

# DATA MINING CLASSIFICATION

---

Nearest Neighbor Classification

Support Vector Machines

Logistic Regression

Naïve Bayes Classifier

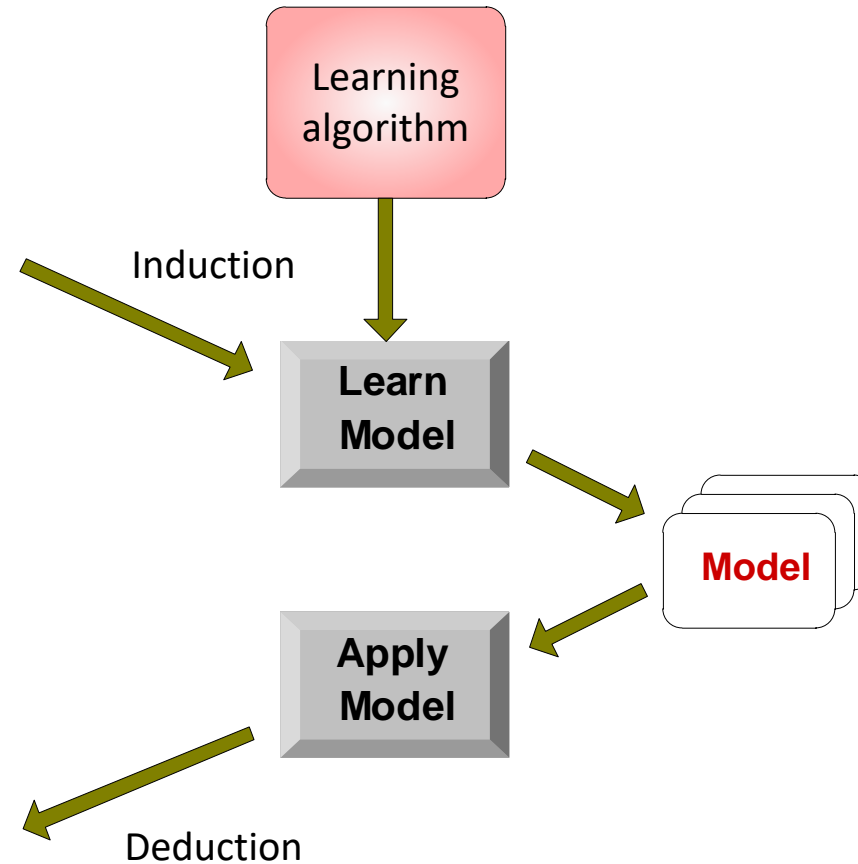
# Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



# NEAREST NEIGHBOR CLASSIFICATION

---

# Instance-Based Classifiers

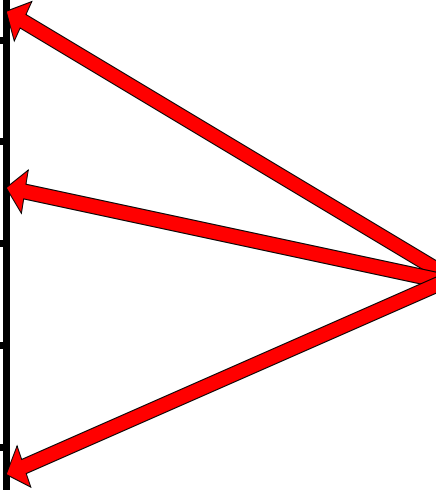
Set of Stored Cases

Atr1	.....	AtrN	Class
			A
			B
			B
			C
			A
			C
			B

- Store the training records
- Use training records to predict the class label of unseen cases

Unseen Case

Atr1	.....	AtrN

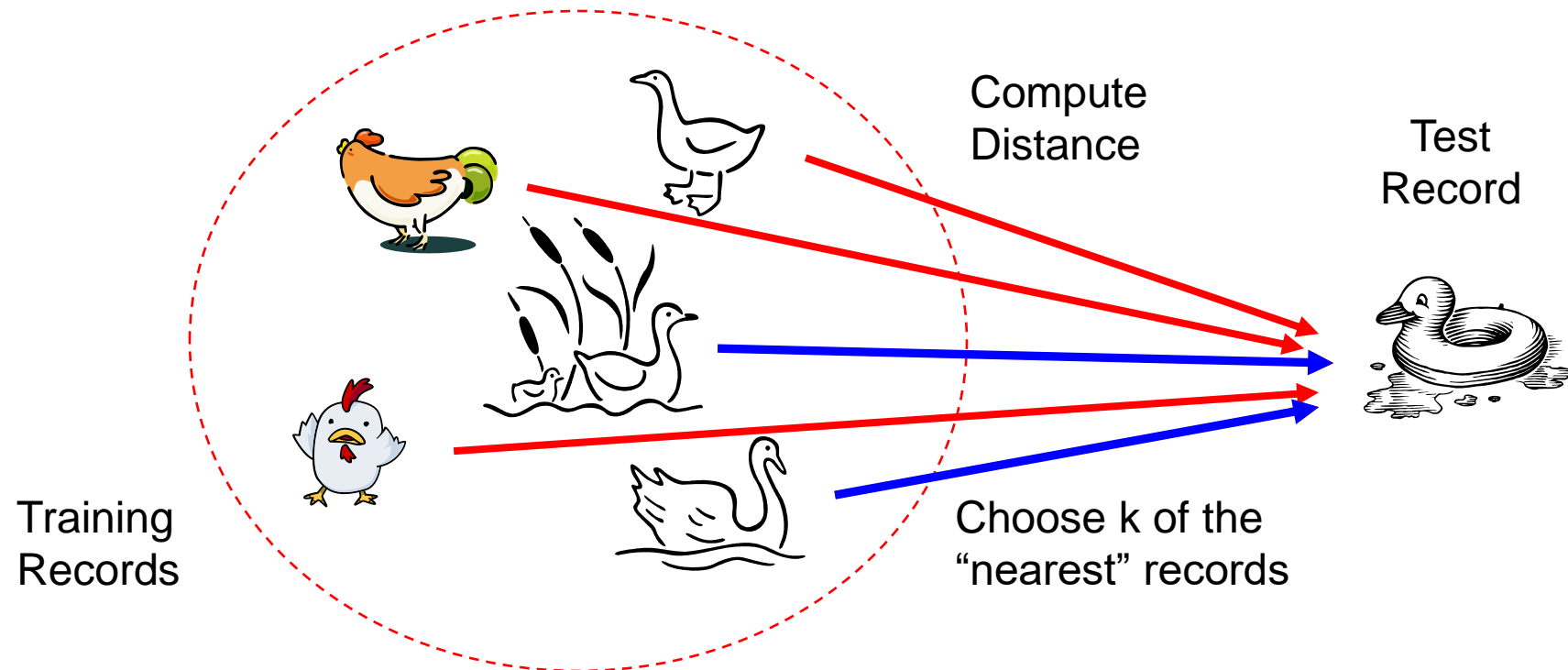


# Instance Based Classifiers

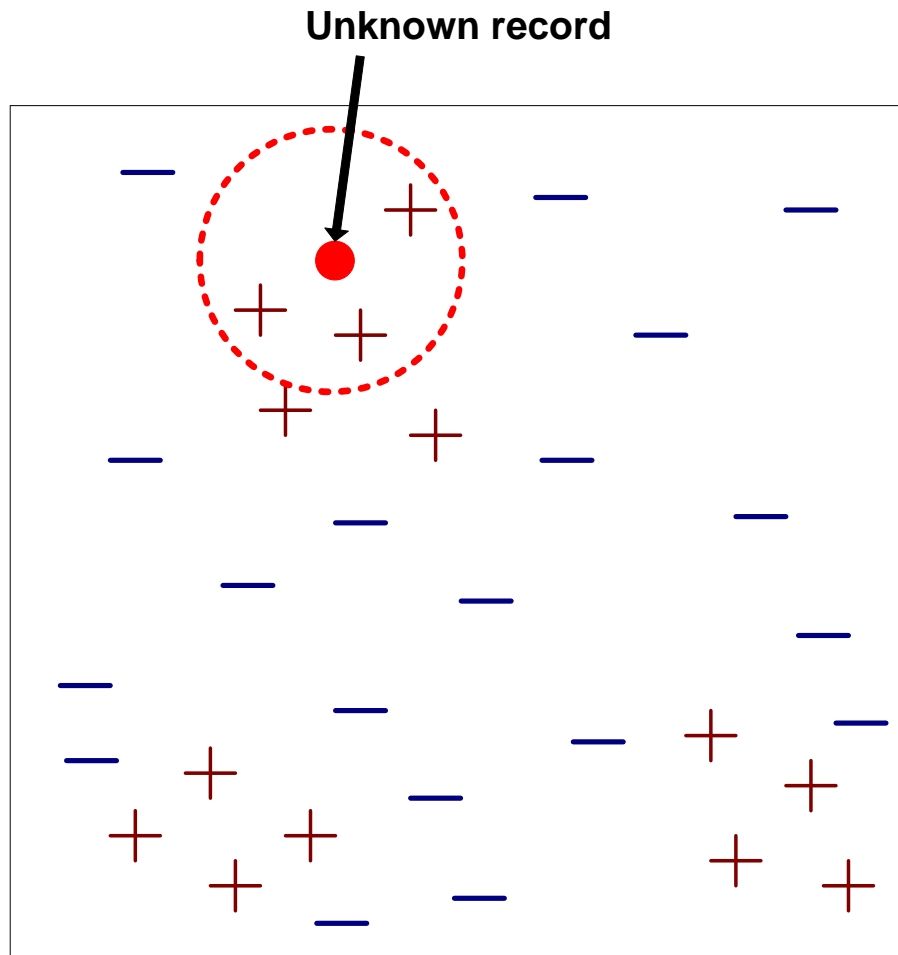
- Examples:
  - Rote-learner
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - Nearest neighbor classifier
    - Uses  $k$  “closest” points (nearest neighbors) for performing classification

# Nearest Neighbor Classifiers

- Basic idea:
  - *“If it walks like a duck, quacks like a duck, then it’s probably a duck”*



# Nearest-Neighbor Classifiers



- Requires three things
  - The set of **stored records**
  - **Distance Metric** to compute distance between records
  - The value of  **$k$** , the **number of nearest neighbors** to retrieve
- To classify an unknown record:
  1. **Compute distance** to other training records
  2. Identify  **$k$  nearest neighbors**
  3. Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking **majority vote**)

# Nearest Neighbor Classification

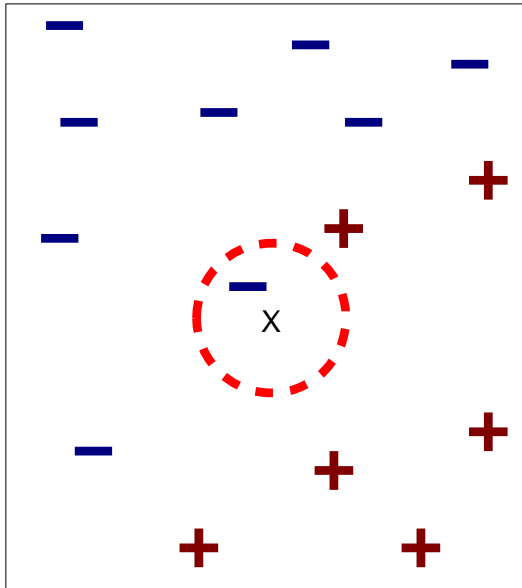
- Compute distance between two points:
  - Euclidean distance

$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

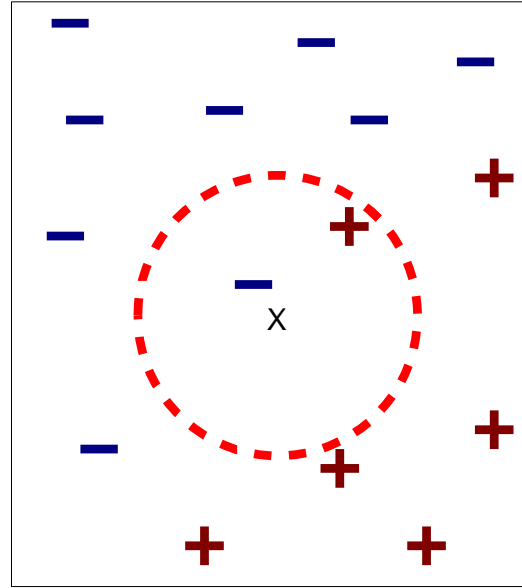
- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor,  $w = 1/d^2$



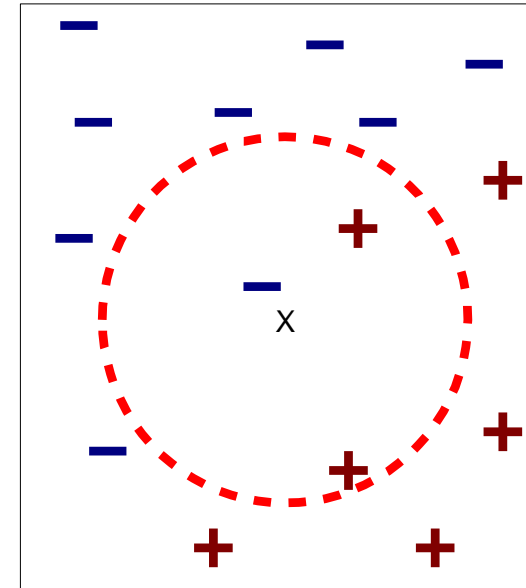
# Definition of Nearest Neighbor



(a) 1-nearest neighbor



(b) 2-nearest neighbor

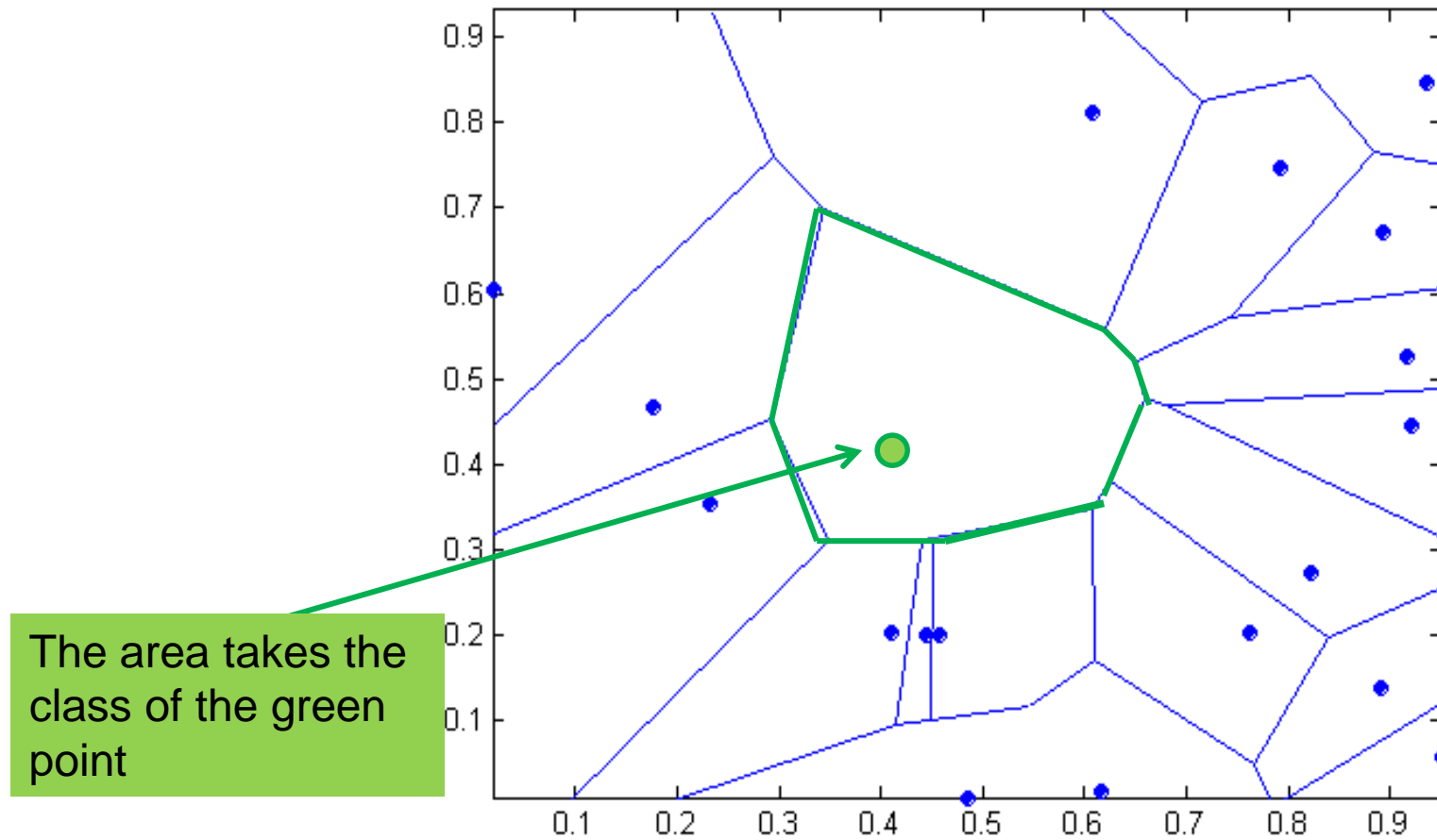


(c) 3-nearest neighbor

K-nearest neighbors of a record  $x$  are data points that have the  $k$  smallest distance to  $x$

