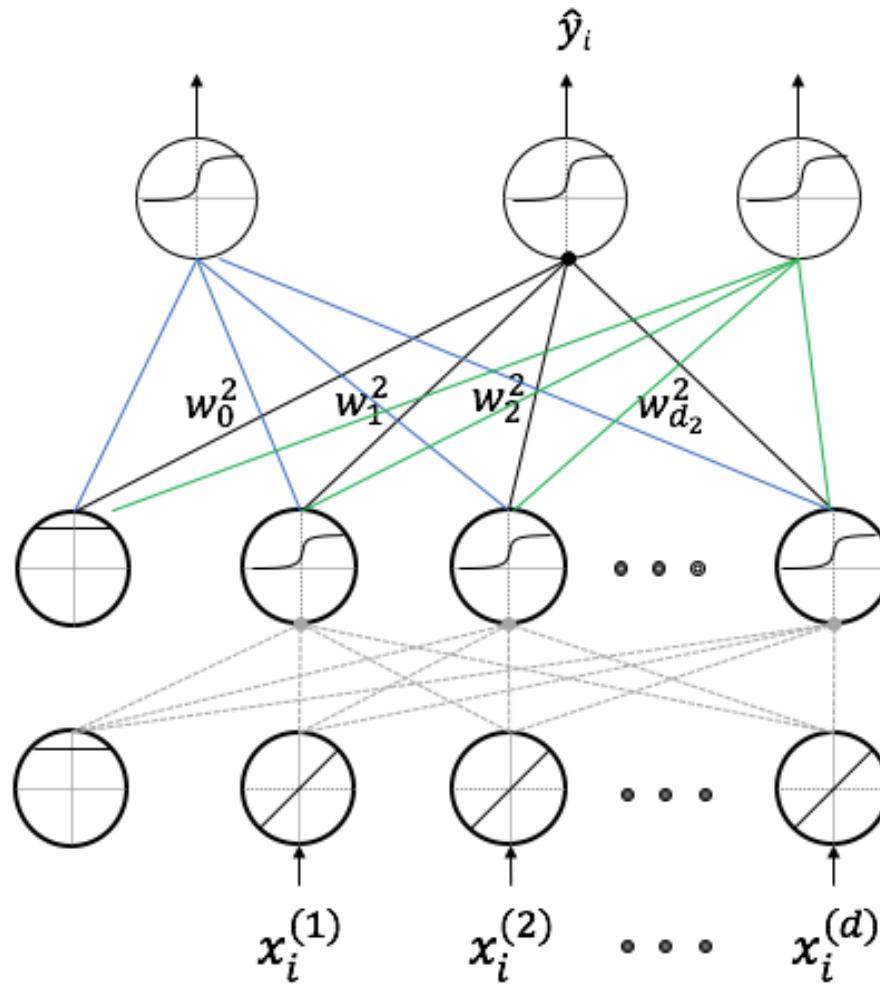
Output layer

Hidden layer

Input layer

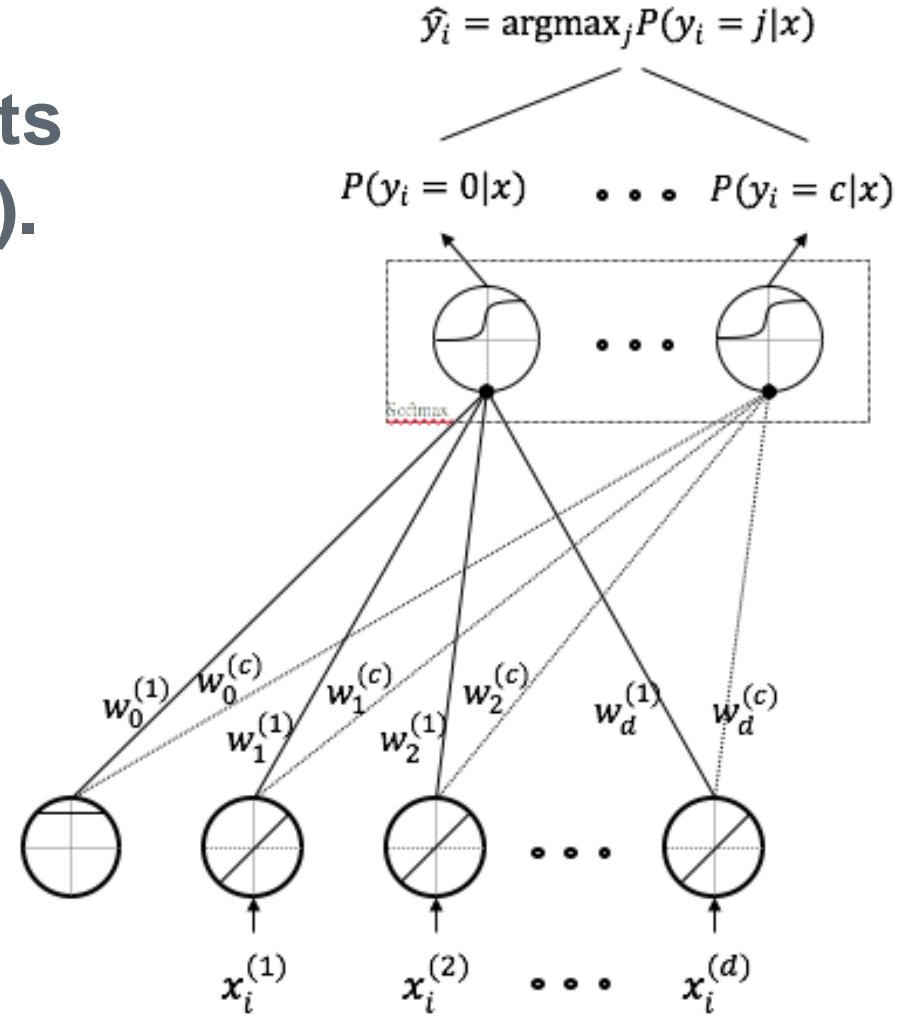


Handling Multiple (>2) Classes



Softmax for Handling Multiple Classes

Using softmax to normalize the outputs (so they add up to 1).



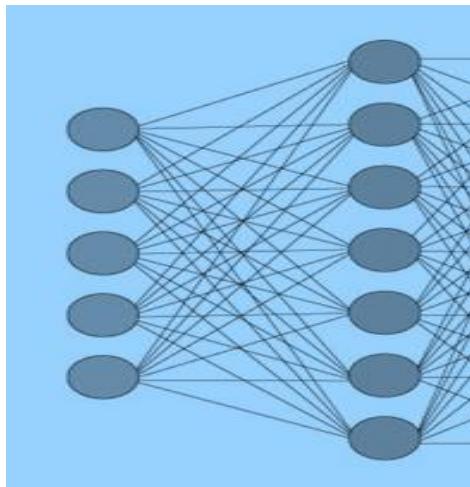
How to Compute “errors” in this Case?

| Consider the cross-entropy as a loss function:

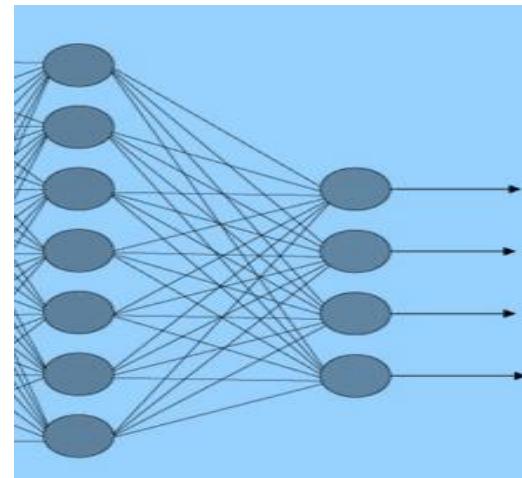
$$l(\mathbf{W}) = \sum_{i=1}^n \sum_{j=1}^c \mathbb{I}_j(y_i) \log P(y_i = j | x_i) ,$$

$$\mathbb{I}_j(y_i) = \begin{cases} 1, & \text{if } y_i = j \\ 0, & \text{otherwise} \end{cases}$$

Neural Networks and Deep Learning



...





Introduction to Deep Learning

Key Techniques Enabling Deep Learning

Objective



Objective

Explain how, in principle, learning is achieved in a deep network



Objective

Explain key techniques that enable efficient learning in deep networks

Overview



| Back-propagation algorithm

| Design of activation functions

| Regularization for improving performance

* Technological advancement in computing hardware is certainly another enabling factor but our discussion will focus on basic, algorithmic techniques.

Back Propagation (BP) Algorithm



| Simple Perceptron algorithm illustrates a path to learning by iterative optimization

- Updating weights based on network errors under current weights, and optimal weights are obtained when errors become 0 (or small enough)

| Gradient descent is a general approach to iterative optimization

- Define a loss function J
- Iteratively update the weights \mathbf{W} according to the gradient of J with respect to \mathbf{W} .

$\mathbf{W} \leftarrow \mathbf{W} - \eta \nabla J(\mathbf{W})$ \mathbf{W} is the parameter of the network; J is the objective function.

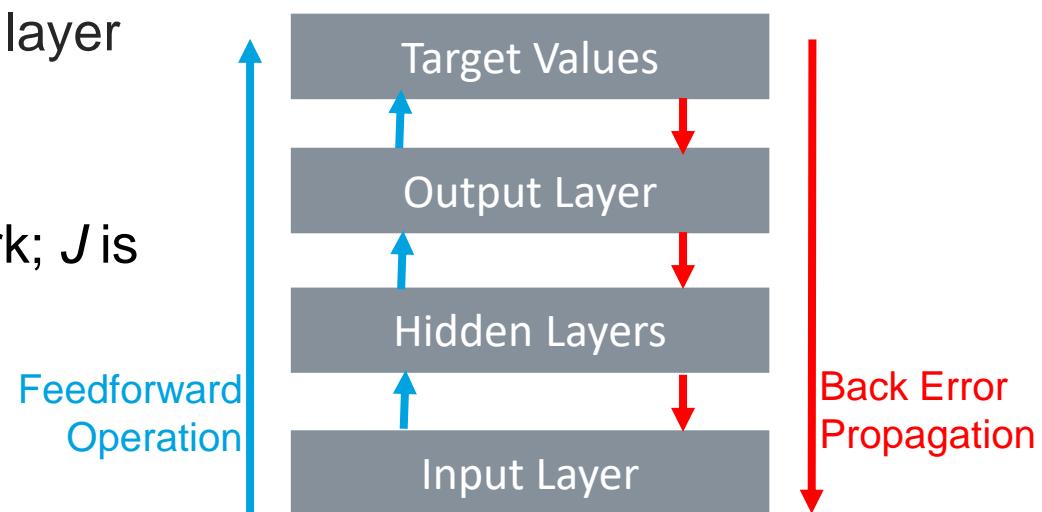
Back Propagation (BP) Algorithm (cont'd)

| Generalizes/Implements the idea for multi-layer networks

- Gradient descent for updating weights in optimizing a loss function
- Propagating gradients back through layers
 - hidden layer weights are linked to loss gradient at output layer

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \bigtriangledown J(\mathbf{W})$$

\mathbf{W} is the parameter of the network; J is the objective function.

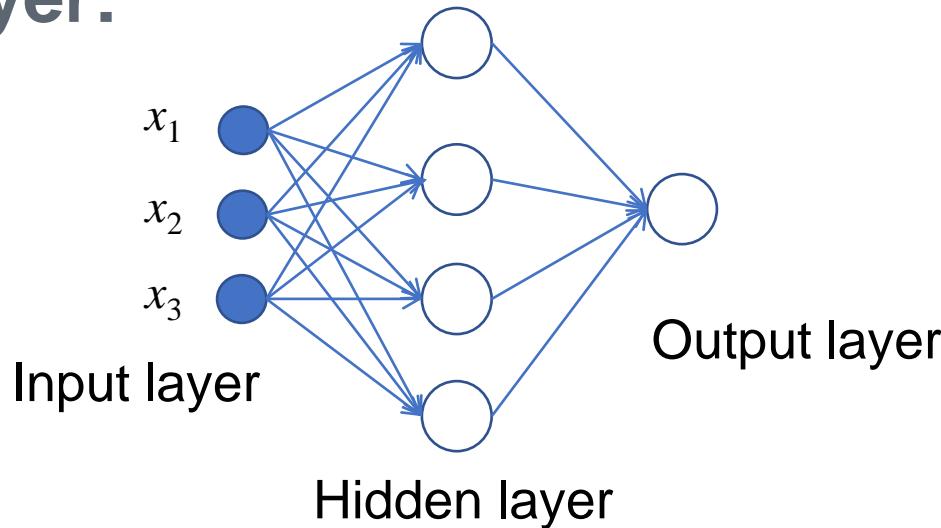


Illustrating the BP Algorithm 1/6

| Let's consider a simple neural network with a single hidden layer. (We will only outline the key steps.)

- Let's write the net input and activation for a hidden node:

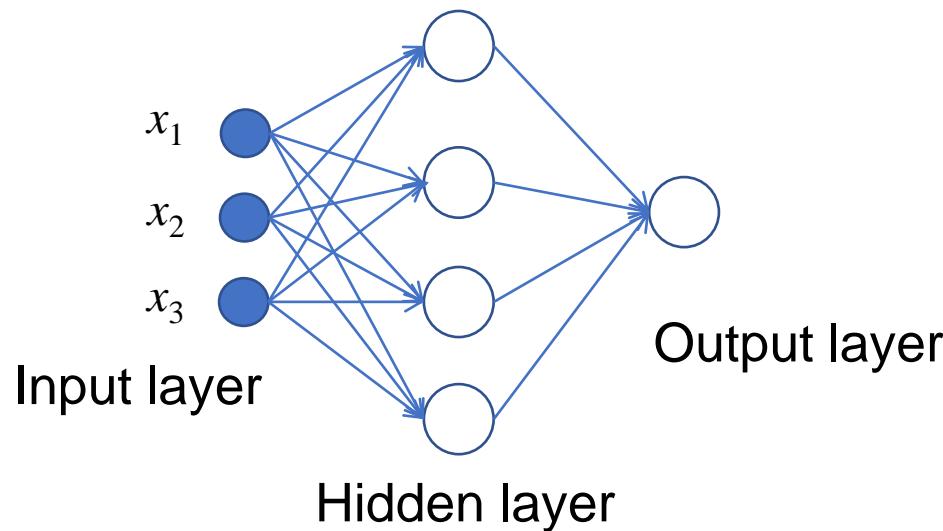
| Let's write the net input and activation for the hidden layer:



Illustrating the BP Algorithm 2/6

| Using matrix/vector notations, for the hidden layer:

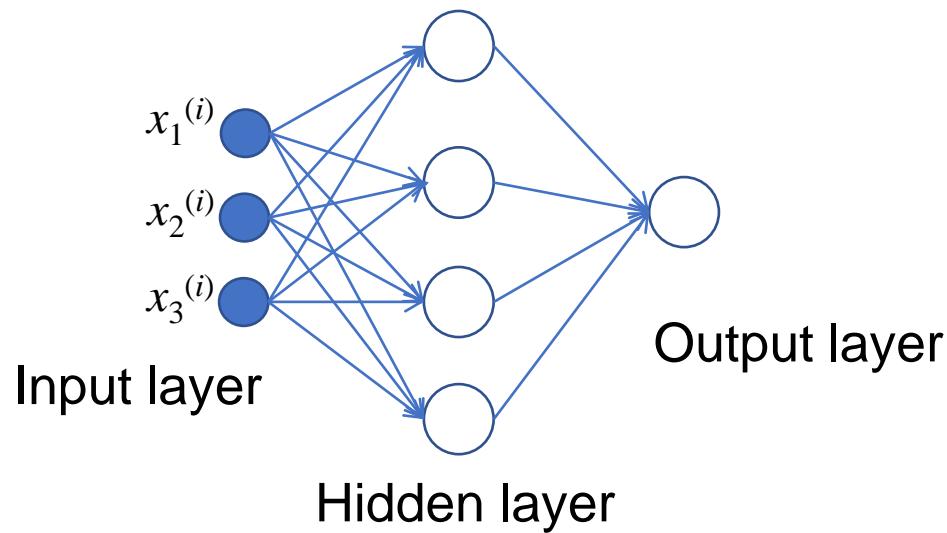
| Similarly, for the output layer → Homework.



