STEVENS
INSTITUTE OF TECHNOLOGY
1870

Artificial Intelligence

Computer Science, CS541 - A

Jonggi Hong

Announcements

- HW #3
  - Due 11:59 pm, Tuesday, April 4

- Midterm project report
  - Grading in progress

STEVENS
INSTITUTE OF TECHNOLOGY
1870

Recap

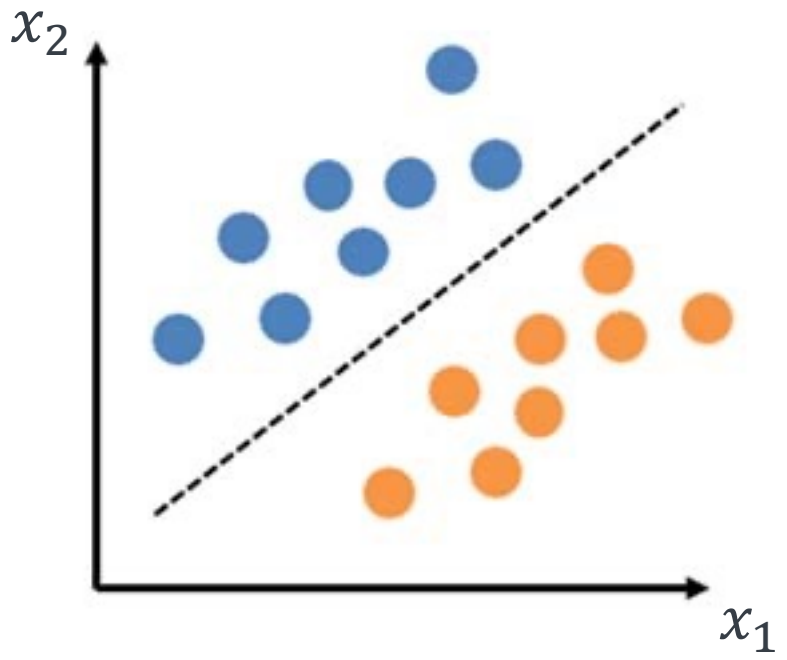
Support Vector Machine

Linear Threshold Classifiers

$$h_w(x) = \begin{cases} 1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} \geq 0 \\ -1 & \text{otherwise.} \end{cases}$$

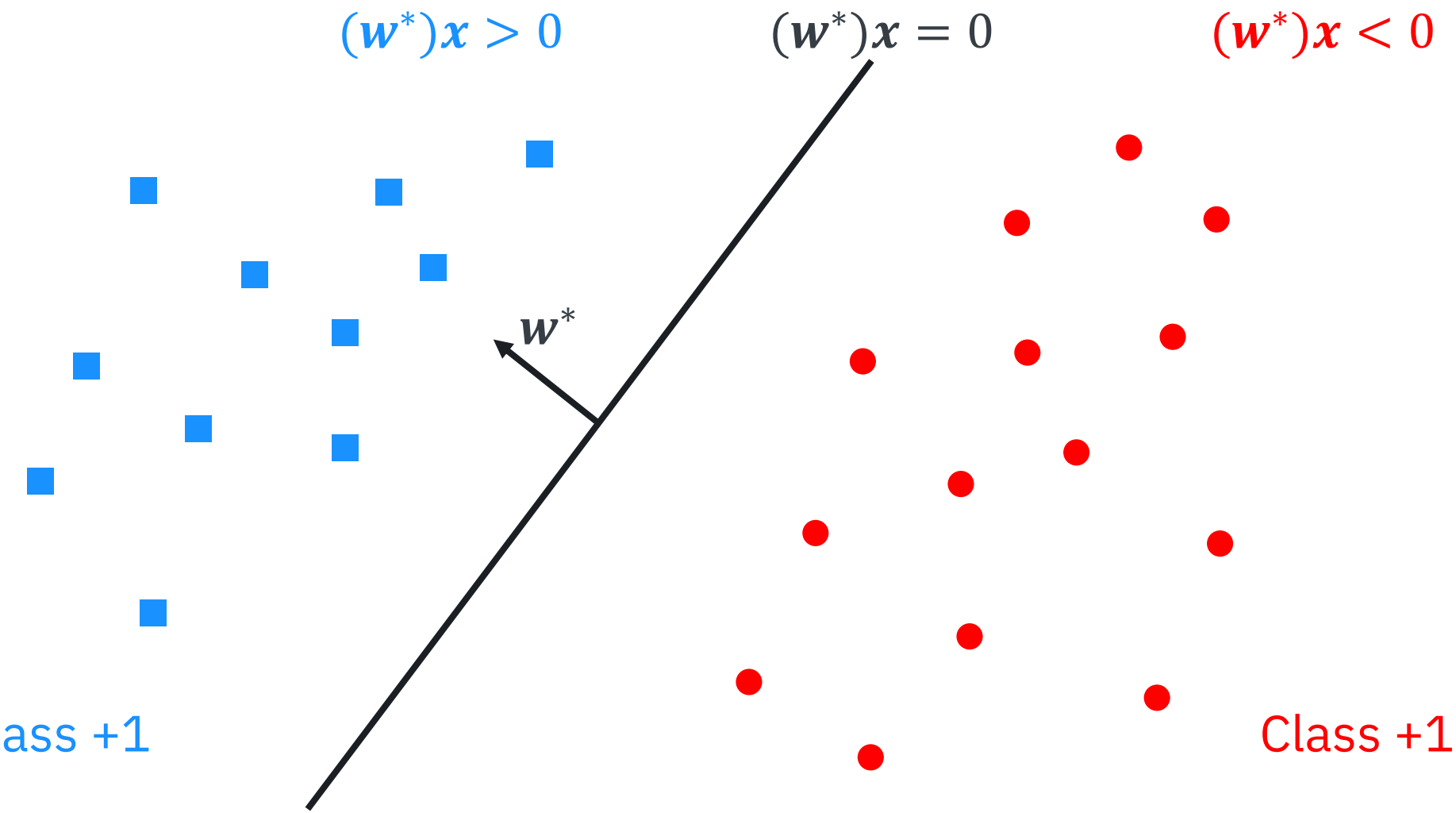
$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} = w_0 + w_1 x_1 + w_2 x_2$$

(bias term)

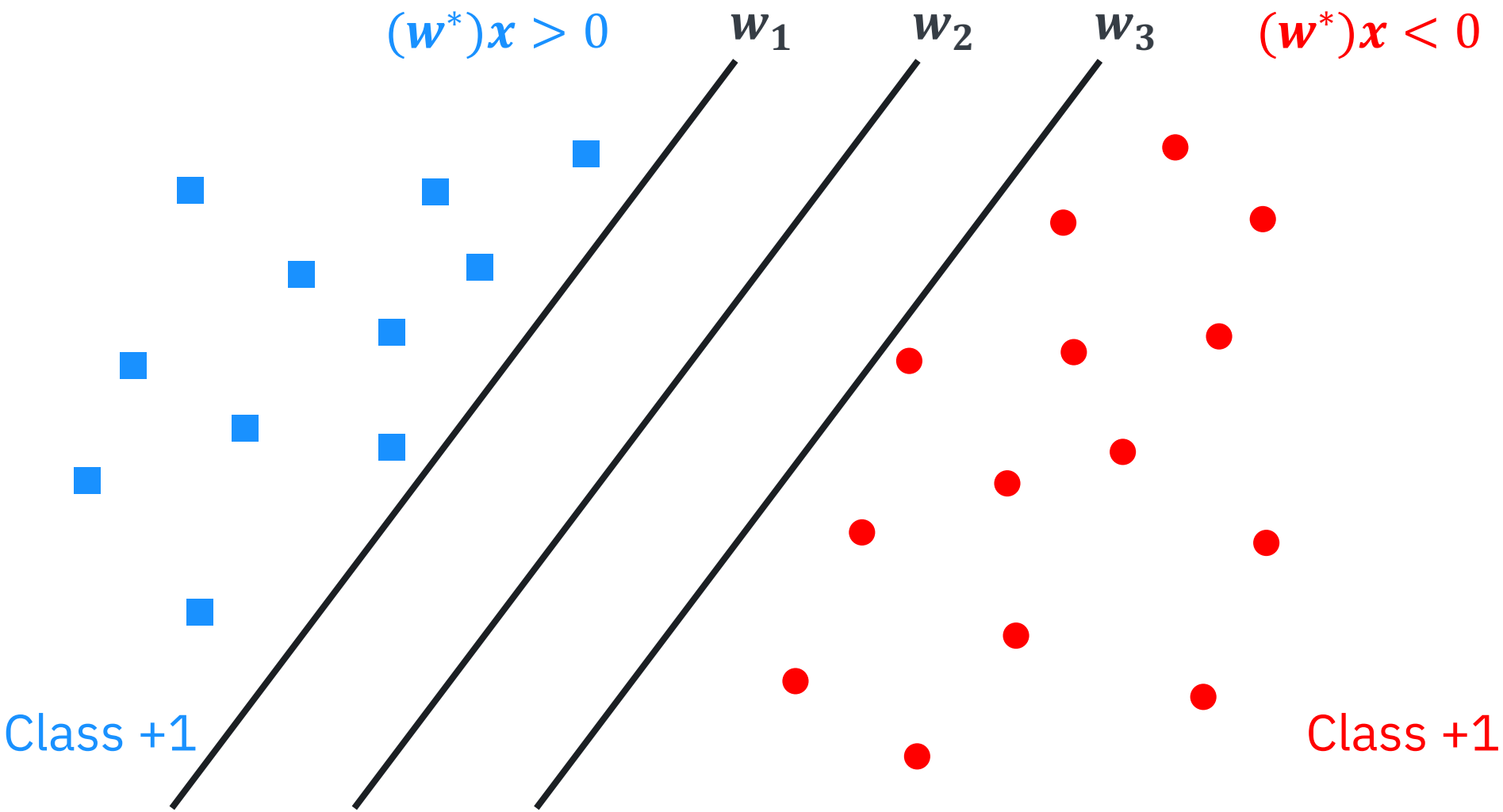


Optimize to find w!

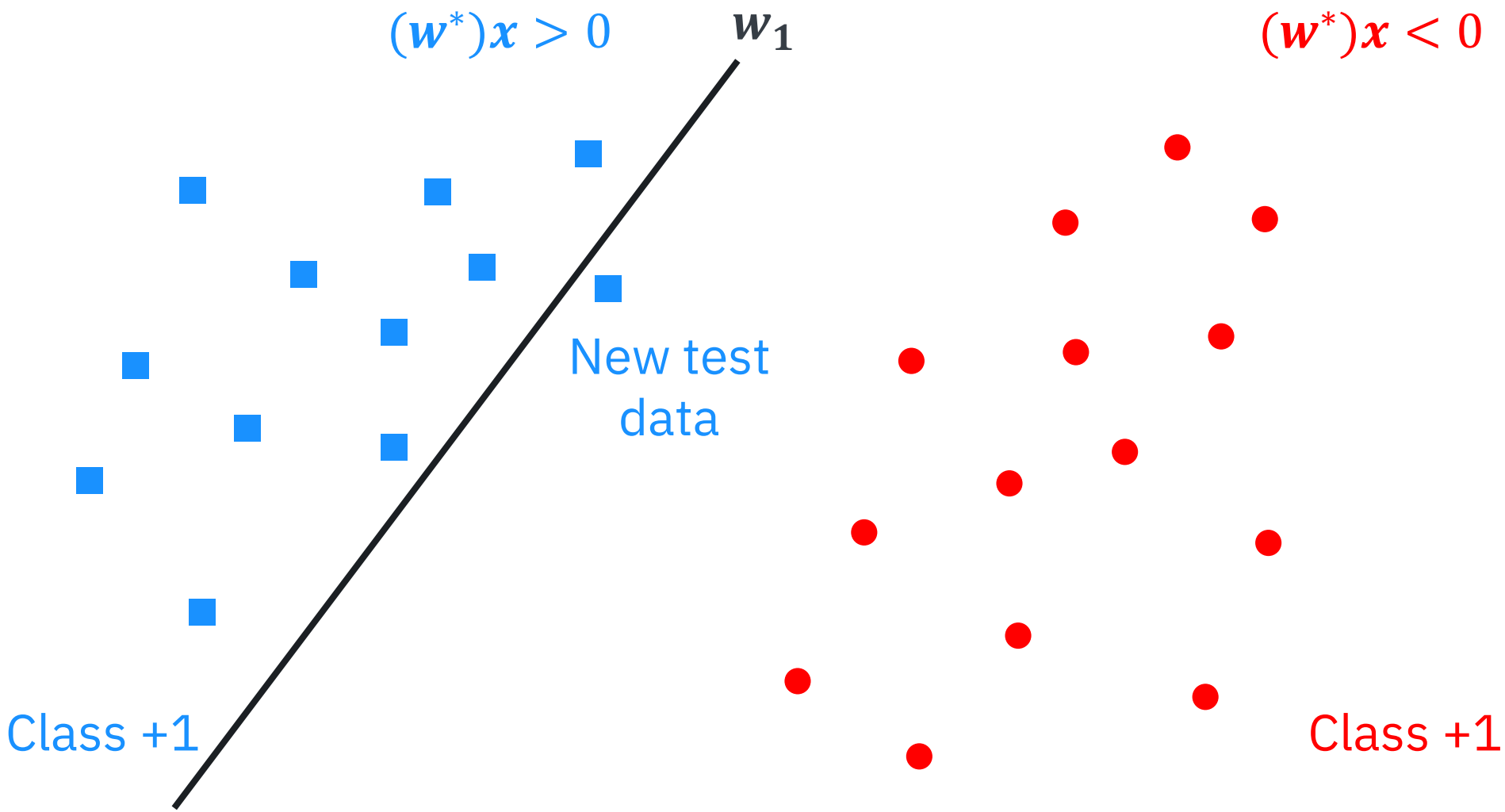
Linear Threshold Classifiers



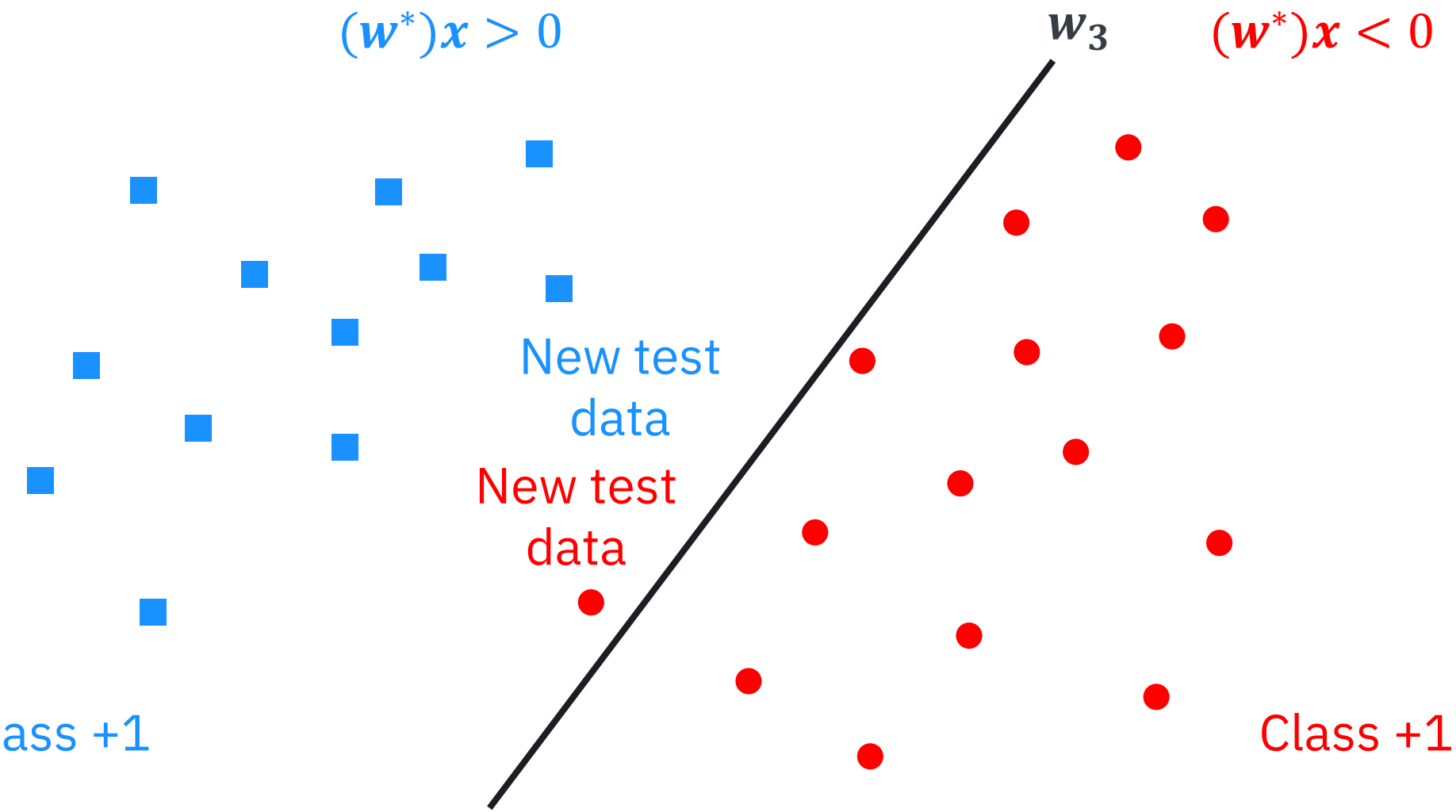
More Optimal Solutions?



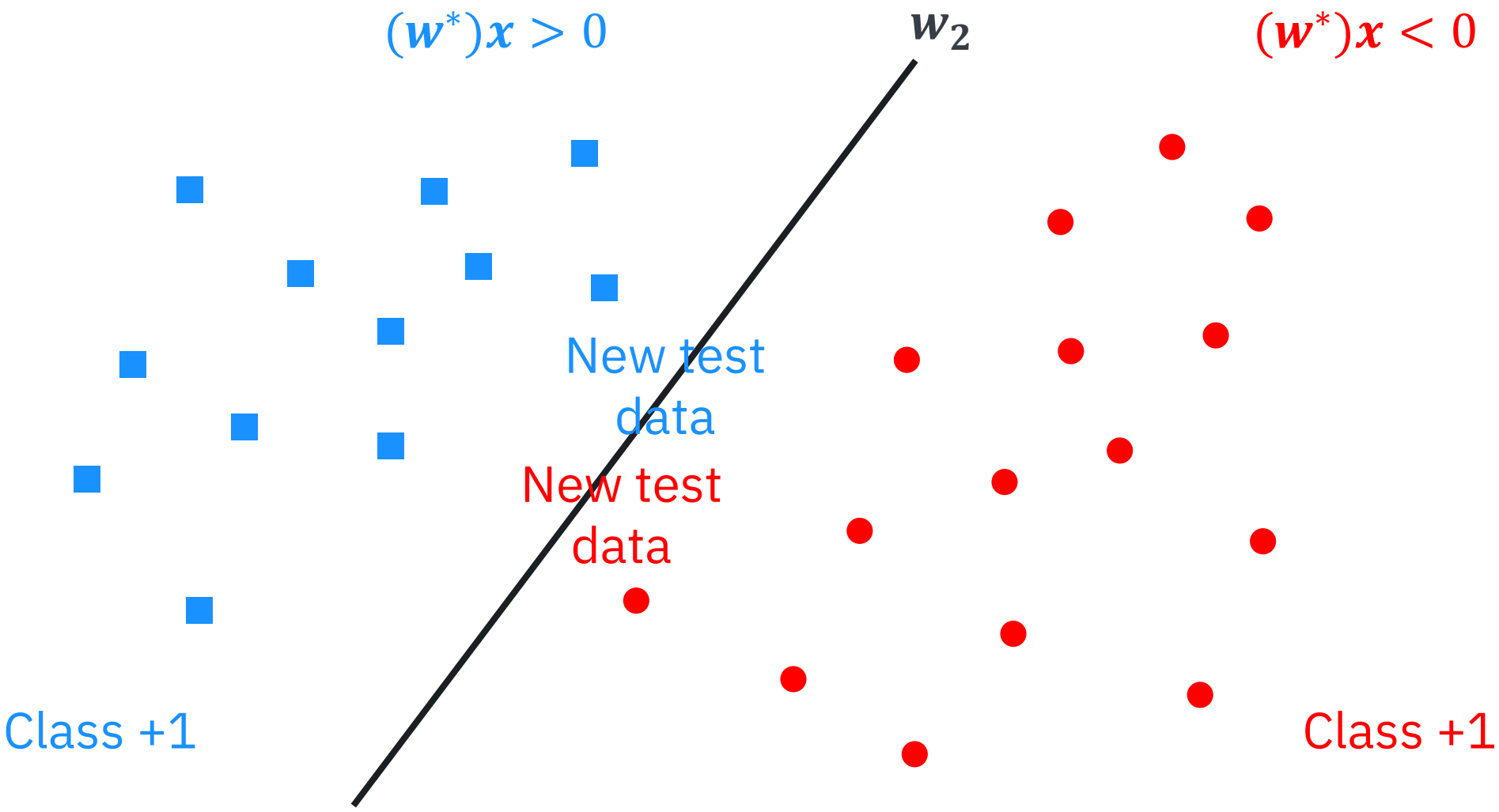
What About w_1?