# 1 nearest-neighbor

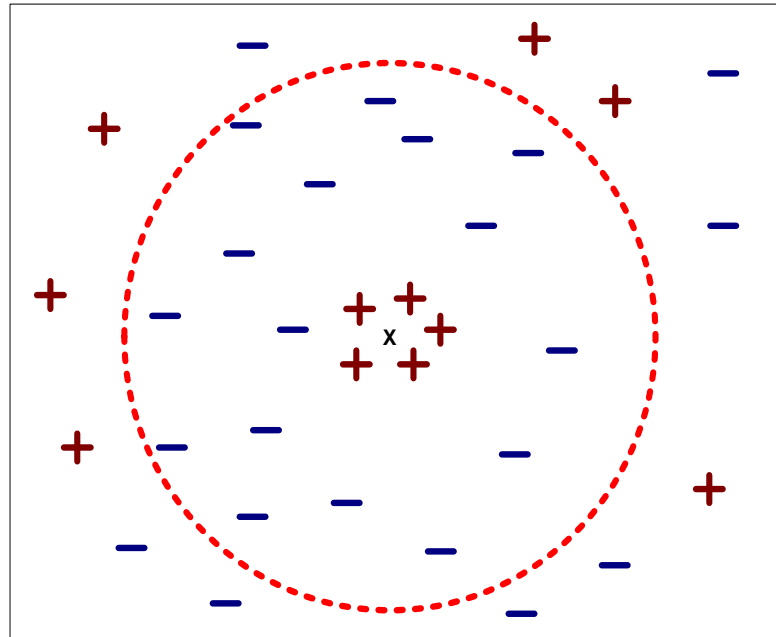
Voronoi Diagram defines the classification boundary



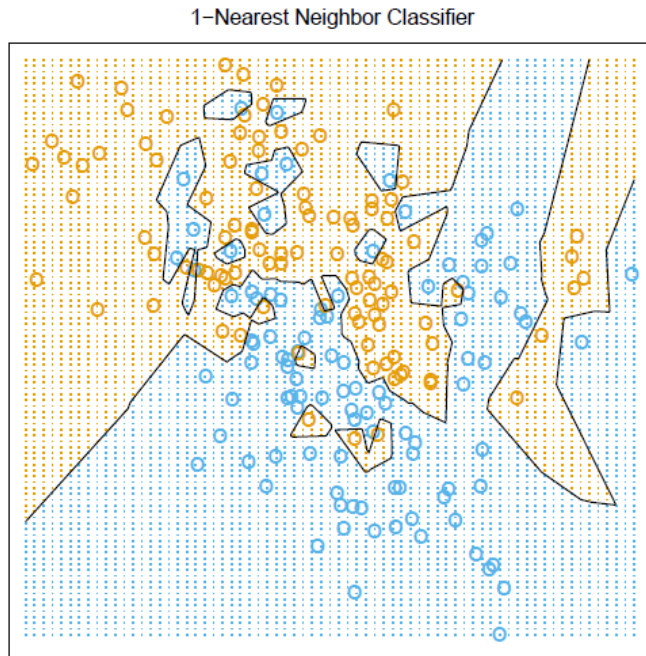
# Nearest Neighbor Classification...

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes

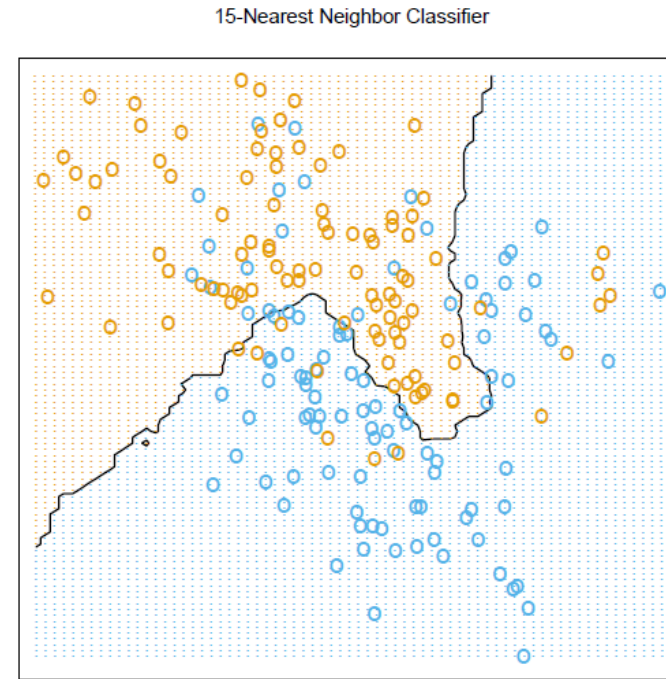
The value of k is the **complexity** of the model



# Example

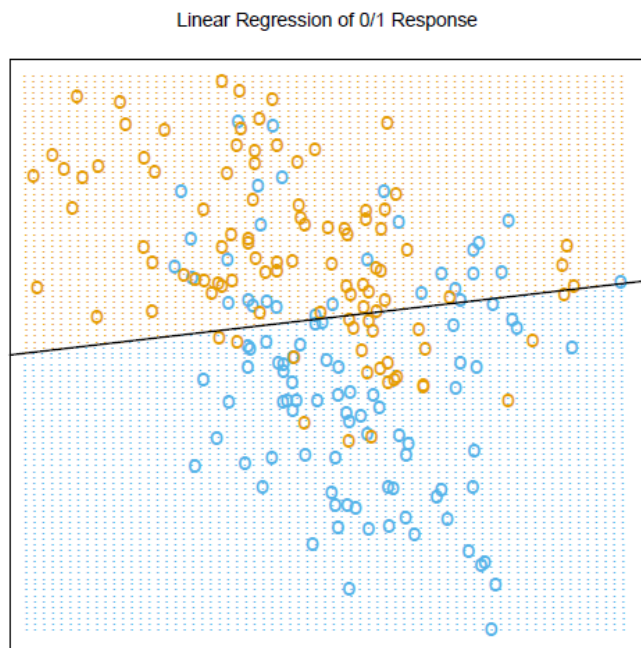


**FIGURE 2.3.** The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1), and then predicted by 1-nearest-neighbor classification.

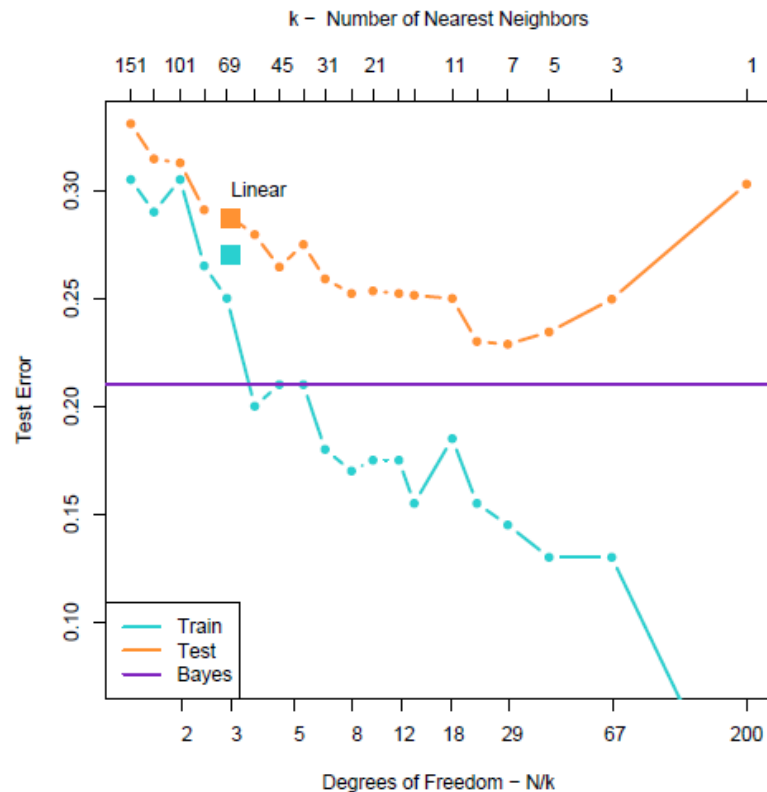


**FIGURE 2.2.** The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1) and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.

# Example



**FIGURE 2.1.** A classification example in two dimensions. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1), and then fit by linear regression. The line is the decision boundary defined by  $x^T \hat{\beta} = 0.5$ . The orange shaded region denotes that part of input space classified as ORANGE, while the blue region is classified as BLUE.



**FIGURE 2.4.** Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size 200 was used, and a test sample of size 10,000. The orange curves are test and the blue are training error for  $k$ -nearest-neighbor classification. The results for linear regression are the bigger orange and blue squares at three degrees of freedom. The purple line is the optimal Bayes error rate.

# Nearest Neighbor Classification...

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb
    - income of a person may vary from \$10K to \$1M

# Nearest Neighbor Classification...

- Problem with Euclidean measure:
  - High dimensional data
    - **curse of dimensionality**
  - Can produce counter-intuitive results

1 1 1 1 1 1 1 1 1 1 0

0 1 1 1 1 1 1 1 1 1 1

$d = 1.4142$

vs

1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1

$d = 1.4142$

- ◆ Solution: Normalize the vectors to unit length

# Nearest neighbor Classification...

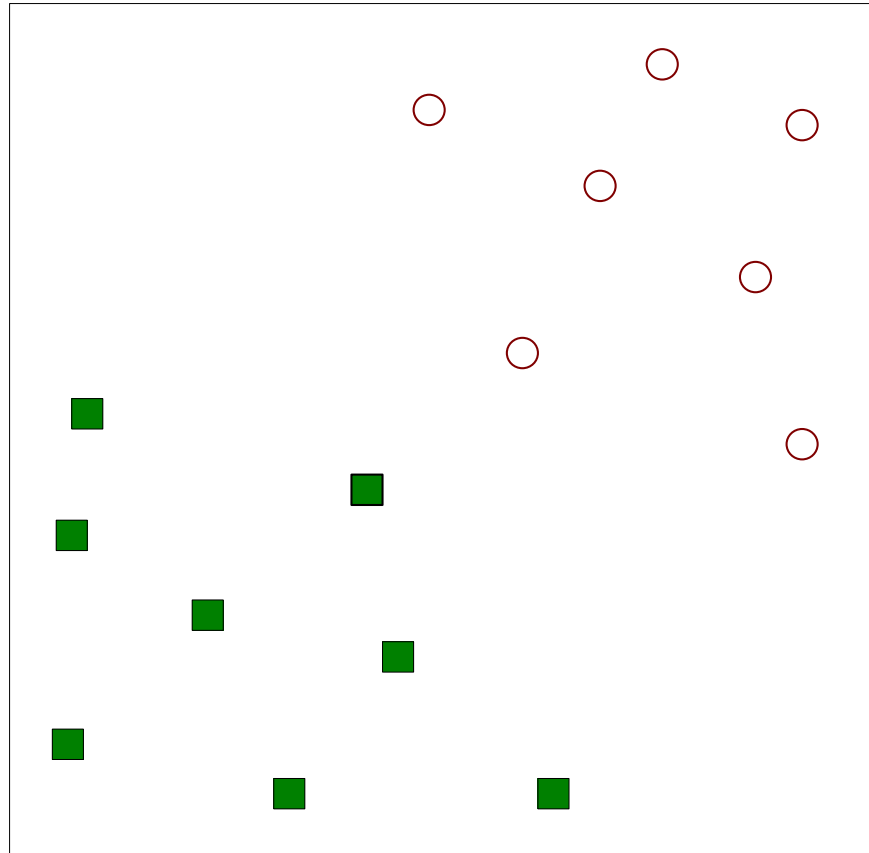
- k-NN classifiers are **lazy learners**
  - It does not build models explicitly
  - Unlike **eager learners** such as decision trees
- Classifying unknown records are relatively expensive
  - Naïve algorithm:  $O(n)$
  - Need for **structures** to retrieve nearest neighbors fast.
    - The **Nearest Neighbor Search** problem.
    - Also, **Approximate Nearest Neighbor Search**