Illustrating the BP Algorithm 3/6

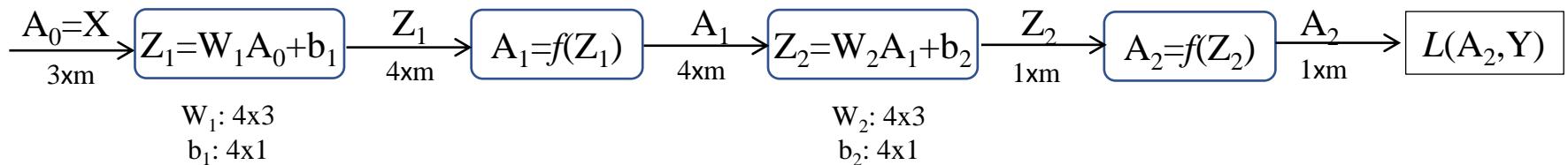
| Now consider m samples as input.

| Output layer is similarly done.

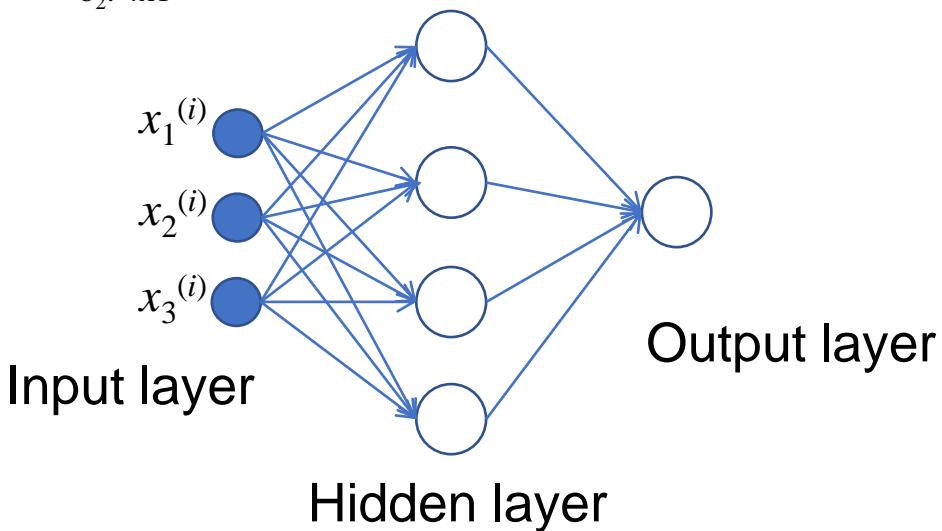


Illustrating the BP Algorithm 4/6

| Overall we have this flow of *feedforward* processing (note the notation change for simplicity: subscripts are for layers):

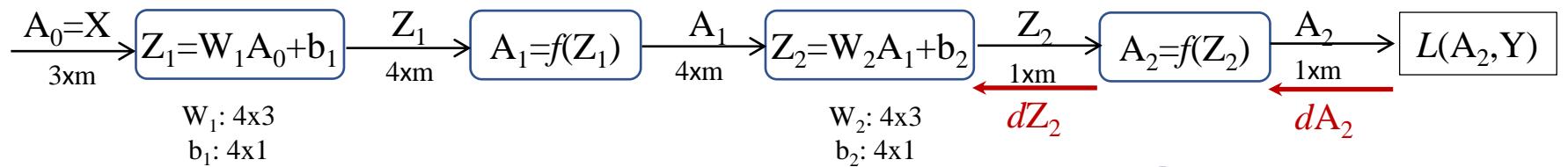


| Consider $dW_2 \triangleq \frac{\partial L}{\partial W_2}$

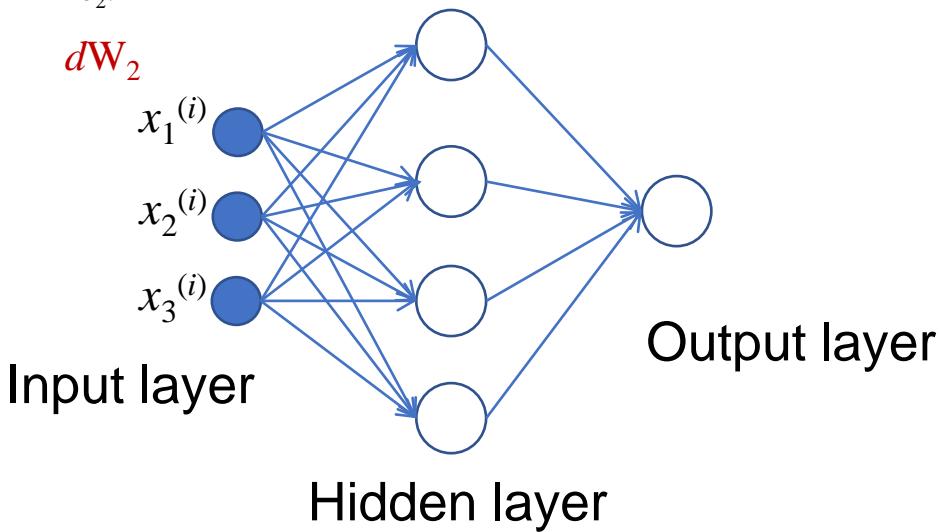


Illustrating the BP Algorithm 5/6

Back-propagation

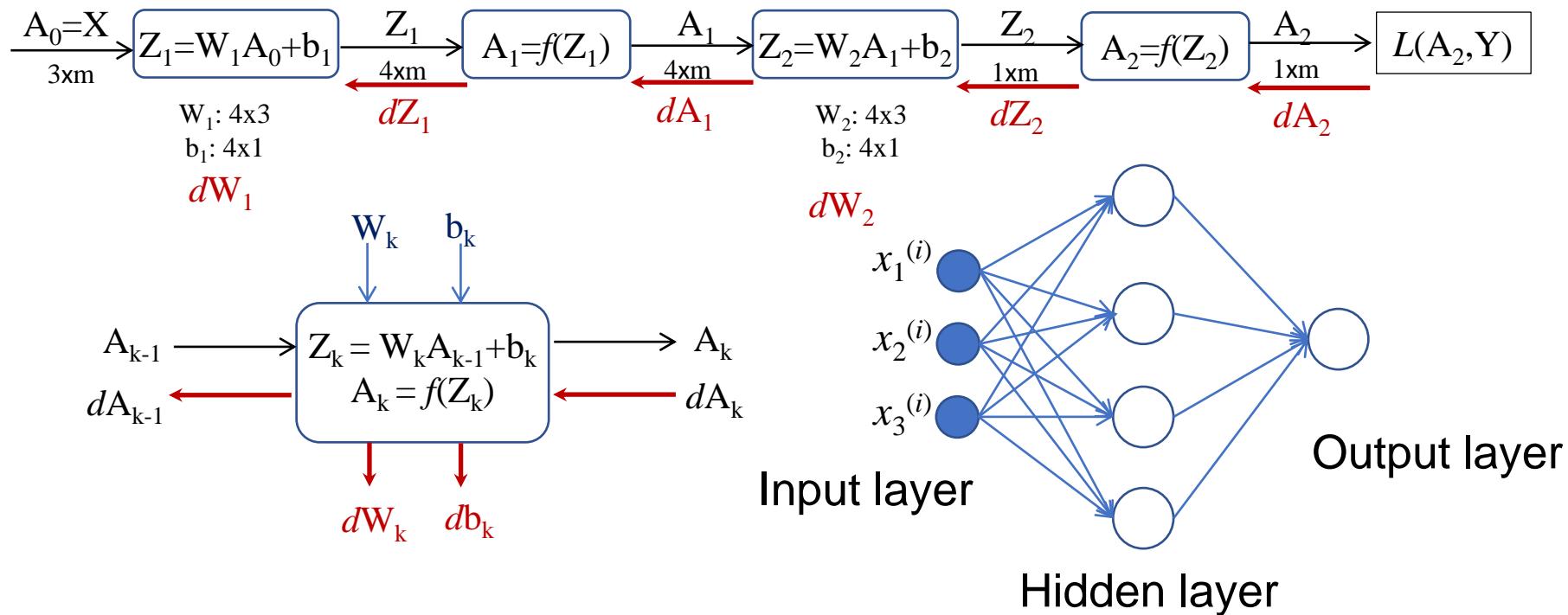


Consider $dW_1 \triangleq \frac{\partial L}{\partial W_1}$



Illustrating the BP Algorithm 6/6

A modular view of the layers



BP Algorithm Recap



| **The feedforward process:** ultimately produce $A^{[K]}$ that leads to the prediction for Y .

| **The backpropagation process:**

- First compute the loss
- Then compute the gradients via back-propagation through layers
- Key: use the chain rule of differentiation

| **Essential to deep networks**

| **Suffers from several practical limitations**

- gradient exploding
- gradient vanishing
- etc.

| **Many techniques were instrumental to enabling learning with BP algorithm for deep neural networks**

Activation Functions: Importance



| Provides non-linearity

| Functional unit of input-output mapping

| Its form impacts on gradients in BP algorithm

Activation Functions: Choices



| Older Types

- Thresholding
- Logistic function
- tanh

| Newer Types

- Rectifier $f(x) = \max(0, x)$ and its variants
- Rectified Linear Unit (ReLU)

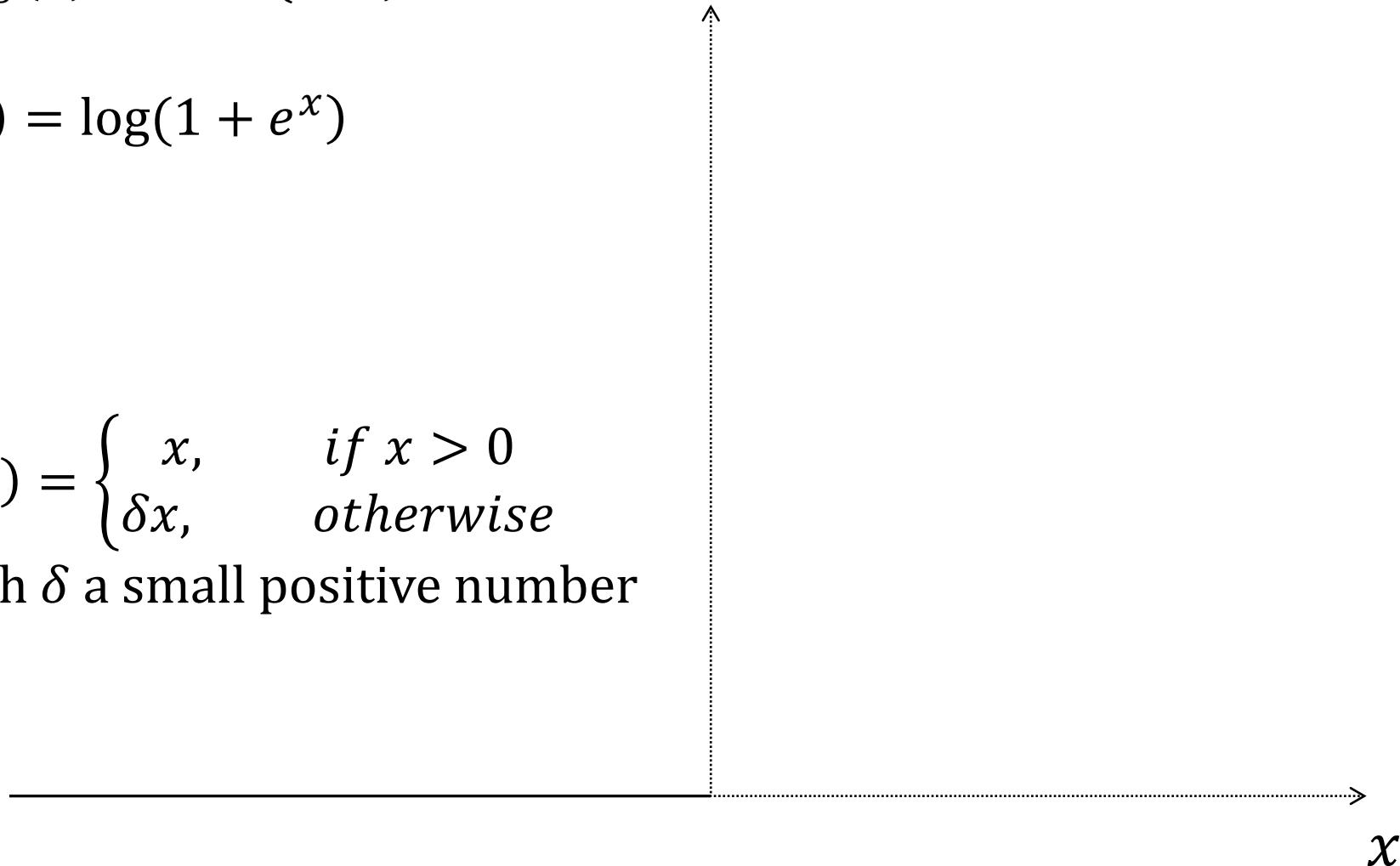
ReLU and Some Variants

$$a_{\text{ReLU}}(x) = \max(0, x)$$

$$a_s(x) = \log(1 + e^x)$$

$$a_L(x) = \begin{cases} x, & \text{if } x > 0 \\ \delta x, & \text{otherwise} \end{cases}$$

with δ a small positive number



The Importance of Regularization



- | The parameter space is huge, if there is no constraint in search for a solution, the algorithm may converge to poor solutions.
- | Overfitting is a typical problem
 - Converging to local minimum good only for the training data

Some Ideas for Regularization



| Favoring a network with small weights

- achieved by adding a term of L2-norm of the weights to original loss function

| Preventing neurons from “co-adaptation” → Drop-out

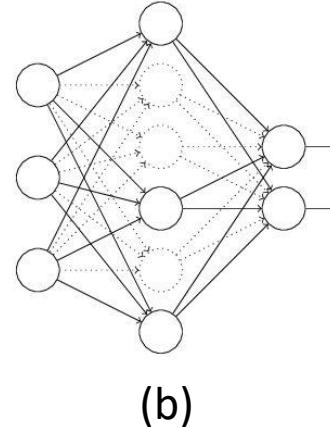
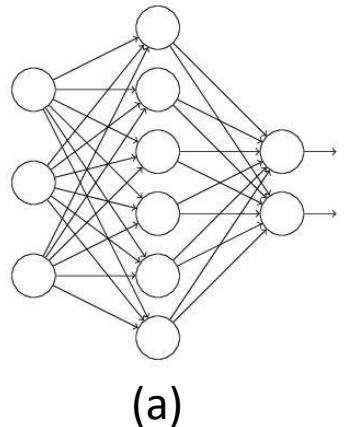
| Making the network less sensitive to initialization/learning rate etc.