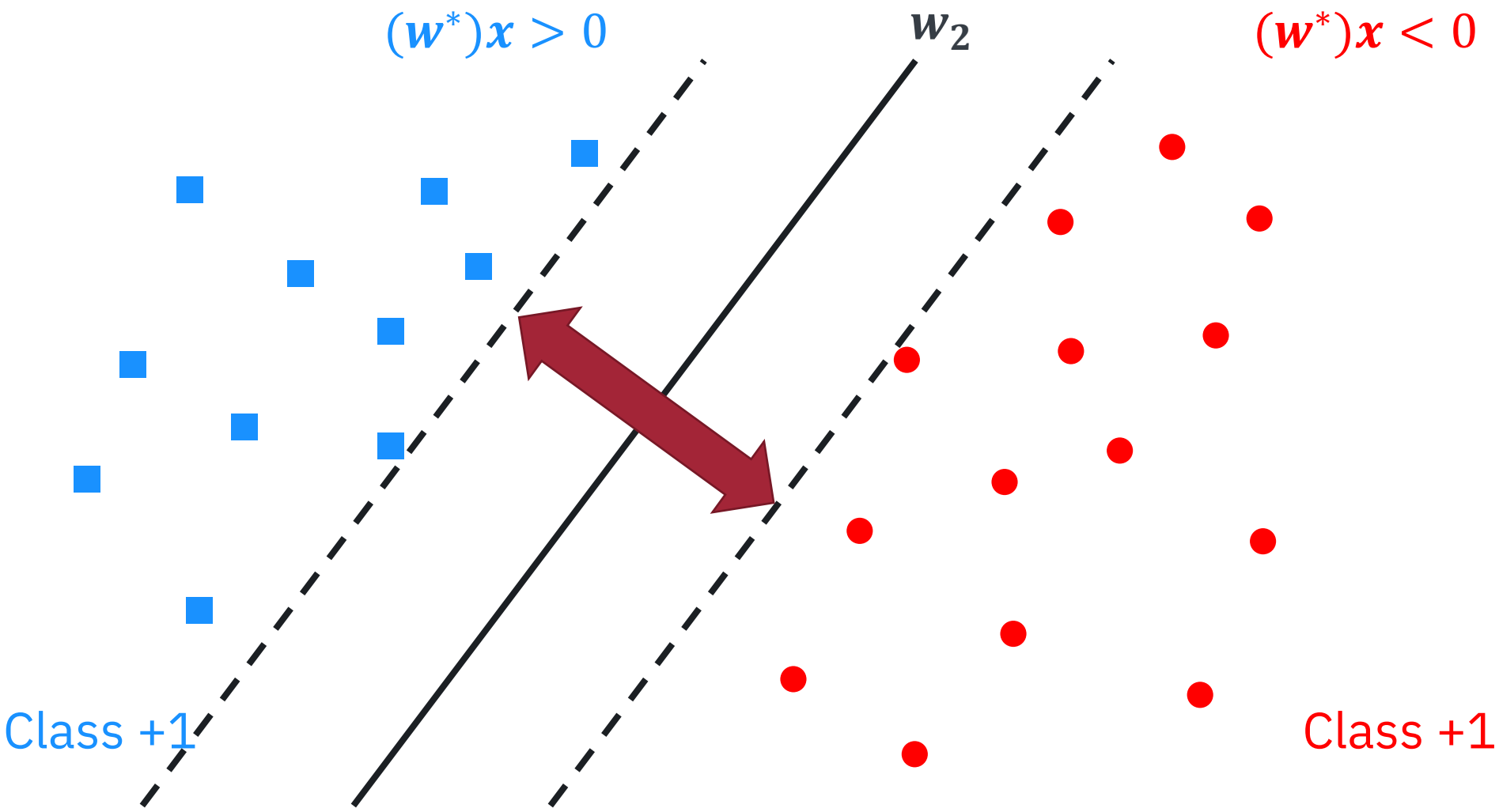
What About w_3?



Most Confident: w_2



Intuition: Margin



SVM: Simplified Objective

- Let's consider a fixed scale such that

$$y_{i^*}(\boldsymbol{w}^T \boldsymbol{x}_{i^*}) = 1$$

  where $\boldsymbol{x}_{i^*}$ is the point closest to the hyperplane.

- Now we have for all data

$$y_{i^*}(\boldsymbol{w}^T \boldsymbol{x}_{i^*}) \geq 1$$

  and at least for one i the equality holds.

- The margin is $\frac{1}{\|\boldsymbol{w}\|}$.

SVM: Simplified Objective

- Optimization simplified to

$$\min_{\boldsymbol{w}} \frac{1}{2} \|\boldsymbol{w}\|^2$$

(Using L2 norm instead of L1 norm because L2 norm is differentiable.)

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i) \geq 1, \forall i$$

- How to find the optimum $\hat{\boldsymbol{w}}^*$?

SVM: Optimization

- Conditions of optimal solution (Karush–Kuhn–Tucker conditions)

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0$$

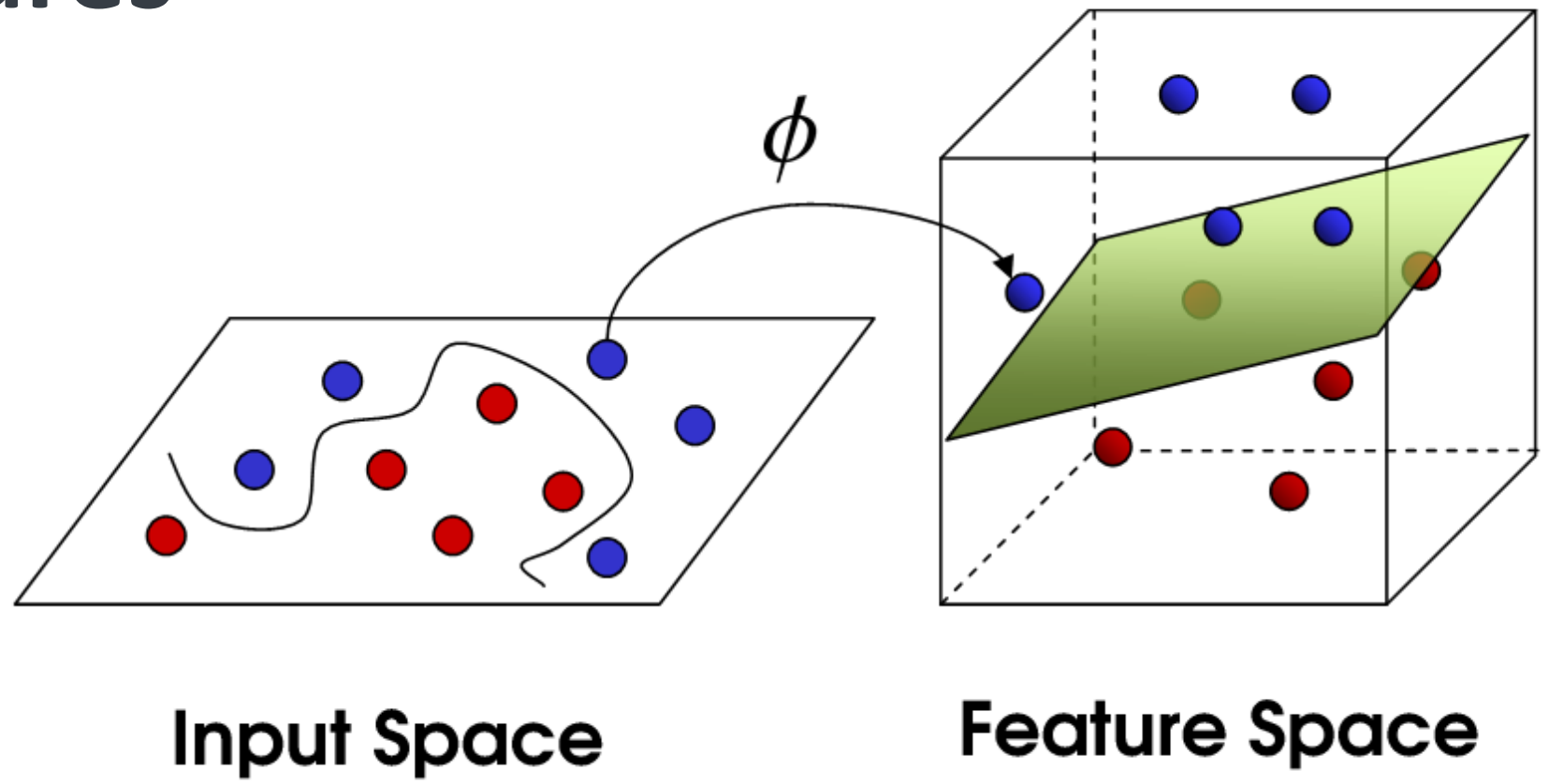
$$\boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i$$

With $\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha})$ and $f_{\boldsymbol{w}}(\boldsymbol{x})$, we can find the solution of the optimization problem.

(The mathematical derivation of the solution is beyond the scope of this class.)

Kernel Methods

Features



$$\phi: (x_1, x_2) \rightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

Features

- Kernel Method
  - Using SVM on the feature space $\{\phi(x_i)\}$: only need $\phi(x_i)^T\phi(x_j)$.
  - No need to design $\phi(x_i)$, only need to design

$$K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$$

Learning Using Kernels

- Conditions of optimal solution (Karush–Kuhn–Tucker conditions)

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0$$

$$\boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i$$

Learning Using Kernels

- Plug into $\mathcal{L}$:

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

- Plug into f:

$$f_{\boldsymbol{w}}(\boldsymbol{x}_i) = \boldsymbol{w}^T \boldsymbol{x}_i = \sum_j \alpha_j y_j \boldsymbol{x}_j^T \boldsymbol{x}_i$$

Only depend on inner product

Learning Using Kernels

- Plug into $\mathcal{L}$:

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\phi}(\boldsymbol{x}_j)$$

- Plug into f:

$$f_{\boldsymbol{w}}(\boldsymbol{\phi}(\boldsymbol{x}_i)) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i) = \sum_j \alpha_j y_j \boldsymbol{\phi}\left(\boldsymbol{x}_j\right)^T \boldsymbol{\phi}(\boldsymbol{x}_i)$$

Learning Using Kernels

- Plug into $\mathcal{L}$:

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

- Plug into f:

$$f_w(\phi(\boldsymbol{x}_i)) = \boldsymbol{w}^T \phi(\boldsymbol{x}_i) = \sum_j \alpha_j y_j K(x_j, x_i)$$

Example

General inner product

$$\boldsymbol{x} = (x_1, x_2)$$

$$\phi(\boldsymbol{x}) = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)$$

$$K(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}') = 1 + x_1 x'_1 + x_2 x'_2 + x_1^2 x_1'^2 + x_2^2 x_2'^2 + x_1 x'_1 x_2 x'_2$$

Kernel trick

$$K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \boldsymbol{x}^T \boldsymbol{x})^2 = (1 + x_1 x'_1 + x_2 x'_2)^2$$

$$= 1 + x_1 x'_1 + x_2 x'_2 + 2x_1^2 x_1'^2 + 2x_2^2 x_2'^2 + 2x_1 x'_1 x_2 x'_2$$

This is an inner product!

$$\phi(\boldsymbol{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2) \qquad \phi(\boldsymbol{x}') = (1, \sqrt{2}x'_1, \sqrt{2}x'_2, x'^2_1, x'^2_2, \sqrt{2}x'_1 x'_2)$$

Learning Using Kernels

- Without a kernel function
  - Generate feature vectors $\phi(\boldsymbol{x})$ from samples $\boldsymbol{x}$.
  - The dimension of the feature vectors is usually **much higher** than the dimension of the original samples.
  - The computing load to find the solution of the optimization problem becomes higher.

- With a kernel function
  - If we can calculate the value of the kernel function, **we do not need to generate the feature vectors**.
  - It is much more efficient to find the solution of optimization problem.

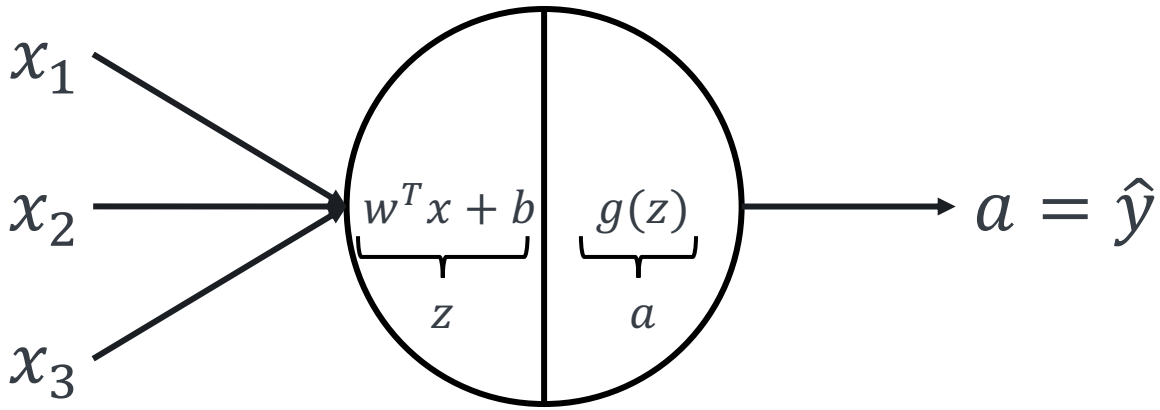
Artificial Neural Network

Artificial Neural Network

- Machine learning algorithms can be viewed as approximations of functions that describe the data.
- In practice, the relationships between input and output can be extremely complex.

Artificial Neural Network

- Collection of simple artificial neurons
- Weights w denote strength of connection from a neuron i to j.
- Input function
  - $z = w^T x + b$
- Activation function
  - $a = g(z)$



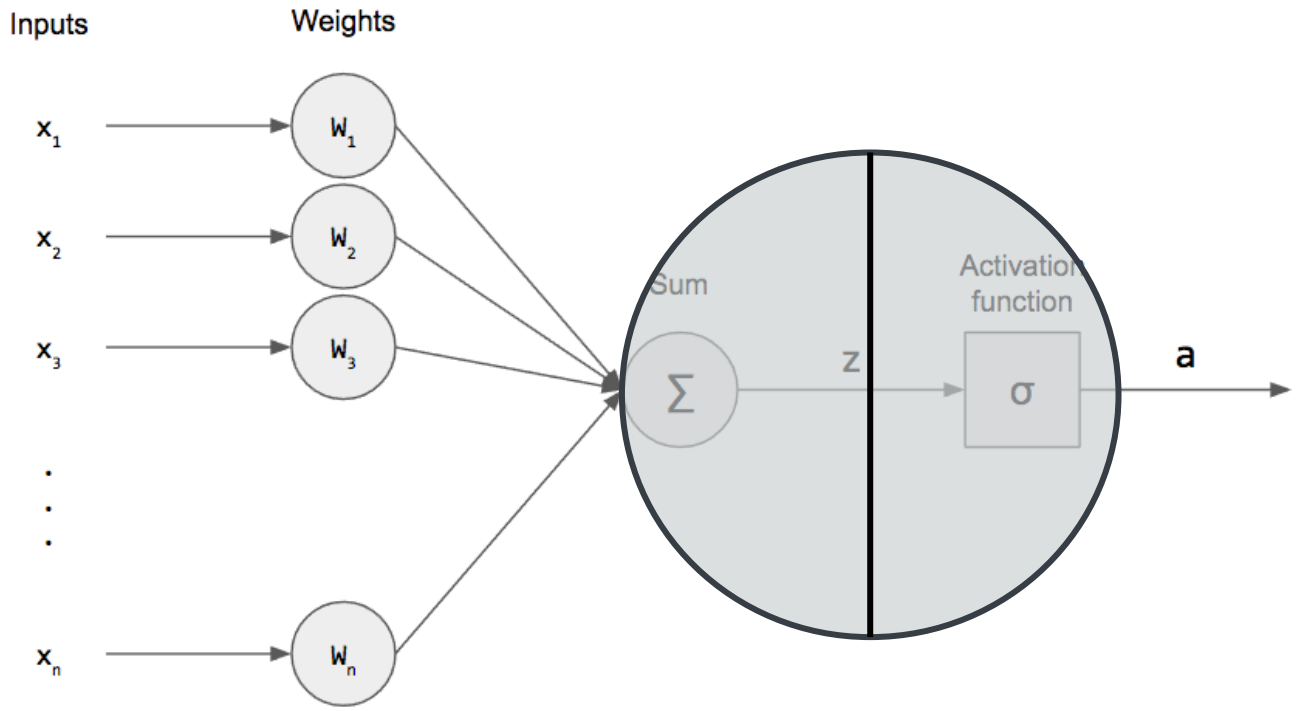
Network Structure

- Feed-forward ANN
  - Direct acyclic graph
  - **No internal state**: maps inputs to outputs
  - Commonly used when input and output are independent (*e.g.*, image recognition)

- Recurrent ANN
  - Directed cyclic graph
  - Dynamical system with an internal state
  - **Can remember information for future use**
  - Commonly used when input and output are dependent (*e.g.*, speech recognition)

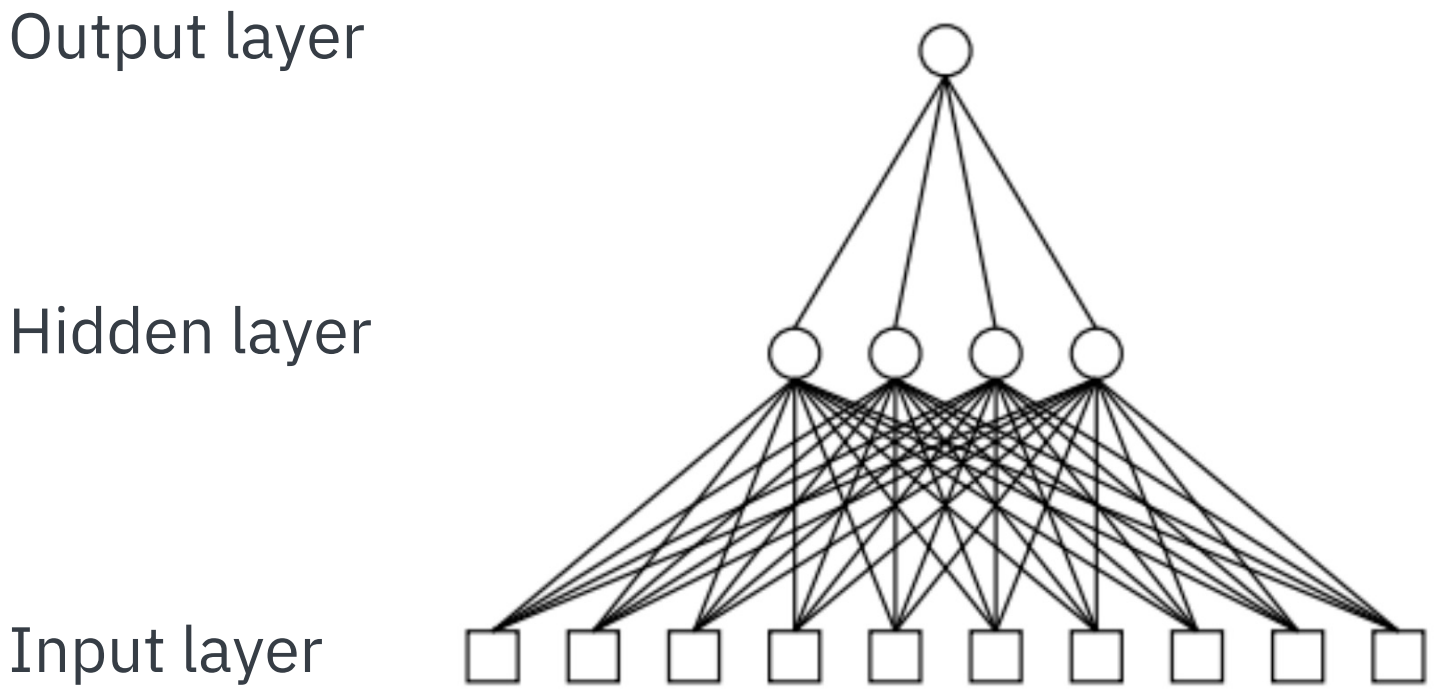
Perceptron

- Single layer feed-forward network

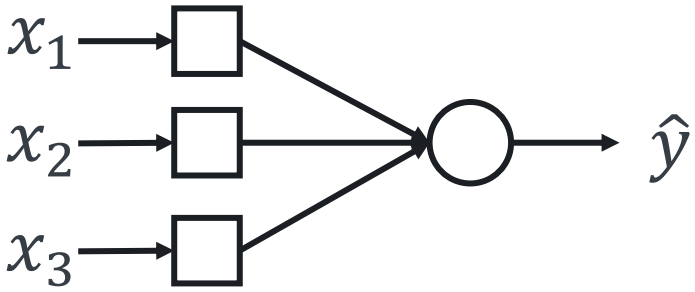


Multilayer Networks

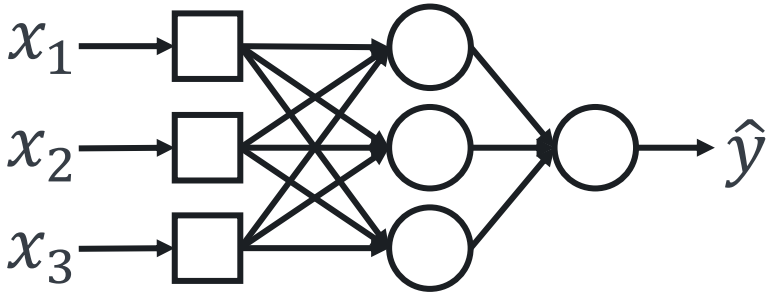
- Any continuous function can be learned by an ANN with just one hidden layer if the layer is large enough.