# SUPPORT VECTOR MACHINES

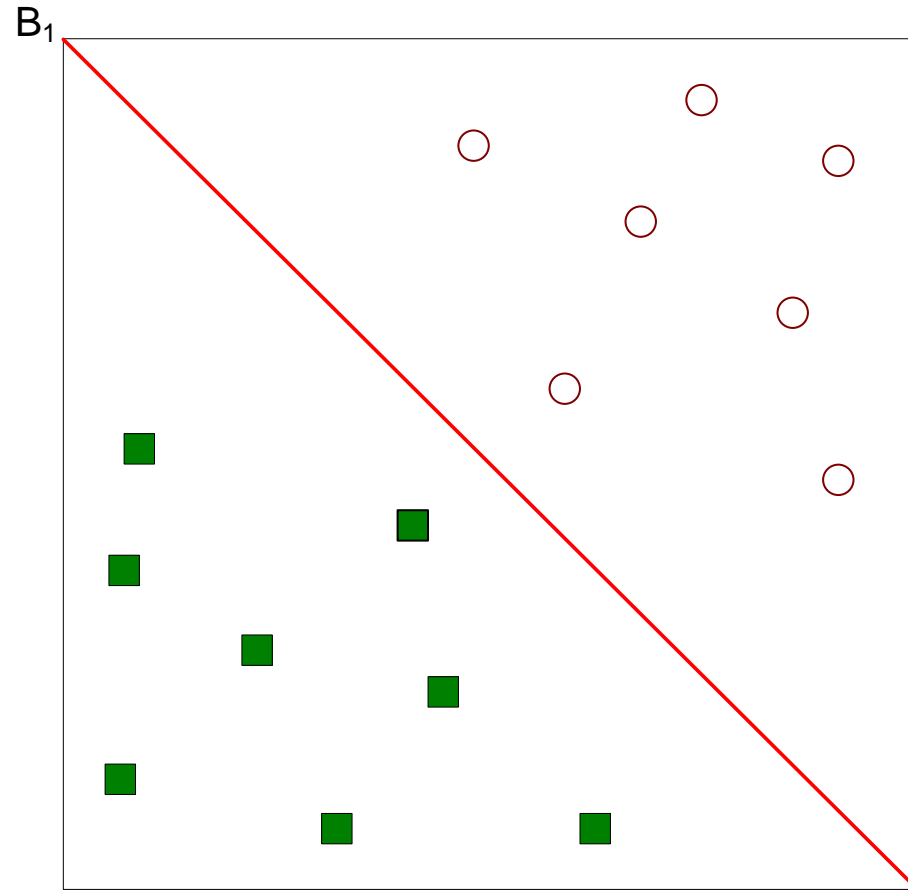
---

# Support Vector Machines



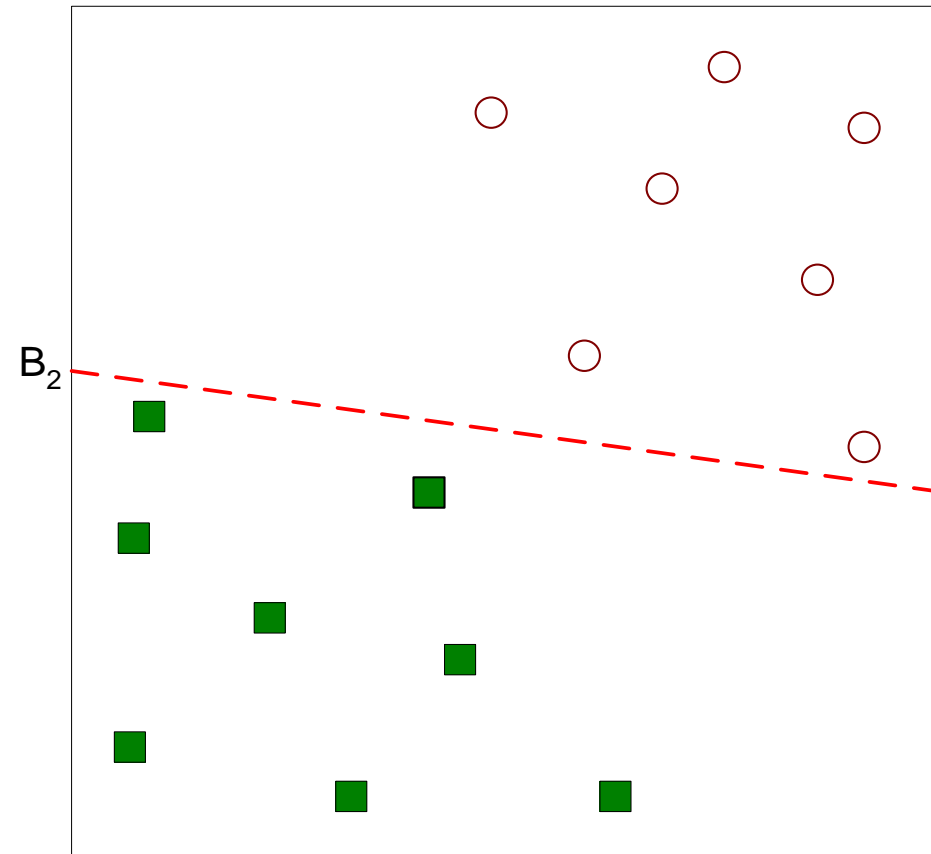
- Find a linear hyperplane (decision boundary) that will separate the data

# Support Vector Machines



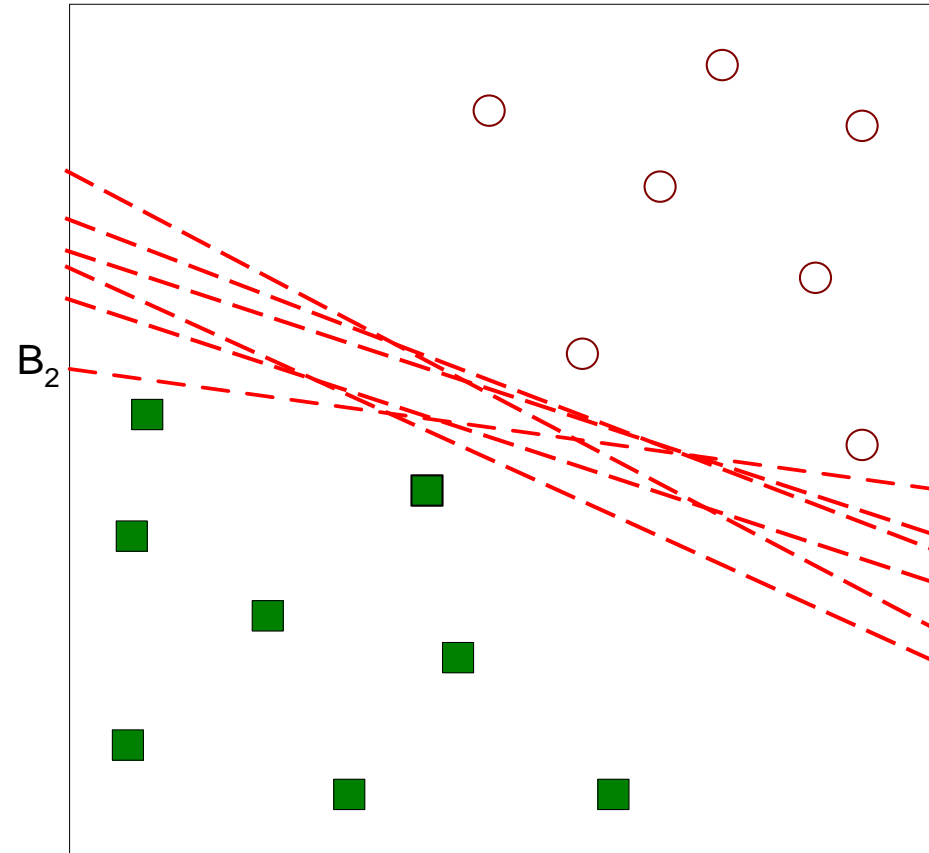
- One Possible Solution

# Support Vector Machines



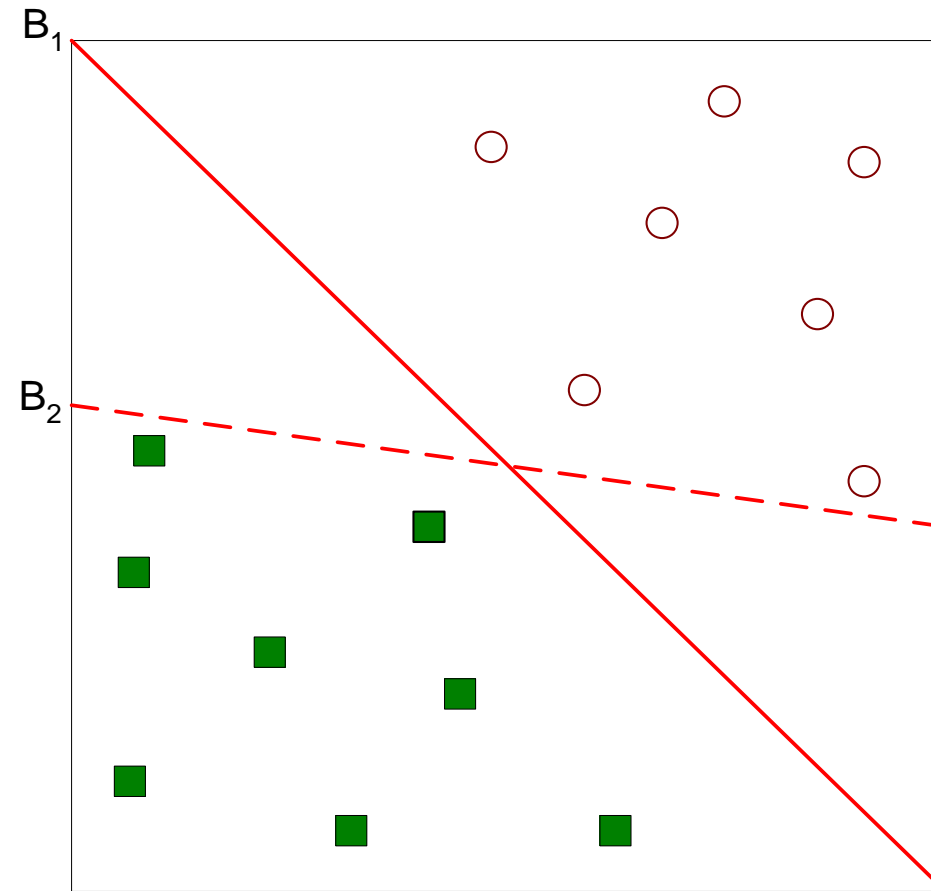
- Another possible solution

# Support Vector Machines



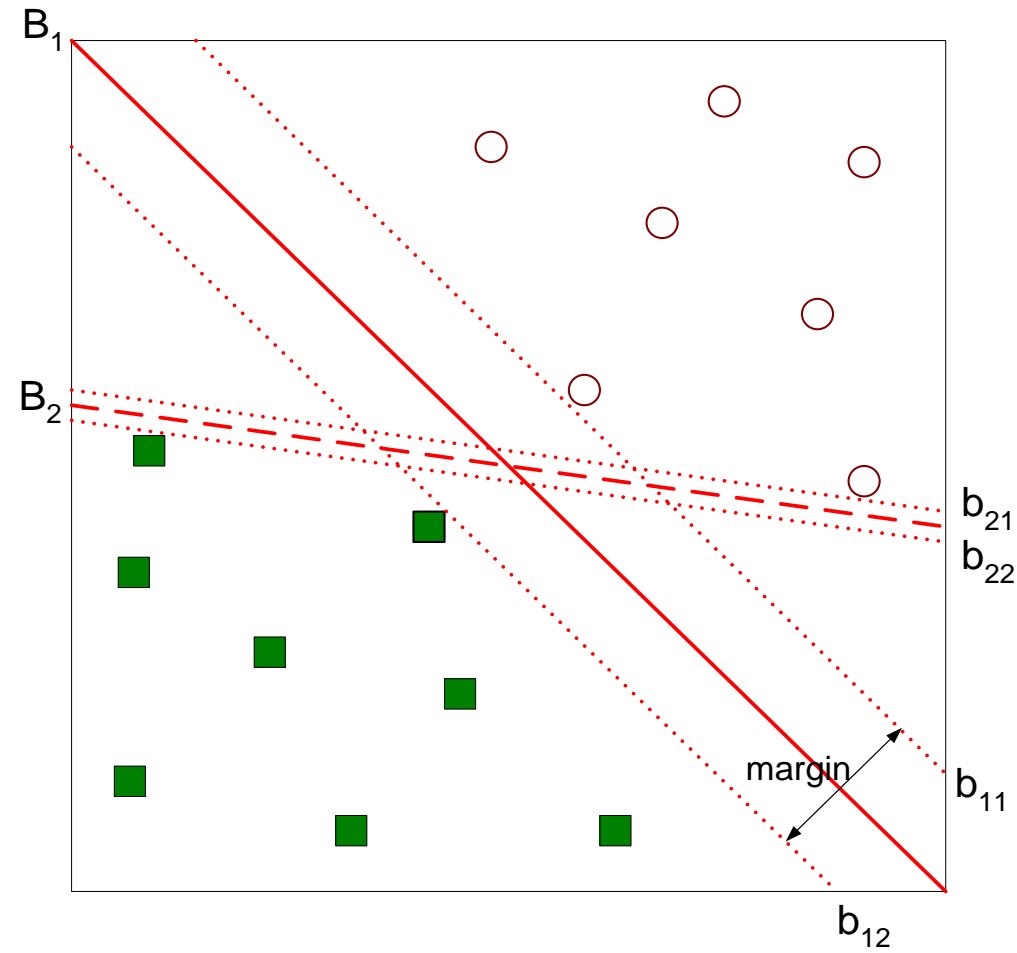
- Other possible solutions

# Support Vector Machines



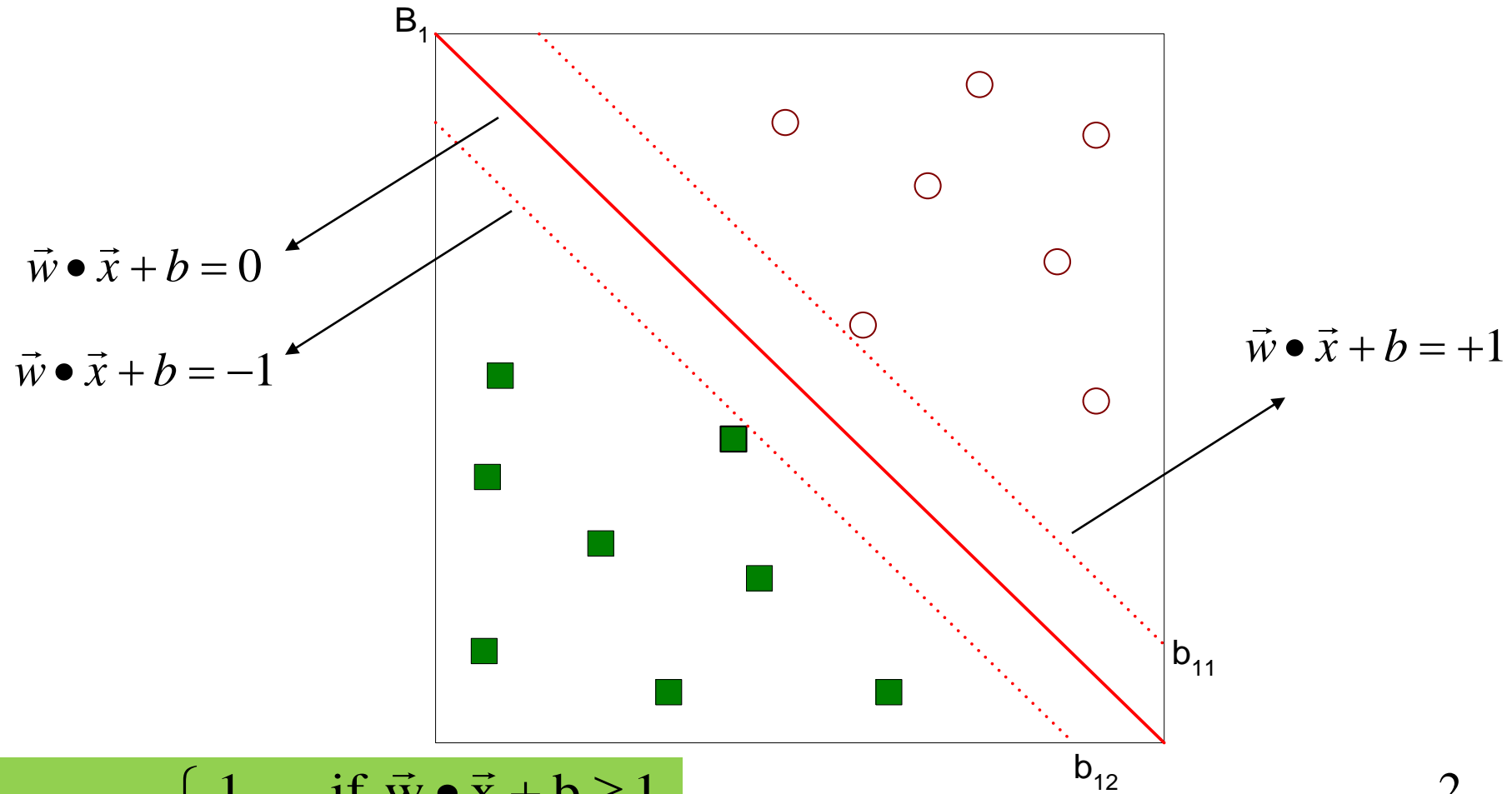
- Which one is better?  $B_1$  or  $B_2$ ?
- How do you define better?