- Batch normalization

| Such regularization techniques have been found to be not only helpful but sometimes critical to learning in deep networks

Drop-out

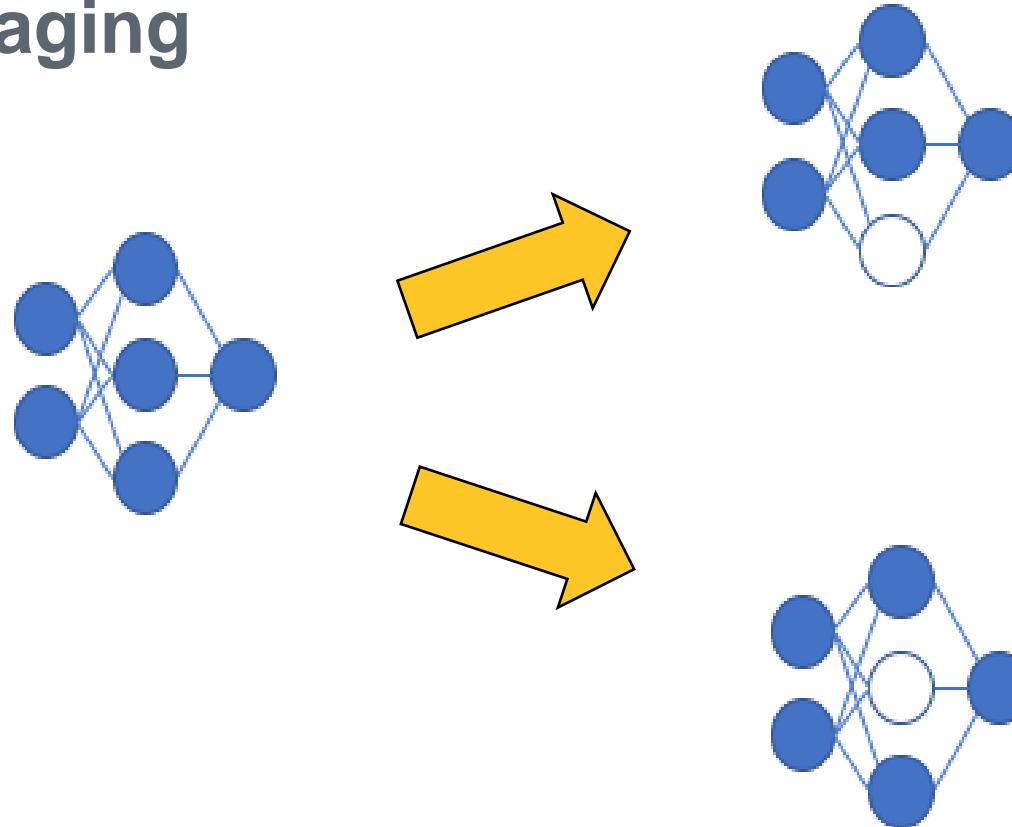
- | Obtain (b) by randomly deactivate some hidden nodes in (a)
- | For input x , calculate output y by using the activated nodes ONLY
- | Use BP to update weights (which connect to the activated nodes) of network
- | Activate all nodes
- | Go back to first step



Why Drop-out?

| Reducing co-adaptation of neuron

| Model averaging



Batch Normalization (BN)



| Inputs to network layers are of varying distributions, the so-called internal covariate shift [Ioffe and Szegedy, 2015]

- Careful parameter initialization and low learning rate are required

| BN was developed to solve this problem by normalizing layer inputs of a batch

The Simple Math of BN

| For a mini-batch with size = m, first calculate

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad \sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad \hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

| Up to this point, \hat{x} has mean = 0 and standard deviation = 1

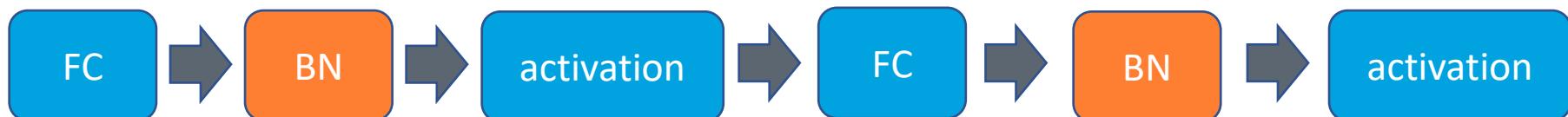
How is BN Used in Learning?

| Define two parameters γ and β so that the output of the BN layer can be calculated as:

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i)$$

| Parameters β and γ can be learned by minimizing the loss function via gradient descent

| Usually used right before the activation functions



Other Regularization Techniques



| Weight sharing

| Training data conditioning

| Sparsity constraints

| Ensemble methods (committee of networks)

→ Some of these will be discussed in later examples of networks.



Introduction to Deep Learning

Some Basic Deep Architecture

Objective



Objective

Appraise the detailed architecture of a basic convolutional neural network



Objective

Explain the basic concepts and corresponding architecture for auto-encoders and recurrent neural networks

Overview



Convolutional Neural Network (CNN)

- will be given the most attention, for its wide range of application

Auto-encoder

Recurrent Neural Networks (RNN)

Convolutional Neural Network (CNN)



- | Most useful for input data defined on grid-like structures, like images or audio
- | Built upon concept of “convolution” for signal/image filtering
- | Invokes other concepts like pooling, weight-sharing, and (visual) receptive field, etc.

Image Filtering via Convolution 1/5

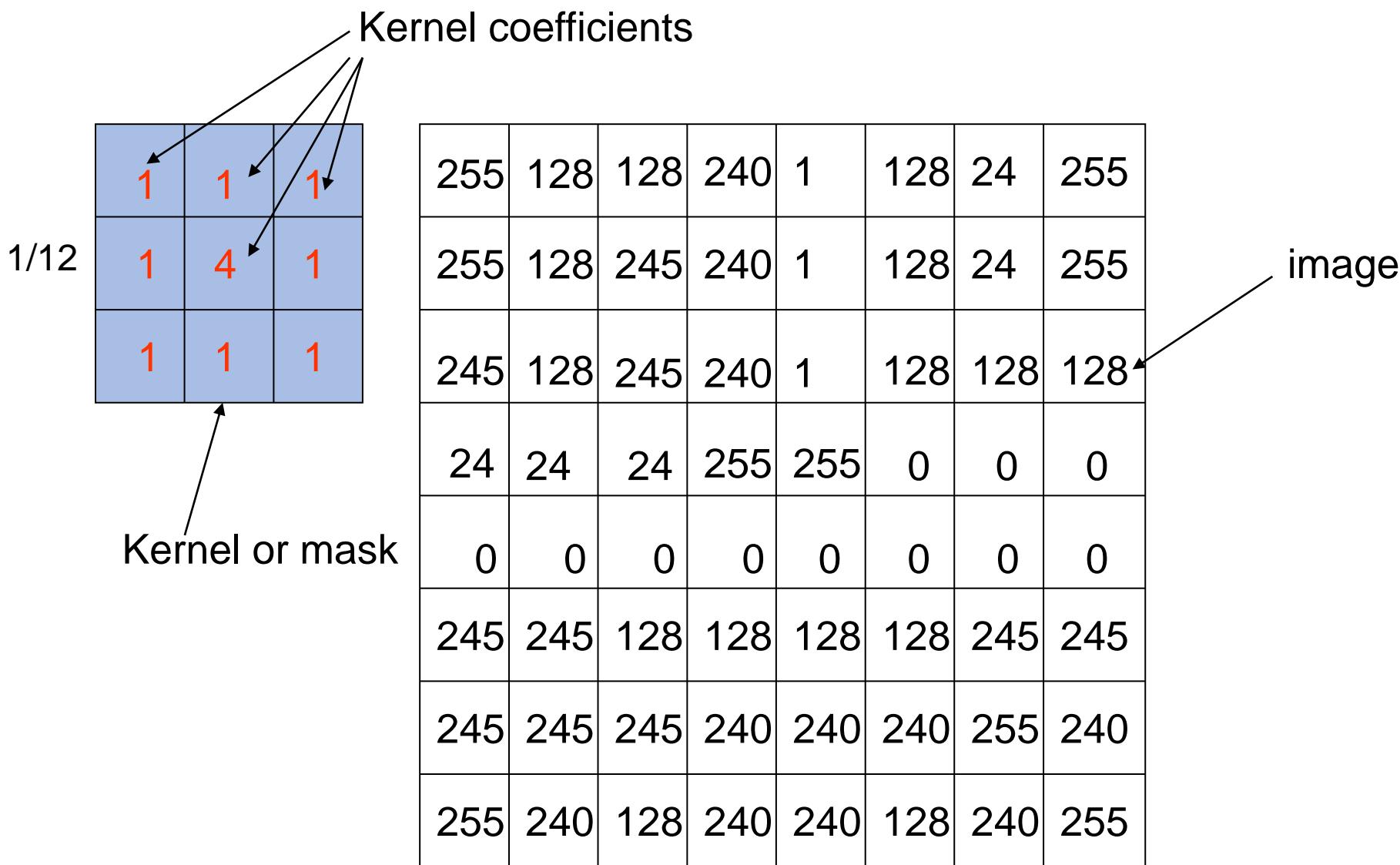


Image Filtering via Convolution 2/5

New pixel
value =
 $(1*255 +$
 $1*128 +$
 $1*128 +$
 $1*255 +$
 $4*128 +$
 $1*245 +$
 $1*245 +$
 $1*128 +$
 $1*245)/12 =$
 $2141/12$
 $=178$

255	128	128	240	1	128	24	255
255	128	245	240	1	128	24	255
245	128	245	240	1	128	128	128
24	24	24	255	255	0	0	0
0	0	0	0	0	0	0	0
245	245	128	128	128	128	245	245
245	245	245	240	240	240	255	240
255	240	128	240	240	128	240	255

Image Filtering via Convolution 3/5

255	128	128	240	1	128	24	255
255	178	245	240	1	128	24	255
245	128	245	240	1	128	128	128
24	24	24	255	255	0	0	0
0	0	0	0	0	0	0	0
245	245	128	128	128	128	245	245
245	245	245	240	240	240	255	240
255	240	128	240	240	128	240	255

Image Filtering via Convolution 4/5

New pixel
value =
 $(1*128 +$
 $1*128 +$
 $1*240 +$
 $1*178 +$
 $4*245 +$
 $1*240 +$
 $1*128 +$
 $1*245 +$
 $1*240)/12 =$
 $2507/12$
 $=209$

255	128	128	240	1	128	24	255
255	178	245	240	1	128	24	255
245	128	245	240	1	128	128	128
24	24	24	255	255	0	0	0
0	0	0	0	0	0	0	0
245	245	128	128	128	128	245	245
245	245	245	240	240	240	255	240
255	240	128	240	240	128	240	255

Image Filtering via Convolution 5/5

255	128	128	240	1	128	24	255
255	<u>178</u>	<u>209</u>	240	1	128	24	255
245	128	245	240	1	128	128	128
24	24	24	255	255	0	0	0
0	0	0	0	0	0	0	0
245	245	128	128	128	128	245	245
245	245	245	240	240	240	255	240
255	240	128	240	240	128	240	255

Image Filtering via Convolution: Kernels

| By varying coefficients of the kernel, we can achieve different goals

- Smoothing, sharpening, detecting edges, etc.

| Better yet: can we learn proper kernels? 
Part of CNN objective

| Examples of Kernels:

1/12

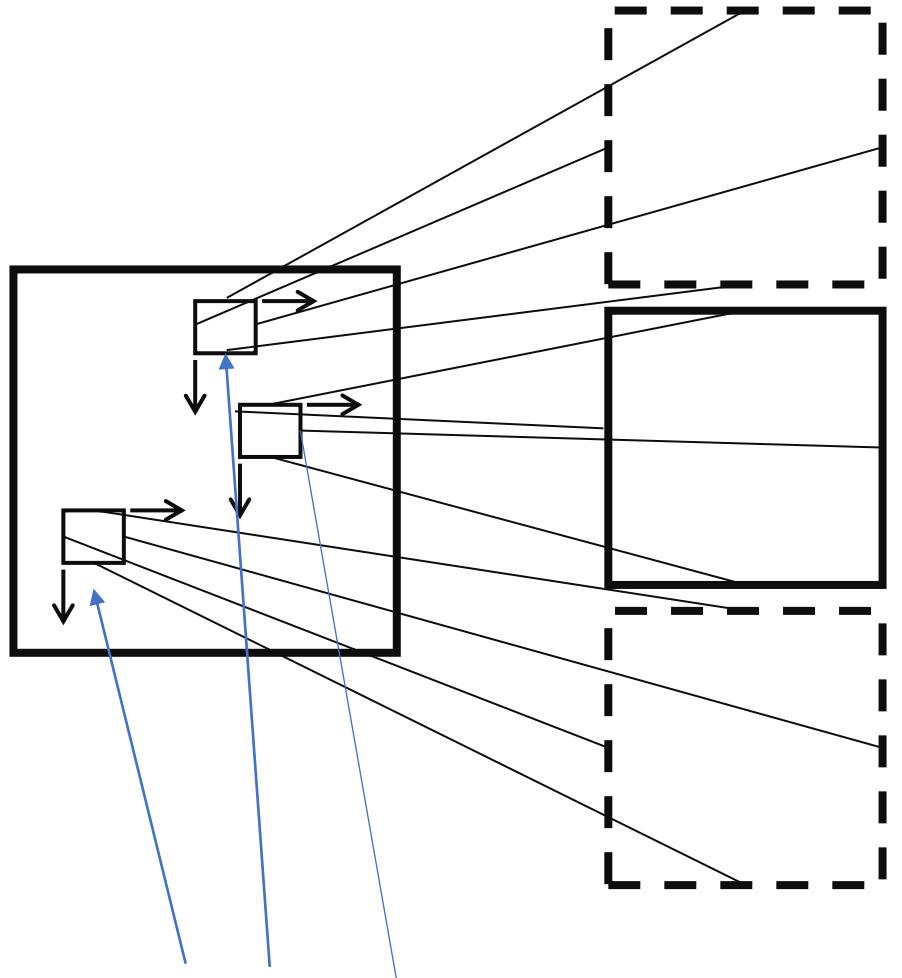
1	1	1
1	4	1
1	1	1

Smoothing/Noise-reduction

1	0	-1
1	0	-1
1	0	-1

(Vertical) Edge detection

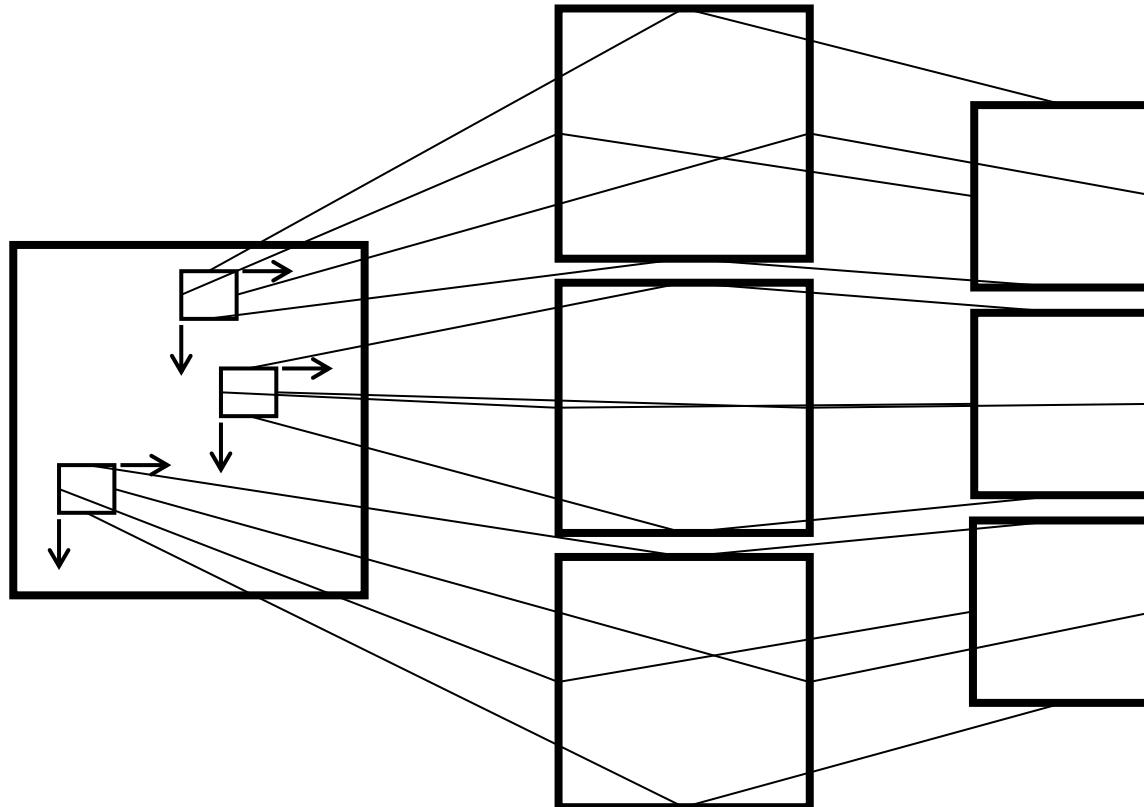
2D Convolutional Neuron



The number of kernels defines the number of *features maps (channels)*.