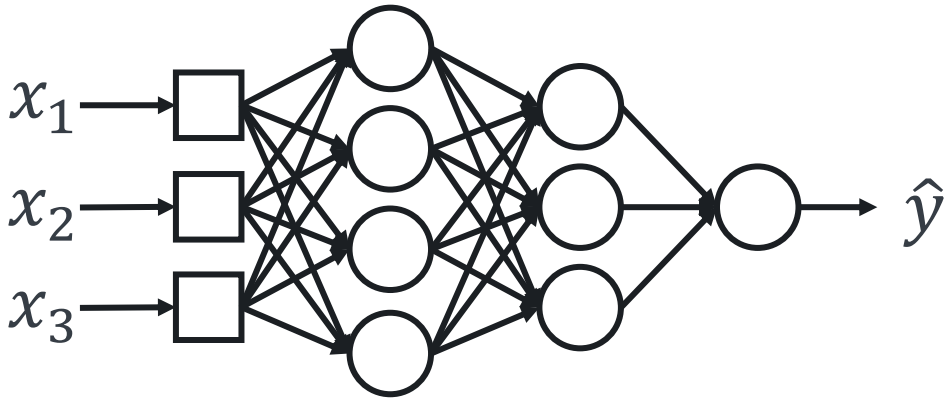
Multilayer Networks



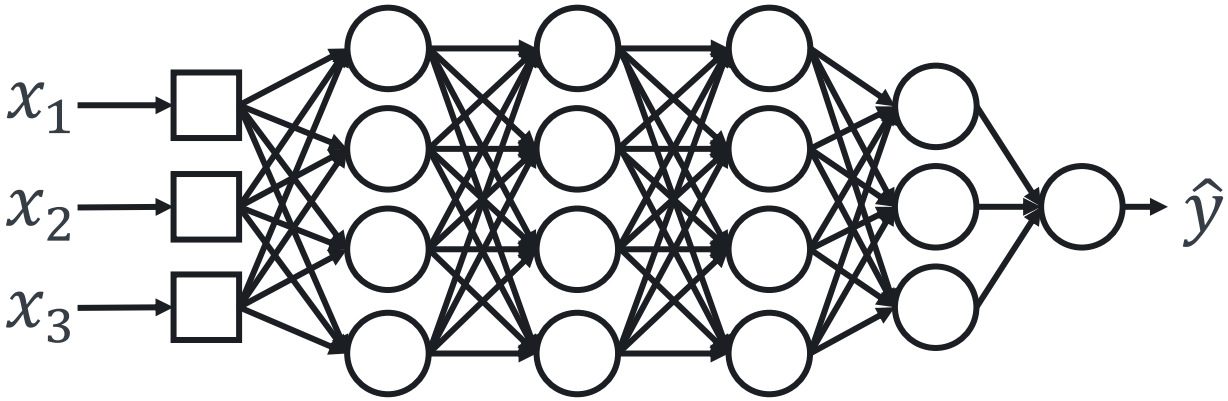
Perceptron



1 hidden layer

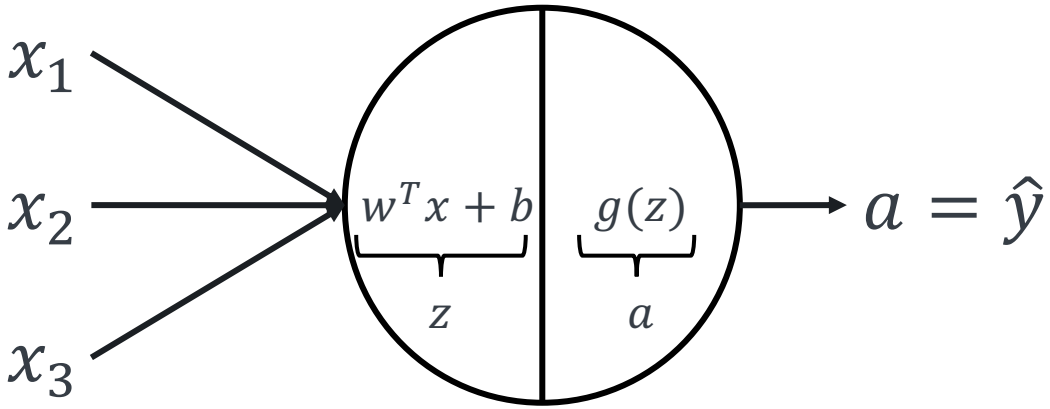


2 hidden layers

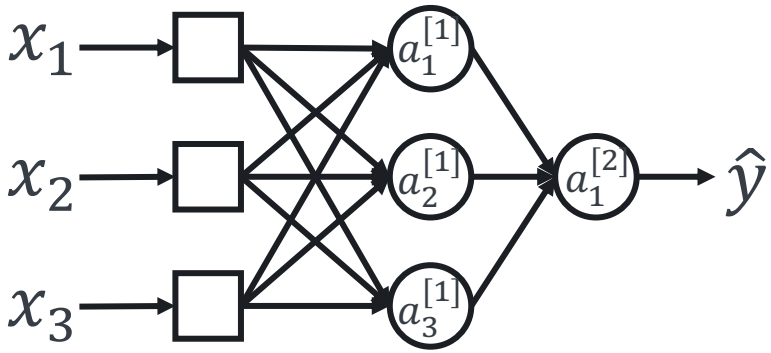


4 hidden layers

Forward Propagation



$$z = w^T x + b$$
$$a = g(z)$$



$$z^{[1]} = w^{[1]T} a^{[0]} + b^{[1]} \qquad a^{[1]} = g(z^{[1]})$$
$$z^{[2]} = w^{[2]T} a^{[1]} + b^{[2]} \qquad a^{[2]} = g(z^{[2]})$$

$$z^{[l]} = w^{[l]T} a^{[l-1]} + b^{[l]} \qquad a^{[l]} = g(z^{[l]})$$

Training Multilayer Nets

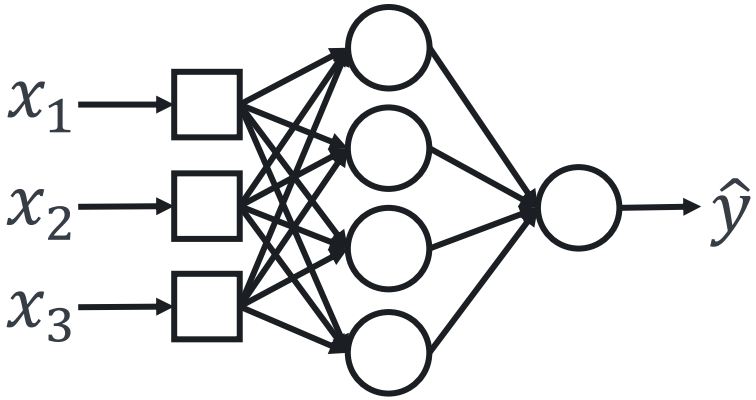
- Back propagation
- Gradient descent algorithm

Gradient Descent for Neural Networks

Parameters: $w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}$

Cost function:

$$J\left(w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}\right) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}, y)$$



Gradient descent

Repeat:

Compute prediction $\hat{y}_i \;\; (i = 1 \ldots m)$

$$dw^{[l]} = \frac{dJ}{dw^{[l]}} \quad , \quad db^{[l]} = \frac{dJ}{db^{[l]}}$$

How do we get the derivatives?

$$w^{[2]} \leftarrow w^{[2]} - \alpha \cdot dw^{[2]} \qquad b^{[2]} \leftarrow b^{[2]} - \alpha \cdot db^{[2]}$$

$$w^{[1]} \leftarrow w^{[1]} - \alpha \cdot dw^{[1]} \qquad b^{[1]} \leftarrow b^{[1]} - \alpha \cdot db^{[1]}$$

Forward and Backward Propagation

Forward Propagation

Input: $a^{[l-1]}$

Output: $a^{[l]}$, cache $(z^{[l]})$

$$z^{[l]} = w^{[l]T} a^{[l-1]} + b^{[l]}$$

$$a^{[l]} = g(z^{[l]})$$

Backward Propagation

Input: $da^{[l]}$

Output: $da^{[l-1]}, dw^{[l]}, db^{[l]}$

$$dz^{[l]} = da^{[l]} * g^{[l]\prime}(z^{[l]})$$

$$dw^{[l]} = dz^{[l]} \cdot a^{[l-1]}$$

$$db^{[l]} = dz^{[l]}$$

$$da^{[l-1]} = w^{[l]T} \cdot dz^{[l]}$$

STEVENS
INSTITUTE OF TECHNOLOGY
1870

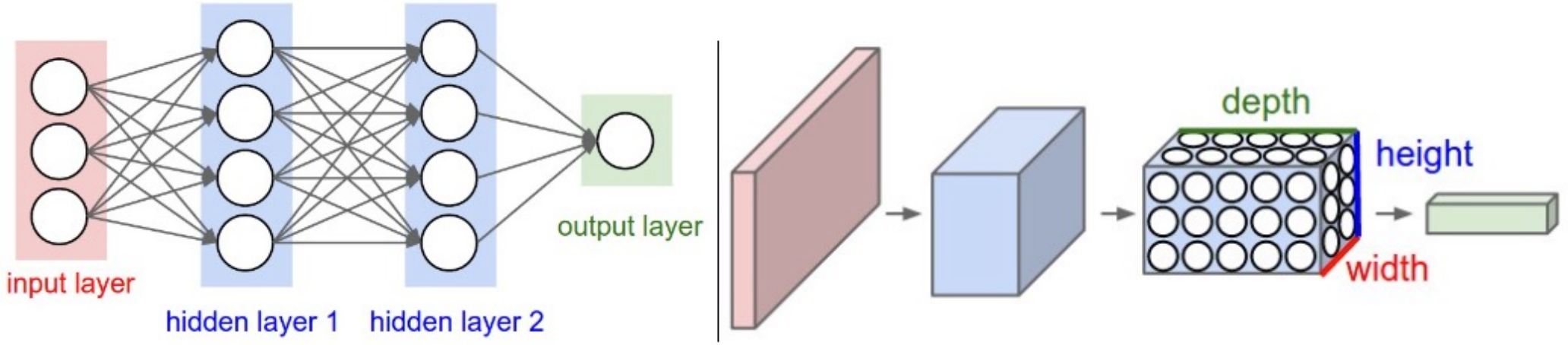
ANN Examples

Convolutional Neural Network

Convolutional Neural Network

- Properties of regular neural network
  - Neurons in a single layer function completely independently and do not share any connections.
  - The neurons between layers are fully connected.
- The fully-connected structure does not scale to larger images.
- The full connectivity is wasteful and the huge number of parameters would quickly lead to overfitting.

Convolutional Neural Network

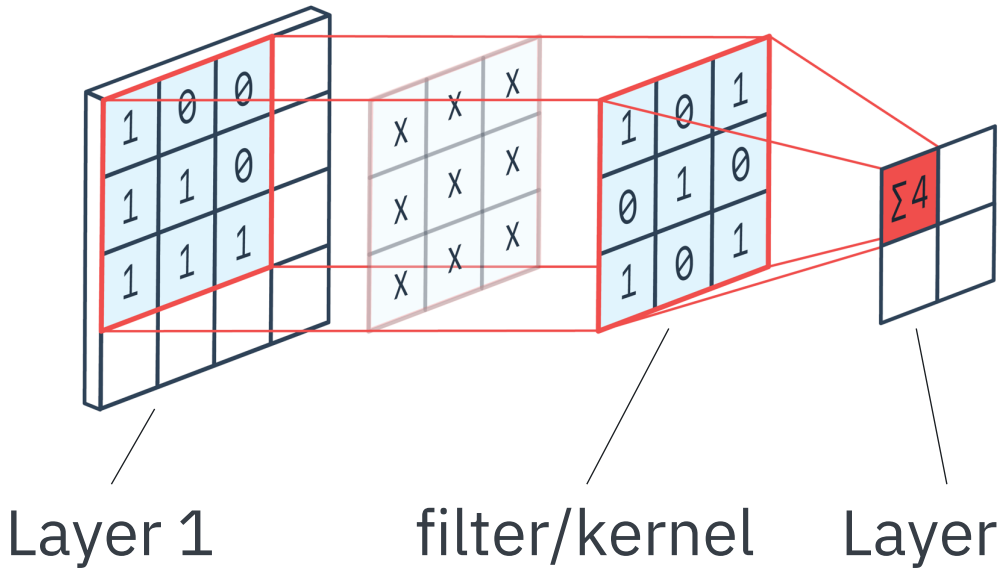


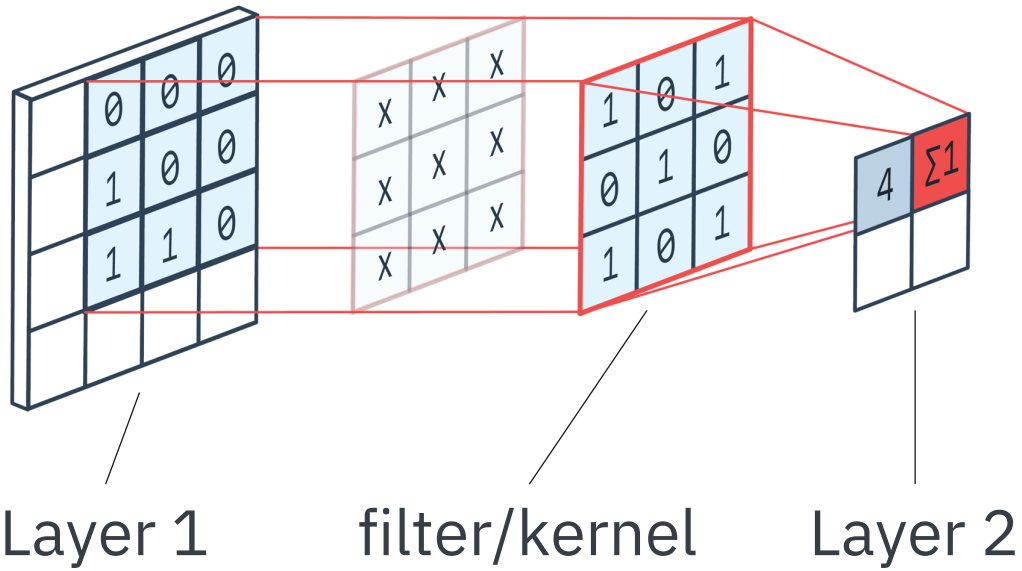
- The layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth.
- The neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner.

Convolutional Neural Network

- Three typical types of layers in CNN
  - Convolutional layer
  - Pooling layer
  - Fully-connected layer
    - Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular neural networks.

Convolutional Layer



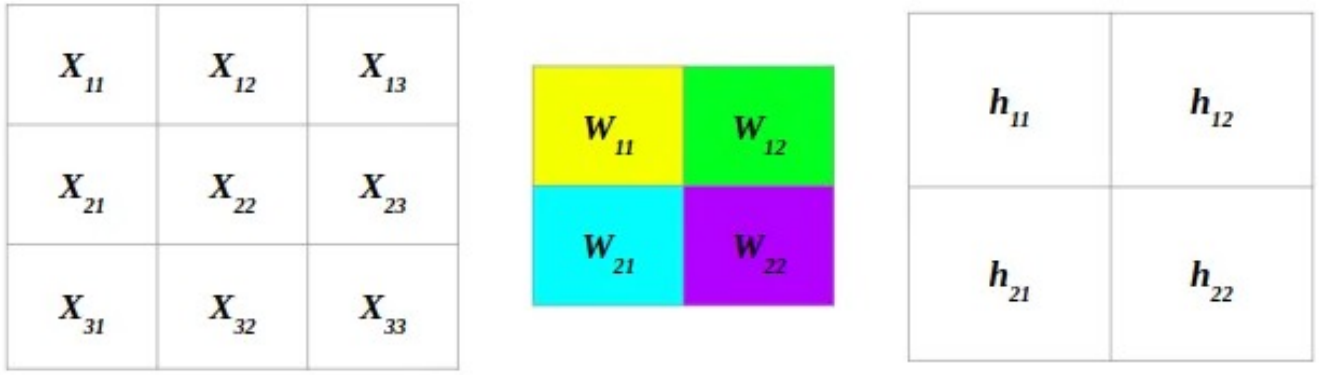


Convolution Operation (Forward Propagation)

X_{11}	X_{12}	X_{13}
X_{21}	X_{22}	X_{23}
X_{31}	X_{32}	X_{33}

W_{11}	W_{12}
W_{21}	W_{22}

Convolution Operation (Forward Propagation)



Input Size : 3×3, **Filter Size** : 2×2, **Output Size** : 2×2

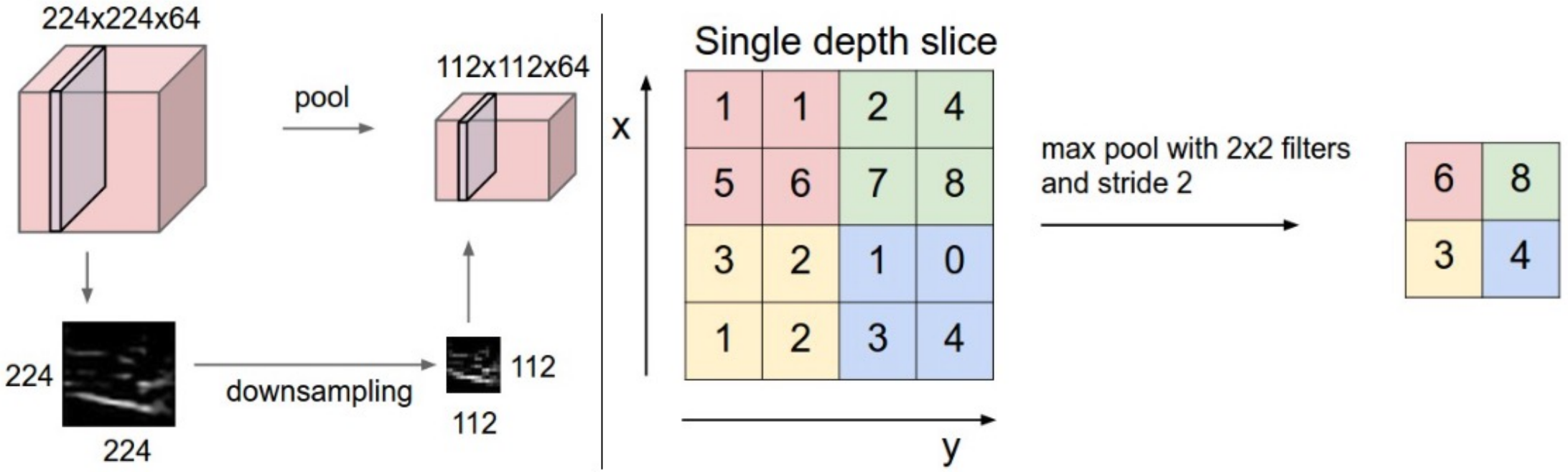
$$h_{11} = W_{11}X_{11} + W_{12}X_{12} + W_{21}X_{21} + W_{22}X_{22}$$

$$h_{12} = W_{11}X_{12} + W_{12}X_{13} + W_{21}X_{22} + W_{22}X_{23}$$

$$h_{21} = W_{11}X_{21} + W_{12}X_{22} + W_{21}X_{31} + W_{22}X_{32}$$

$$h_{22} = W_{11}X_{22} + W_{12}X_{23} + W_{21}X_{32} + W_{22}X_{33}$$

Pooling Layer



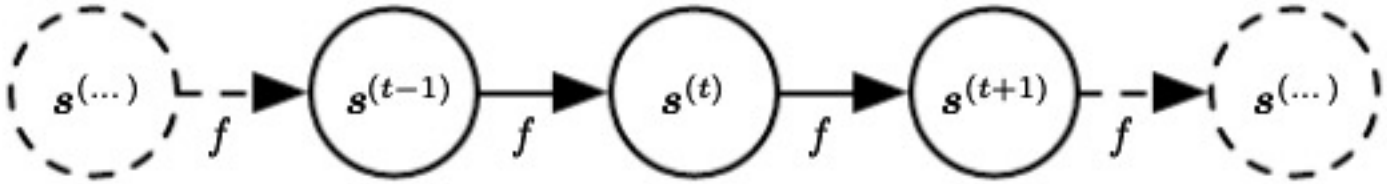
Recurrent Neural Network

Recurrent Neural Network

- Recurrent neural networks are a family of neural networks for processing **sequential data**.
- Specialized for sequential data
  - Convolutional networks: images with large width and height
  - Recurrent networks: long sequences

Computational Graph

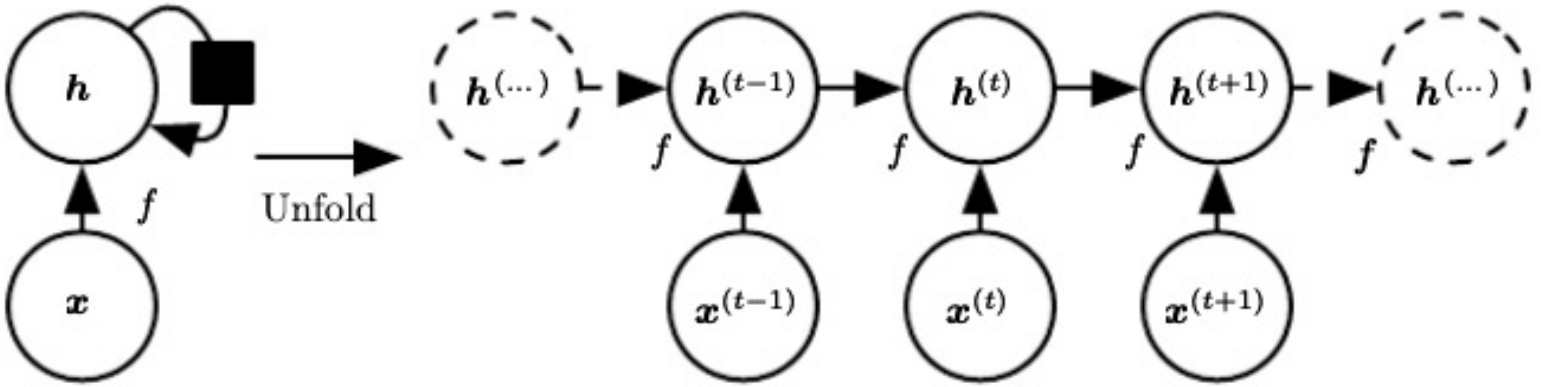
- A computational graph is a way to formalize the structure of a set of computations.



$$s^{(t)} = f(s^{(t-1)}; \theta)$$

Computational Graph

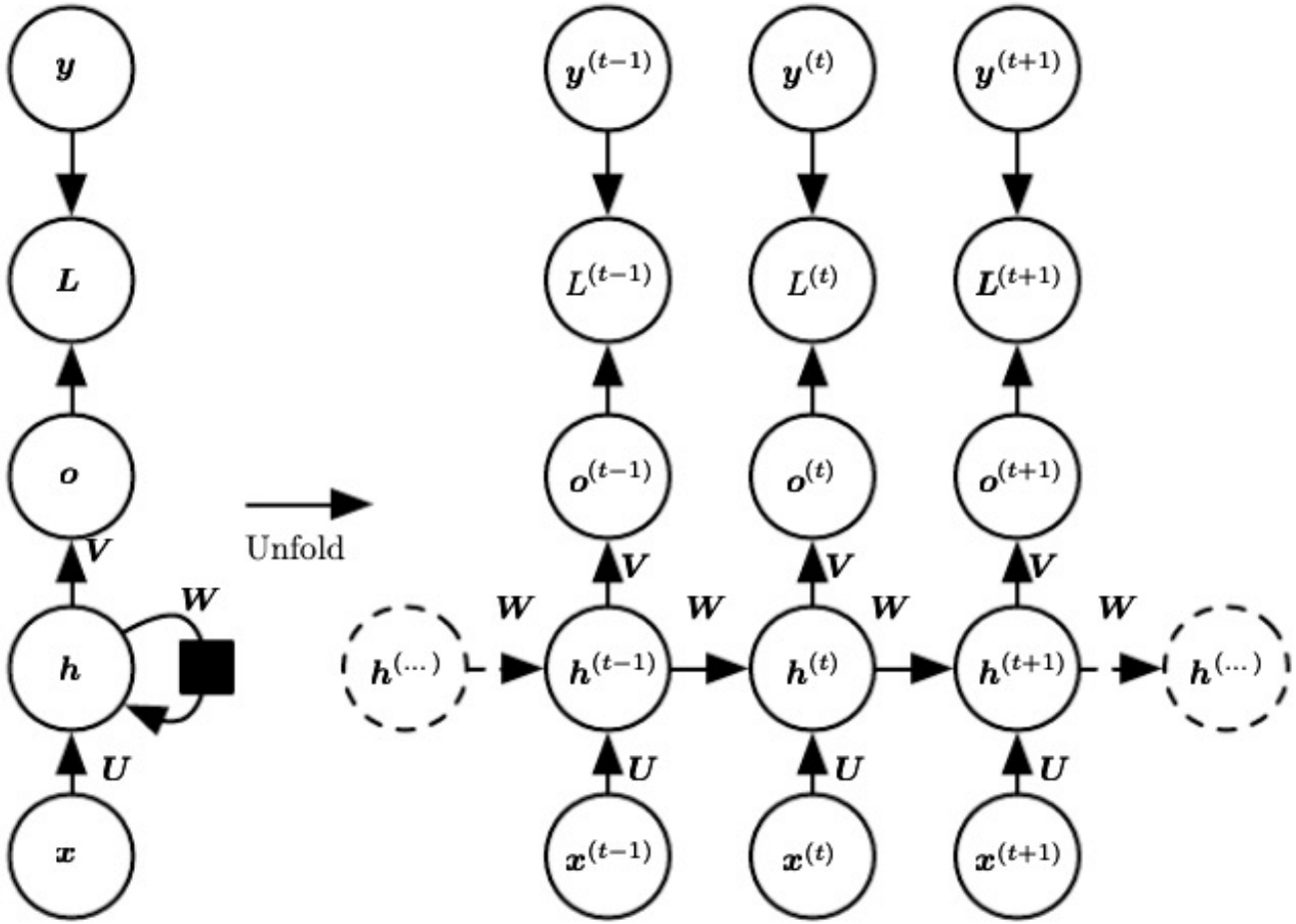
- A computational graph is a way to formalize the structure of a set of computations.