# Support Vector Machines



- Find hyperplane **maximizes** the margin =>  $B_1$  is better than  $B_2$

# Support Vector Machines



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\|\vec{w}\|}$$

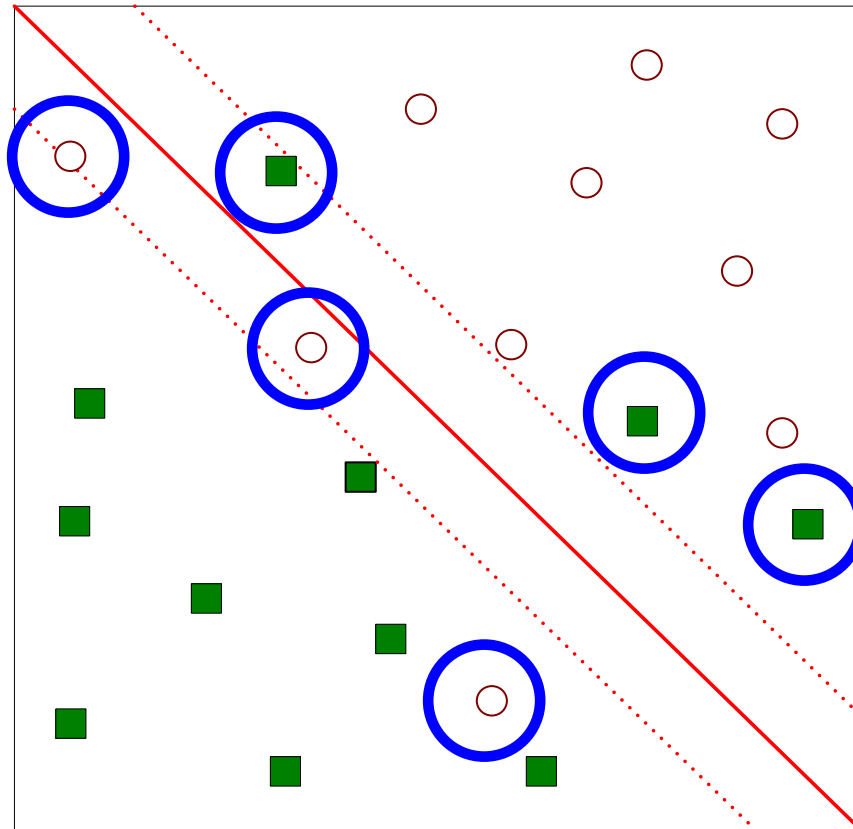


# Support Vector Machines

- We want to **maximize**:  $Margin = \frac{2}{\|\vec{w}\|}$
- Which is equivalent to **minimizing**:  $L(\vec{w}) = \frac{\|\vec{w}\|}{2}$
- But subjected to the following **constraints**:
$$\vec{w} \cdot \vec{x}_i + b \geq 1 \text{ if } y_i = 1$$
$$\vec{w} \cdot \vec{x}_i + b \leq -1 \text{ if } y_i = -1$$
- This is a **constrained optimization problem**
  - Numerical approaches to solve it (e.g., **quadratic programming**)

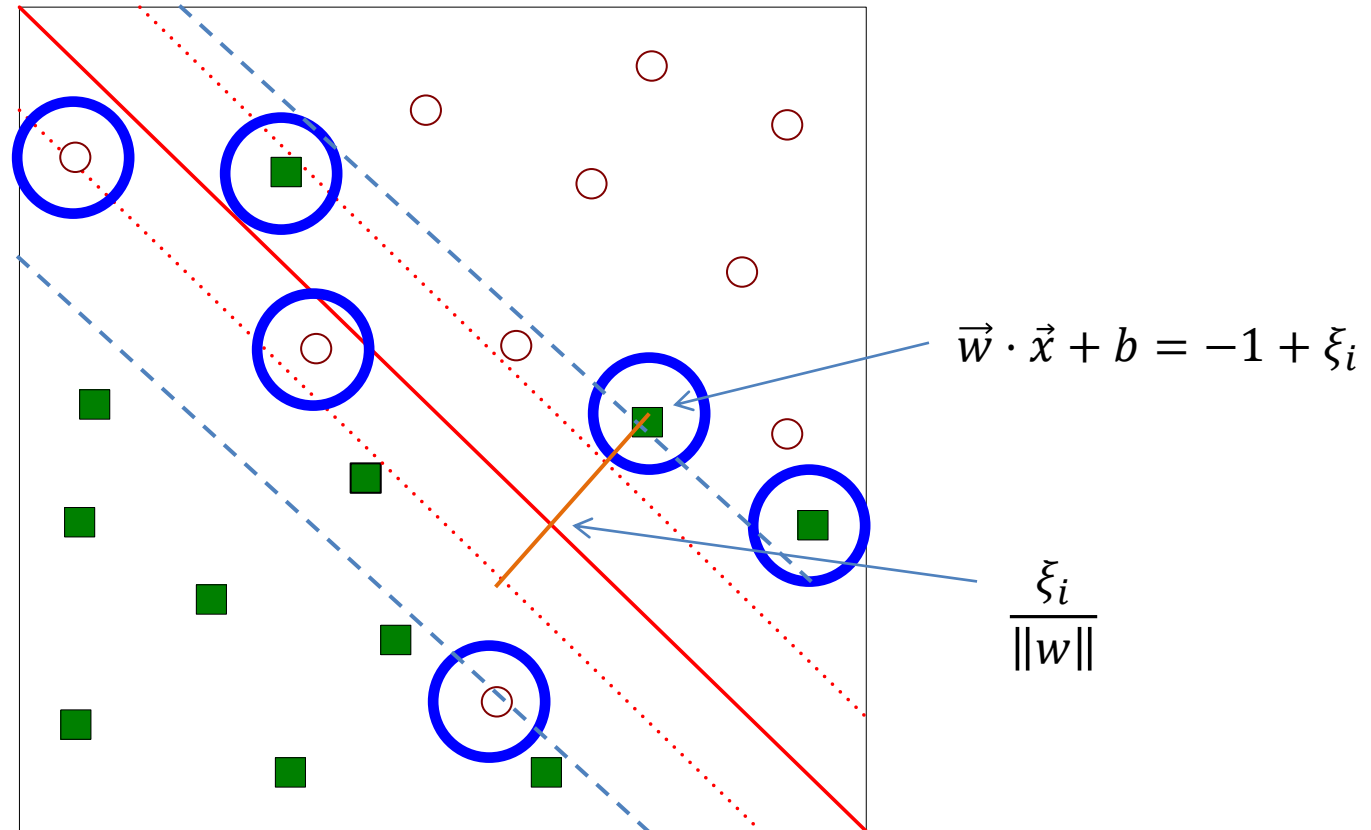
# Support Vector Machines

- What if the problem is **not linearly separable**?



# Support Vector Machines

- What if the problem is not linearly separable?



# Support Vector Machines

- What if the problem is not linearly separable?
- Introduce **slack variables**
  - Minimize:

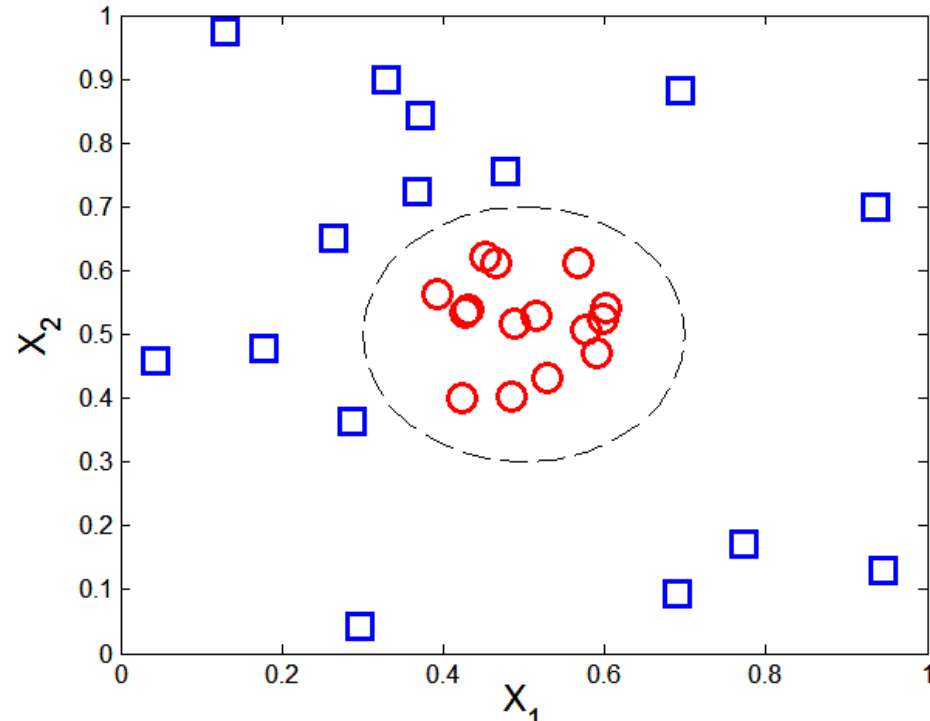
$$L(w) = \frac{\|\vec{w}\|}{2} + C \left( \sum_{i=1}^N \xi_i^k \right)$$

- Subject to:

$$\begin{aligned} \vec{w} \cdot \vec{x}_i + b &\geq 1 - \xi_i \text{ if } y_i = 1 \\ \vec{w} \cdot \vec{x}_i + b &\leq -1 + \xi_i \text{ if } y_i = -1 \end{aligned}$$

# Nonlinear Support Vector Machines

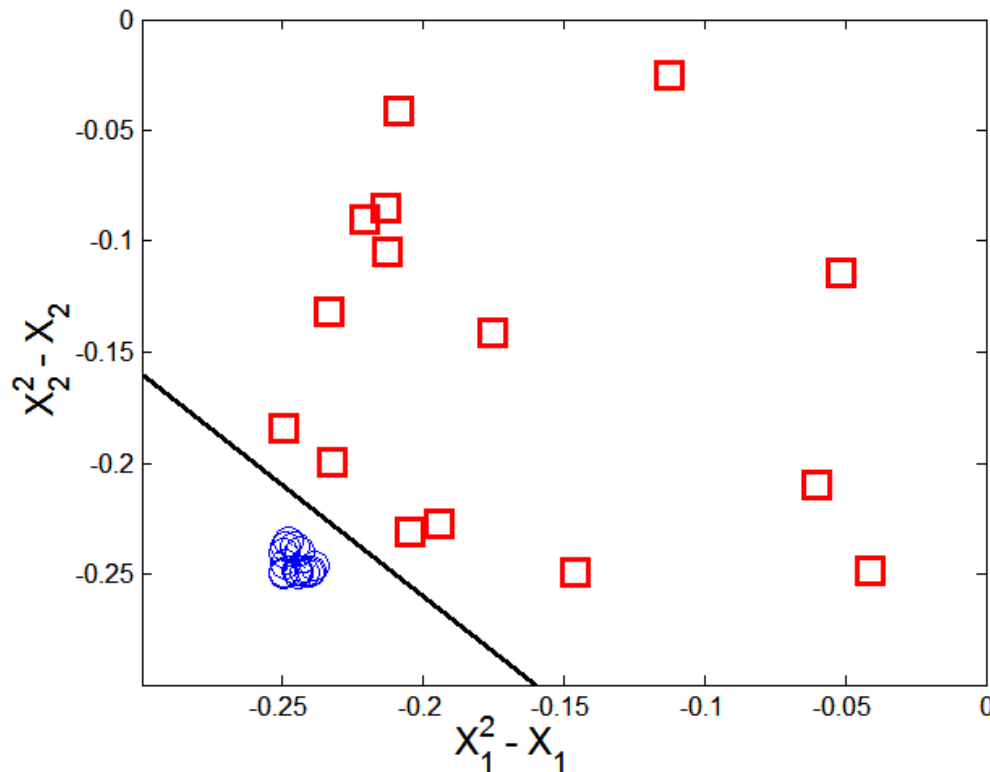
- What if decision boundary is not linear?



$$y(x_1, x_2) = \begin{cases} 1 & \text{if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 0.2 \\ -1 & \text{otherwise} \end{cases}$$

# Nonlinear Support Vector Machines

- Trick: Transform data into higher dimensional space



$$x_1^2 - x_1 + x_2^2 - x_2 = -0.46.$$

$$\Phi : (x_1, x_2) \longrightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1).$$

$$w_4x_1^2 + w_3x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$

Decision boundary:

$$\vec{w} \cdot \Phi(\vec{x}) + b = 0$$

# Learning Nonlinear SVM

- Optimization problem:

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

*subject to*  $y_i(\mathbf{w} \cdot \Phi(\mathbf{x}_i) + b) \geq 1, \forall \{(\mathbf{x}_i, y_i)\}$

- Which leads to the same set of equations (but involve  $\Phi(x)$  instead of  $x$ )

$$L_D = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \quad \mathbf{w} = \sum_i \lambda_i y_i \Phi(\mathbf{x}_i)$$
$$\lambda_i \{ y_i (\sum_j \lambda_j y_j \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i) + b) - 1 \} = 0,$$

$$f(\mathbf{z}) = \text{sign}(\mathbf{w} \cdot \Phi(\mathbf{z}) + b) = \text{sign}\left(\sum_{i=1}^n \lambda_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{z}) + b\right).$$

# Learning NonLinear SVM

- Issues:
  - What type of mapping function  $\Phi$  should be used?
  - How to do the computation in high dimensional space?
    - Most computations involve dot product  $\Phi(x_i) \cdot \Phi(x_j)$
    - Curse of dimensionality?



# Learning Nonlinear SVM

- **Kernel Trick:**

- $\Phi(x_i) \cdot \Phi(x_j) = K(x_i, x_j)$

- $K(x_i, x_j)$  is a **kernel function** (expressed in terms of the coordinates in the original space)

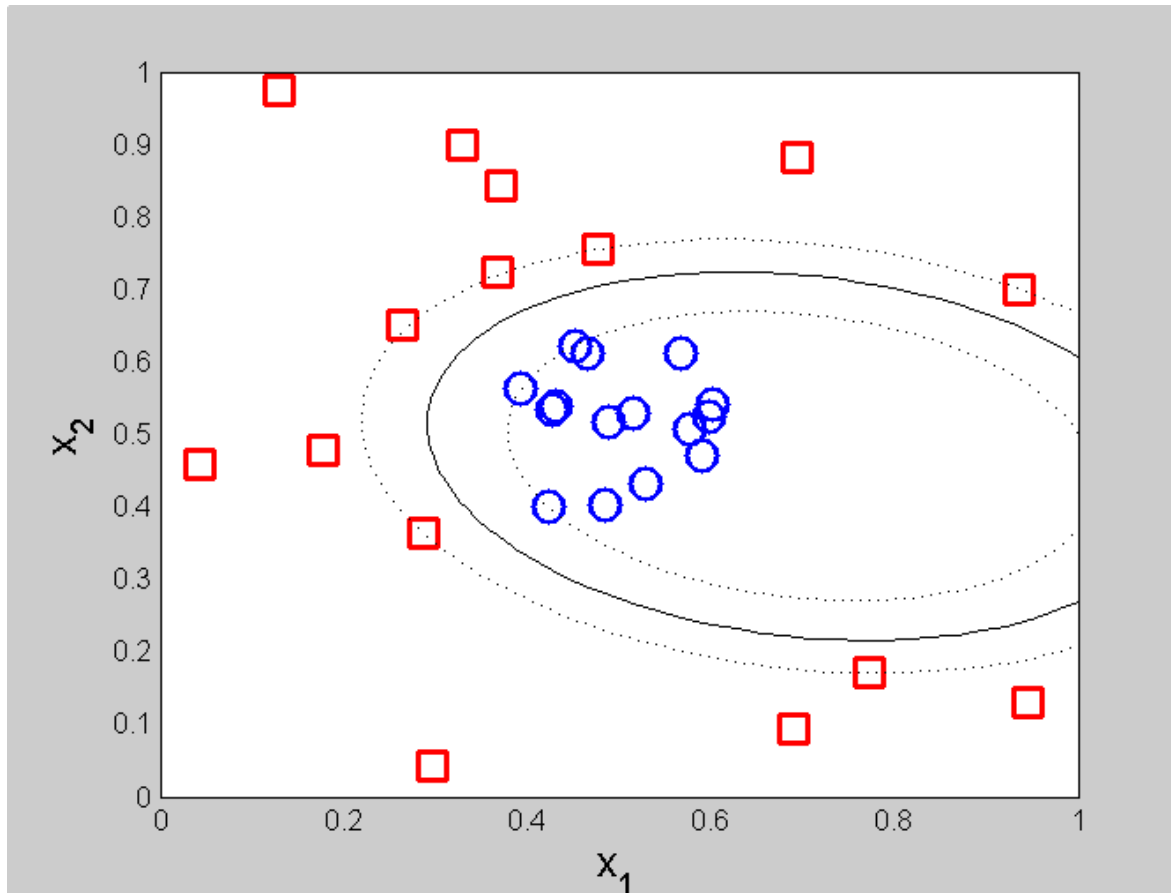
- Examples:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$$

$$K(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x} - \mathbf{y}\|^2 / (2\sigma^2)}$$

$$K(\mathbf{x}, \mathbf{y}) = \tanh(k\mathbf{x} \cdot \mathbf{y} - \delta)$$

# Example of Nonlinear SVM



SVM with polynomial  
degree 2 kernel

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^2$$

# Learning Nonlinear SVM

- Advantages of using kernel:
  - Don't have to know the mapping function  $\Phi$
  - Computing dot product  $\Phi(x_i) \cdot \Phi(x_j)$  in the original space avoids curse of dimensionality
- Not all functions can be kernels
  - Must make sure there is a corresponding  $\Phi$  in some high-dimensional space
  - Mercer's theorem (see textbook)

# LOGISTIC REGRESSION

---

# Classification via regression

- Instead of predicting the **class** of a record we want to **predict the probability of the class** given the record
- Transform the **classification** problem into a **regression** problem.
- But how do you define the probability that you want to predict?