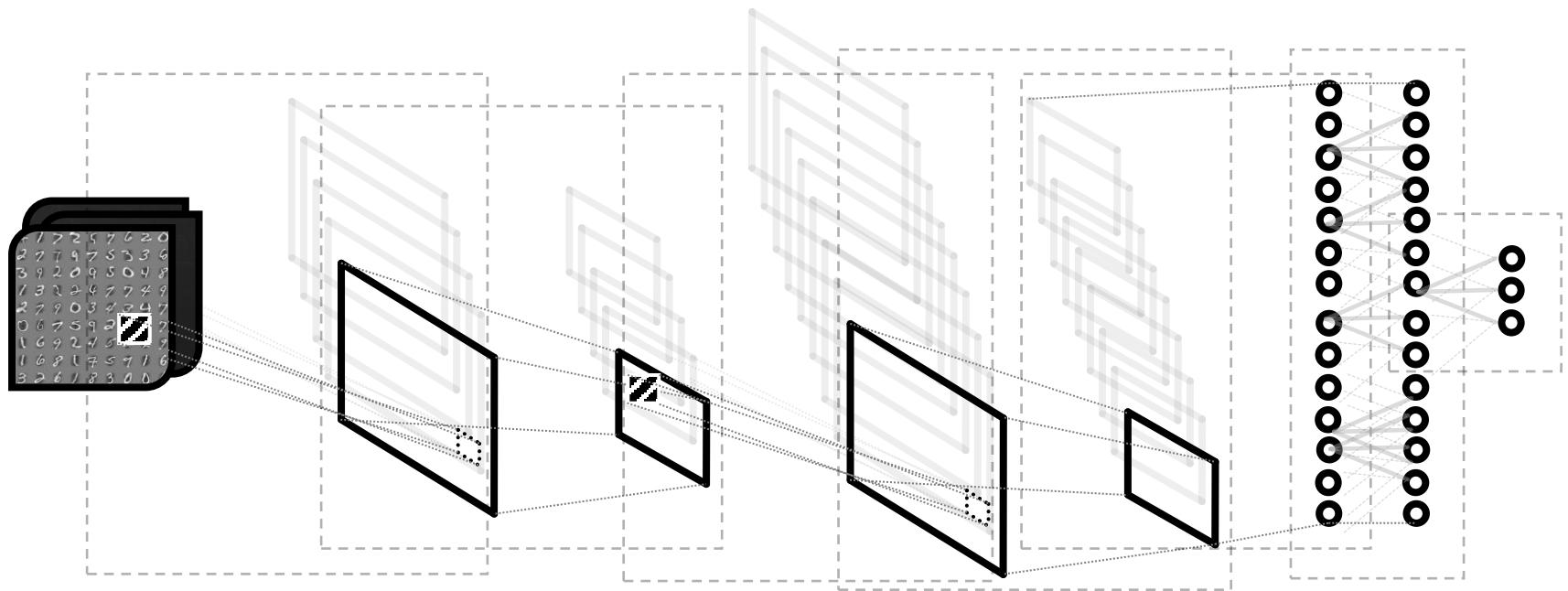
The sizes of the kernels define the *receptive fields*.

Convpool Layer



Convolution, pooling, and going through some activations

Illustrating A Simple CNN



Some convpool layers plus some fully-connected layers

CNN Examples - Different Complexities



| LeNet

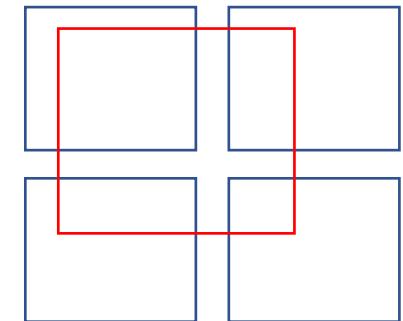
| AlexNet

LeNet

| Each pixel in layer 3 corresponds to 7/3 of a pixel in the input Second level

| Receptive field of layer 1 is 5X5

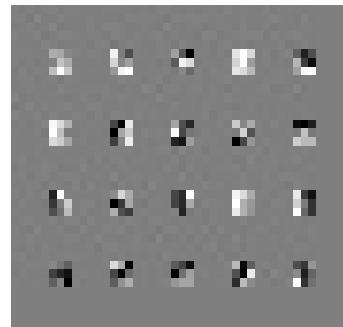
Layer Number	Input Shape	Receptive Field	Number of Feature Maps	Type of Neuron
1	28 X 28 X 1	5 X 5	20	Convolutional
2	24 X 24 X 20	2 X 2		Pooling
3	12 X 12 X 20	5 X 5	50	Convolutional
4	8 X 8 X 50	2 X 2		Pooling
5	800	1 X 1	500	Fully Connected
6	500		10	Softmax



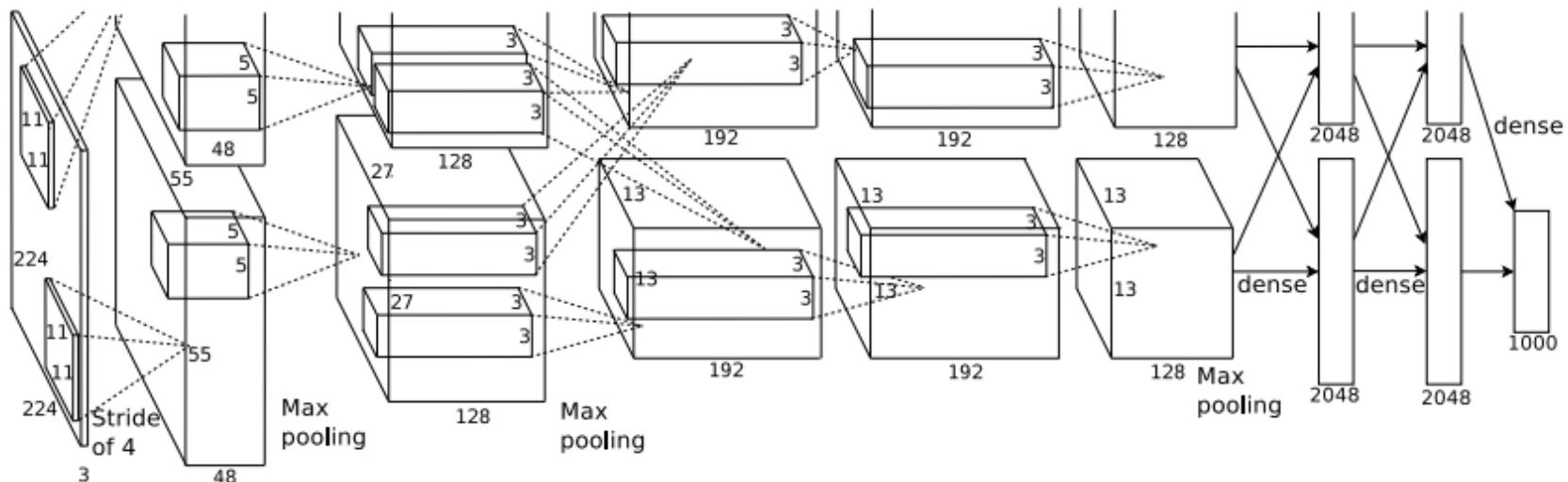
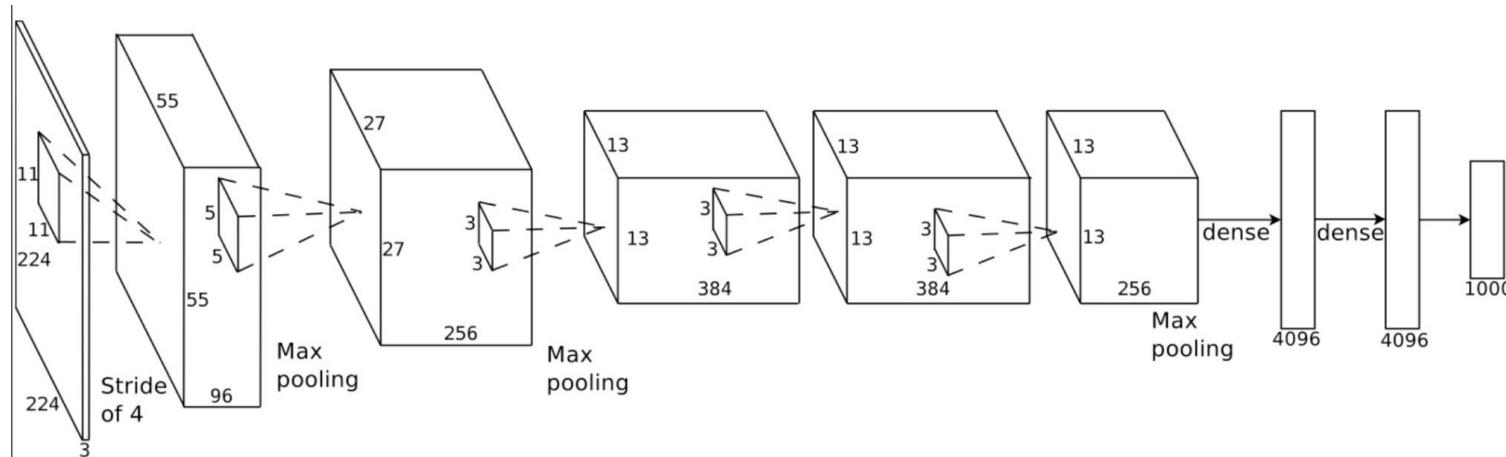
Case Study: LeNet

- | This is after training the network for 75 epochs with a learning rate of 0.01
- | Produces an accuracy of 99.38% on the MNIST dataset.

278203380
247197072
928407514
331966928
567931904
911346194
122044312
431874139
833024017



Case Study: AlexNet



Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

Case Study: AlexNet 1/3

Layer Number	Input Shape	Receptive Field	Number of Kernels	Type of Neuron
1	224 X 224 X 3	11 X 11, stride 4	96	Convolutional
2		3 X 3, stride 2		Pooling
3	55 X 55 X 96	5 X 5	256	Convolutional
4		3 X 3, stride 2		Pooling
5	13 X 13 X 256	3 X 3, padded	384	Convolutional
6	13 X 13 X 384	3 X 3, padded	384	Convolutional
7	13 X 13 X 384	3 X 3	256	Convolutional
8	43264	1 X 1	4096	Fully Connected
9	4096	1 X 1	4096	Fully Connected
10	4096		1000	Softmax

| Receptive field of the layer 7 is

~ 52 pixels !! which is almost as big as an object part (about one – fourth of the input image)

Case Study: AlexNet 2/3



Imagenet -15 million images in over 22,000 categories

(ILSVRC), used about 1000 of these categories

Imagenet categories are much more complicated than other datasets

- Often difficult even for humans to categorize perfectly
- Average human-level performance is about 96% on this dataset

AlexNet was the earliest systems to break the 80% mark

- Non-neural conventional techniques were unable to achieve such performance

Case Study: AlexNet 3/3



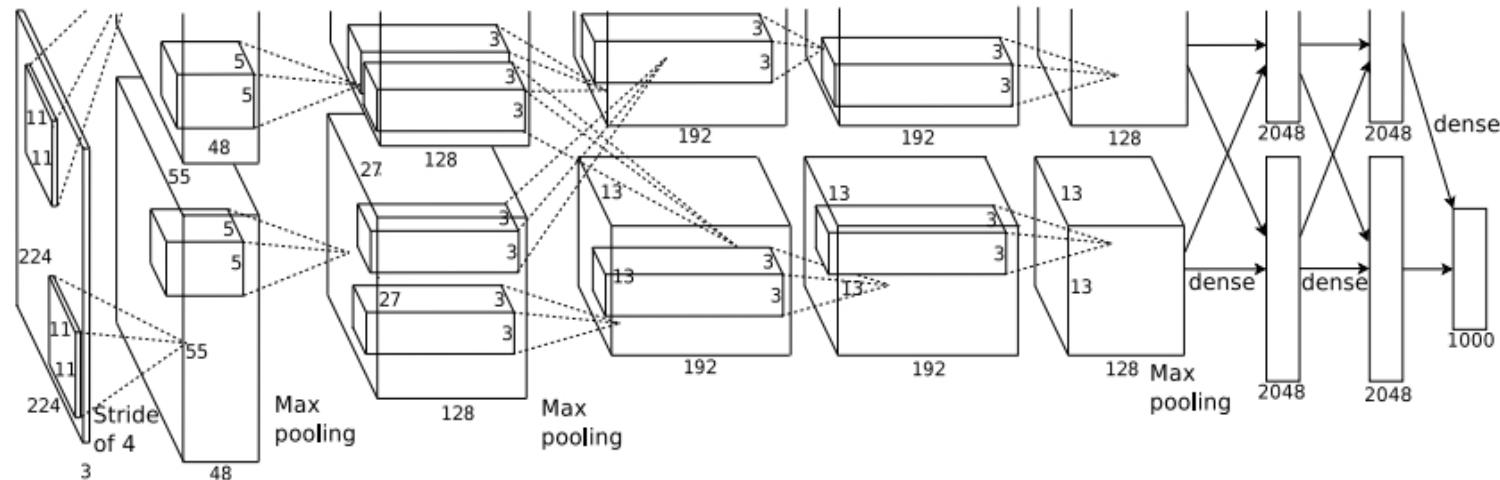
AlexNet was huge at the time.