$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

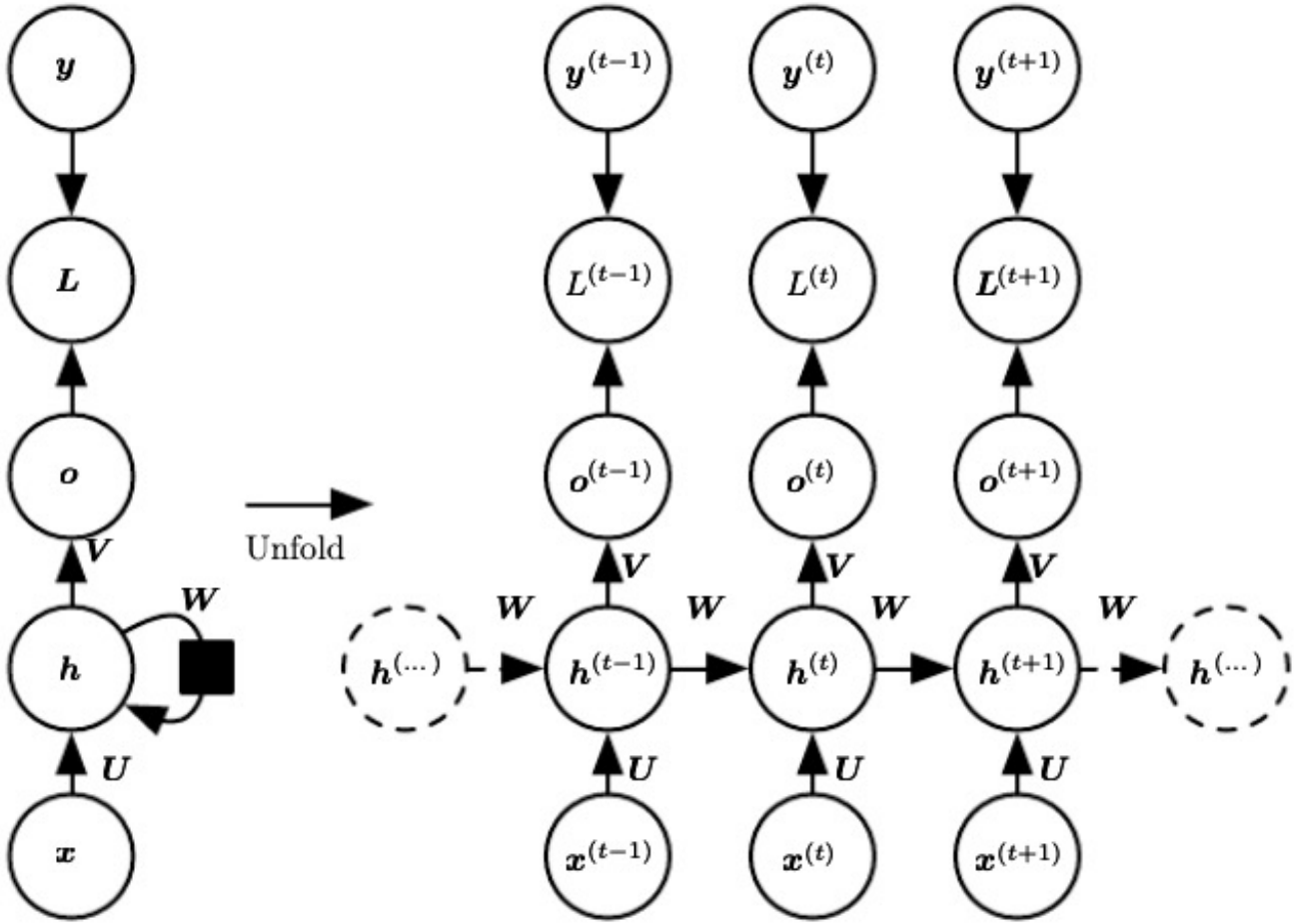
Variations of RNN

- Graph
  - y: training label
  - L: loss function
  - o: output value
  - h: the state of the system
  - x: input



Variations of RNN

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}$$
$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)})$$
$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}$$
$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}(o^{(t)})$$



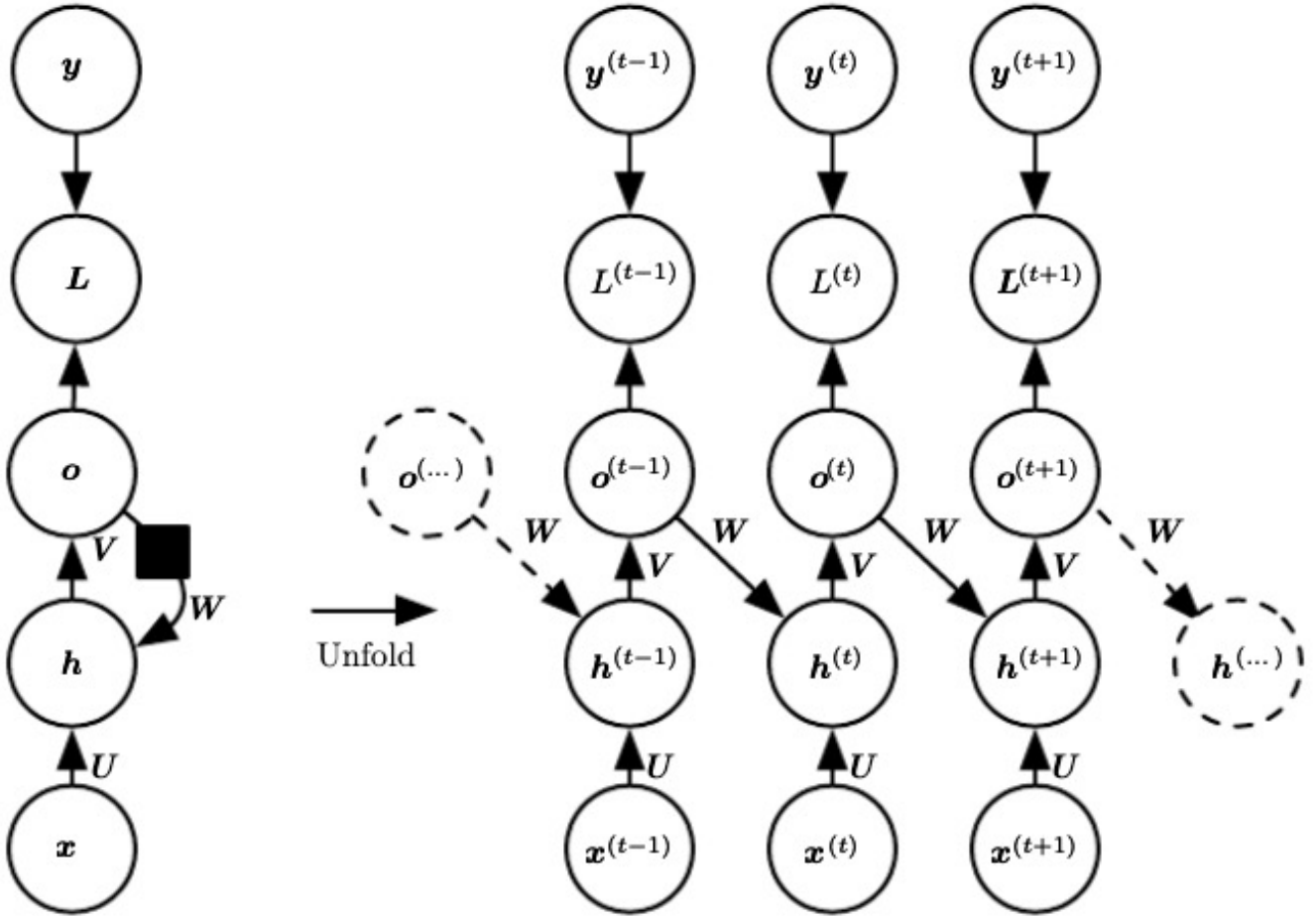
Variations of RNN

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{o}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}$$

$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)})$$

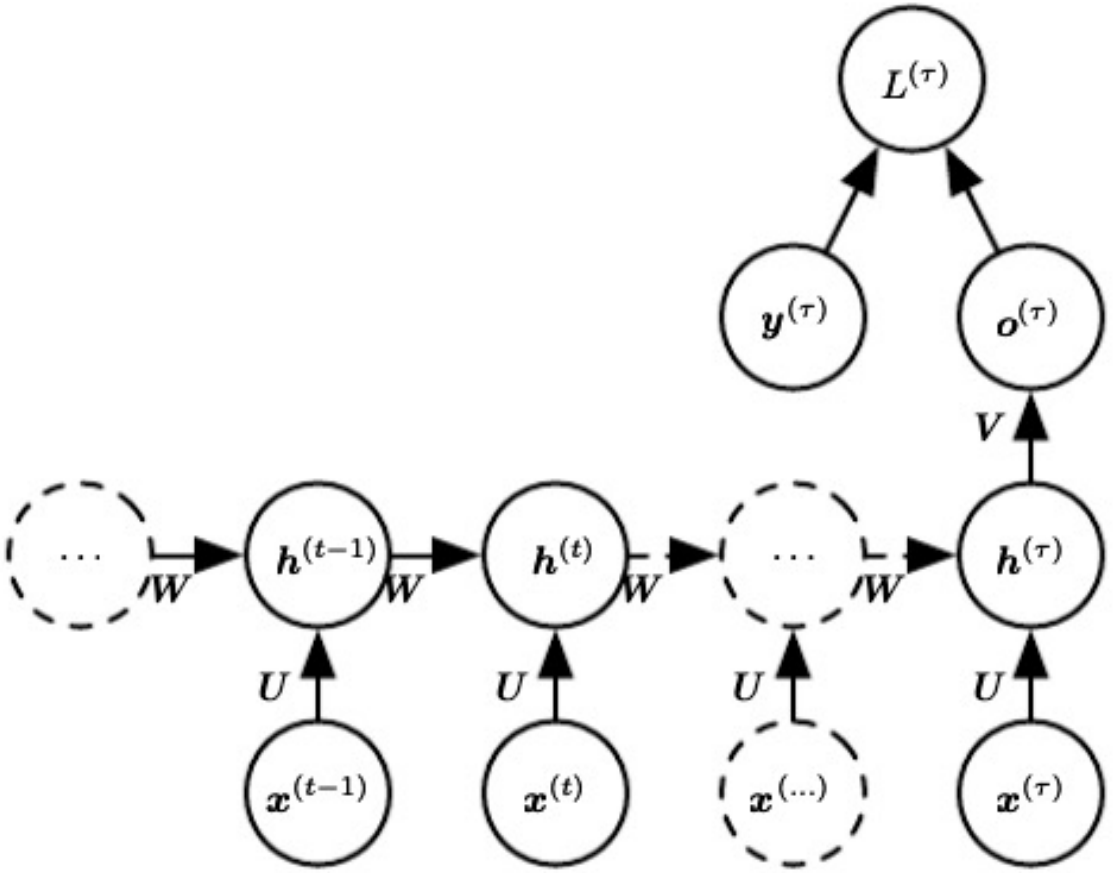
$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}$$

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}(o^{(t)})$$



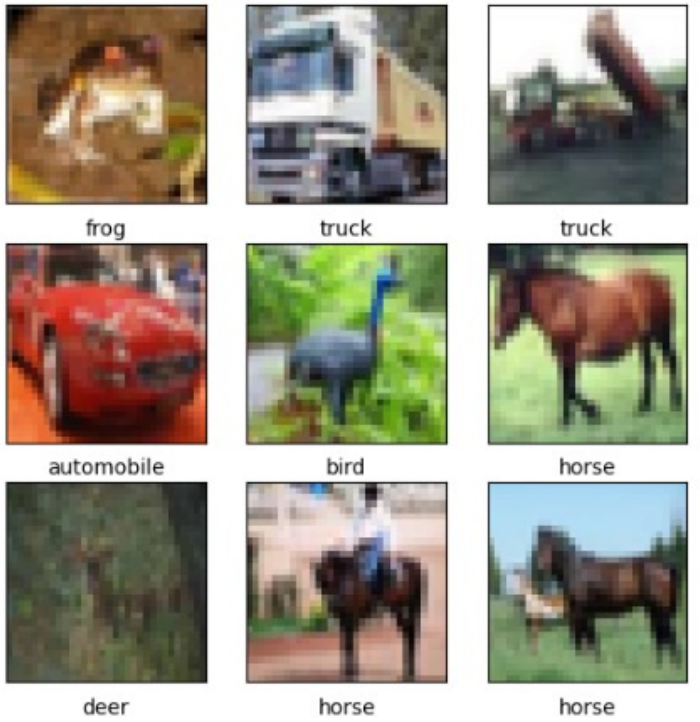
Variations of RNN

- RNN with single output at the end of the sequence
  - Summarize a sequence
  - Produce a fixed-size representation



Sample Codes

- Convolutional neural network
  - Image classification
  - https://www.tensorflow.org/tutorials/images/cnn



- Recurrent neural network
  - Generate music with an RNN
  - https://www.tensorflow.org/tutorials/audio/music_generation

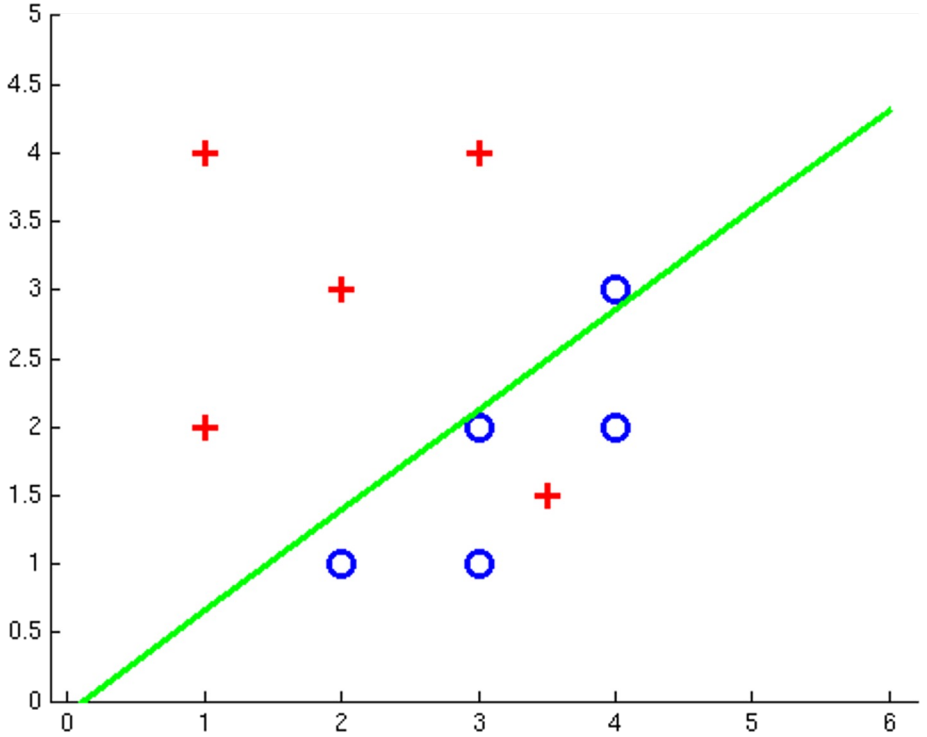
STEVENS
INSTITUTE OF TECHNOLOGY
1870

Nonparametric Models

Chapter 18

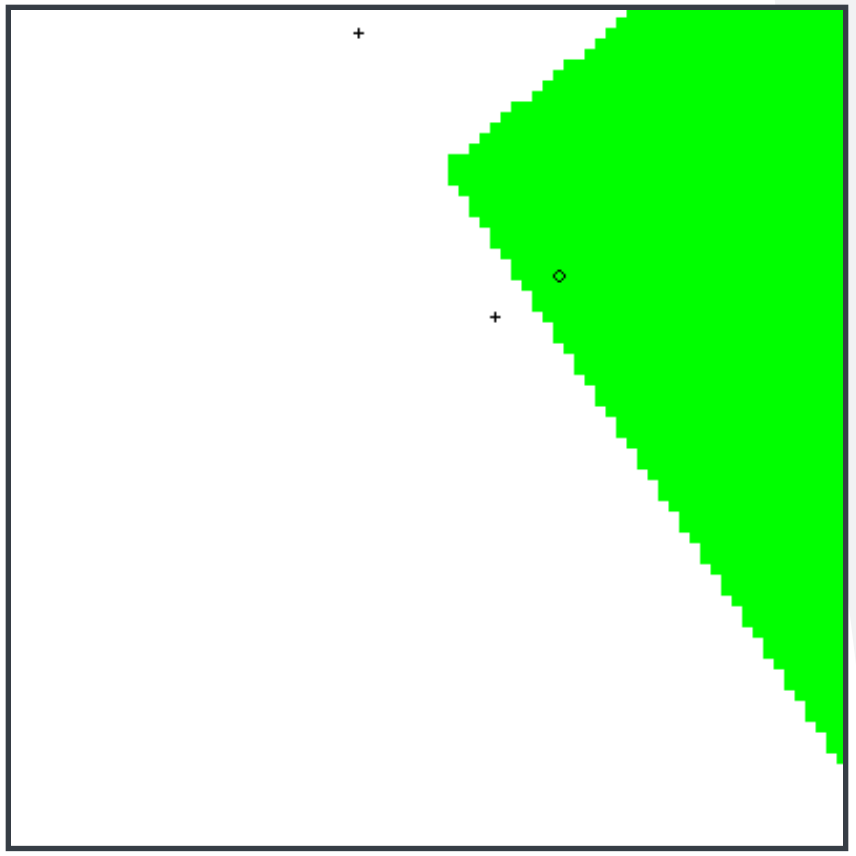
Case-Based Learning

Non-Separable Data



Case-Based Reasoning

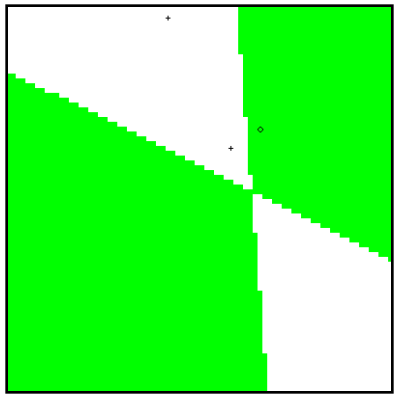
- Classification from similarity
  - Case-based reasoning
  - Predict an instance's label using similar instances
- Nearest-neighbor classification
  - 1-NN: copy the label of the most similar data point
  - K-NN: vote the k nearest neighbors (need a weighting scheme)
  - Key issue: how to define similarity
  - Trade-offs: Small k gives relevant neighbors, Large k gives smoother functions



http://www.cs.cmu.edu/~zhuxj/courseproject/knndemo/KNN.html

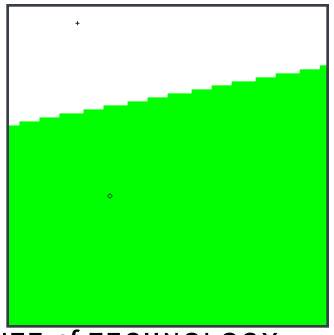
Parametric / Non-Parametric

- Parametric models:
  - Fixed set of parameters
  - More data means better settings
- Non-parametric models:
  - Complexity of the classifier increases with data
  - Better in the limit, often worse in the non-limit
- (K)NN is **non-parametric**

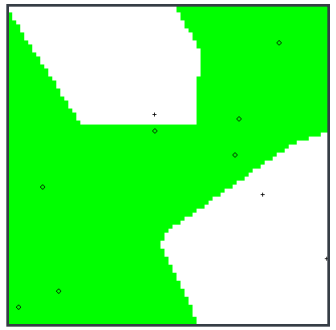


Truth

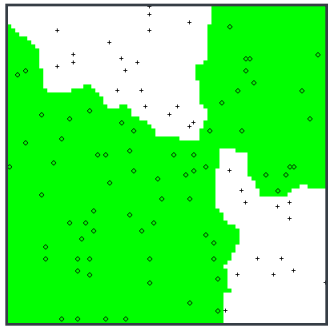
2 Examples



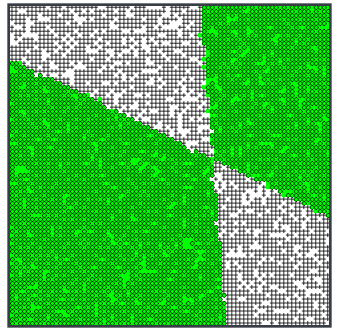
10 Examples



100 Examples



10000 Examples



Nearest-Neighbor Classification

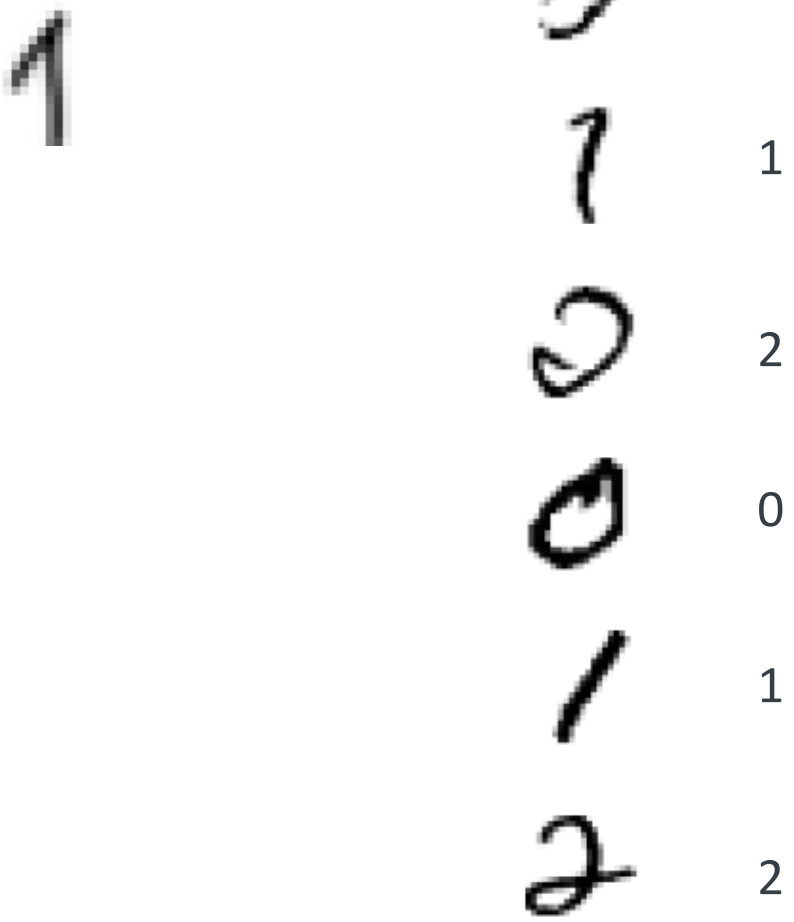
- Nearest neighbor for digits:
  - Take new image
  - Compare to all training images
  - Assign based on closest example
- Encoding: image is vector of intensities:

 $= \langle 0.0 \;\; 0.0 \;\; 0.3 \;\; 0.8 \;\; 0.7 \;\; 0.1 \ldots 0.0 \rangle$

- What's the similarity function?
  - Dot product of two images vectors?

$$\text{sim}(x, x') = x \cdot x' = \sum_i x_i x'_i$$

  - Usually normalize vectors so ||x|| = 1
  - min = 0 (when?), max = 1 (when?)