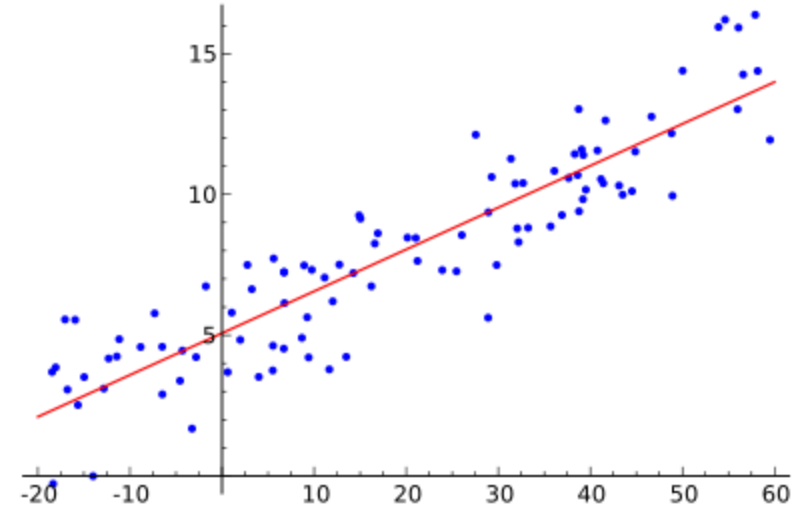
# Linear regression

- Given a dataset of the form  $\{(x_1, y_1), \dots, (x_n, y_n)\}$  find a linear function that given the vector  $x_i$  predicts the  $y_i$  value as  $y'_i = w^T x_i$ 
  - Find a vector of weights  $w$  that **minimizes the sum of square errors**

$$\sum_i (y'_i - y_i)^2$$

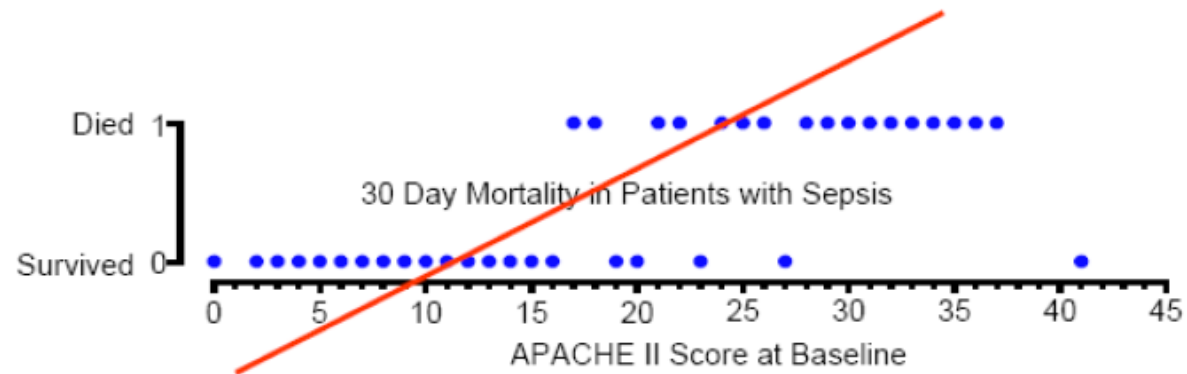
- Closed form solution:

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



# Linear regression

- A simple approach: use linear regression to learn a linear function that predicts 0/1 values
  - Not good: It may produce negative probabilities, or probabilities that are greater than 1.



# Class probabilities

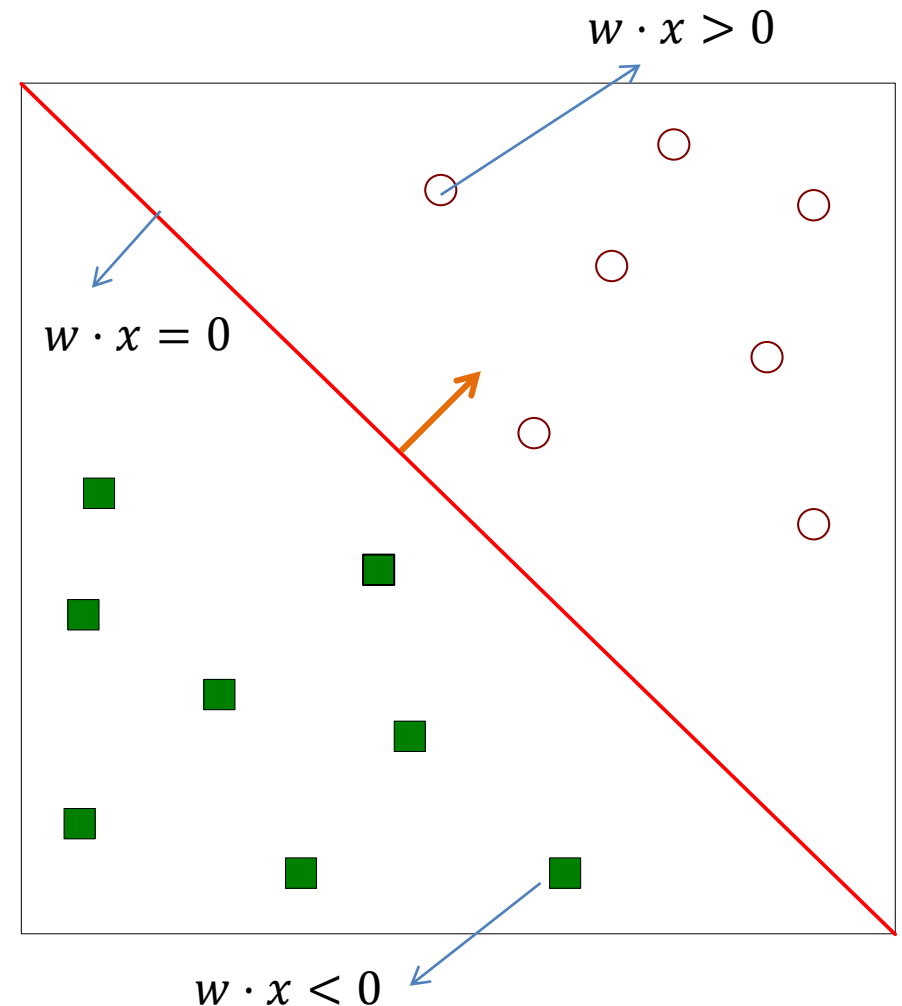
- Assume a **linear classification boundary**

For the positive class the **bigger** the **value** of  $w \cdot x$ , the further the point is from the classification boundary, the higher our **certainty** for the membership to the **positive class**

- Define  $P(C_+|x)$  as an **increasing** function of  $w \cdot x$

For the negative class the **smaller** the **value** of  $w \cdot x$ , the further the point is from the classification boundary, the higher our **certainty** for the membership to the **negative class**

- Define  $P(C_-|x)$  as a **decreasing** function of  $w \cdot x$





# Logistic Regression

The **logistic function**

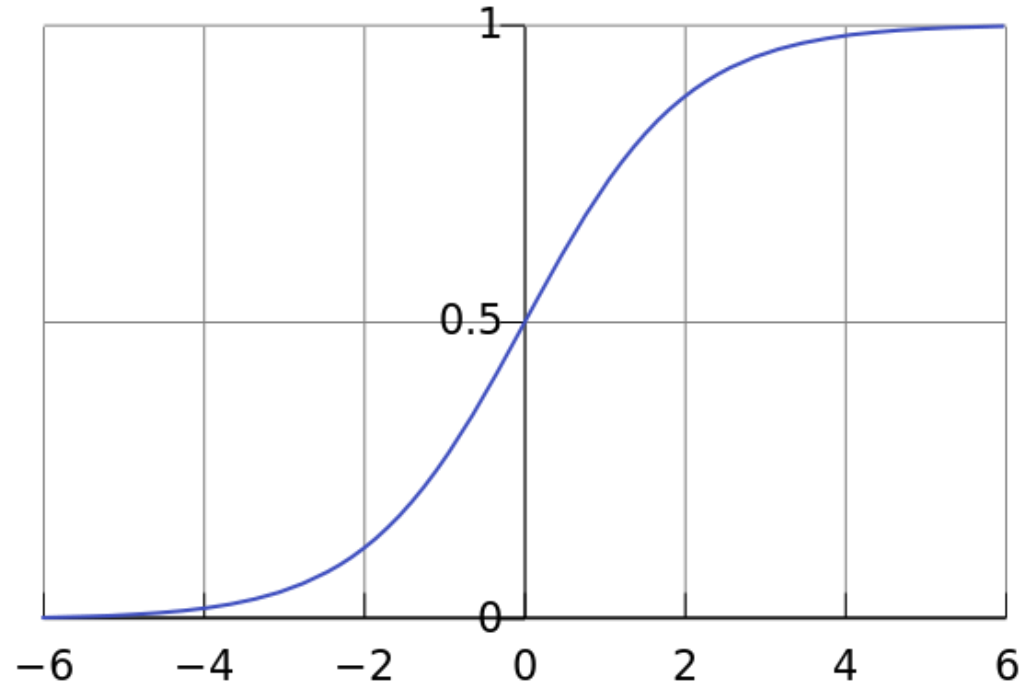
$$f(t) = \frac{1}{1 + e^{-t}}$$

$$P(C_+|x) = \frac{1}{1 + e^{-w \cdot x - a}}$$

$$P(C_-|x) = \frac{e^{-w \cdot x - a}}{1 + e^{-w \cdot x - a}}$$

$$\log \frac{P(C_+|x)}{P(C_-|x)} = w \cdot x + a$$

Linear regression on the **log-odds ratio**



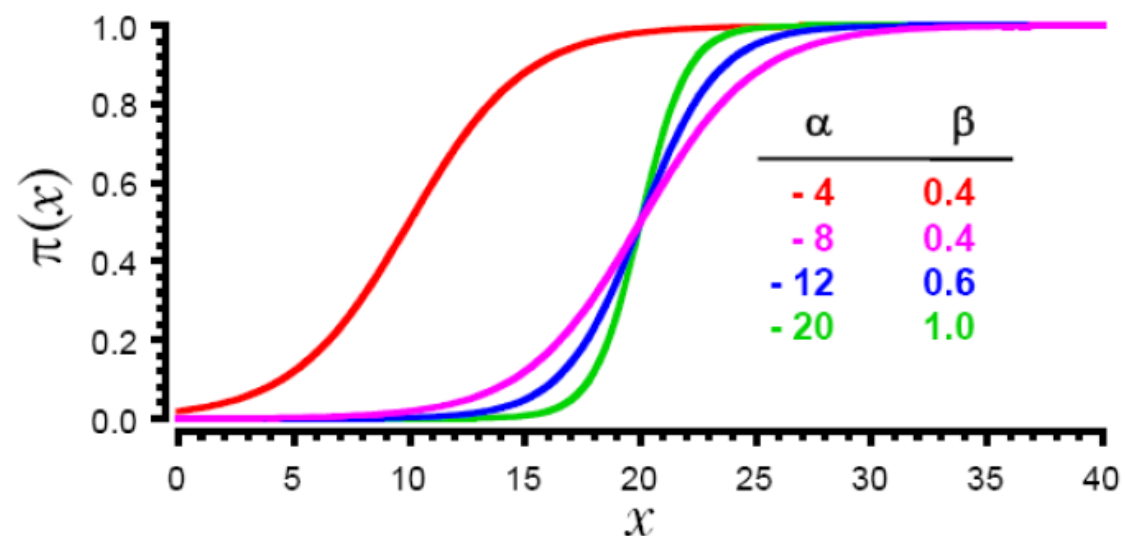
**Logistic Regression:** Find the vector  $w$  that **maximizes the probability** of the observed data

# The logistic function

$\beta$  controls the slope

$a$  controls the position of the turning point

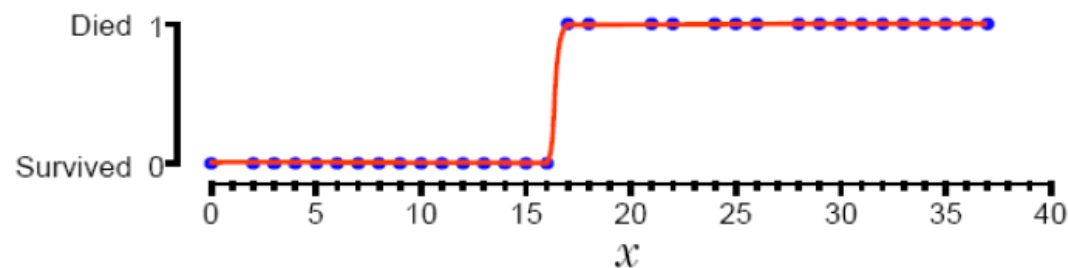
$$\pi(x) = \exp(\alpha + \beta x) / (1 + \exp(\alpha + \beta x))$$



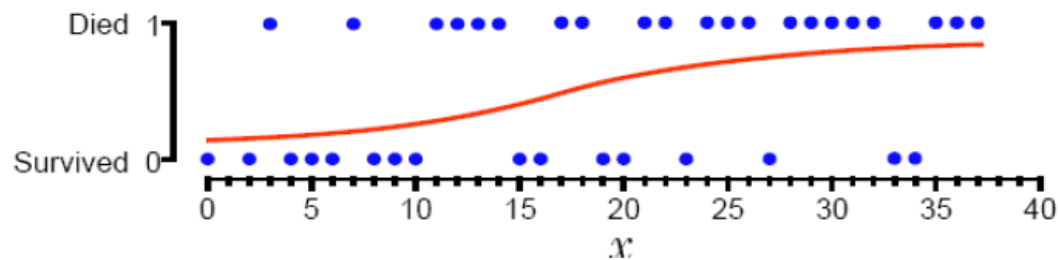
When  $x = -\alpha / \beta$ ,  $\alpha + \beta x = 0$  and hence  $\pi(x) = 1 / (1 + 1) = 0.5$

# Logistic Regression in one dimension

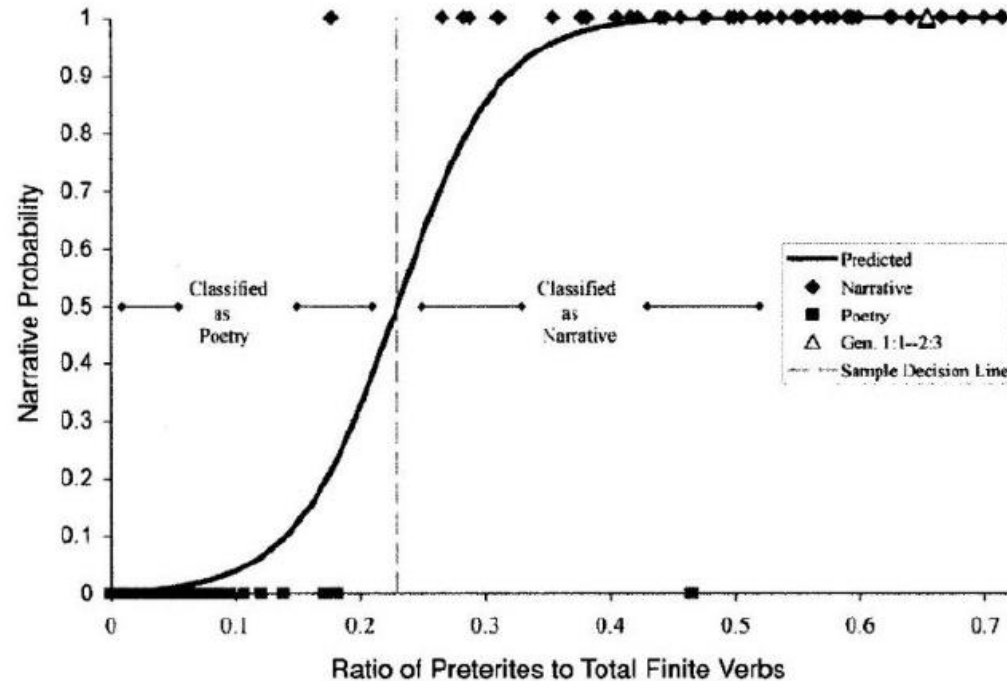
Data that has a sharp survival cut off point between patients who live or die should have a large value of  $\beta$ .