- The size could lead to instability during training or inability to learn, if without proper regularization

Some techniques were used to make it trainable

- AlexNet was the first prominent network to feature ReLU
- Features multi-GPU training (originally trained the networks on two Nvidia GTX 580 GPUs with 3GB)

Case Study: AlexNet Filters



Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

CNN Recap



| The CNNs are similar to the basic MLP architecture illustrated earlier, but some key extensions include:

- The concept of weight-sharing through kernels
- Weight-sharing enables learnable kernels, which in turn define feature maps
- The idea of pooling

Auto-encoder 1/4

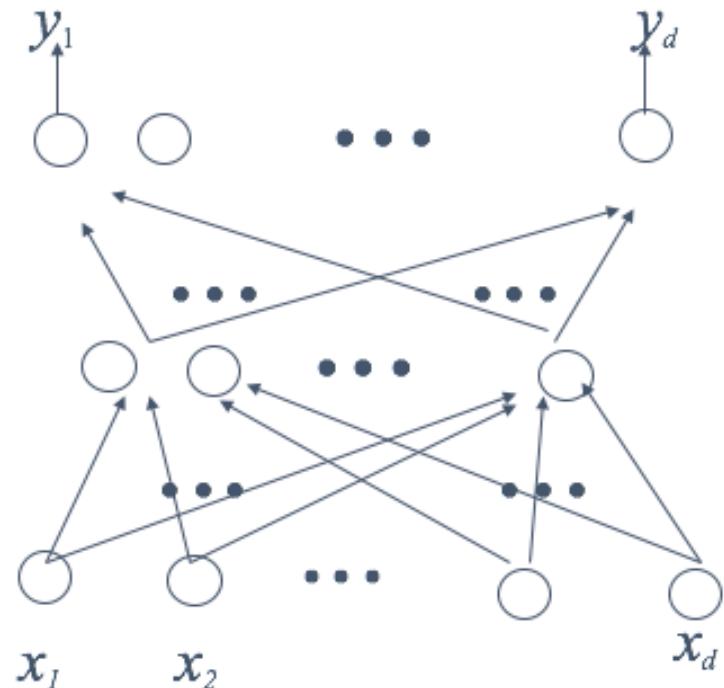
| Networks seen thus far are all trained via supervised learning

| Sometimes we may need to train a network without supervision:

→ Unsupervised learning

| Auto-encoder is a such example

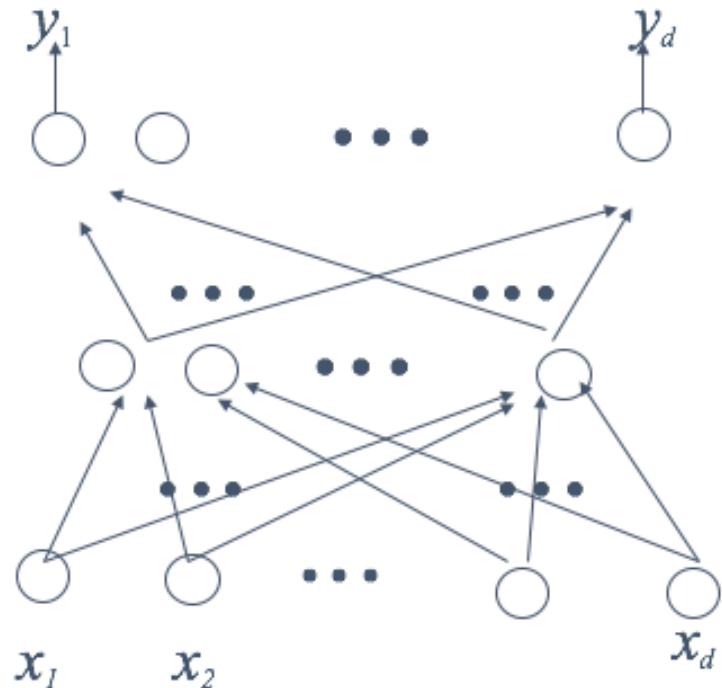
– Consider y_i being an approximation of x_i .



Auto-encoder 2/4

| Perfect auto-encoder
would map x_i to x_i

| Learn good
representations in the
hidden layer



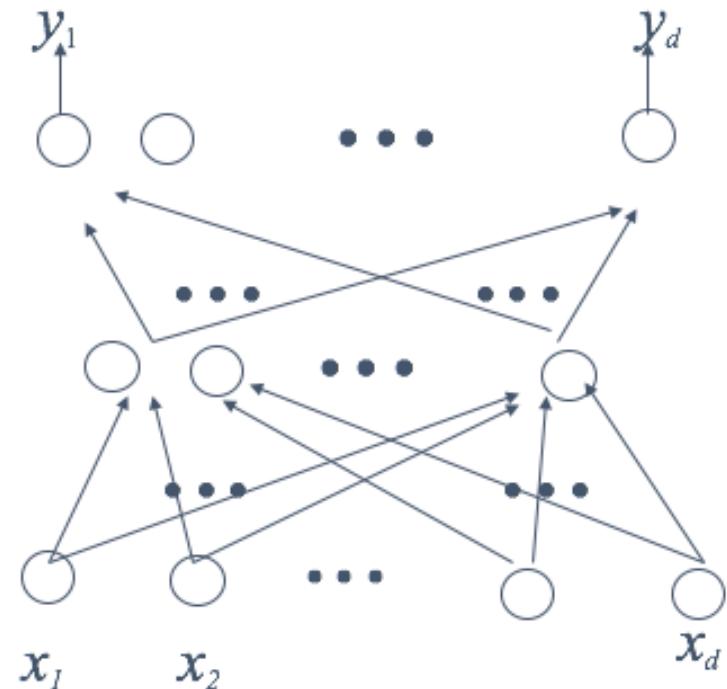
Auto-encoder 3/4

| Consider two cases

- Much fewer hidden nodes than input nodes
- Many hidden nodes or more hidden nodes than input nodes

| Case 1: Encoder for compressing input and compressed data should still be able to reconstruct the input

- Similar to, e.g., PCA



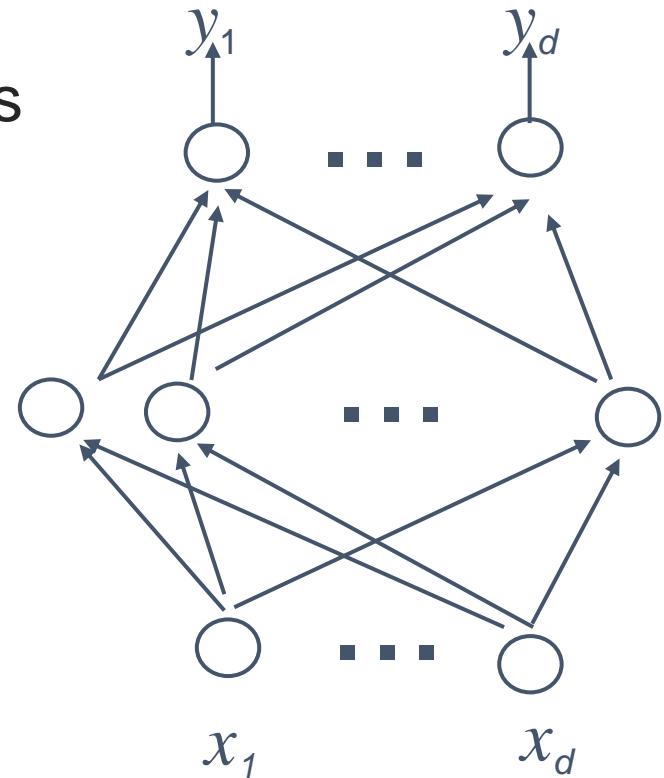
Auto-encoder 4/4

Consider two cases

- Fewer hidden nodes than input nodes
- More hidden nodes than input nodes

Case 2: Allow more hidden nodes than input

- Allow more freedom for the input-to-hidden layer mapping in exploring structure of the input
- Additional “regularization” will be needed in order to find meaningful results



Recurrent Neural Networks (RNNs) 1/4



| *Feedforward networks:* Neurons are interconnected without any cycle in the connection

| *Recurrent neural networks:* Allow directed cycles in connections between neurons

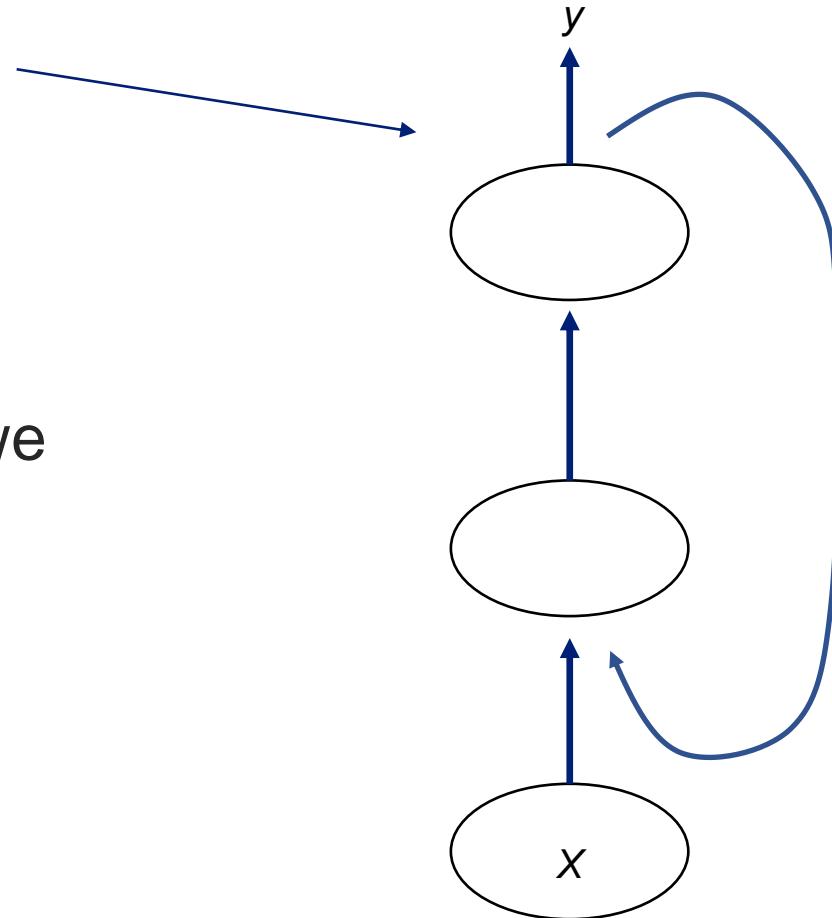
- Notion of “state” or temporal dynamics
- Necessity of internal memory

| One clear benefit: Such networks could naturally model variable-length sequential data

Recurrent Neural Networks (RNNs) 2/4

A basic, illustrative architecture for RNN
(showing only one node each layer)

- **QUESTION:** What is this network equivalent to, if we “unfold” the cycles for a given sequence of data?



Recurrent Neural Networks (RNNs) 3/4



- | Training with BP algorithm may suffer from so-called *vanishing gradient problem*
- | Some RNN variants have sophisticated “recurrence” structures, invented in part to address such difficulties faced by basic RNN models

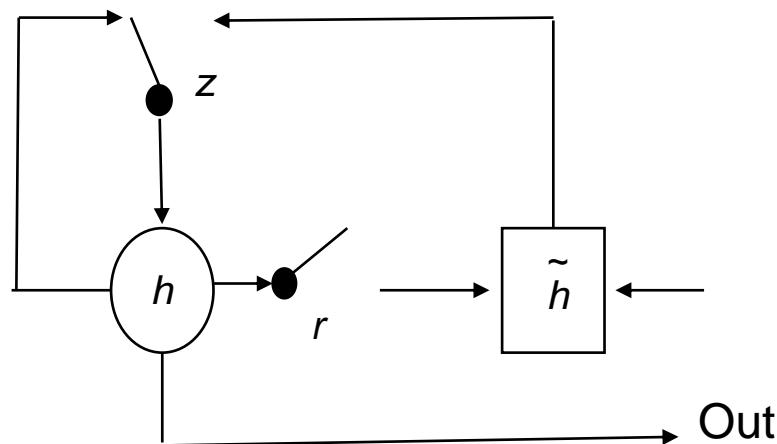
Recurrent Neural Networks (RNNs) 4/4

| Examples:

| The “Long short-term memory” (LSTM) model

- used to produce state-of-the-art results in speech and language applications

| The Gated Recurrent Unit model, illustrated here:





Exemplar Deep Learning Applications

Image Classification

Objective



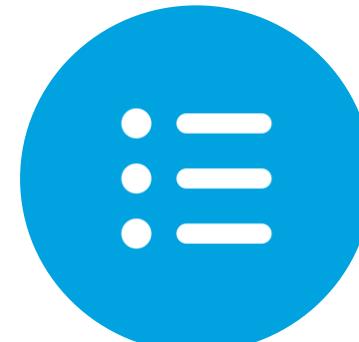
Objective

Describe an example
network for image
classification



Objective

Explain the
parameters defining
the network



Objective

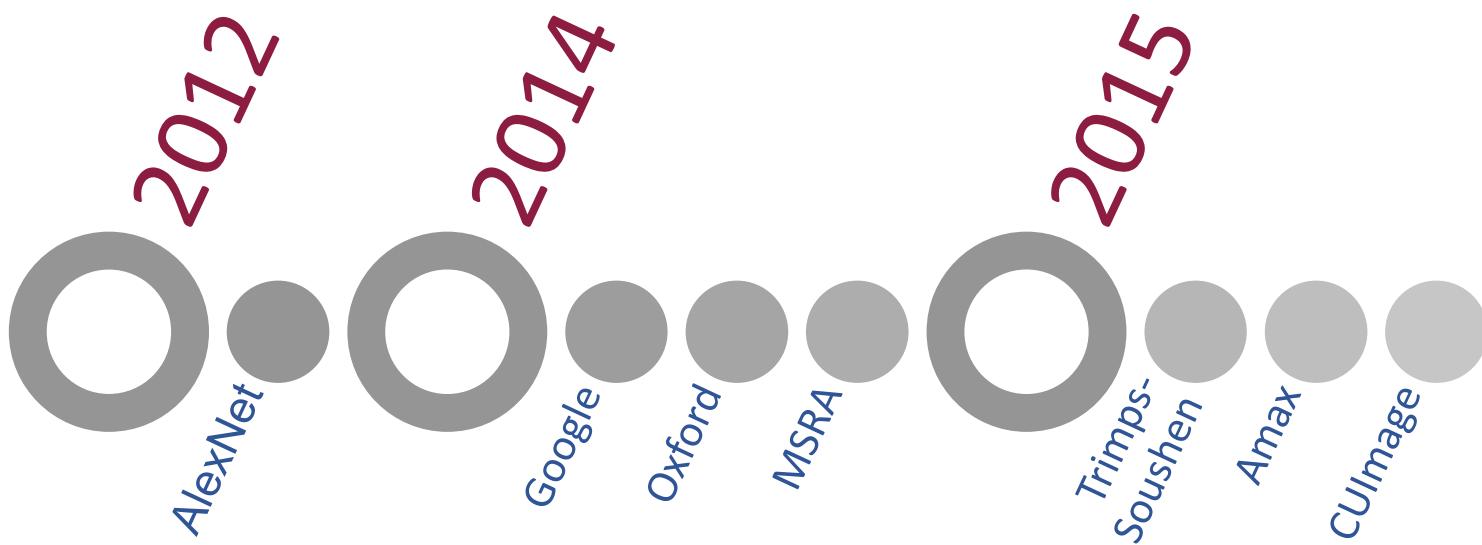
Identify common
tricks for improving
classification
performance

Deep Learning for Image-based Recognition



- | Visual recognition is an important part of human intelligence.
- | ILSVRC (ImageNet Large-scale Visual Recognition Challenge) illustrates such a task.
- | Many ImageNet images are difficult for conventional algorithms to classify.