Similarity Functions

Basic Similarity

- Many similarities based on **feature dot products**:

$$\text{sim}(x, x') = f(x) \cdot f(x') = \sum_i f_i(x) f_i(x')$$

- If features are just the pixels:

$$\text{sim}(x, x') = x \cdot x' = \sum_i x_i x'_i$$

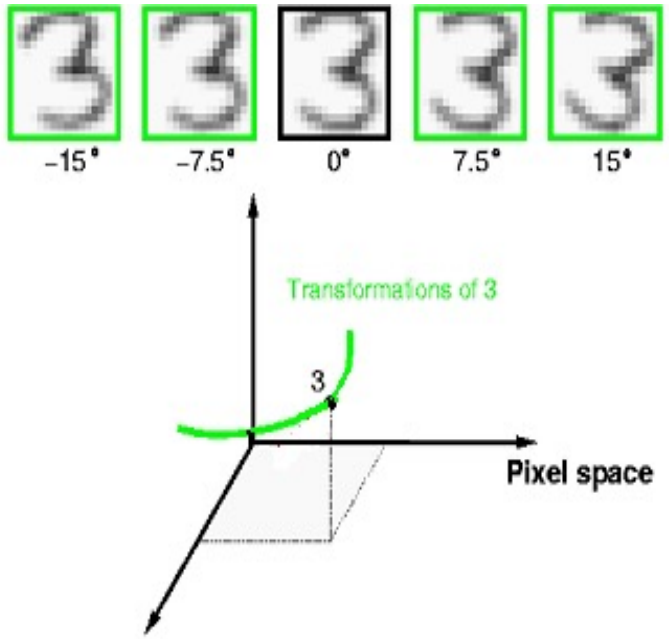
- Note: not all similarities are of this form

Invariant Metrics

- Better similarity functions use knowledge about vision
- Example: invariant metrics:
  - Similarities are invariant under certain transformations
  - Rotation, scaling, translation, stroke-thickness…
  - E.g:

    - 16 x 16 = 256 pixels; a point in 256-dim space
    - These points have small similarity in R^{256} (why?)
  - How can we incorporate such invariances into our similarities?

Rotation Invariant Metrics



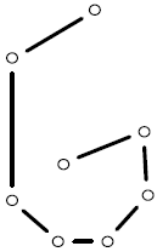
- Each example is now a curve in R^{256}
- Rotation invariant similarity:

s'=max s(r(), r())

- E.g. highest similarity between images' rotation lines

Template Deformation

- Deformable templates:
  - An “ideal” version of each category
  - Best-fit to image using min variance
  - Cost for high distortion of template
  - Cost for image points being far from distorted template
- Used in many commercial digit recognizers



Examples from [Hastie 94]

STEVENS
INSTITUTE OF TECHNOLOGY
1870

Unsupervised Learning

Chapter 20

Recap: Classification

- Classification systems:
  - **Supervised learning**
  - Make a **prediction** given evidence
  - We've seen several methods for this
  - Useful when you have **labeled data**

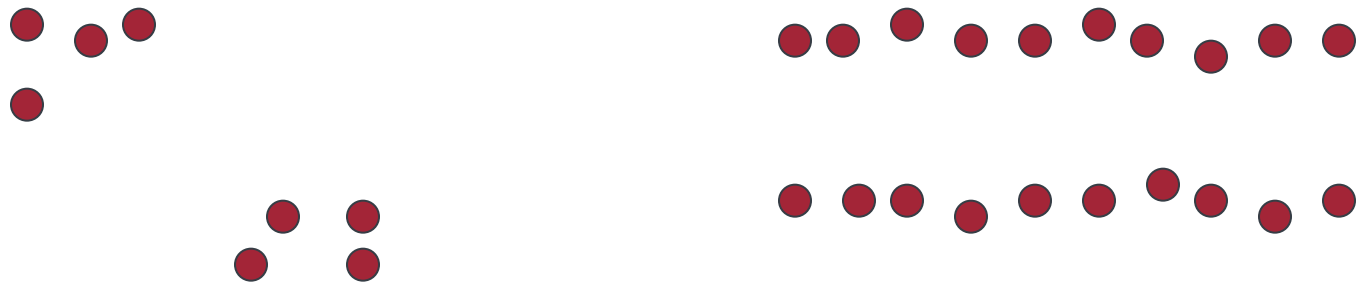
Clustering

- Clustering systems:
  - **Unsupervised learning**
  - **Detect patterns** in unlabeled data
    - E.g. group emails or search results
    - E.g. find categories of customers
    - E.g. detect anomalous program executions
  - Useful when don’t know what you’re looking for
  - Requires data, but no labels
  - Often get gibberish

Clustering

Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



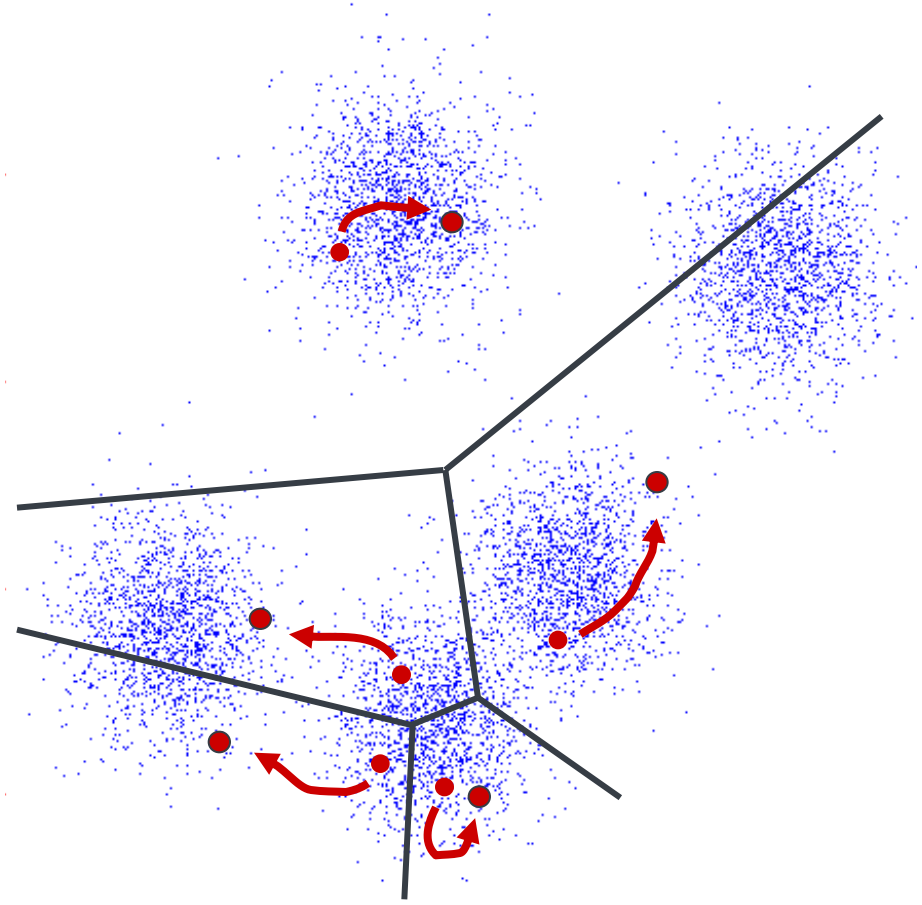
- What could “similar” mean?
  - One option: small (squared) Euclidean distance

$$\text{dist}(x, y) = (x - y)^{\mathsf{T}}(x - y) = \sum_i (x_i - y_i)^2$$

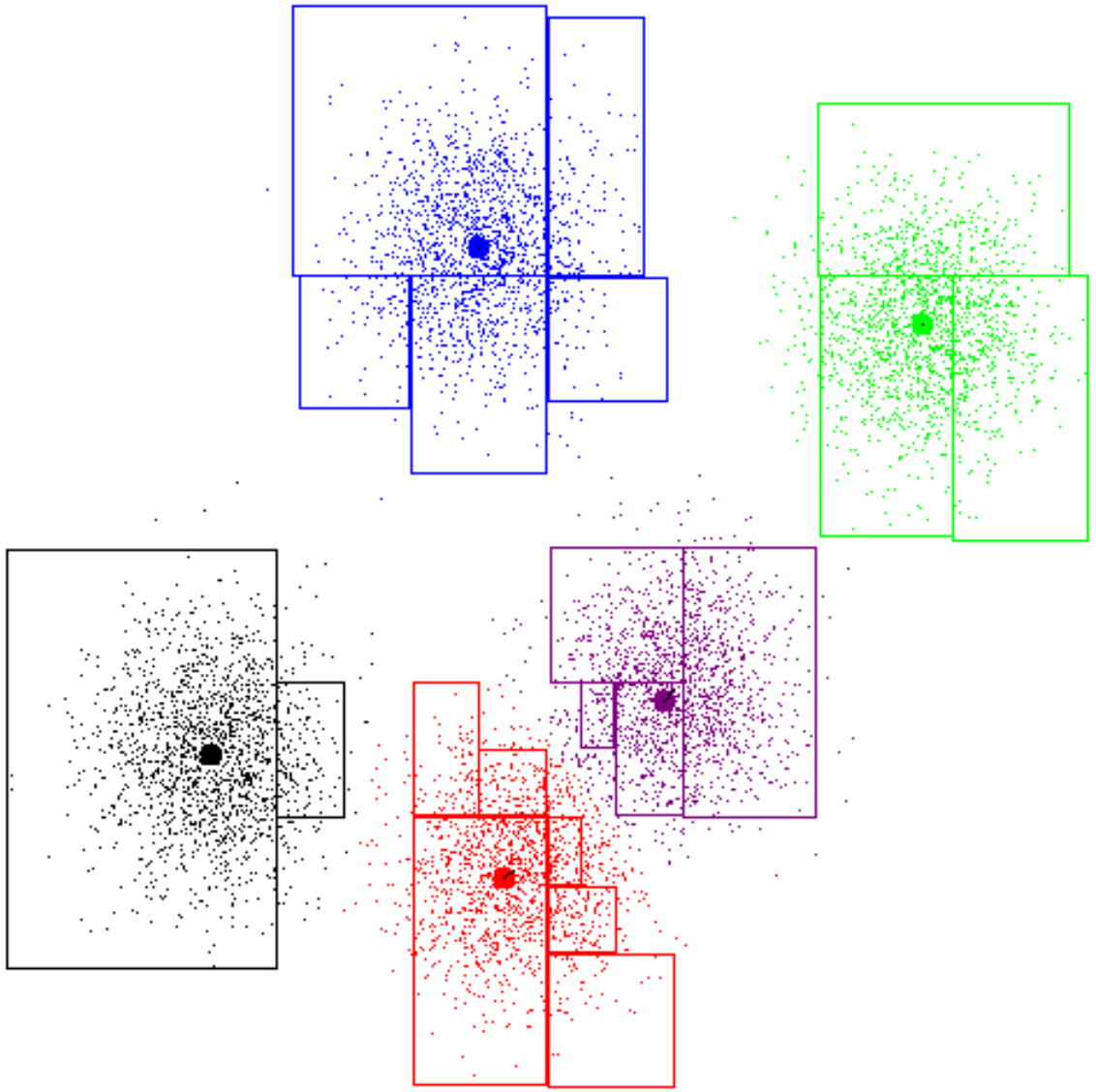
K-Means

K-Means

- An iterative clustering algorithm
  - Pick K random points as cluster centers (means)
  - Alternate:
    - Assign data instances to closest mean
    - Assign each mean to the average of its assigned points
  - Stop when no points' assignments change



K-Means Example

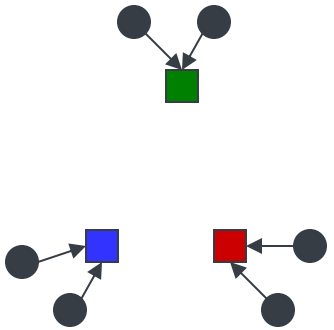


K-Means as Optimization

- Consider the total distance to the means:

$$\phi(\{x_i\}, \{a_i\}, \{c_k\}) = \sum_i \text{dist}(x_i, c_{a_i})$$

points assignments means



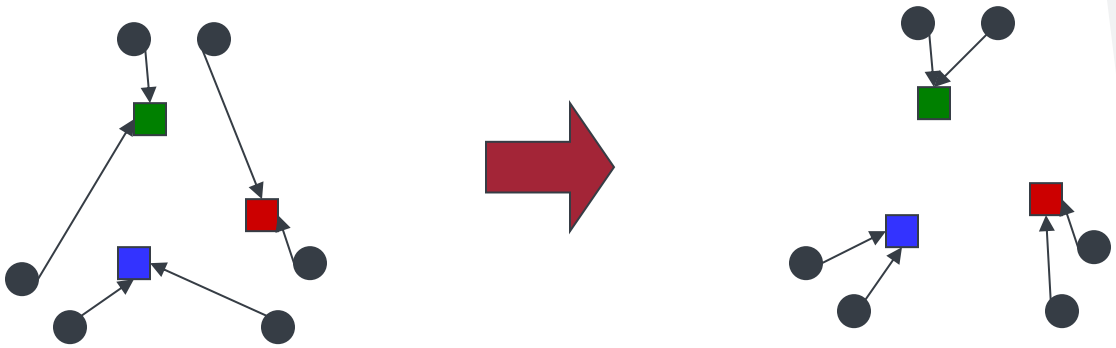
- Each iteration reduces ϕ

- Two stages each iteration:
  - Update assignments: fix means c, change assignments a
  - Update means: fix assignments a, change means c

Phase I: Update Assignments

- For each point, re-assign to closest mean:

$$a_i = \operatorname*{argmin}_k \text{dist}(x_i, c_k)$$



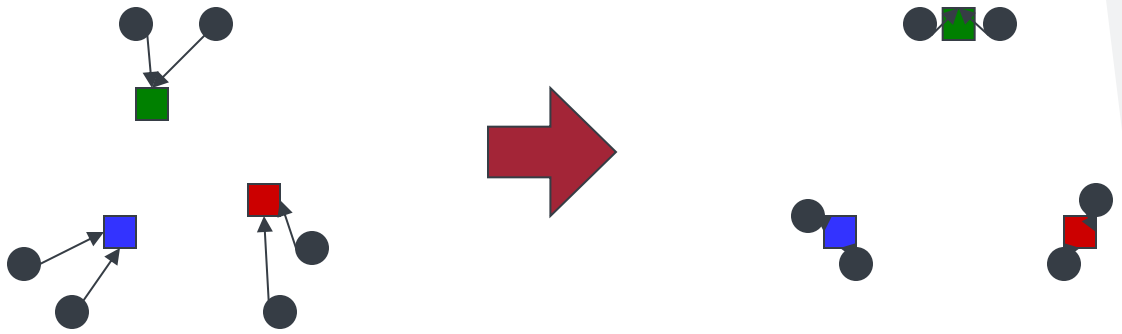
- Can only decrease total distance ϕ!

$$\phi(\{x_i\}, \{a_i\}, \{c_k\}) = \sum_i \text{dist}(x_i, c_{a_i})$$

Phase II: Update Means

- Move each mean to the average of its assigned points:

$$c_k = \frac{1}{|\{i : a_i = k\}|} \sum_{i:a_i=k} x_i$$



- Also, can only decrease total distance… (Why?)

- Fun fact: the point y with minimum squared Euclidean distance to a set of points {x} is their mean

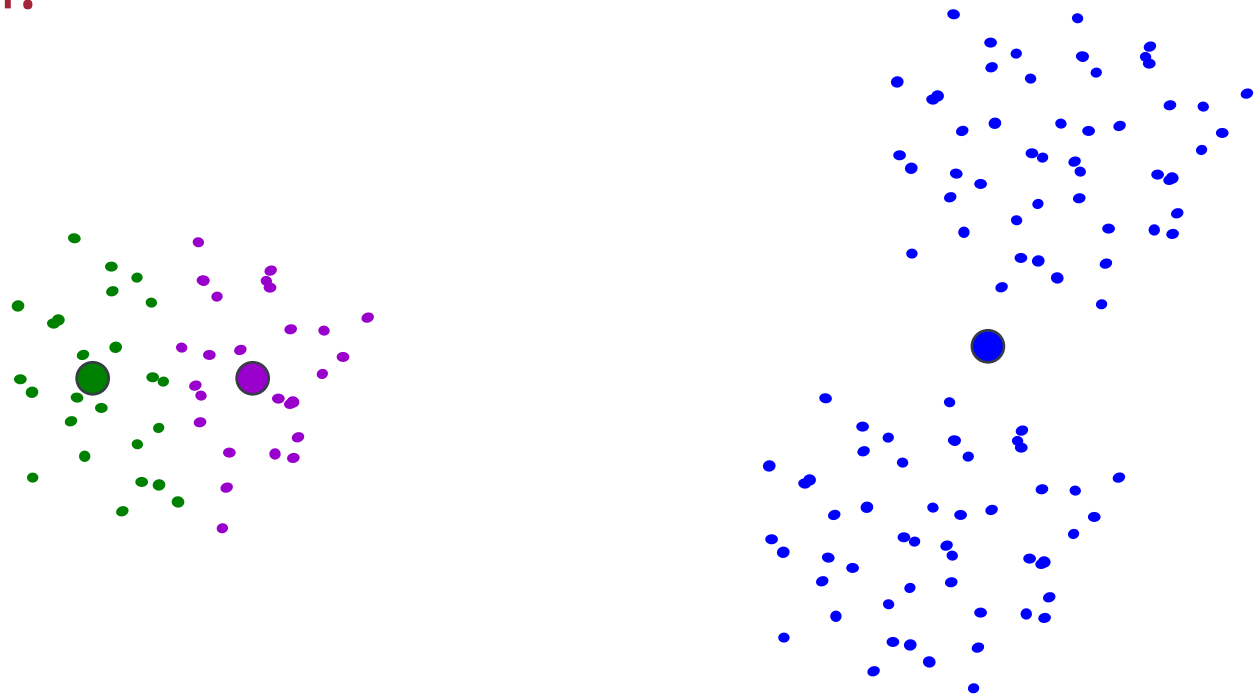
Initialization

- K-means is non-deterministic
  - Requires initial means
  - It does matter what you pick!
  - What can go wrong?

  - Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics

K-Means Getting Stuck

- A local optimum:



Why doesn't this work out like the earlier example, with the purple taking over half the blue?

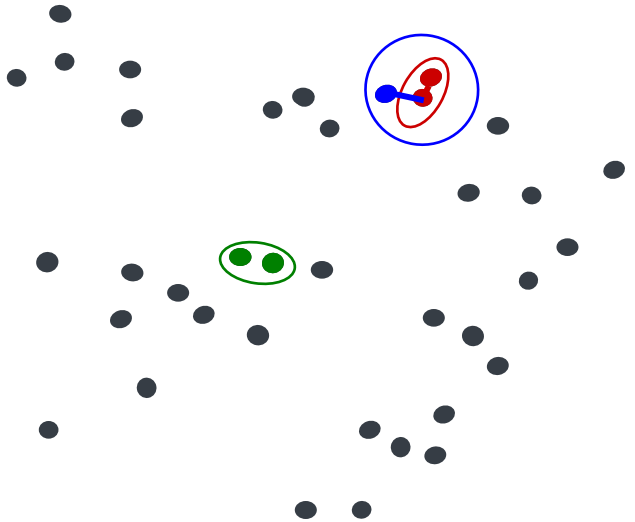
K-Means Questions

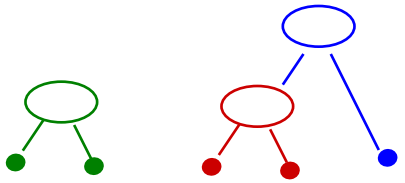
- Will K-means converge?
  - To a global optimum?
- Will it always find the true patterns in the data?
  - If the patterns are very very clear?
- Will it find something interesting?
- How many clusters to pick?

Agglomerative Clustering

Agglomerative Clustering

- Agglomerative clustering:
  - First merge very similar instances
  - Incrementally build larger clusters out of smaller clusters
- Algorithm:
  - Maintain a set of clusters
  - Initially, each instance in its own cluster
  - Repeat:
    - Pick the two **closest** clusters
    - Merge them into a new cluster
    - Stop when there's only one cluster left
- Produces not one clustering, but a family of clusterings represented by a **dendrogram**



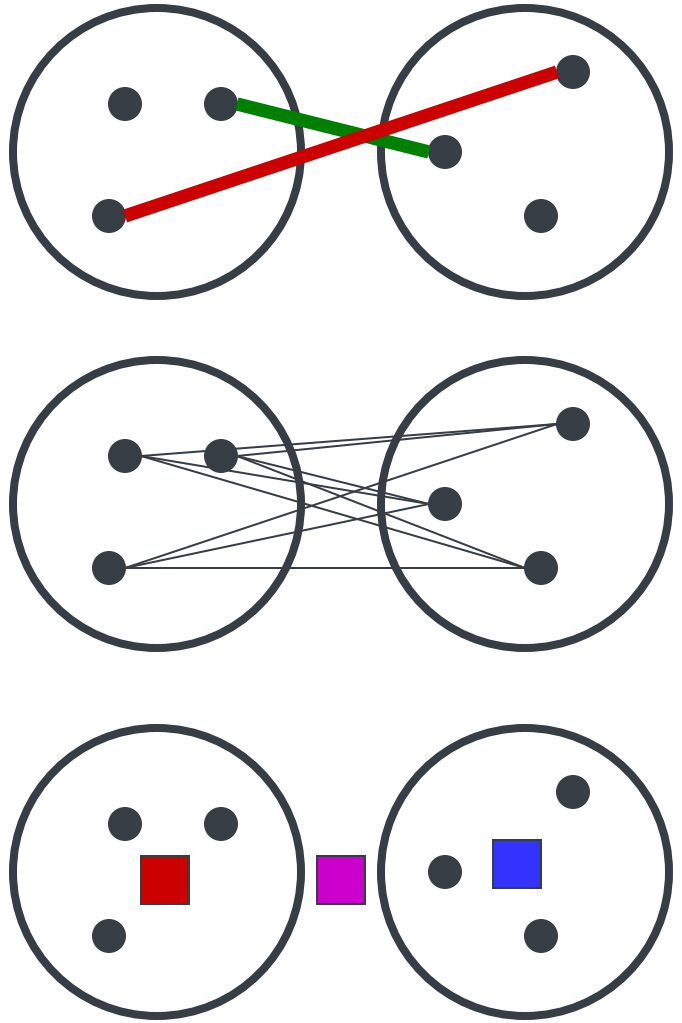


Agglomerative Clustering

- How should we define “closest” for clusters with multiple elements?