Data with a lengthy transition from survival to death should have a low value of  $\beta$ .



# Logistic Regression in one dimension



**Figure 10-3.** The solid curved line is called a logistic regression curve. The vertical axis measures the probability that an Old Testament passage is narrative, based on the use of preterite verbs. The probability is zero for poetry and unity or one for narrative. Passages with high preterite verb counts, falling to the right of the vertical dotted line, are likely narrative. The triangle on the upper right represents Genesis 1:1-2:3, which is clearly literal, narrative history.

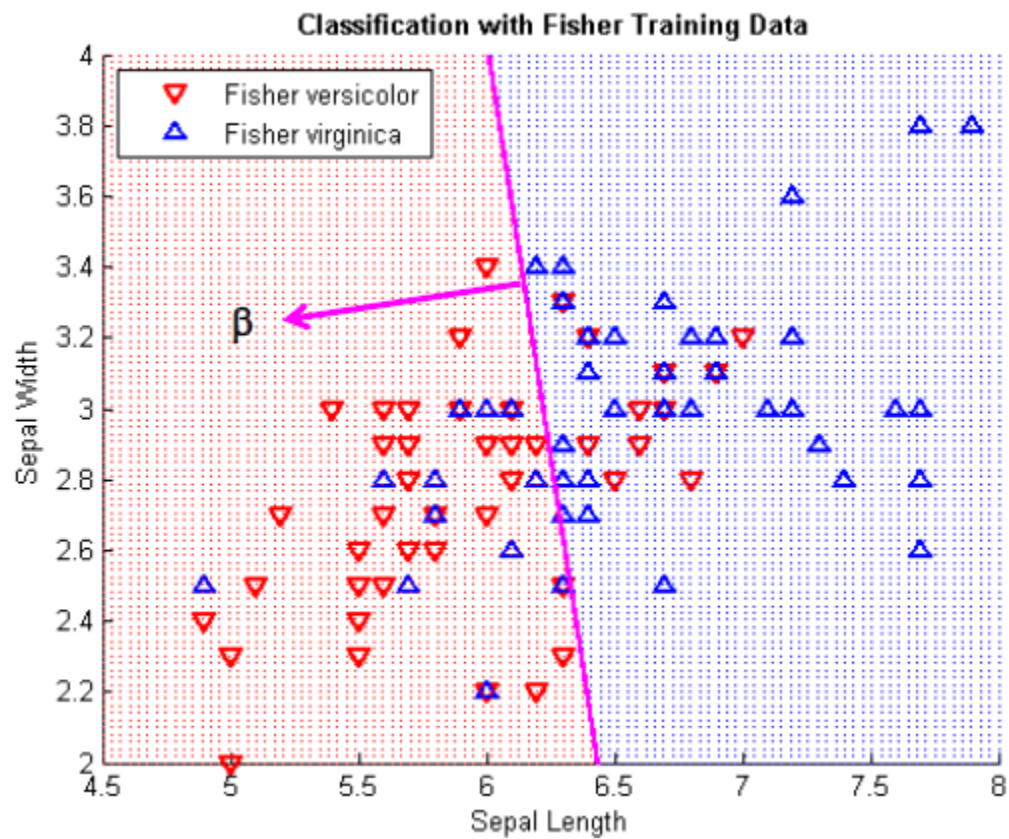
# Logistic regression in 2-d

Coefficients

$$\beta_1 = -1.9$$

$$\beta_2 = -0.4$$

$$\alpha = 13.04$$



# Estimating the coefficients

- **Maximum Likelihood Estimation:**
  - We have pairs of the form  $(x_i, y_i)$
- **Log Likelihood function**

$$L(w) = \sum_i [y_i \log P(y_i|x_i, w) + (1 - y_i) \log(1 - P(y_i|x_i, w))]$$

- Unfortunately, it does not have a closed form solution
  - Use **gradient descend** to find local minimum

# Logistic Regression

- Produces a **probability estimate** for the **class membership** which is often very useful.
- The **weights** can be useful for understanding the **feature importance**.
- Works for relatively large datasets
- Fast to apply.

# INTRODUCTION TO NEURAL NETWORKS

---

(Thanks to Philipp Koehn for the material borrowed from his slides)

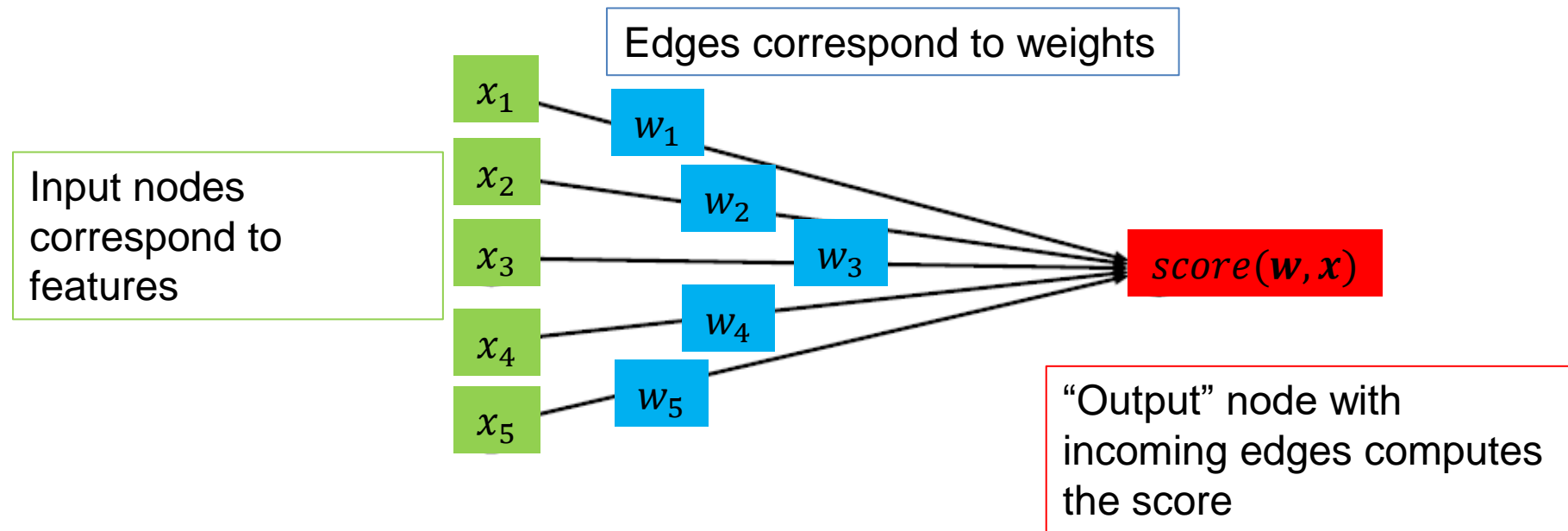


# Linear Classification

- A simple model for classification is to take a **linear combination** of the feature values and compute a score.
- Input: Feature vector  $\mathbf{x} = (x_1, \dots, x_n)$
- Model: Weights  $\mathbf{w} = (w_1, \dots, w_n)$
- Output:  $score(\mathbf{w}, \mathbf{x}) = \sum_i w_i x_i$
- Make a decision depending on the output score.
  - E.g.: Decide “Yes” if  $score(\mathbf{w}, \mathbf{x}) > 0$  and “No” if  $score(\mathbf{w}, \mathbf{x}) < 0$
- The **perceptron** classification algorithm

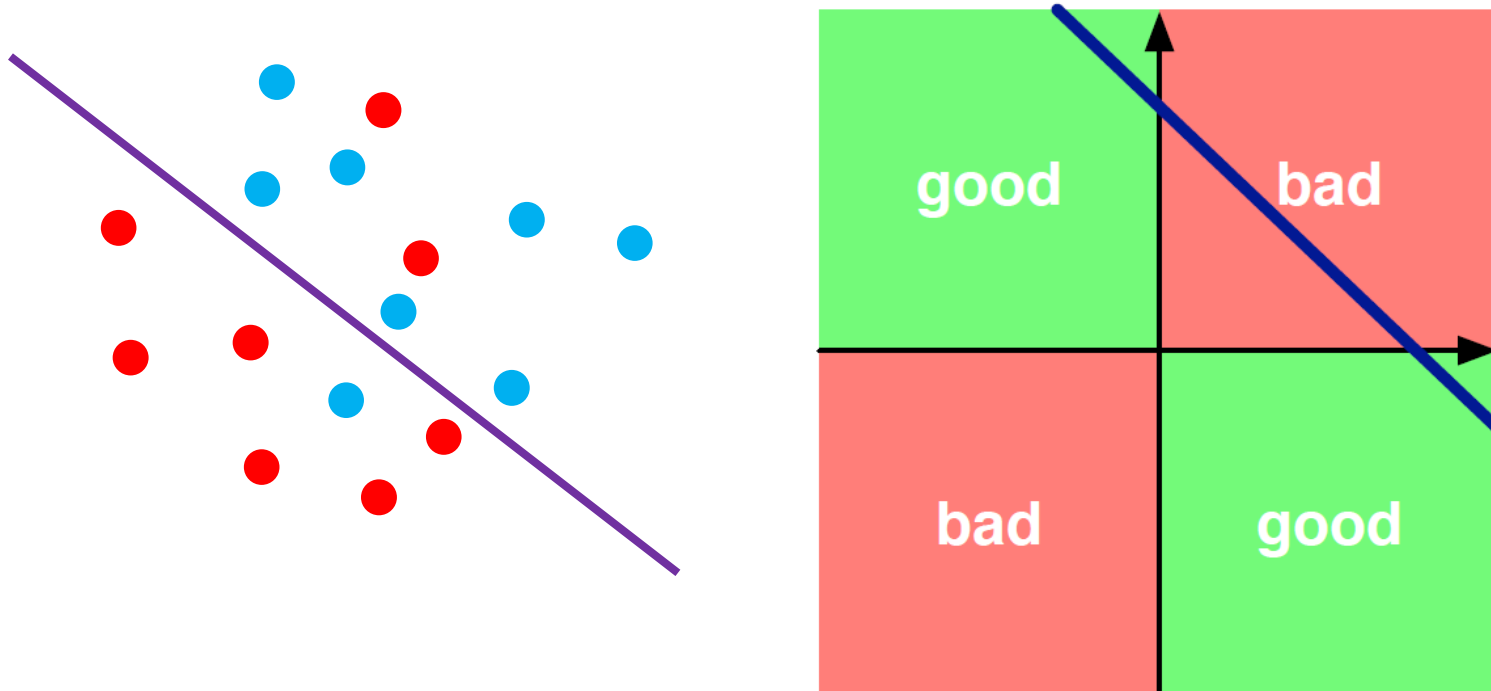
# Linear Classification

- We can represent this as a network



# Linear models

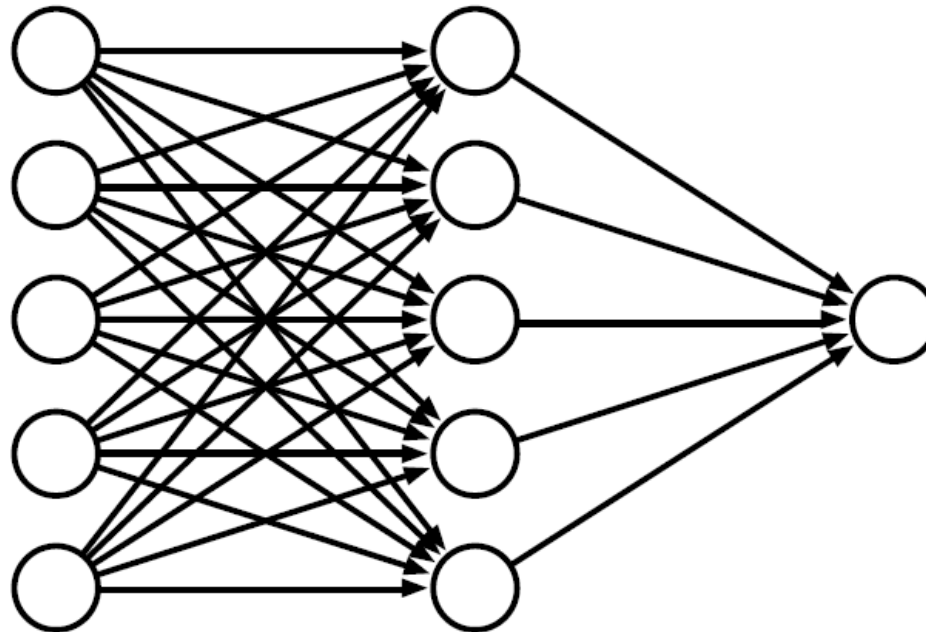
- Linear models partition the space according to a hyperplane



- But they cannot model everything

# Multiple layers

- We can add more **layers**:
  - Each arrow has a weight
  - Nodes compute scores from incoming edges and give input to outgoing edges



Did we gain anything?