Success Stories



ImageNet.org Samples

Golden retriever

An English breed having a long silky golden coat

1607 pictures

64.99%
Popularity
Percentile



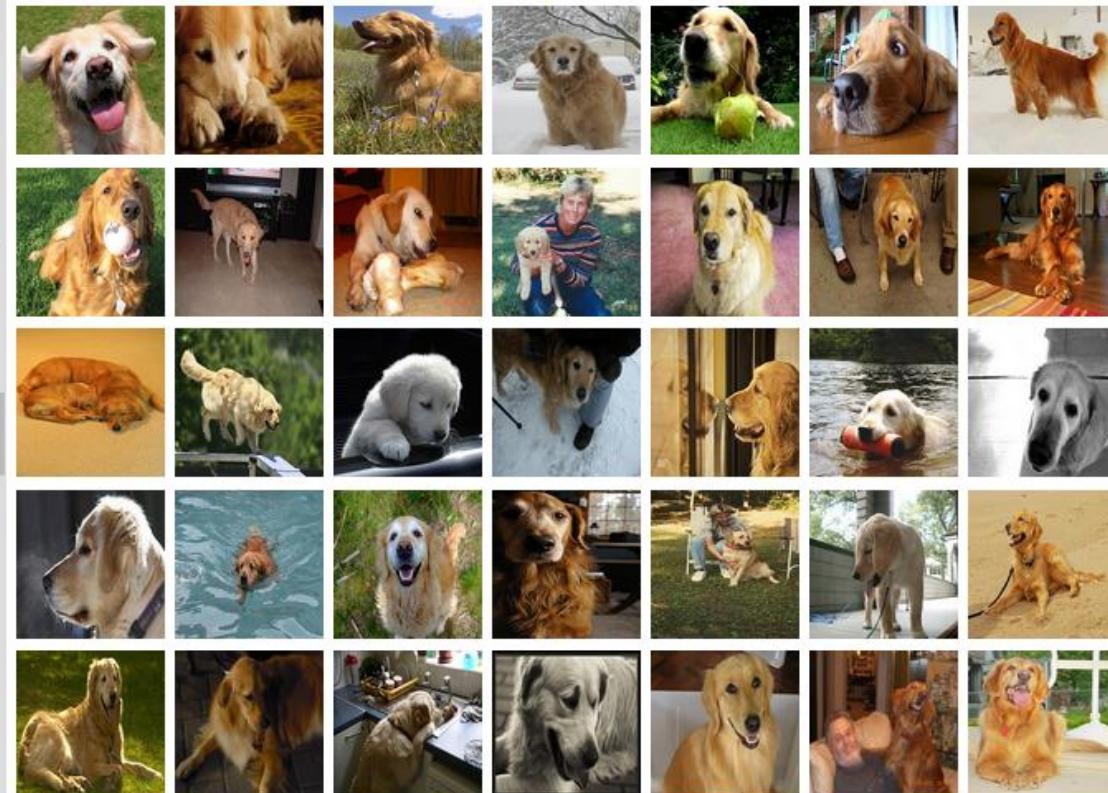
Numbers in brackets: (the number of synsets in the subtree).

- ↳ ImageNet 2011 Fall Release (32326)
 - ↳ plant, flora, plant life (4486)
 - ↳ geological formation, formation (1)
 - ↳ natural object (1112)
 - ↳ sport, athletics (176)
 - ↳ artifact, artefact (10504)
 - ↳ fungus (308)
 - ↳ person, individual, someone, som
↳ animal, animate being, beast, bru
 - ↳ invertebrate (766)
 - ↳ homeotherm, homoiotherm, ho
 - ↳ work animal (4)
 - ↳ darter (0)
 - ↳ survivor (0)
 - ↳ range animal (0)
 - ↳ creepy-crawly (0)
 - ↳ domestic animal, domesticate
 - ↳ domestic cat, house cat, Fe
 - ↳ dog, domestic dog, Canis f:
 - ↳ pooch, doggie, doggy, b: - ↳ hunting dog (101)
 - ↳ sporting dog, gun do
 - ↳ pointer, Spanish p:
 - ↳ setter (3)
 - ↳ bird dog (0)
 - ↳ spaniel (11)
 - ↳ griffon, wire-haire
 - ↳ water dog (0)
 - ↳ retriever (5)
 - ↳ golden retriever

Treemap Visualization

Images of the Synset

Downloads



*Images of children synsets are not included. All images shown are thumbnails. Images may be subject to copyright.

Prev 1 2 3 4 5 6 7 8 9 10 ... 67 68 Next

SOURCE: ImageNet.org

Success Stories: 2014 – Top Three

Rank	Team	Error
1	Google	0.06656
2	Oxford	0.07325
3	MSRA	0.08062

Success Stories: 2015 – Top Three

Team Name	Entry Description	Description of Outside Data Used	Localization Error	Classification Error
Trimps-Soushen	Extra annotations collected by ourselves	Extra annotations collected by ourselves	0.122285	0.04581
Amax	Validate the classification model we used in DET Entry1	Share proposal procedure with DET for convenience	0.14574	0.04354
CUIimage	Average multiple models – validation accuracy is 79.78%	3000-class classification images from ImageNet are used to pre-train CNN	0.198272	0.05858

Example Application 1: DR Detection

DR: Diabetic Retinopathy

A recent work: **Gulshan et al.** "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs." *JAMA* 316.22 (2016): 2402-2410

- Employed large datasets
- A specific CNN architecture (Inception-v3) taking the entire image as input (as opposed to lesion/structure-specific CNNs)
- High performance: Comparable to a panel of 7 board-certified ophthalmologists



Example Application 2: Visual Aesthetics



| While being subjective, computational modes are possible since there are patterns in visually-appealing pictures.

- E.g., photographic rules.

| Huge on-line datasets available. If ratings are also available, the problem becomes supervised learning.

- Conventional approaches still face the bottleneck of feature extraction.

Example Application 2: Visual Aesthetics



| While being subjective, computational modes are possible since there are patterns in visually-appealing pictures.

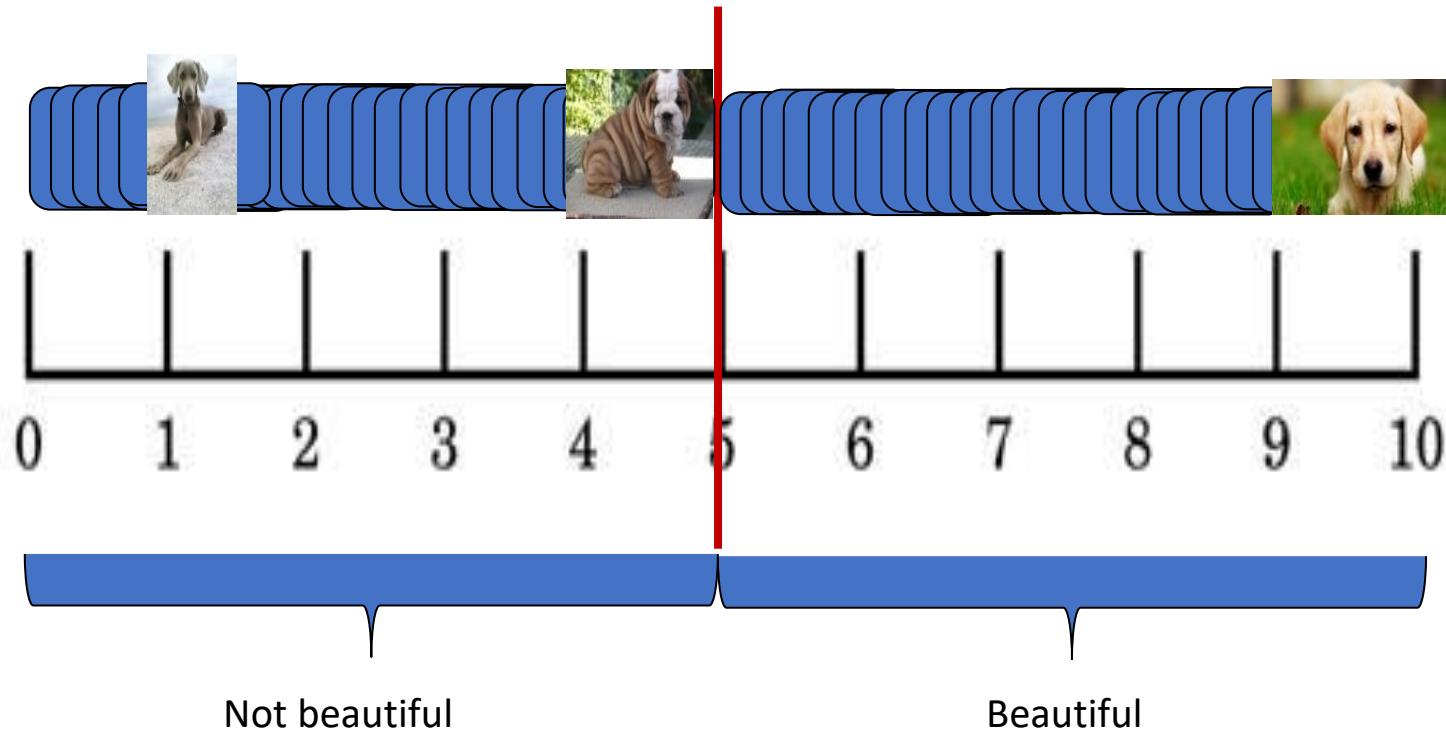
- E.g., photographic rules.

| Huge on-line datasets available. If ratings are also available, the problem becomes supervised learning.

- Conventional approaches still face the bottleneck of feature extraction.

Related Approaches

| Solving the task as binary classification



Related Approaches: Examples



- | *RAPID: Rating Pictorial Aesthetics using Deep Learning (Lu et al.)*
- | *Deep Multi-Patch Aggregation Network for Image Style, Aesthetics, and Quality Estimation (Lu et al.)*
- | *Image Aesthetic Evaluation Using Paralleled Deep Convolution Neural Network (Guo & Li)*

A New Task: Relative Aesthetics



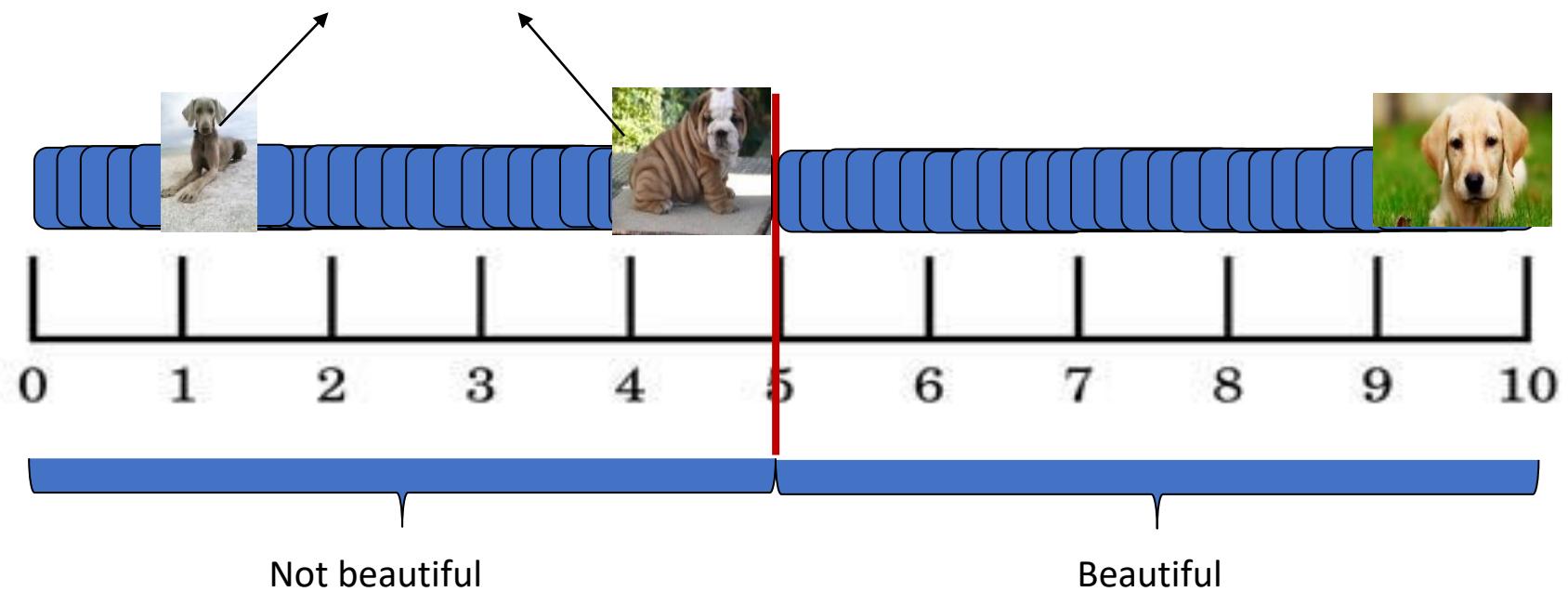
| Image retrieval

| Image enhancement

Which is more beautiful?



Most approaches do not solve



A Deep Learning Approach



- | Dual-channeled CNN trained using relative learning
- | Siamese Network characteristics (weight sharing) and hinge-loss function
- | A custom data-set with relative labels – pairs formed based on aesthetic rating

SOURCE: Gattupalli et al. "A Computational Approach to Relative Aesthetics", *International Conference on Pattern Recognition (ICPR)* 2016.

Constructing a Useful Data Set 1/2

| Total of 250,000 images
extracted from
dpchallenge.com



| Challenges under which
users post their submission



| Peers rate and a final winner
is selected based on the
average rating



| Belong to a wide variety of
semantic categories



Constructing a Useful Data Set 2/2

| The minimum gap between the average rating of the two images is one

–e.g., 3.4 and 4.5, 6.3, and 7.8



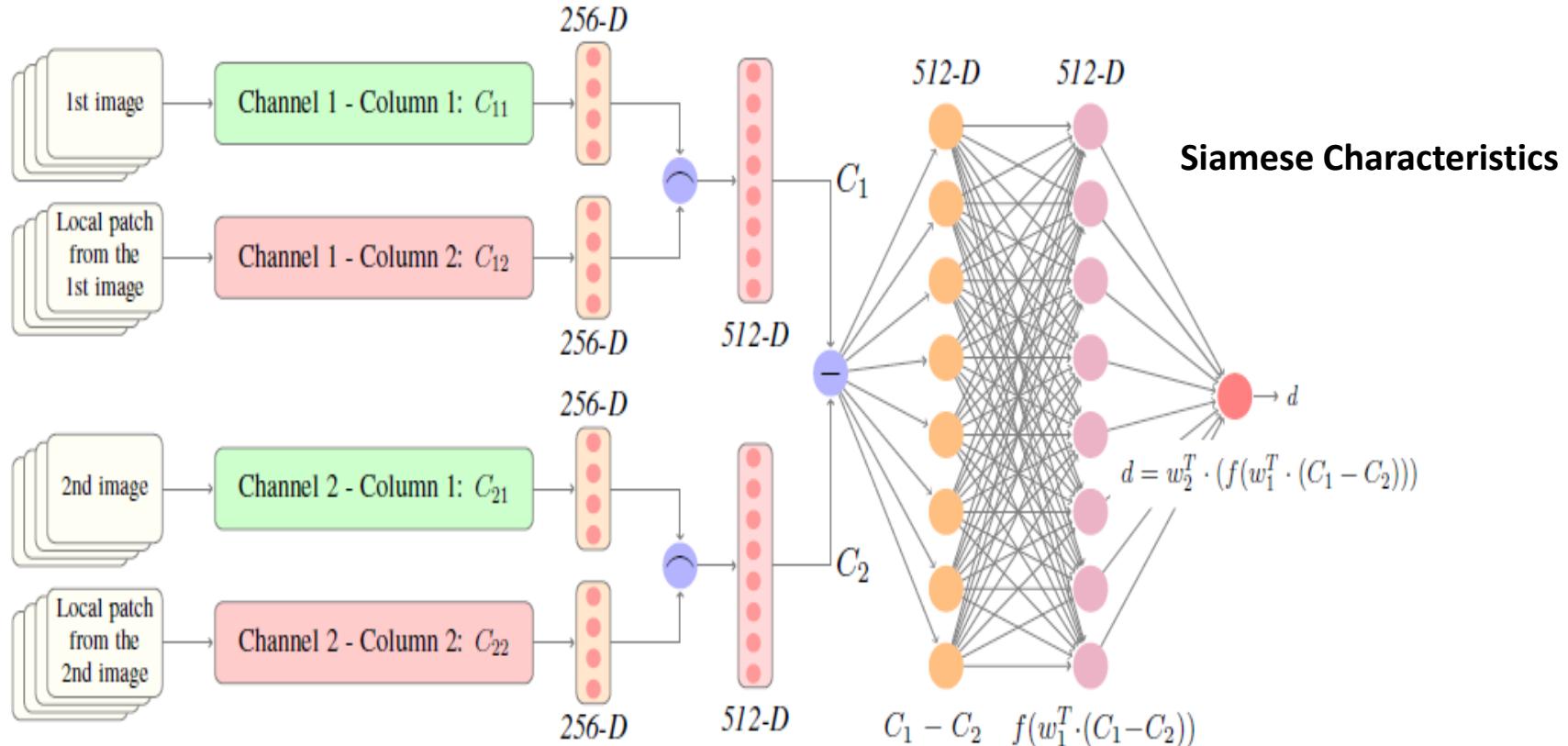
| The maximum variance allowed between the ratings of different voters is 2.6

| Pick pairs from the same category only

–e.g., cannot compare an image of a car and a building



The Network Architecture & Other Characteristics



Padded Input	Conv	Max-pooling	Conv	Max-pooling	Conv	Conv	Dropout	Dense	Dropout	Dense	Dropout
$3 \times 230 \times 230$	$2, 64, 11, 2$	2×2	$1, 64, 5, 1$	2×2	$1, 64, 3, 1$	$- , 64, 3, 1$	0.5	1000	0.5	256	0.5