- Many options
  - Closest pair (single-link clustering)
  - Farthest pair (complete-link clustering)
  - Average of all pairs
  - Ward’s method (min variance, like k-means)

- Different choices create different clustering behaviors



Example: Google News

Google News

Search News | Search the Web | Advanced news search | Preferences

Search and browse 25,000 news sources updated continuously.

World » edit

Heavy Fighting Continues As Pakistan Army Battles Taliban
Voice of America - **10 hours ago**
By Barry Newhouse Pakistan's military said its forces have killed 55 to 60 Taliban militants in the last 24 hours in heavy fighting in Taliban-held areas of the northwest.
Pakistani troops battle Taliban militants for fourth day guardian.co.uk
Army: 55 militants killed in Pakistan fighting The Associated Press
Christian Science Monitor - CNN International - Bloomberg - New York Times
all 3,824 news articles »


ABC News

Sri Lanka admits bombing safe haven
guardian.co.uk - **3 hours ago**
Sri Lanka has admitted bombing a "safe haven" created for up to 150000 civilians fleeing fighting between Tamil Tiger fighters and the army.
Chinese billions in Sri Lanka fund battle against Tamil Tigers Times Online
Huge Humanitarian Operation Under Way in Sri Lanka Voice of America
BBC News - Reuters - AFP - Xinhua
all 2,492 news articles »


WA today

U.S. » edi

Weekend Opinionator: Souter, Specter and the Future of the GOP
New York Times - **48 minutes ago**
By Tobin Harshaw An odd week. While Barack Obama celebrated his 100th day in office, the headlines were pretty much dominated by the opposition party, albeit not in the way many Republicans would have liked.
US Supreme Court Vacancy An Early Test For Sen Specter Wall Street Journal
Letters: Arlen Specter, Notre Dame, Chrysler Houston Chronicle
The Associated Press - Kansas City Star - Philadelphia Inquirer - Bangor Daily News
all 401 news articles »


FOXNew

Joe Biden, the Flu and You
New York Times - **48 minutes ago**
By GAIL COLLINS The swine flu scare has made it clear why Barack Obama picked Joe Biden for vice president. David Brooks and Gail Collins talk between columns.
After his flu warning, Biden takes the train home The Associated Press
Biden to visit Balkan states in mid-May Washington Post
AFP - Christian Science Monitor - Bizjournals.com - Voice of America
all 1,506 news articles »


TIME

Business » edit

Buffett Calls Investment Candidates' 2008 Performance Subpar
Bloomberg - **2 hours ago**
By Hugh Son, Erik Holm and Andrew Frye May 2 (Bloomberg) -- Billionaire Warren Buffett said all of the candidates to replace him as chief investment officer of Berkshire Hathaway Inc. failed to beat the 38 percent decline of the Standard & Poor's 500 ...
Buffett offers bleak outlook for US newspapers Reuters
Buffett: Limit CEO pay through embarrassment MarketWatch
CNBC - The Associated Press - guardian.co.uk
all 1,454 news articles » BRK.A

Chrysler's Fall May Help Administration Reshape GM
New York Times - **5 hours ago**
Auto task force members, from left: Treasury's Ron Bloom and Gene Sperling, Labor's Edward Montgomery, and Steve Rattner. BY DAVID E. SANGER and BILL VLASIC WASHINGTON - Fresh from pushing Chrysler into bankruptcy, President Obama and his economic team ...
Comment by Gary Chaison Prof. of Industrial Relations, Clark University
Bankruptcy reality sets in for Chrysler, workers Detroit Free Press
Washington Post - Bloomberg - CNNMoney.com
all 11,028 news articles » OTC:FIATY – BIT:FR – GM

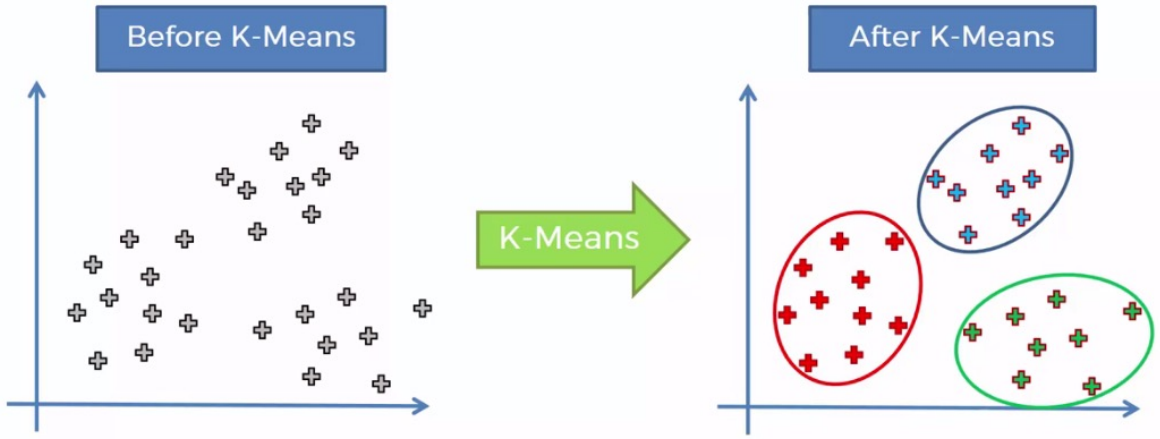

guardian.co.uk

Top-level categories:
supervised classification

Story groupings:
unsupervised clustering

Sample Codes

- Kaggle
  - K-Means Clustering with Python
  - https://www.kaggle.com/code/prashant111/k-means-clustering-with-python



STEVENS
INSTITUTE OF TECHNOLOGY
1870

15 min. break

STEVENS
INSTITUTE OF TECHNOLOGY
1870

Expectation Maximization

Chapter 20

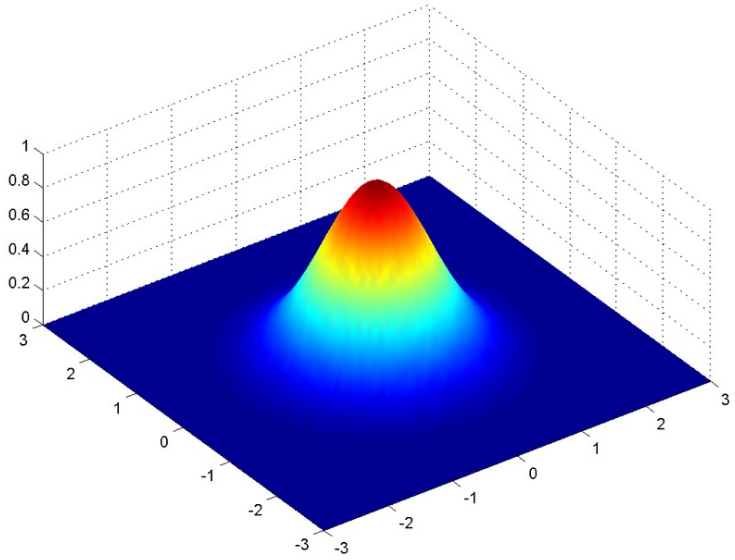
Expectation-Maximization (EM)

- EM algorithm
  - Guess h_{ML}
  - Iterate
    - Expectation: based on h_{ML} compute expectation of (missing) values
    - Maximization: based on expected (missing) values, compute new h_{ML}
- K-means algorithm is a special case of EM
  - EM with one hidden variable (*i.e.*, assignment of clusters to points) and one parameter (*i.e.*, mans of clusters)

EM Example: Learning Mixtures of Gaussians

- Gaussian mixture models
  - A probabilistic model for representing normally distributed subpopulations within an overall population.

K=1

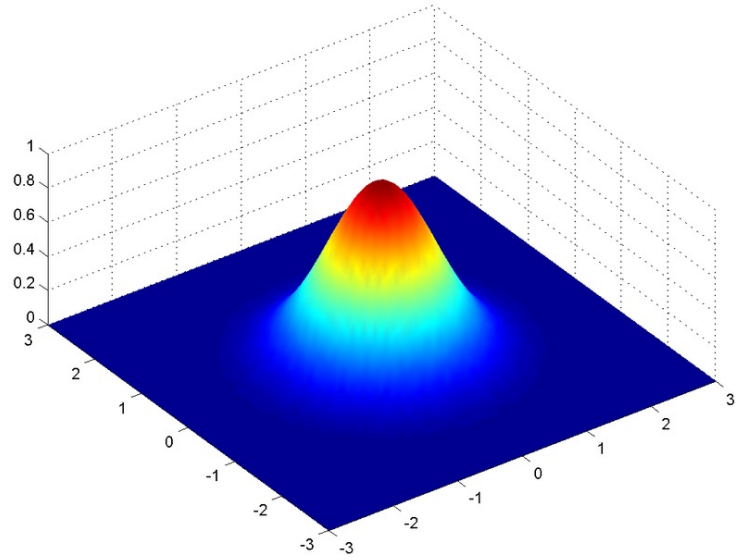


$$\mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{\sqrt{(2\pi)^K |\boldsymbol{\Sigma}_i|}} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i) \right)$$

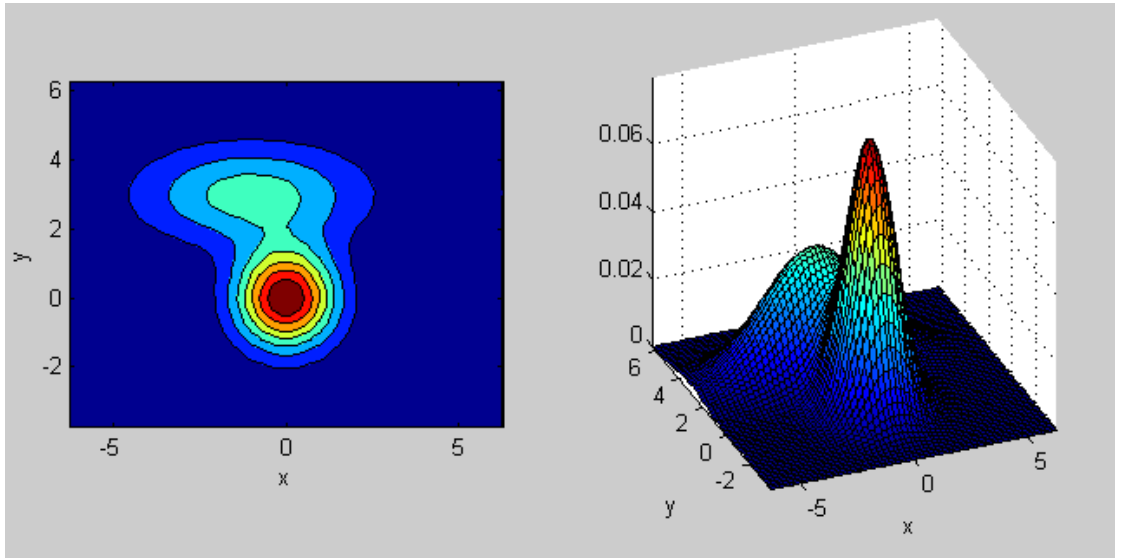
EM Example: Learning Mixtures of Gaussians

- Gaussian mixture models
  - A probabilistic model for representing normally distributed subpopulations within an overall population.

K=1



K=2



Gaussian Mixture Models

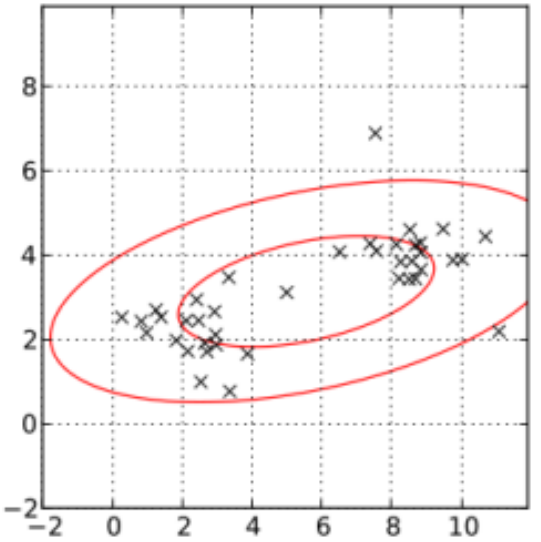
- Gaussian mixture models
  - The probability of points $p(\boldsymbol{x})$ are from a mix of Gaussian probability distributions ($\mathcal{N}$).

$$p(\boldsymbol{x}) = \sum_{i=1}^{K} w_i \mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

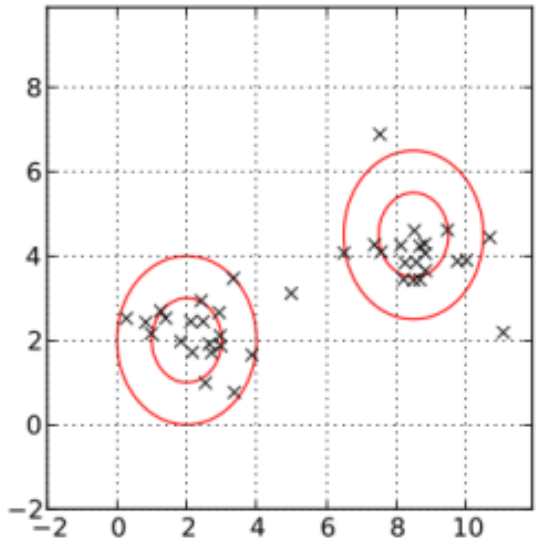
$$\mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{\sqrt{(2\pi)^K |\boldsymbol{\Sigma}_i|}} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i) \right)$$

$$\sum_{i=1}^{K} w_i = 1$$

K=1



K=2



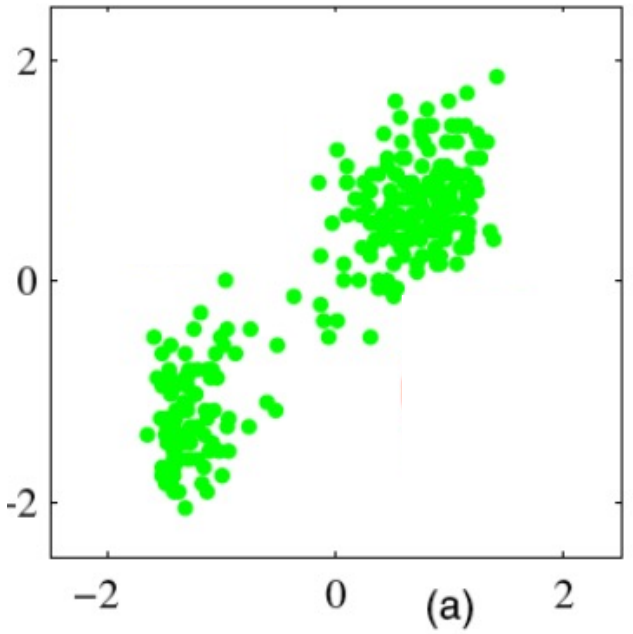
Gaussian Mixture Models

- Gaussian mixture models
  - The probability of points $p(\boldsymbol{x})$ are from a mix of Gaussian probability distributions ($\mathcal{N}$).

$$p(\boldsymbol{x}) = \sum_{i=1}^{K} w_i \mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

$$\mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{\sqrt{(2\pi)^K |\boldsymbol{\Sigma}_i|}} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i) \right)$$

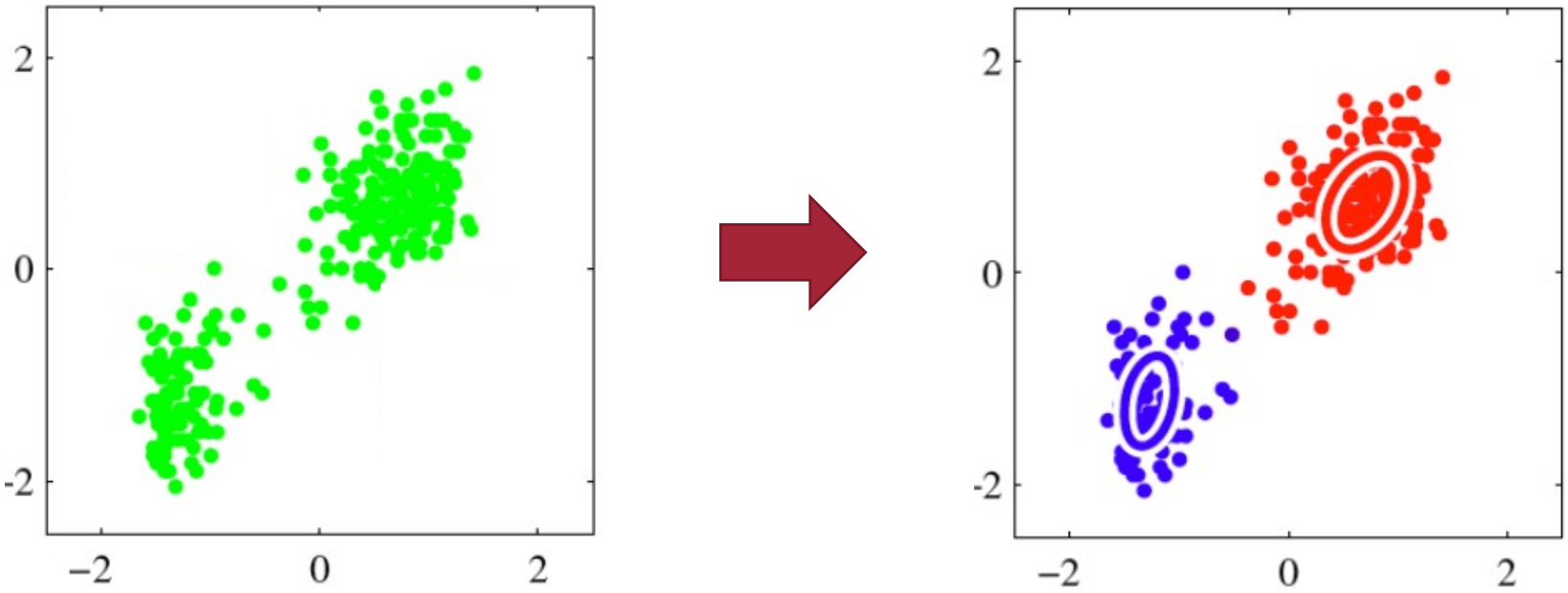
$$\sum_{i=1}^{K} w_i = 1$$



K=2

Learn $w_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$!

EM Example: Learning Mixtures of Gaussians



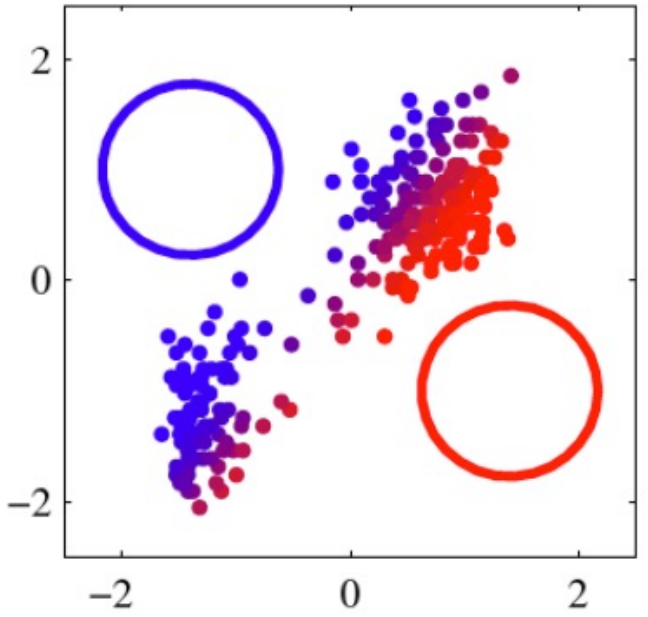
K=2

Maximize the log likelihood

$$w, \boldsymbol{\mu}, \boldsymbol{\Sigma} = \underset{w,\boldsymbol{\mu},\boldsymbol{\Sigma}}{\operatorname{argmax}} \log p(\boldsymbol{X}|w, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \underset{w,\boldsymbol{\mu},\boldsymbol{\Sigma}}{\operatorname{argmax}} \sum_{n=1}^{N} \log \left(\sum_{i=1}^{K} w_i \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \right)$$

EM Example: Learning Mixtures of Gaussians

- Initialization
  - Randomly assign values to the component **mean** estimates $\hat{\mu}_1, \hat{\mu}_2, \ldots, \hat{\mu}_K$.
    - e.g., K=2, $\hat{\mu}_1 = (-1.5, 1), \hat{\mu}_2 = (1.5, -1)$
  - Set all component variance estimates to the sample **variance** $\hat{\Sigma}_1^2, \hat{\Sigma}_2^2, \ldots, \hat{\Sigma}_K^2 = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n - \overline{\boldsymbol{x}})^2$, where $\overline{\boldsymbol{x}}$ is the sample mean $\overline{\boldsymbol{x}} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n$.
  - Set all component distribution **prior** estimates to the uniform distribution $\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_K = \frac{1}{K}$

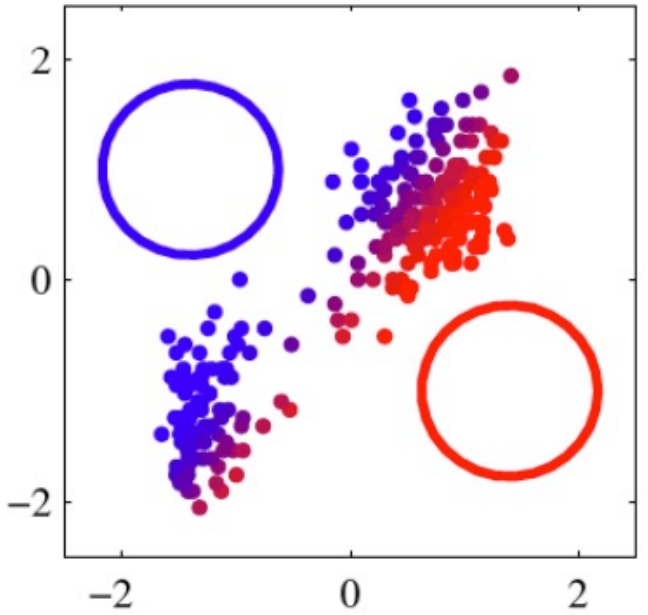


EM Example: Learning Mixtures of Gaussians

- Expectation (E) step
  - Calculate probability ($p_{i,n}$) for all clusters ($\forall i$) and points ($\forall n$)

$$p_{i,n} = P(C = i|\boldsymbol{x}_n, w, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$= \frac{w_i \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{j=1}^{K} w_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