# Non-linearity

- Instead of computing a linear combination

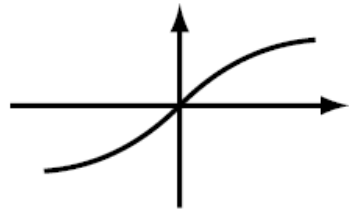
$$\text{score}(\mathbf{w}, \mathbf{x}) = \sum_i w_i x_i$$

- Apply a non-linear function on top:

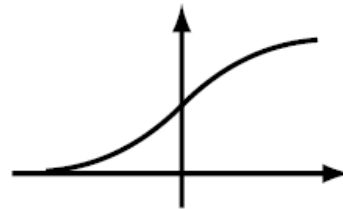
$$\text{score}(\mathbf{w}, \mathbf{x}) = g\left(\sum_i w_i x_i\right)$$

- Popular functions:

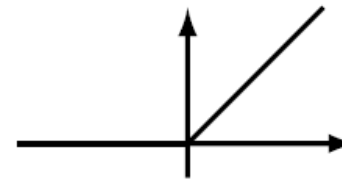
$\tanh(x)$



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



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

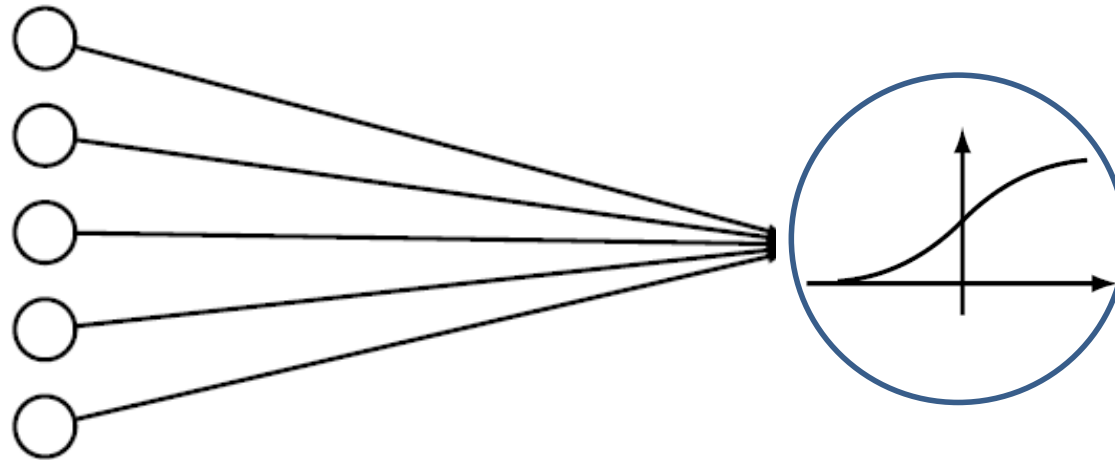


(sigmoid is also called the "logistic function")

These functions play the role of a soft "switch" (threshold function)

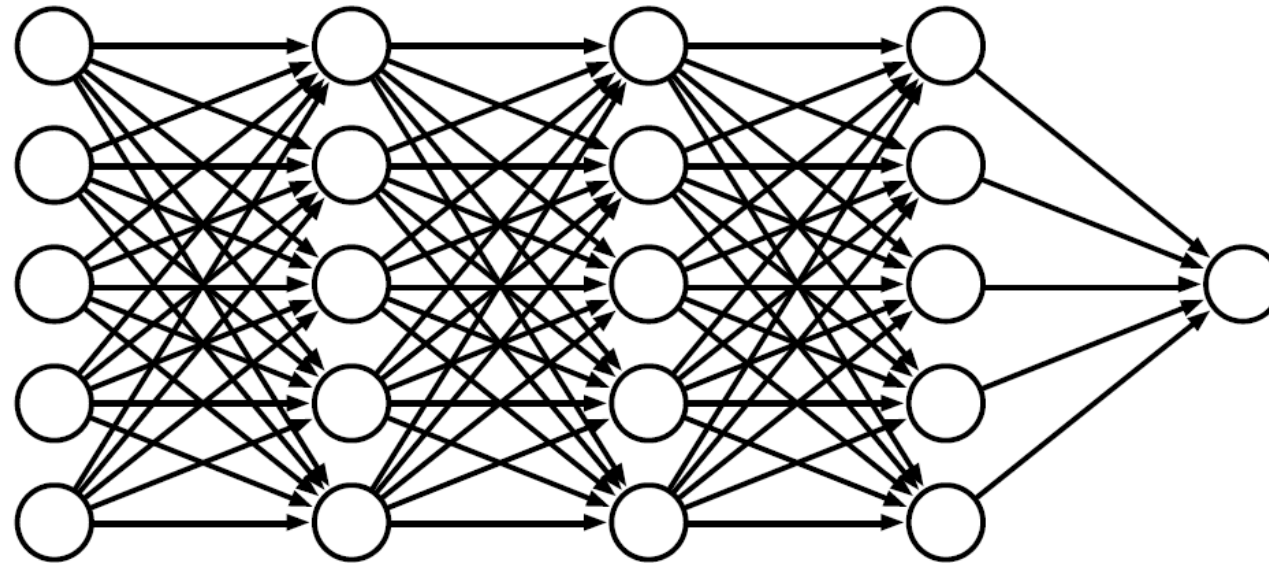
# Side note

- Logistic regression classifier:
  - Single layer with a logistic function



# Deep learning

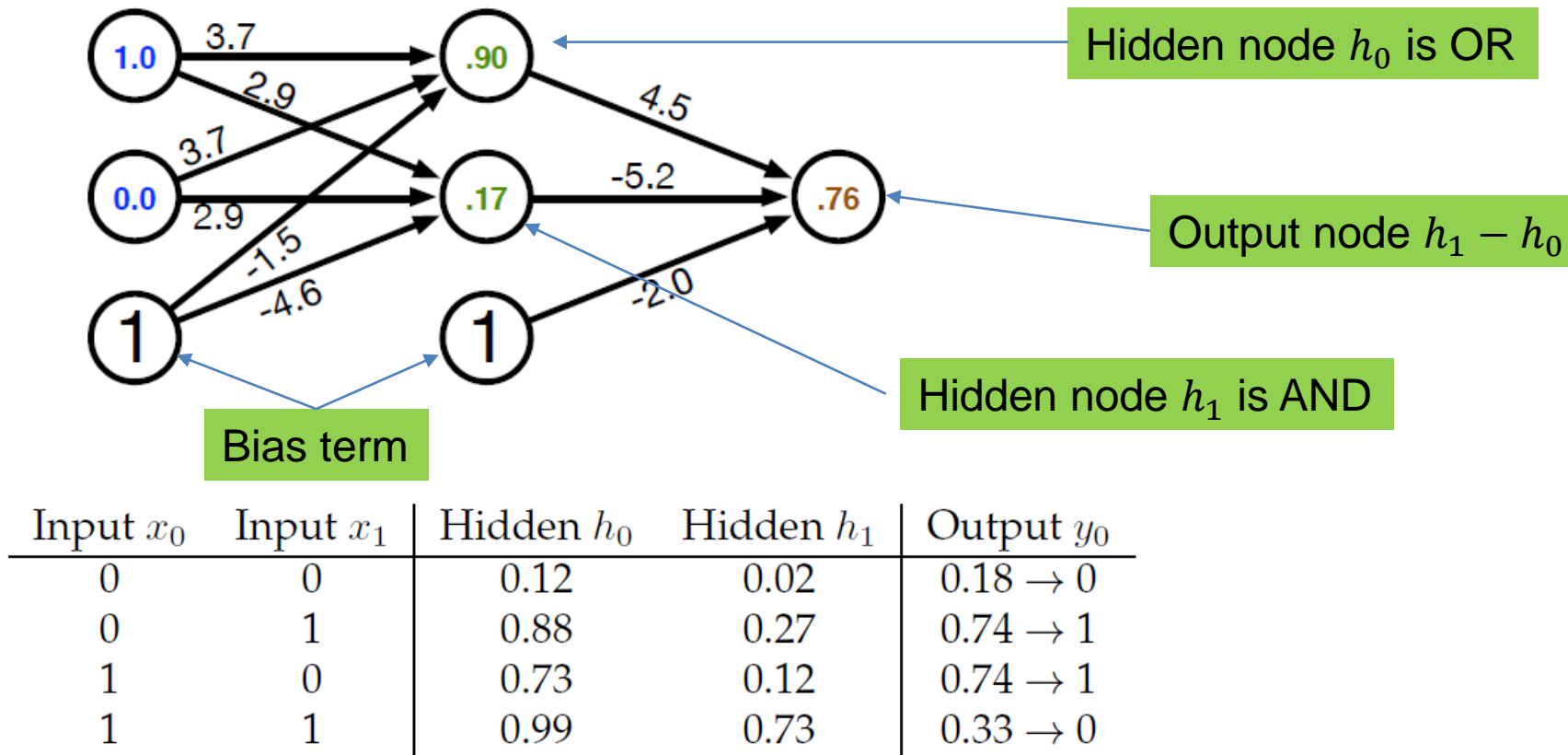
- Networks with **multiple layers**



- Each layer can be thought of as a processing step
- Multiple layers allow for the computation of more complex functions

# Example

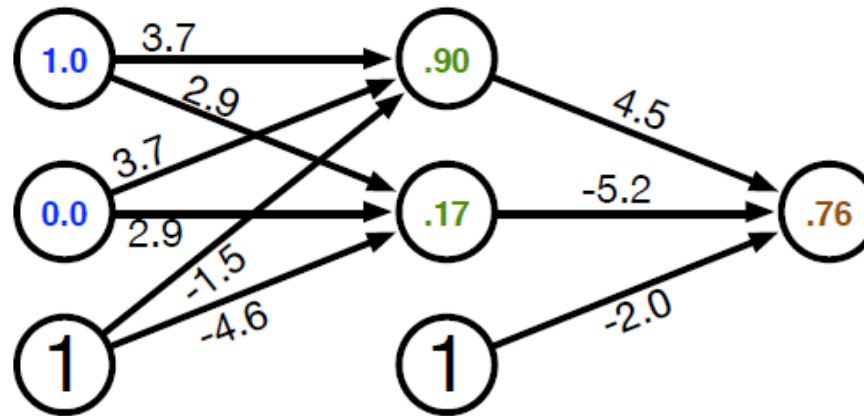
- A network that implements XOR





# Error

- The computed value is 0.76 but the correct value is 1
  - There is an **error** in the computation

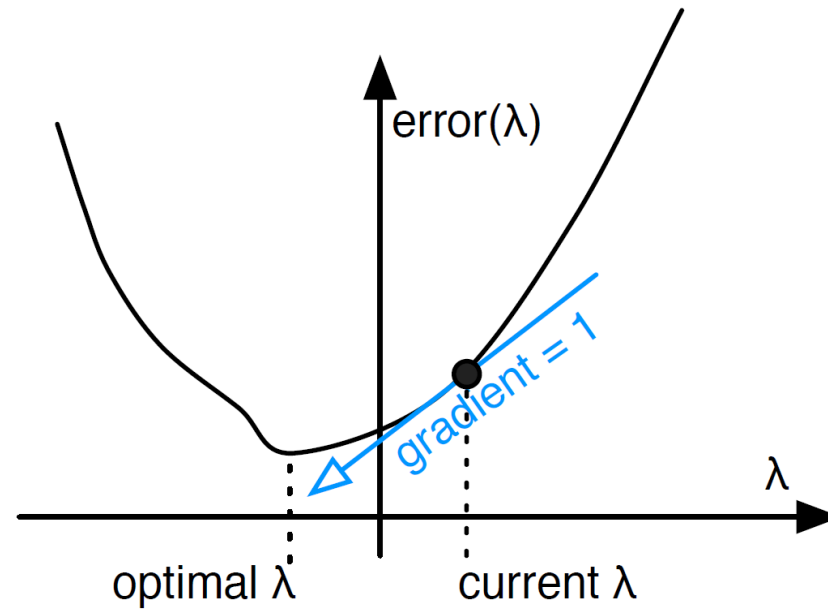


- How do we set the weights so as to minimize this error?

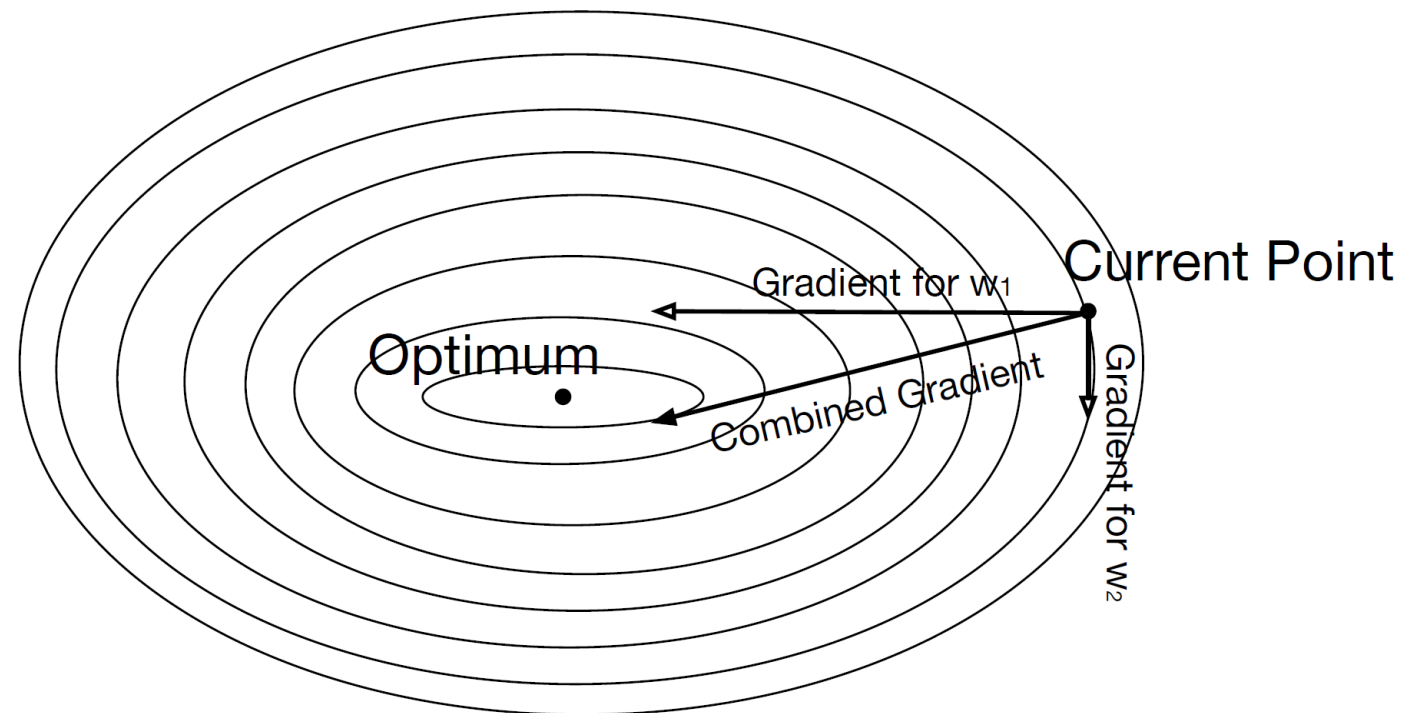
# Gradient Descent

- The **error** is a **function of the weights**
- We want to find the weights that minimize the error
- Compute **gradient**: gives the direction to the minimum
- Adjust weights, **moving at the direction of the gradient**.

# Gradient Descent

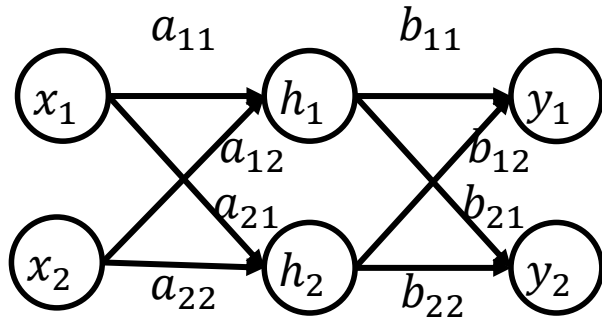


# Gradient Descent



# Backpropagation

- How can we compute the gradients? **Backpropagation!**
- Main idea:
  - Start from the final layer: compute the gradients for the weights of the final layer.
  - Use these gradients to compute the gradients of previous layers using the chain rule
  - Propagate the error backwards
- Backpropagation essentially is an application of the **chain rule** for differentiation.