Further Implementation Details



- | Each channel contains two streams of processing: column 1 for global, and column 2 for local
- | Global Patch
 - e.g., rule of thirds, golden ration
- | Local Patch
 - e.g., smoothness/graininess

The Loss Function



$$L = \max(0, \delta - y \cdot d(I_1, I_2)) \quad \longrightarrow \text{Hinge Loss}$$

$$d(I_1, I_2) = f(C_1 - C_2)$$

| where,

y = True label of the image pair,

i.e., 1 if $I_1 > I_2$ and

-1 otherwise

| C_1, C_2 = Outputs of channel 1 and channel 2 respectively

Sample Results



| Two ways of training

- Using binary labels
- Using relative labels

| Tested for two tasks

- For Binary Classification task
- For Ranking task

Eight Experiments Total

	Ranking (custom test-set)	Ranking (standard test-set)	Classification (custom test- set)	Classification (standard test-set)
Base-line	62.21	65.87	59.92	69.18
Relative aesthetics	70.51	76.77	59.41	71.60





Exemplar Deep Learning Applications

Video-Based Inference

Objective



Objective

Describe unique challenges in using deep networks for sequential data



Objective

Describe the difference between image-based and video-based classification tasks



Objective

Explain the value of using video action recognition to contrast the difference between image-based and video-based classification tasks



Objective

Evaluate a video-based classification example using deep learning

Going from Image to Video



Solution

Naïve



Solution

Better

| Processing each frame
of a video as an
independent image and
then aggregating the
frame-level results

| Extracting spatio-
temporal features and
an inference task will
be based on such
features

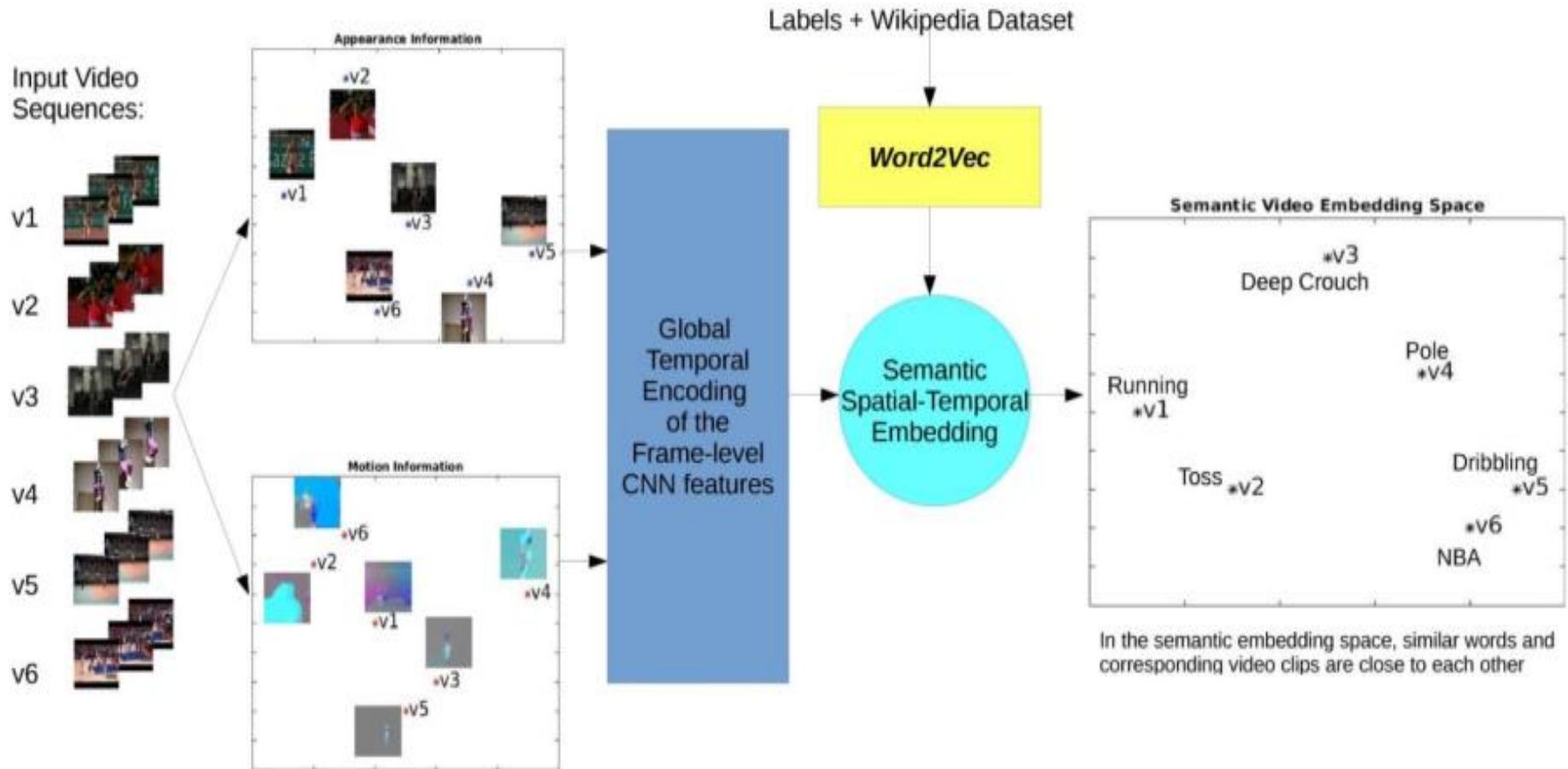
Video2Vec: Sample Applications



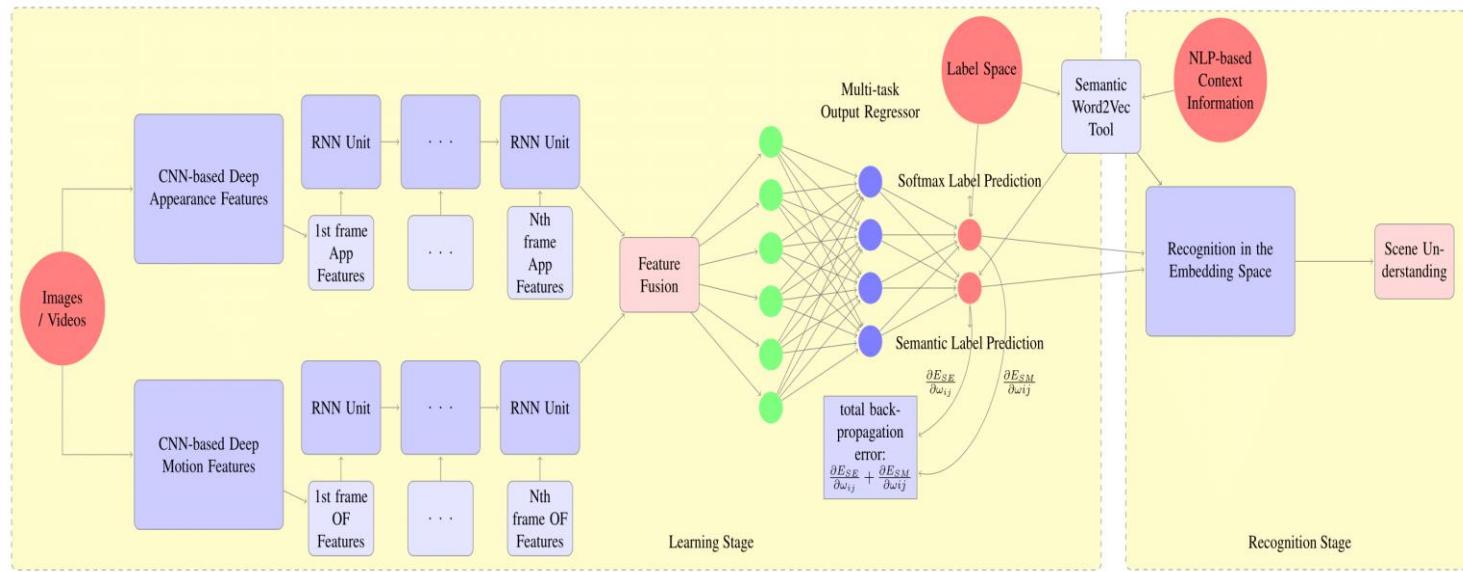
- | We examine a deep learning approach for finding video representations that naturally encode spatial-temporal semantics.
- | Mostly based on the following papers:

- Yikang Li, Sheng-hung Hu, Baoxin Li, “Recognizing Unseen Actions in a Domain-Adapted Embedding Space”, ICIP, Sep 2016.
- Yikang Li, Sheng-hung Hu, Baoxin Li, “*Video2Vec: Learning Semantic Spatio-Temporal Embeddings for Video Representations*”, ICPR, Dec 2016.

Video2Vec Deep Learning Model: Key Idea



Video2Vec Deep Learning Model: Implementation



- | A two-stream CNN for extracting appearance and optical flow features
- | RNNs for further global spatial-temporal encoding
- | A MLP for final semantic embedding space

Applications of the Model



| Visual tasks:

- Video Action Recognition
- Zero-Shot Learning
- Semantic Video Retrieval

| Dataset: UCF101 dataset (13320 video clips from 101 categories; training/testing ratio is 7:3; the split list is provided by its own web)

Additional Implementation Details 1/4



| Pretraining for the component models:

- **Pre-trained Spatial CNN Model:** VGG-f trained on ImageNet
- **Pre-trained OF CNN Model:** Flow-net trained on UCF Sports
- **Pre-trained Word2Vec Model:** Wikipedia corpus contained 1 billion words

Additional Implementation Details 2/4



| Deep model parameter settings:

- **CNNs:** Pretrained model + the last layer (fc7) features (dimension: 4096x1)
- **RNNs:** Hidden layer size is 1024x1
- **MLP:** Input layer size (2048x1), hidden layer size (1200x1), output layer size (500x1)

| Loss function:

- Hinge loss function for semantic embedding
- Softmax loss function for fine-tuning and classification

Additional Implementation Details 3/4



| Video processing settings:

- Dense Optical Flow and RGB frames are extracted at 10fps.
- Building Video Sequence Mask for each training batch to make each sequence the same length.

Additional Implementation Details 4/4



| Training parameter settings:

- Learning rate: initialized as 0.0001 and reduced by half each 15 epochs
- Total epoch: 60 epochs
- Batch size: 30 video clips
- Margin value for Hinge Loss function:
 - a. For zero-shot learning, 0.4
 - b. For video retrieval and action recognition, 0.55

Summaries of Key Results



| Dataset: UCF101 dataset

Zero-shot learning results

- . The model achieved state-of-the-art performance on ZSL even without any domain-adapted strategy.

Video action recognition

- . The performance was on par with those with sophisticated fusion strategies or deeper networks.

Additional Results

- | The task is to retrieve videos from training dataset by using query words that never appear in the training stage but share some information with training labels.
- | The results show the top 10 retrieval video clips among video dataset.

Query Labels	Top10 Retrieve Results	Query Labels	Top10 Retrieve Results
NBA	Basketball Dunk (10)	Extreme	Rock Climbing Indoor (5), Uneven Bars (2), Soccer Juggling (2), Pole Vault (1)
Orchestra	Playing Cello (9), Playing Piano (1)	Tide	Cliff Diving (4), Surfing (2), Throw Discus (2), Sky Diving (1), Rafting (1)
Army	Military Parade (10)	India	Paying Tabla (4), Playing Sitar (2), Head Massage (1), Cricket Shot (1), Mixing (1)
Music	Playing Sitar (9), Playing Piano (1)	Celebrate	Military Parade (6), Long Jump (1), Band Marching (1), Ice Dancing (1), Blowing Candles (1)
Computer	Typing (10)	Home-run	Baseball Pitch (5), Basketball Dunk (3), Field Hockey Penalty (1), Frisbee Catch (1)
Park	Biking (9), Golf Swing (1)	Boat	Kayaking (4), Rafting (2), Rowing (2), Cliff Diving (1), Push Ups (1)
Summit	Cliff Diving (7), Skiing (2), Rope Climbing (1)	Toy	Yo-yo (4), Nun chucks (4), Pull Ups (1), Juggling Ball (1)
School	Skate Boarding (10)	Snow	Skiing (2), Ice Dancing (2), Cricket Bowling (1), Pole Vault (1), Blowing Candles (1), Blow Dry Hair (1), Rafting (1), Sky Diving (1)
Park	Biking (9), Golf Swing (1)	Acrobatics	Juggling Balls (5), Soccer Juggling (5)
Water	kayaking (10)	Ocean	Cliff Diving (4), Sky Diving (3), Kayaking (2), Rafting (1)
FIFA	Soccer Penalty (8), Soccer Juggling (2)	Hurl	Throw Discus (2), Mopping Floor (2), Baby Crawling (1), Javelin Throw (1), Cricket Shot (1), Blowing Candles (1), Pull Ups (1)
Club	Golf Swing (8), Soccer Juggling (2)	Hiking	Biking (5), Kayaking (4), Rafting (1)
Nature	Tai Chi (7), Hammering (2), Walking with Dog (1)	Swim	Diving (5), kayaking (3), Cricket Bowling (1), Sky Diving (1)
Beethoven	Playing Cello (8), Playing Voilin (2)	Jogging	Biking (5), Skate Boarding (2), Soccer Juggling (1), Skiing (1), Ice Dancing (1)
Classical	Playing Cello (7), Playing Voilin (3)	Foam	Blowing Candles (7), Pull Ups (1), Rope Climbing (1), Juggling Balls (1)
Yankees	Baseball Pitch (10)	Hip-hop	Trampoline Jumping (6), Swing (4)
Duel	Boxing Punching Bag (8), Punch(2)	Scramble	Pull Ups (6), Trampoline Jumping (2), Rope Climbing (1), Cricket Shot (1)
Lifting	Body Weight Squats (4), Rope Climbing (4), Pull Ups (2)	Mat	Rope Climbing (4), Pommel Horse (3), Trampoline Jumping (2), Javelin Throw (1)
Martial	Fencing (3), Archery (3), Boxing Punching Bag (3), Balance Beam (1)	Parachuting	Diving (6), Cricket Bowling (2), Hand Stand Walking (1), Sky Diving (1)
Tumbling	Trampoline Jumping (8), Throw Discus (1), Frisbee Catch (1)	Hunting	Horse Riding (3), Kayaking (3), Nun chucks (3), Frisbee Catch (1)



Exemplar Deep Learning Applications

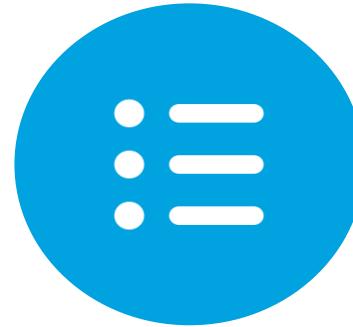
Generative Adversarial Networks (GANs)

Objective



Objective

Describe basic
concepts and
architecture for GANs



Objective

Illustrate variants of
GANs and their
applications

Generative Adversarial Networks (GANs)



| Proposed in 2014 by Goodfellow *et al.*

| An architecture with two neural networks gaming against each other.