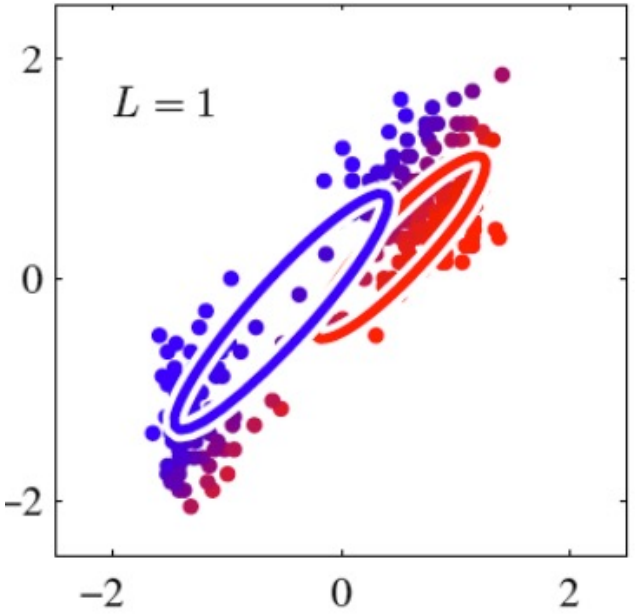


EM Example: Learning Mixtures of Gaussians

- Maximization (M) step
  - Using the $p_{i,n}$ in the expectation step, calculate the following, $\forall i$:

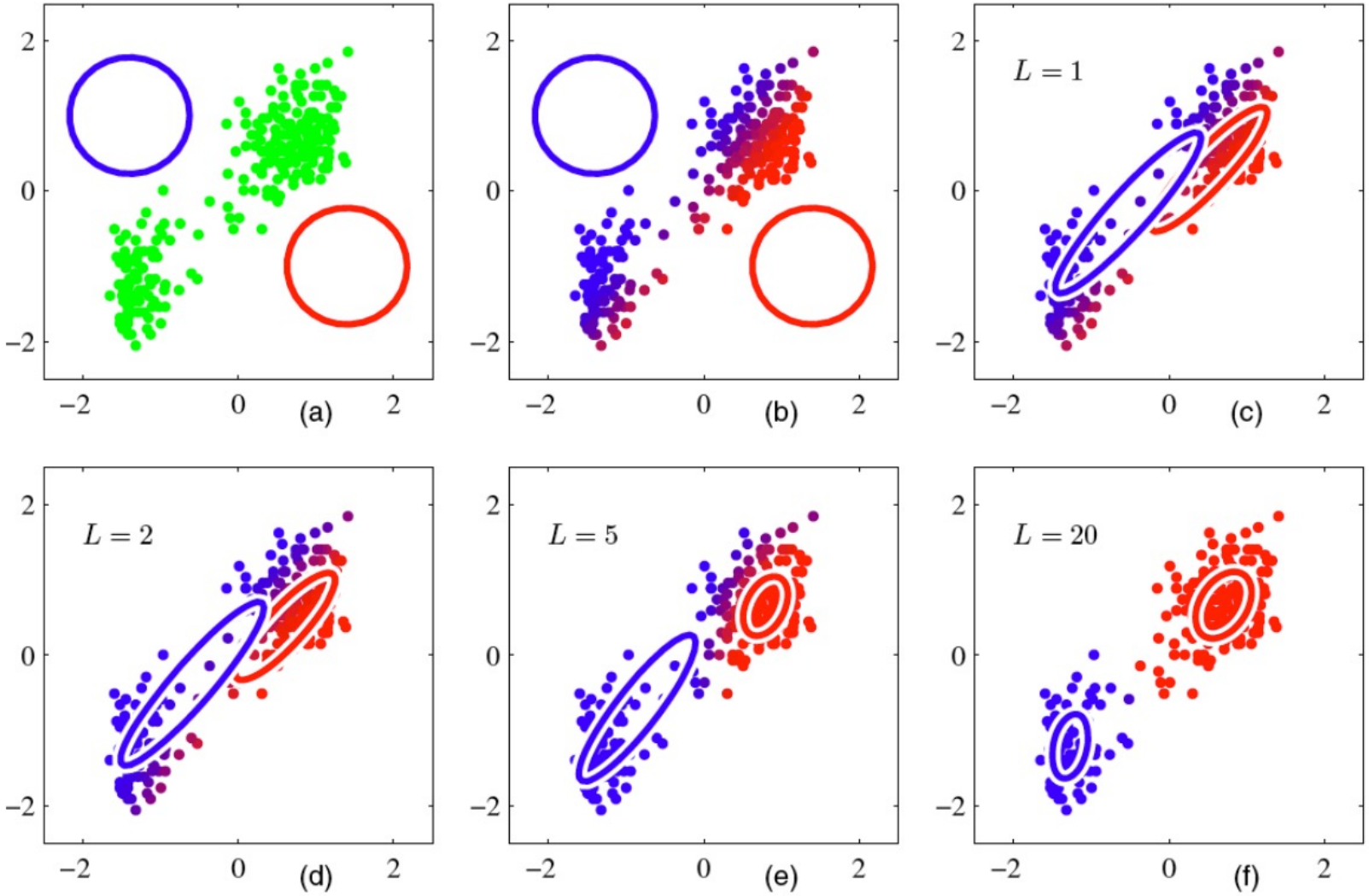
$$\widehat{w}_i = \sum_{n=1}^{N} \frac{p_{i,n}}{N} \qquad \widehat{\boldsymbol{\mu}}_i = \frac{\sum_{n=1}^{N} p_{i,n} \boldsymbol{x}_n}{\sum_{n=1}^{N} p_{i,n}}$$

$$\boldsymbol{\Sigma}_i = \frac{\sum_{n=1}^{N} p_{i,n} (\boldsymbol{x}_n - \widehat{\boldsymbol{\mu}}_i)^2}{\sum_{n=1}^{N} p_{i,n}}$$

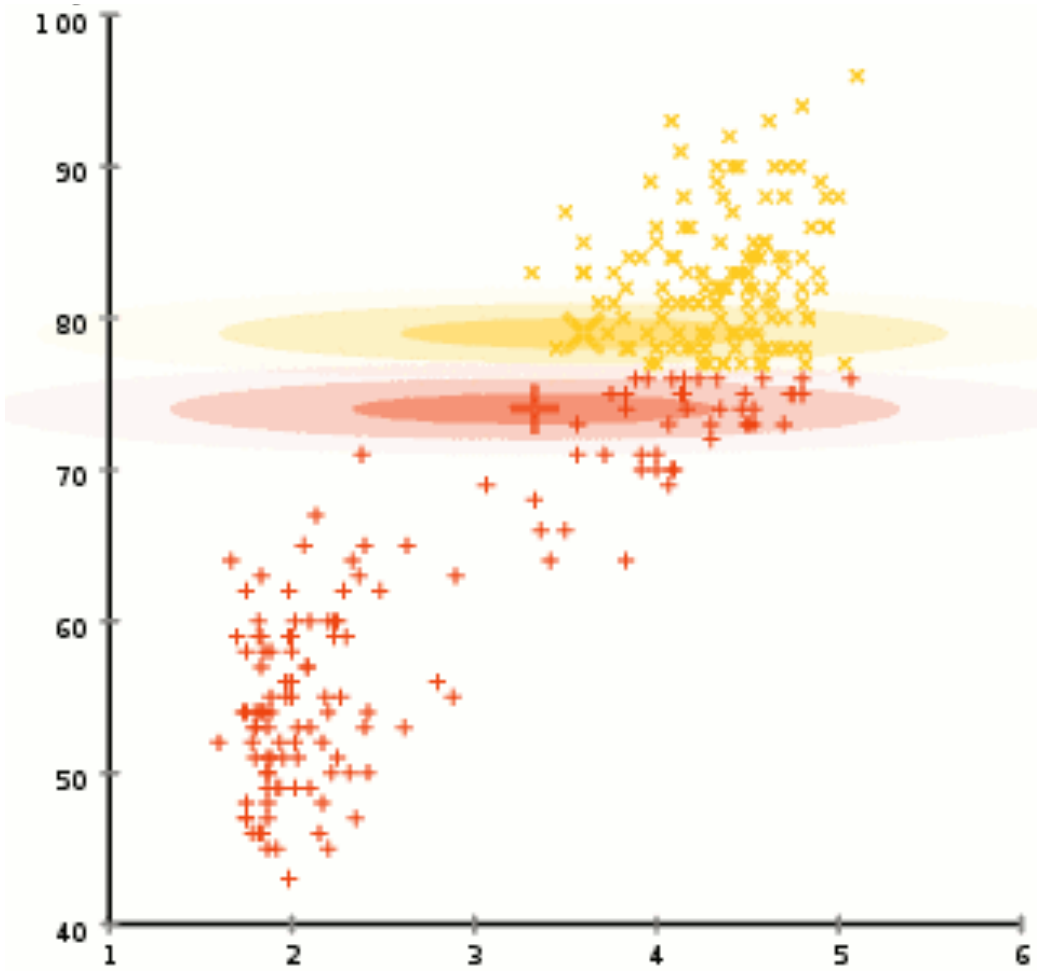


EM Example: Learning Mixtures of Gaussians

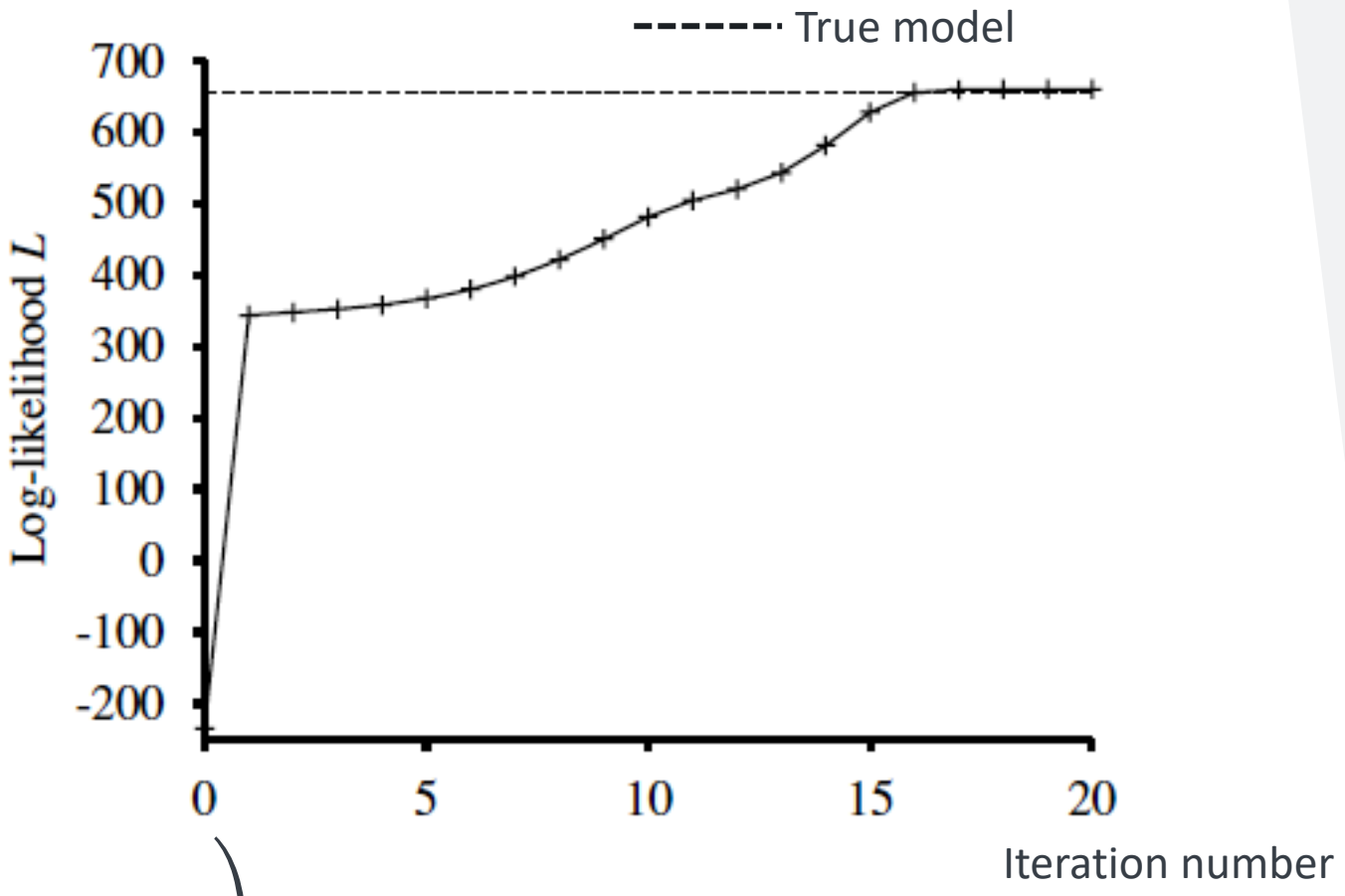
- Iterate
  - Expectation (E) step
  - Maximization (M) step



EM Example: Learning Mixtures of Gaussians



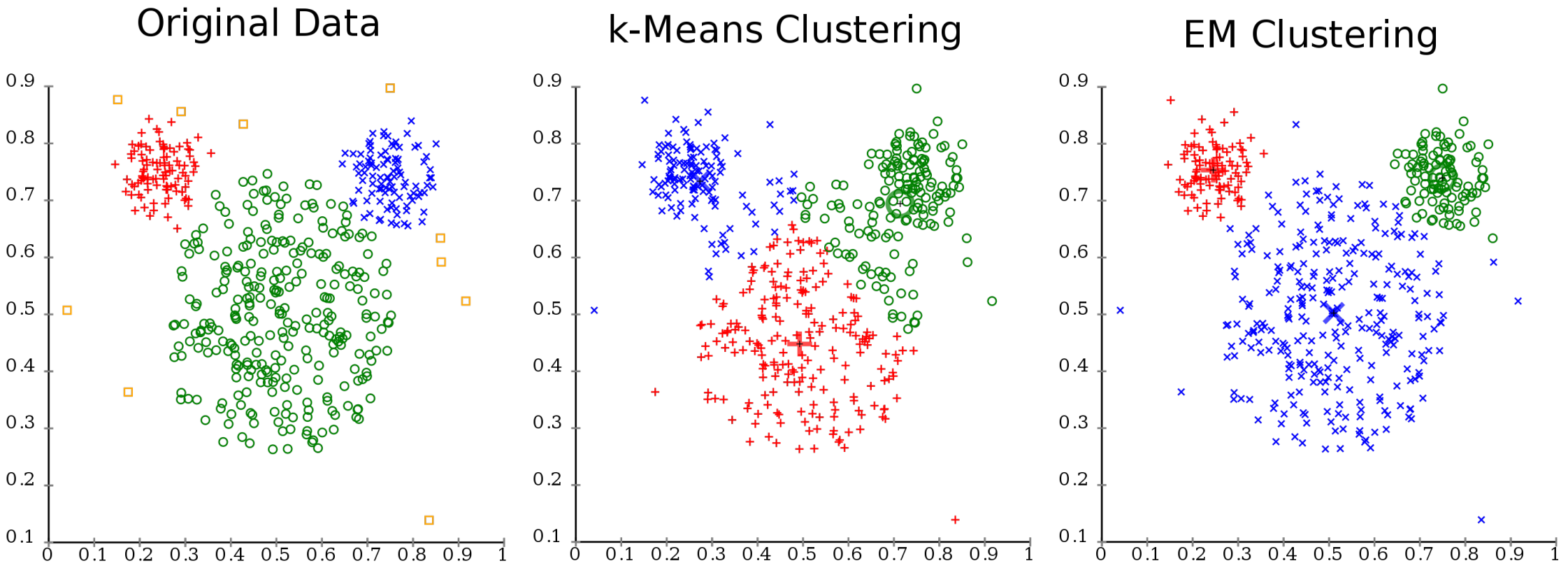
EM Example: Learning Mixtures of Gaussians



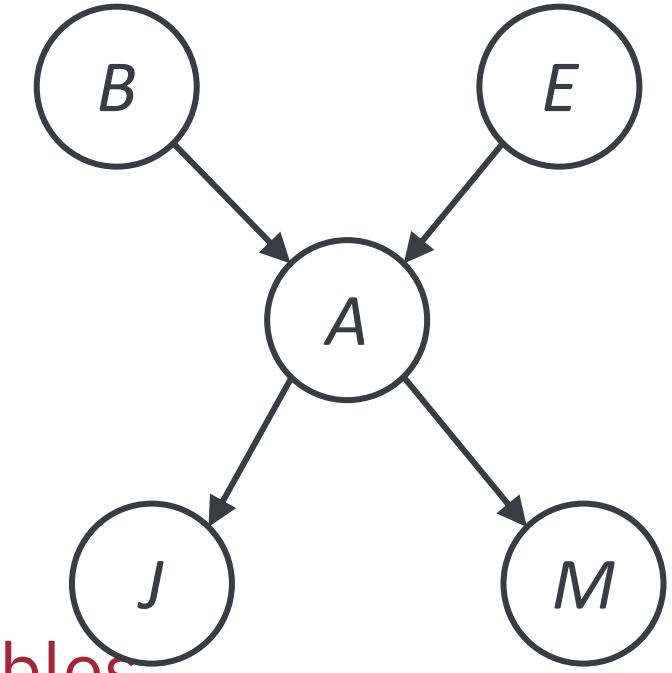
Maximize the log likelihood

$$\log p(\boldsymbol{X}|w, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \log \left(\sum_{i=1}^{K} w_i \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \right)$$

K-Means vs EM


Original Data
k-Means Clustering
EM Clustering
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

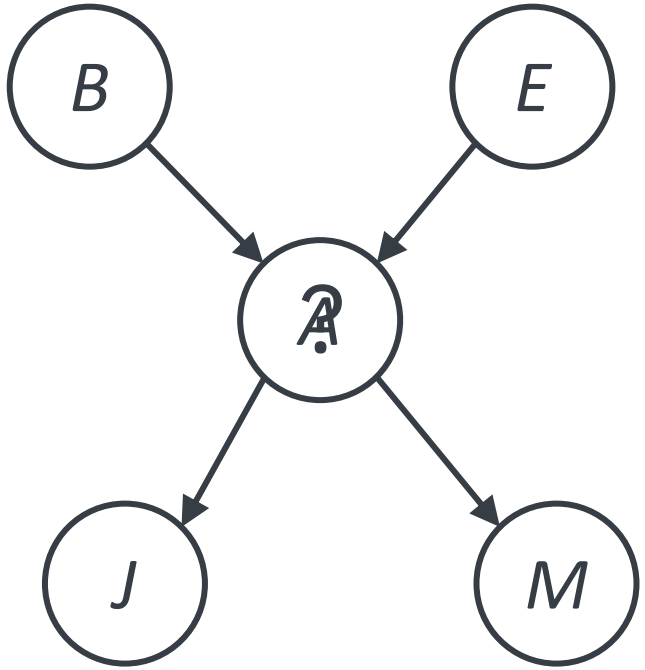
EM Example: Bayes' Net



- So far we have seen problems where
  - Values of all attributes are known
  - Learning is relatively easy
- Many real-world problems have hidden variables
  - Incomplete data
  - Missing attribute values

Burglars and Earthquakes

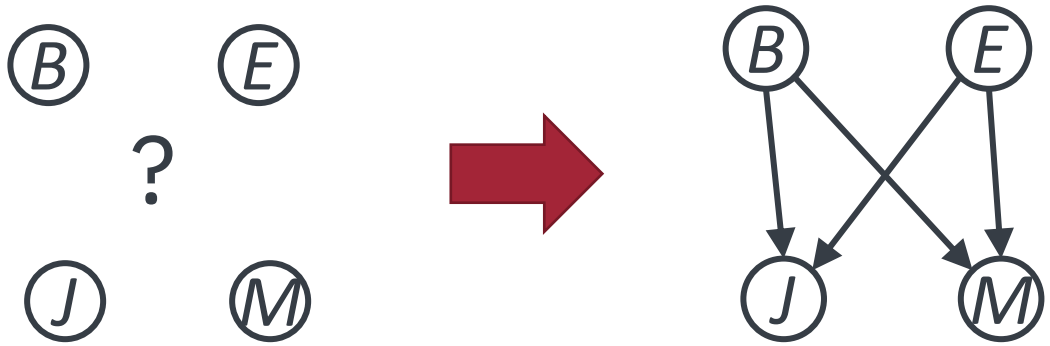
Earthquake	Burglars	#
0	0	1000
0	0	10
0	1	20
0	1	100
1	0	200
1	0	50
1	1	0
1	1	5



	P(+a|E, B)
+e, +b	
+e, -b	?
-e, +b	
-e, -b	

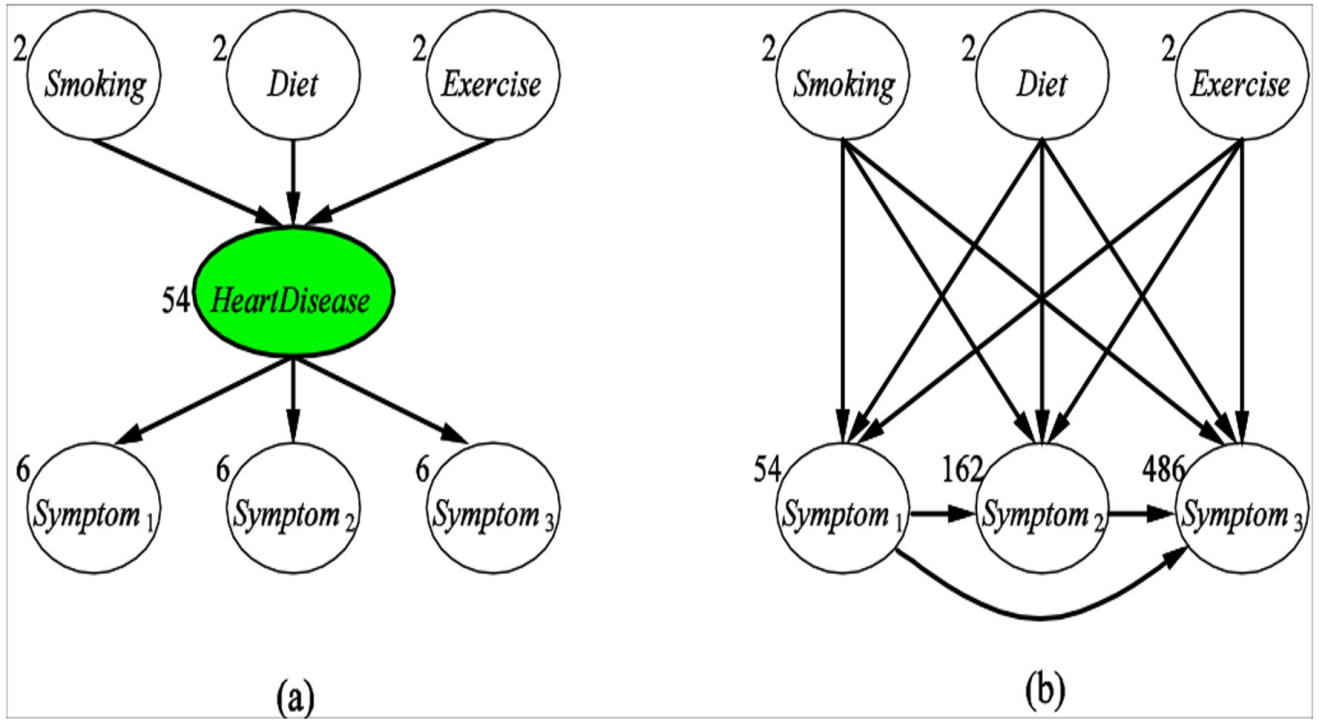
Solutions

- Ignore hidden variables
  - Model might become much more complex



- Use expectation-maximization

Hidden Variables: Heart Disease Example



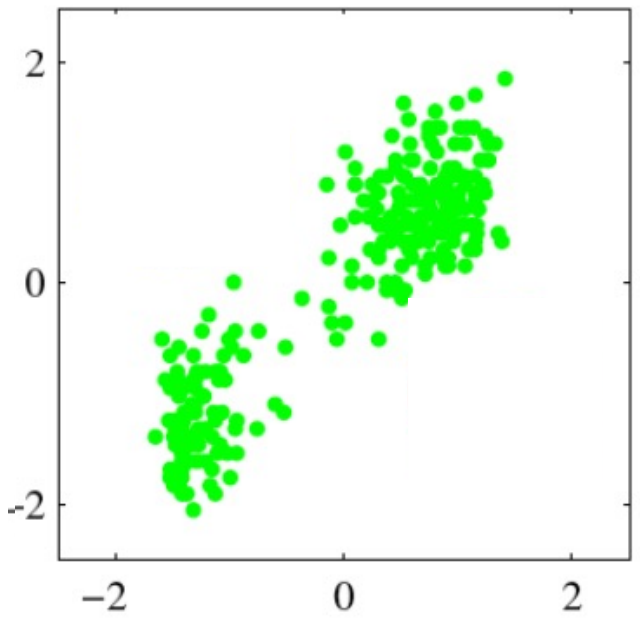
- (a) uses a hidden variable, simpler (fewer CPT parameters)
- (b) no hidden variable, complex (many CPT parameters)

Maximum Likelihood Learning

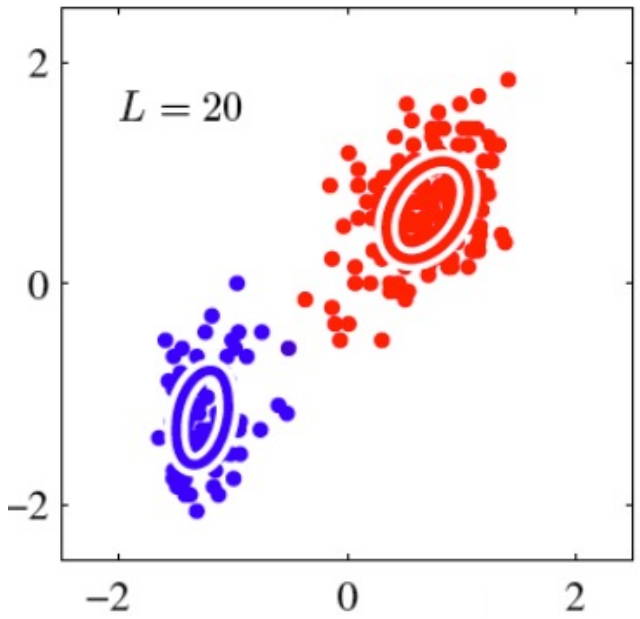
- Learning of Bayes nets parameters
  - $P(+a|+e,-b) = \frac{Number\ of\ instances\ where\ A=+a,E=+a,B=-b}{Number\ of\ instances\ where\ E=+a,B=-b}$

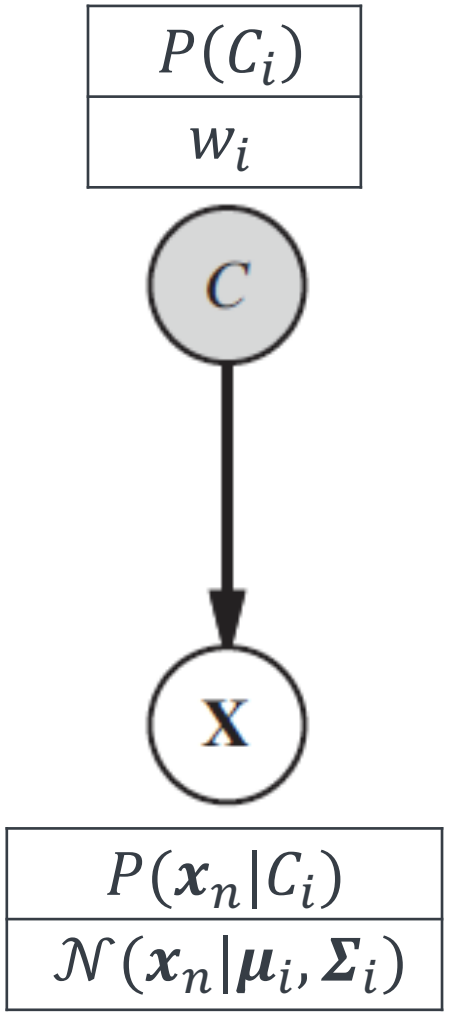
- Assumes the Bayes nets have all variables
  - What if some variables are missing?

Hidden Variables: Gaussian Mixture Models



Hidden Variables: Gaussian Mixture Models



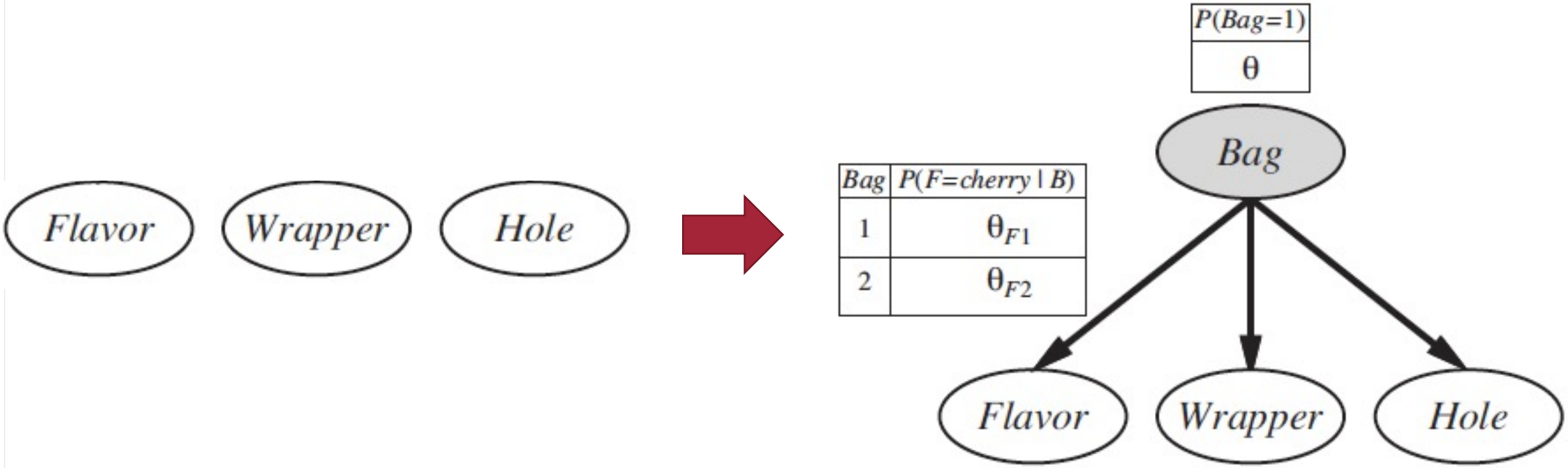


$$p_{i,n} = P(C = i|\boldsymbol{x}_n, w, \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$= \frac{w_i \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{j=1}^{K} w_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

$$= \frac{P(C_i)P(\boldsymbol{x}_n|C_i)}{P(\boldsymbol{x}_n)}$$

Hidden Variables: A Mixture Model for Candy



A Mixture Model for Candy

- Assumption
  - The two bags of candy have noticeably different distributions of flavors and wrappers, with some candies having holes and others not.
- Parameters
  - θ: a candy is from Bag 1
  - θ_{F1}, θ_{F1}: the probabilities that the flavor is cherry, given that the candy comes from Bag 1 or Bag 2 respectively
  - θ_{W1}, θ_{W1}: the wrapper is red
  - θ_{H1}, θ_{H1}: the candy has a hole

Learning Bayes' Nets with Hidden Variables

- Ground truth

$$\theta = 0.5, \quad \theta_{F1} = \theta_{W1} = \theta_{H1} = 0.8, \quad \theta_{F2} = \theta_{W2} = \theta_{H2} = 0.3$$

- Counts for the eight possible kinds of candy

	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	273	93	104	90
F=lime	97	100	94	167

Learning Bayes' Nets with Hidden Variables

- Initialization

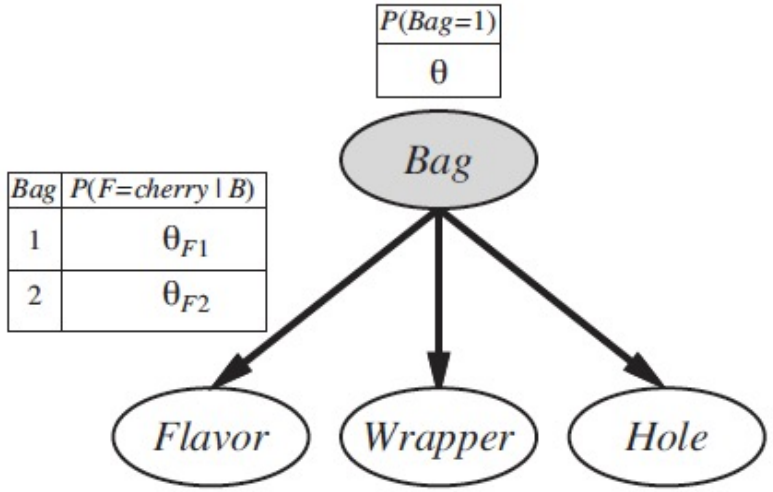
$$\theta^{(0)} = 0.6$$

$$\theta_{F1}^{(0)} = \theta_{W1}^{(0)} = \theta_{H1}^{(0)} = 0.6$$

$$\theta_{F2}^{(0)} = \theta_{W2}^{(0)} = \theta_{H2}^{(0)} = 0.4$$

Learning Bayes' Nets with Hidden Variables

- Expectation (E) step
  - Calculate for all candies ($\forall j$)



$$p_j = P(Bag = 1 | flabor_j, wrapper_j, holes_j)$$

$$= \frac{P(Bag = 1, flabor_j, wrapper_j, holes_j)}{\sum_i P(Bag = i, flabor_j, wrapper_j, holes_j)}$$

$$= \frac{P(flabor_j | Bag = 1) P(wrapper_j | Bag = 1) P(holes_j | Bag = 1) P(Bag = 1)}{\sum_i P(flabor_j | Bag = i) P(wrapper_j | Bag = i) P(holes_j | Bag = i) P(Bag = i)}$$

$$= \frac{\theta_{F1}^{(0)} \theta_{W1}^{(0)} \theta_{W1}^{(0)} \theta^{(0)}}{\theta_{F1}^{(0)} \theta_{W1}^{(0)} \theta_{W1}^{(0)} \theta^{(0)} + \theta_{F2}^{(0)} \theta_{W2}^{(0)} \theta_{W2}^{(0)} \theta^{(0)}}$$

Learning Bayes' Nets with Hidden Variables

- Maximization (M) step
  - Using the p_j calculated in the expectation step, calculate the parameters

$$\theta^{(1)} = \frac{\widehat{N}(Bag = 1)}{N} = \frac{\sum_{j=1}^{N} p_j}{N} = \frac{1}{N}\sum_{j=1}^{N} P(Bag = 1 | flabor_j, wrapper_j, holes_j)$$

  - Applying this formula to the 273 red-wrapped cherry candies, we get a contribution of

$$\frac{273}{1000} p_j \approx 0.22796$$

  - Continuing with the other seven kinds of candy in the table of counts, we obtain

$$\theta^{(1)} = 0.6124$$

Learning Bayes' Nets with Hidden Variables

- Maximization (M) step
  - Other parameters
    1. Compute the expected count of candies with p_j

$$\sum_{j:F=cherry,W=red,H=1}^{N} p_j = 273 * p_j$$

	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	273	93	104	90
F=lime	97	100	94	167

Bag 1

	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	227.96	64.38	72.0	45
F=lime	67.15	50	47	51.38

	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	45.04	28.62	32	45
F=lime	29.85	50	47	115.62

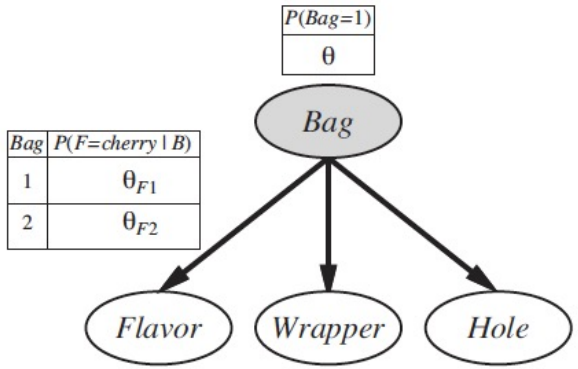
Bag 2

Learning Bayes' Nets with Hidden Variables

- Maximization (M) step
  - Other parameters
    2. Calculate parameters

$$\theta_{F1}^{(0)} = P(flabor = cherry | Bag = 1)$$
$$= \frac{P(Bag = 1, flabor = cherry)}{P(Bag = 1)}$$
$$= \frac{227.96 + 64.38 + 72.0 + 45}{\theta^{(1)} * 1000}$$
$$= 0.6684$$

(The process to calculate the remaining parameters is similar.)



Bag 1

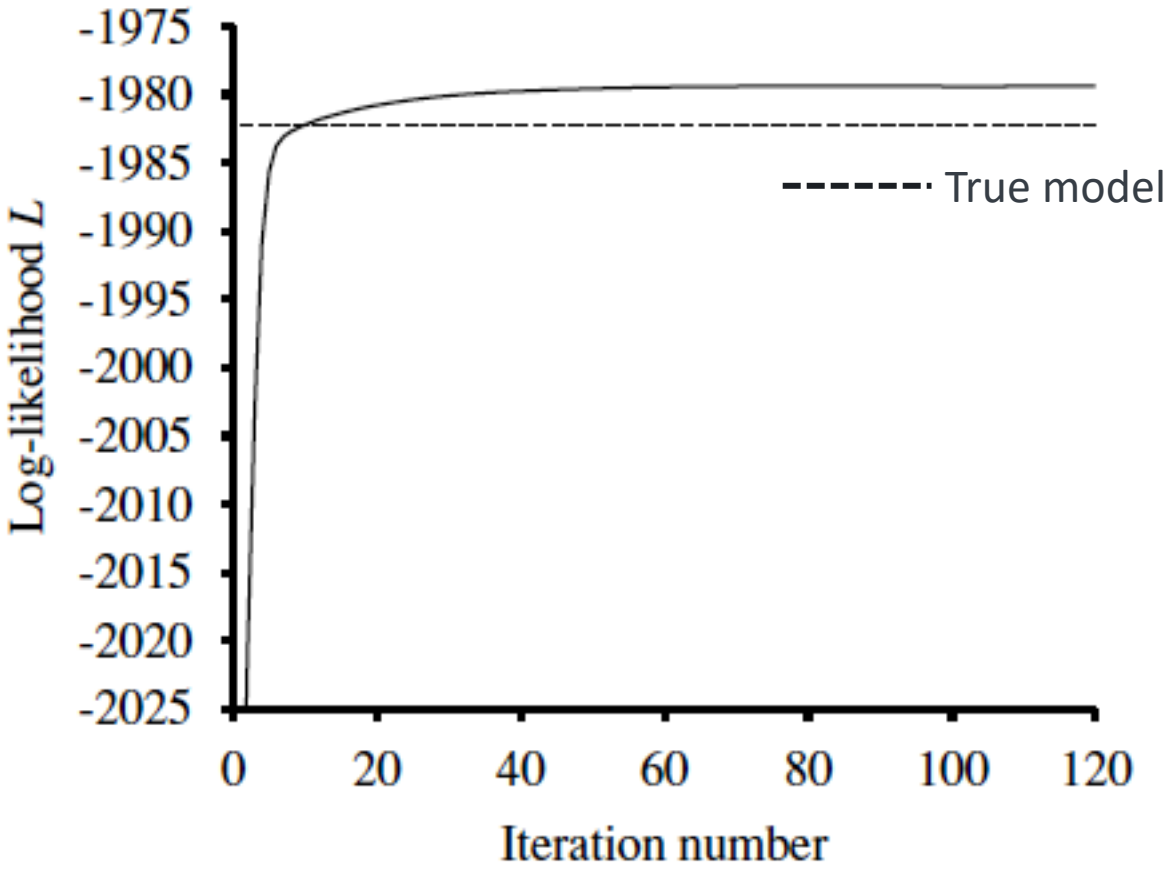
	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	227.96	64.38	72.0	45
F=lime	67.15	50	47	51.38

	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	45.04	28.62	32	45
F=lime	29.85	50	47	115.62

Bag 2

EM Example: Learning Mixtures of Gaussians

- Iterate
  - Expectation (E) step
  - Maximization (M) step



Maximize the log likelihood

$$\log p(\boldsymbol{X}|\boldsymbol{\theta}) = \sum_{j=1}^{N} \log \left(\sum_{i=1}^{2} P(flabor_j, wrapper_j, holes_j | Bag = i) \right)$$

The General Form of the EM Algorithm

The General Form of the EM Algorithm

- The examples involve two steps:
  - Step 1 (E): computing expected values of hidden variables for each example
  - Step 2 (M): recomputing the parameters, using the expected values as if they were observed values.

The General Form of the EM Algorithm

- The general form

$$\theta^{(i+1)} = \underset{\theta}{\operatorname{argmax}} \sum_{\boldsymbol{Z}} P\left(\boldsymbol{Z} = \boldsymbol{z} \middle| \boldsymbol{x}, \boldsymbol{\theta}^{(i)}\right) L(\boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z} | \boldsymbol{\theta})$$

  where $\boldsymbol{Z}$ denote all the hidden variables for all the examples, L is log likelihood, and $\boldsymbol{\theta}$ be all the parameters for the probability model.

- e.g., Gaussian Mixture Models
  - Hidden variables ($\boldsymbol{Z}$): Z_{ij} where Z_{ij} is 1 if example j was generated by component I
  - Parameters ($\boldsymbol{\theta}$): $w, \boldsymbol{\mu}, \boldsymbol{\Sigma}$

The General Form of the EM Algorithm

- The general form

$$\theta^{(i+1)} = \underset{\theta}{\operatorname{argmax}} \underbrace{\sum_{\boldsymbol{Z}} P\left(\boldsymbol{Z} = \boldsymbol{z} \middle| \boldsymbol{x}, \boldsymbol{\theta}^{(i)}\right) L(\boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z} | \boldsymbol{\theta})}$$

- The E-step is the computation of summation
  - The expectation of the log likelihood of the "completed" data with respect to the distribution $P\left(\boldsymbol{Z} = \boldsymbol{z} \middle| \boldsymbol{x}, \boldsymbol{\theta}^{(i)}\right)$

The General Form of the EM Algorithm

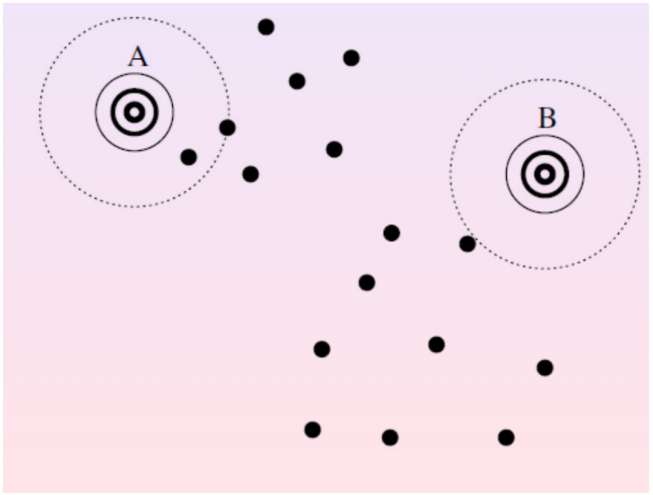
- The general form

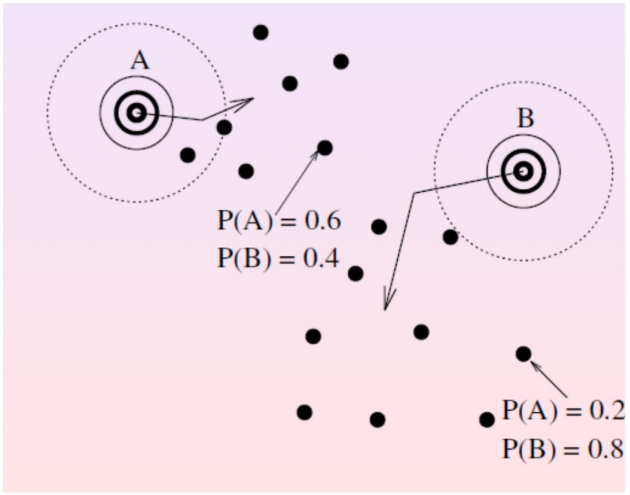
$$\theta^{(i+1)} = \underset{\theta}{\operatorname{argmax}} \sum_{\boldsymbol{Z}} P\left(\boldsymbol{Z} = \boldsymbol{z} \middle| \boldsymbol{x}, \boldsymbol{\theta}^{(i)}\right) L(\boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z} | \boldsymbol{\theta})$$

- The M-step is maximizing the summation with θ

Sample Codes

- Kaggle
  - Gaussian Mixture Models
  - https://www.kaggle.com/code/charel/learn-by-example-expectation-maximization/notebook





STEVENS
INSTITUTE OF TECHNOLOGY
1870

Questions?

Acknowledgement

Fahiem Bacchus, University of Toronto
Dan Klein, UC Berkeley
Kate Larson, University of Waterloo