- One attempting to learn a *generative model*

| Many variants have been proposed since the initial model.

Discriminative vs Generative Models 1/4

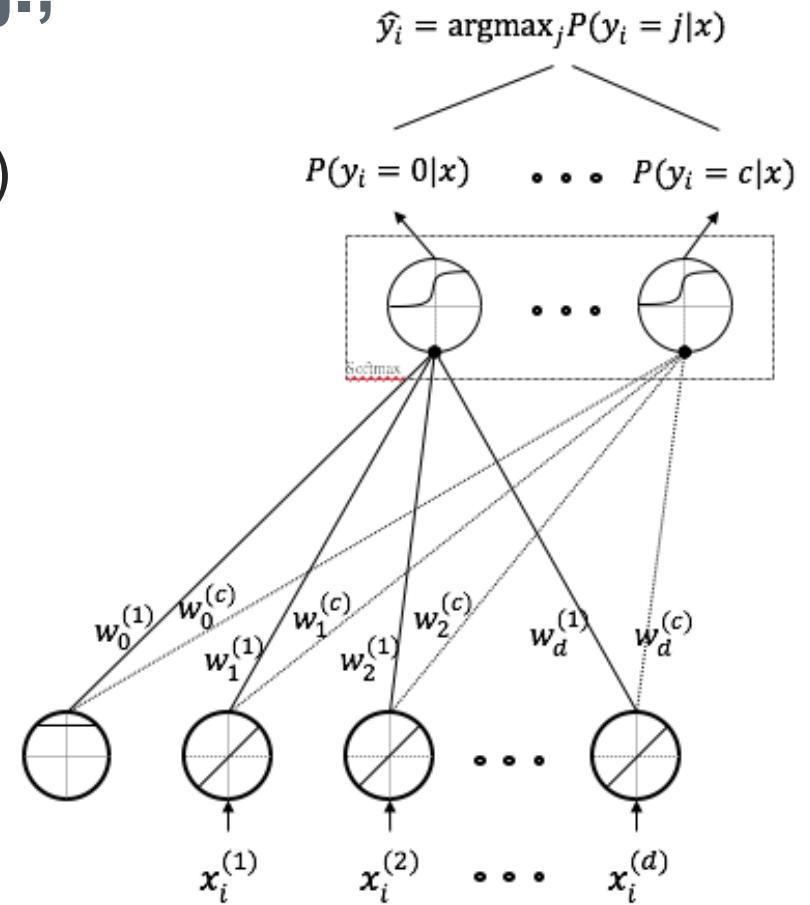
| Discriminative models: E.g.,
the familiar MLP

- Given $\{(x_i, y_i)\}$, to learn $P(y_i | x)$

| More generally, we try to
learn a *posterior
distribution* of y given x ,
 $p(y|x)$

- Usually reduced to posterior
probabilities for classification
problems

→ See also earlier discussion on
Naïve Bayes vs Logistic Regression.



Discriminative vs Generative Models 2/4



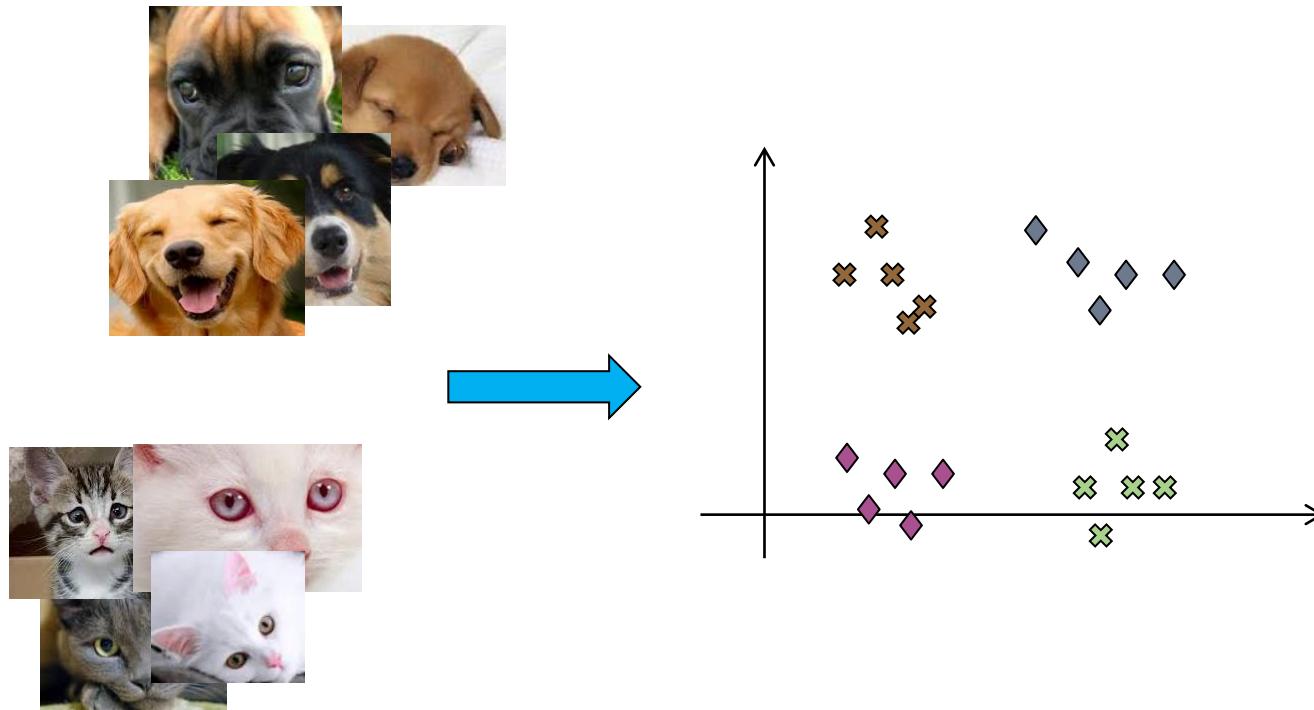
| Generative models think the other direction: how to generate x given y

- E.g., $x_i=?$ if $y_i=2$?

| More generally, we try to learn a *conditional distribution* of x given y , $p(x|y)$

Discriminative vs Generative Models 3/4

Illustrating the ideas



Discriminative vs Generative Models 4/4

| Estimating $p(x|y)$ (or, in general any $p(x)$, if we drop y by assuming it is given)

- Explicit density estimation: assuming some parametric or non-parametric models.
- Implicit density estimation: learn (essentially equivalent) models that may create good samples (as if from the “true” model), without explicitly defining the true model.

→ **GAN is such an approach**

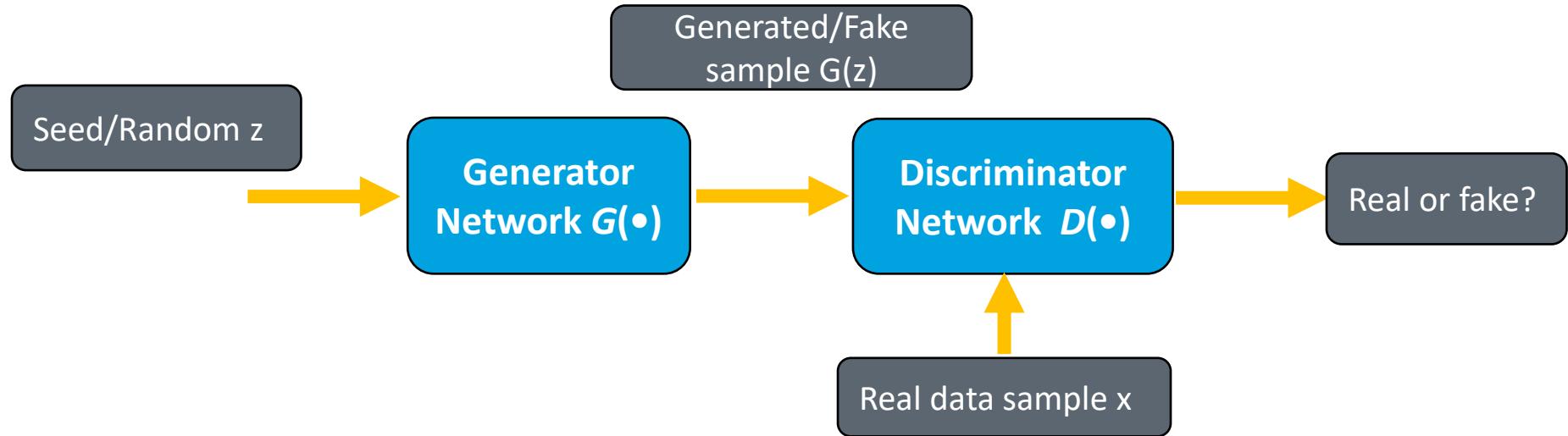
Discriminative vs Generative Models 4/4

| Estimating $p(x|y)$ (or, in general any $p(x)$, if we drop y by assuming it is given)

- Explicit density estimation: assuming some parametric or non-parametric models.
- Implicit density estimation: learn (essentially equivalent) models that may create good samples (as if from the “true” model), without explicitly defining the true model.

→ **GAN is such an approach**

Basic GAN Architecture



| **Objective of the Discriminator Network:**

making $D(x) \rightarrow 1$, $D(G(z)) \rightarrow 0$

| **Objective of the Generator Network:**

making $D(G(z)) \rightarrow 1$

Basic GAN Training Algorithm



for number of training iterations **do**

for k steps **do**

 Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise distribution $p_g(\mathbf{z})$

 Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data distribution $p_{data}(\mathbf{x})$

 Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m [\log D(\mathbf{x}^{(i)}) + \log(1 - D(G(\mathbf{z}^{(i)})))]$$

end for

 Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise distribution $p_g(\mathbf{z})$

 Update the generator by descending its stochastic gradient

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(\mathbf{z}^{(i)})))$$

end for

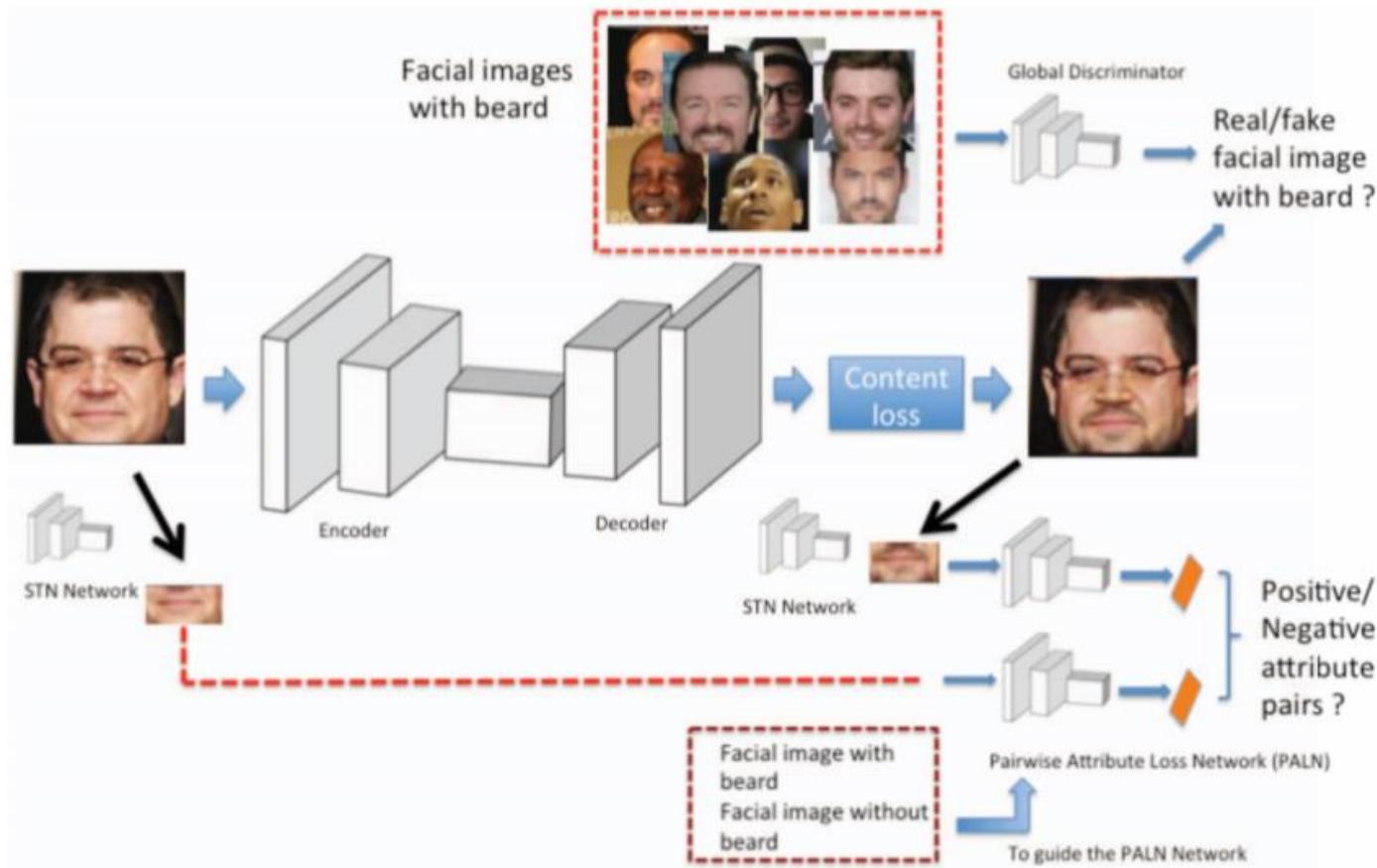
θ_d and θ_g are the parameters of the discriminator and generator respectively.

Applications of GAN



- | GAN enabled many novel/interesting/fun applications.
- | Many GAN-based models have been proposed, following the initial paper.
- | Consider one example: Facial attribute manipulation
 - Y. Wang *et al.* “*Weakly Supervised Facial Attribute Manipulation via Deep Adversarial Network*”, WACV 2018.

Facial Attribute Manipulation 1/2



SOURCE: Y. Wang et al. "Weakly Supervised Facial Attribute Manipulation via Deep Adversarial Network", WACV 2018.

Facial Attribute Manipulation 2/2