Notation:

Activation function:  $g$

$$s_{y_1} = b_{11}h_1 + b_{12}h_2, y_1 = g(s_{y_1})$$

$$s_{y_2} = b_{21}h_1 + b_{22}h_2, y_2 = g(s_{y_2})$$

$$s_{h_1} = a_{11}x_1 + a_{12}x_2, h_1 = g(s_{h_1})$$

$$s_{h_2} = a_{21}x_1 + a_{22}x_2, h_2 = g(s_{h_2})$$

$$\text{Error: } E = \|y - t\|^2 = (y_1 - t_1)^2 + (y_2 - t_2)^2$$

$$\frac{\partial E}{\partial b_{11}} = \frac{\partial E}{\partial s_{y_1}} \frac{\partial s_{y_1}}{\partial b_{11}} = \delta_{y_1} h_1$$

$$\delta_{y_1} = \frac{\partial E}{\partial s_{y_1}} = \frac{\partial E}{\partial y_1} \frac{\partial y_1}{\partial s_{y_1}} = 2(y_1 - t_1)g'(s_{y_1})$$

$$\frac{\partial E}{\partial b_{21}} = \delta_{y_2} h_1$$

$$\delta_{y_2} = \frac{\partial E}{\partial s_{y_2}} = 2(y_2 - t_2)g'(s_{y_2})$$

$$\frac{\partial E}{\partial b_{12}} = \delta_{y_1} h_2$$

$$\frac{\partial E}{\partial b_{22}} = \delta_{y_2} h_2$$

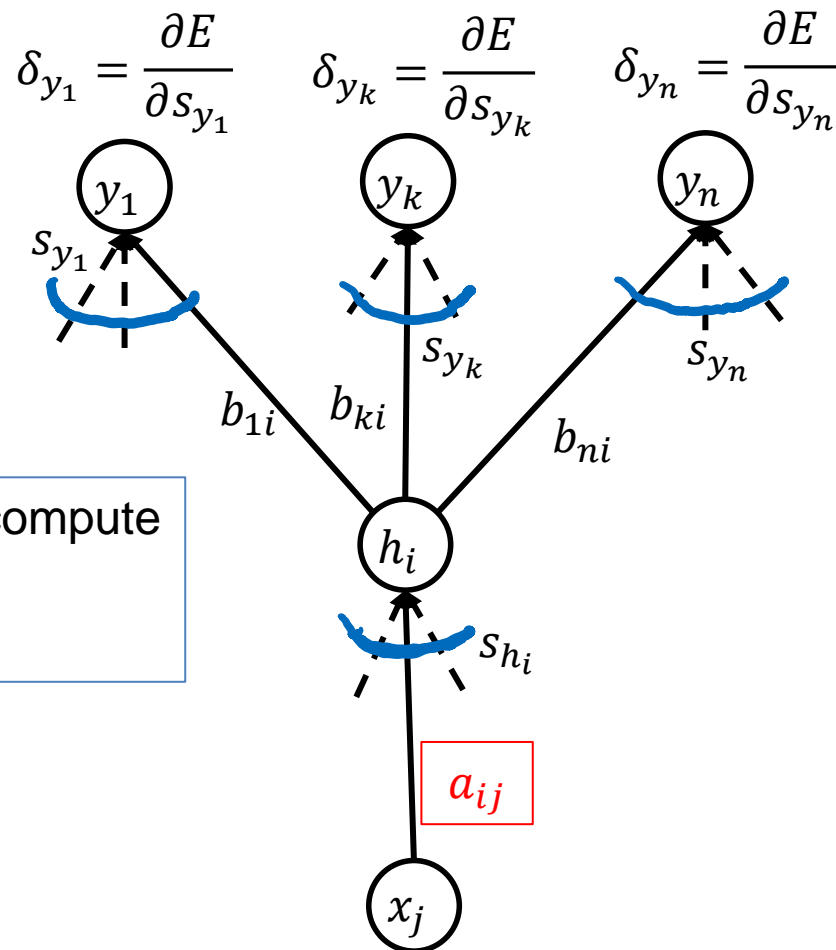
$$\frac{\partial E}{\partial a_{11}} = \frac{\partial E}{\partial s_{h_1}} \frac{\partial s_{h_1}}{\partial a_{11}} = \delta_{h_1} x_1 \quad \frac{\partial E}{\partial a_{22}} = \frac{\partial E}{\partial s_{h_2}} \frac{\partial s_{h_2}}{\partial a_{22}} = \delta_{h_2} x_2 \quad \frac{\partial E}{\partial a_{21}} = \delta_{h_1} x_2 \quad \frac{\partial E}{\partial a_{12}} = \delta_{h_2} x_1$$

$$\delta_{h_1} = \frac{\partial E}{\partial s_{h_1}} = \frac{\partial E}{\partial h_1} \frac{\partial h_1}{\partial s_{h_1}} = \left( \frac{\partial E}{\partial s_{y_1}} \frac{\partial s_{y_1}}{\partial h_1} + \frac{\partial E}{\partial s_{y_2}} \frac{\partial s_{y_2}}{\partial h_1} \right) g'(s_{h_1}) = (\delta_{y_1} b_{11} + \delta_{y_2} b_{21}) g'(s_{h_1})$$

$$\delta_{h_2} = (\delta_{y_1} b_{12} + \delta_{y_2} b_{22}) g'(s_{h_2})$$

# Backpropagation

We have already computed the  $\delta_{y_k}$ 's



We want to compute  
 $\frac{\partial E}{\partial a_{ij}}$

$$\frac{\partial E}{\partial a_{ij}} = \sum_{k=1}^n \delta_{y_k} b_{ki} g'(s_{h_i}) x_j$$

For the **sigmoid activation function**:

$$g(x) = \frac{1}{1 + e^{-x}}$$

The derivative is:

$$g'(x) = g(x)(1 - g(x))$$

This makes it easy to compute it. We have:

$$g'(s_{h_i}) = h_i(1 - h_i)$$

Therefore

$$\frac{\partial E}{\partial a_{ij}} = \sum_{k=1}^n \delta_{y_k} b_{ki} h_i(1 - h_i) x_j$$

# Stochastic gradient descent

- Ideally the loss should be the average loss over all training data.
- We would need to compute the loss for all training data every time we update the gradients.
  - However, this is expensive.
- **Stochastic gradient descent**: Consider one input point at the time. Each point is considered only once.
- Intermediate solution: Use **mini-batches** of data points.



# WORD EMBEDDINGS

---

Thanks to Chris Manning for the slides

# Basic Idea

- You can get a lot of value by representing a word by means of its neighbors
- “You shall know a word by the company it keeps”
  - (J. R. Firth 1957: 11)
- One of the most successful ideas of modern statistical NLP

government debt problems turning into banking crises as has happened in  
saying that Europe needs unified banking regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

# Basic idea

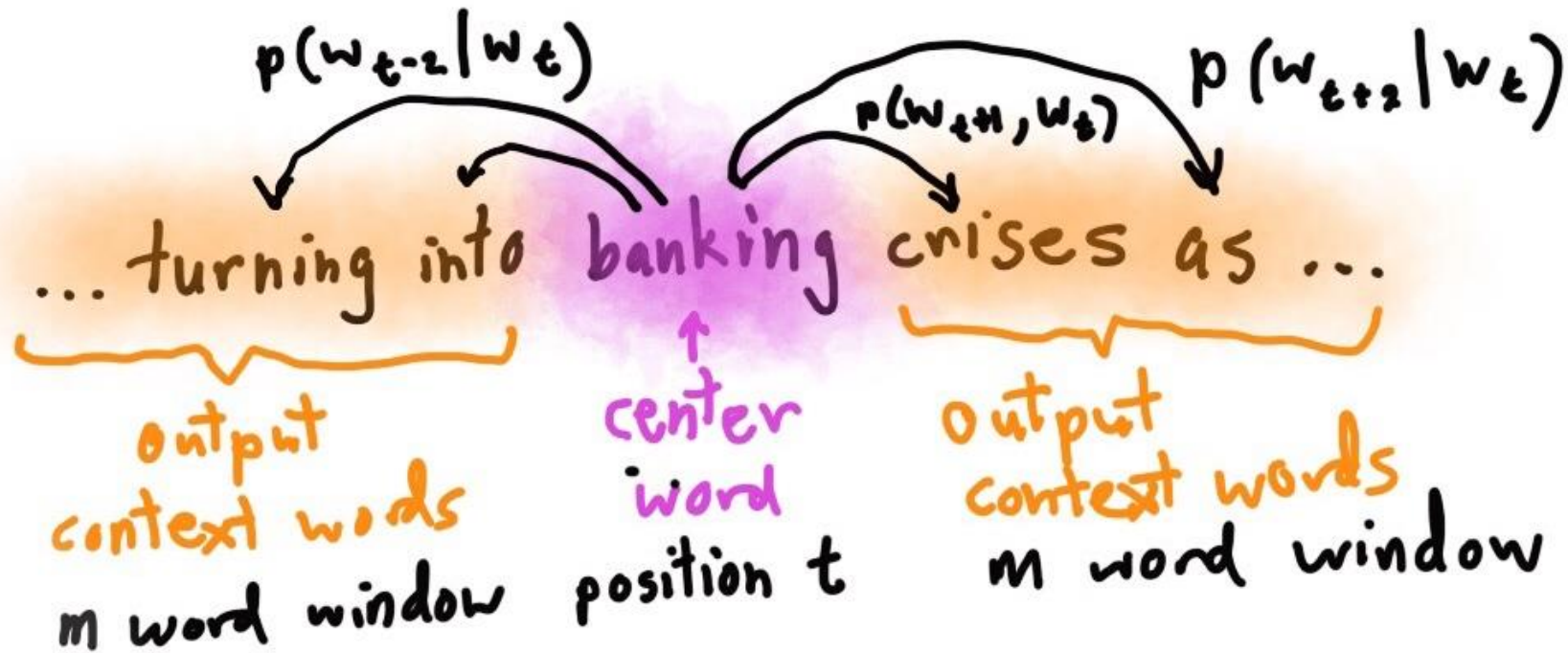
Define a model that aims to predict between a **center word**  $w_c$  and **context words** in some window of **length**  $m$  in terms of word vectors

$$P(w_c | w_{c-m}, \dots, w_{c-1}, w_{c+1}, \dots, w_{c+m})$$

*Pairwise probabilities*

Independence assumption (bigram model)

$$P(w_1, w_2, \dots, w_n) = \prod_{i=2}^n P(w_i | w_{i-1})$$



# Word2Vec

Predict between every word and its context words

## Two algorithms

### 1. Skip-grams (SG)

Predict context words given the center word

### 2. Continuous Bag of Words (CBOW)

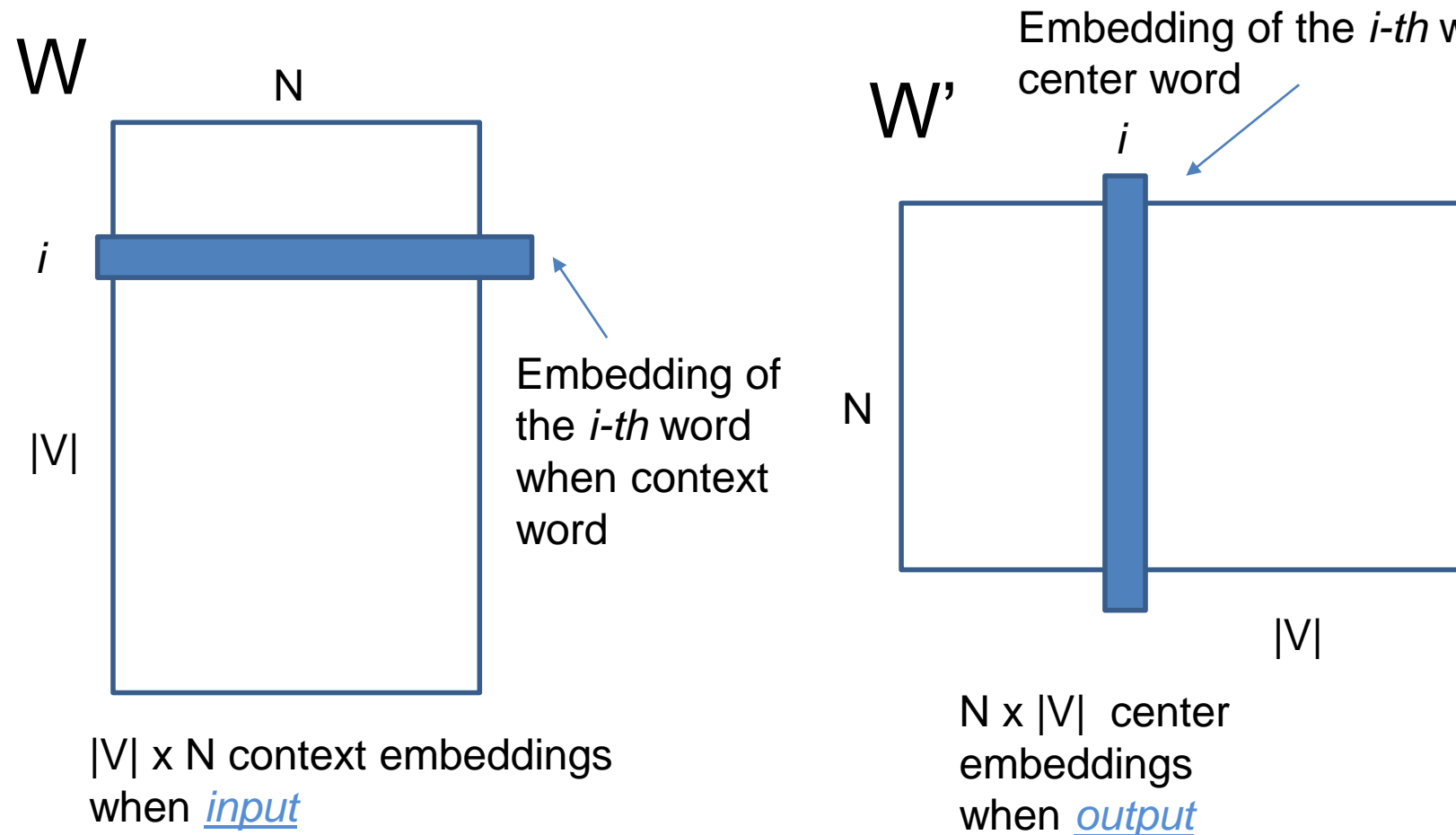
Predict center word from a bag-of-words context

*Position independent* (do not account for distance from center)

# CBOW

Use a window of context words to predict the center word

Learn **two matrices** ( $N$  size of embedding,  $|V|$  number of words)



# CBOW

Given **window size**  $m$

$x^{(c)}$  one hot vector for context words,  $y$  one hot vector for the center word

1. Input: the **one hot vectors** for the  $2m$  context words

$$x^{(c-m)}, \dots, x^{(c-1)}, x^{(c+1)}, \dots, x^{(c+m)}$$

2. Compute the **embeddings** of the context words

$$v_{c-m} = Wx^{(c-m)}, \dots, v_{c-1} = Wx^{(c-1)}, v_{c+1} = Wx^{(c+1)}, \dots, v_{c+m} = Wx^{(c+m)}$$

3. **Average** these vectors

$$\hat{v} = \frac{v_{c-m} + v_{c-m+1} + \dots + v_{c+m}}{2m}, \hat{v} \in R^N$$

4. Generate a **score vector**

$$z = W' \hat{v}$$

dot product, (embedding of center word) similar vectors close to each other

5. Turn the **score vector to probabilities**

$$\hat{y} = \text{softmax}(z)$$

We want this to be close to 1 for the center word

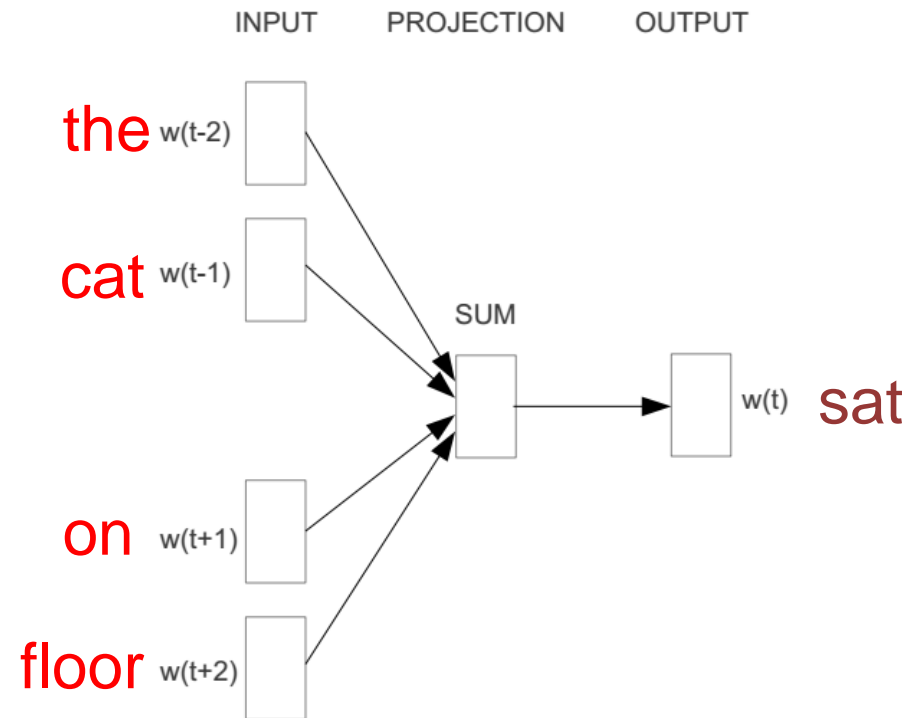
*Exponentiate to  
make positive* **Softmax**

*Normalize to  
give probability*

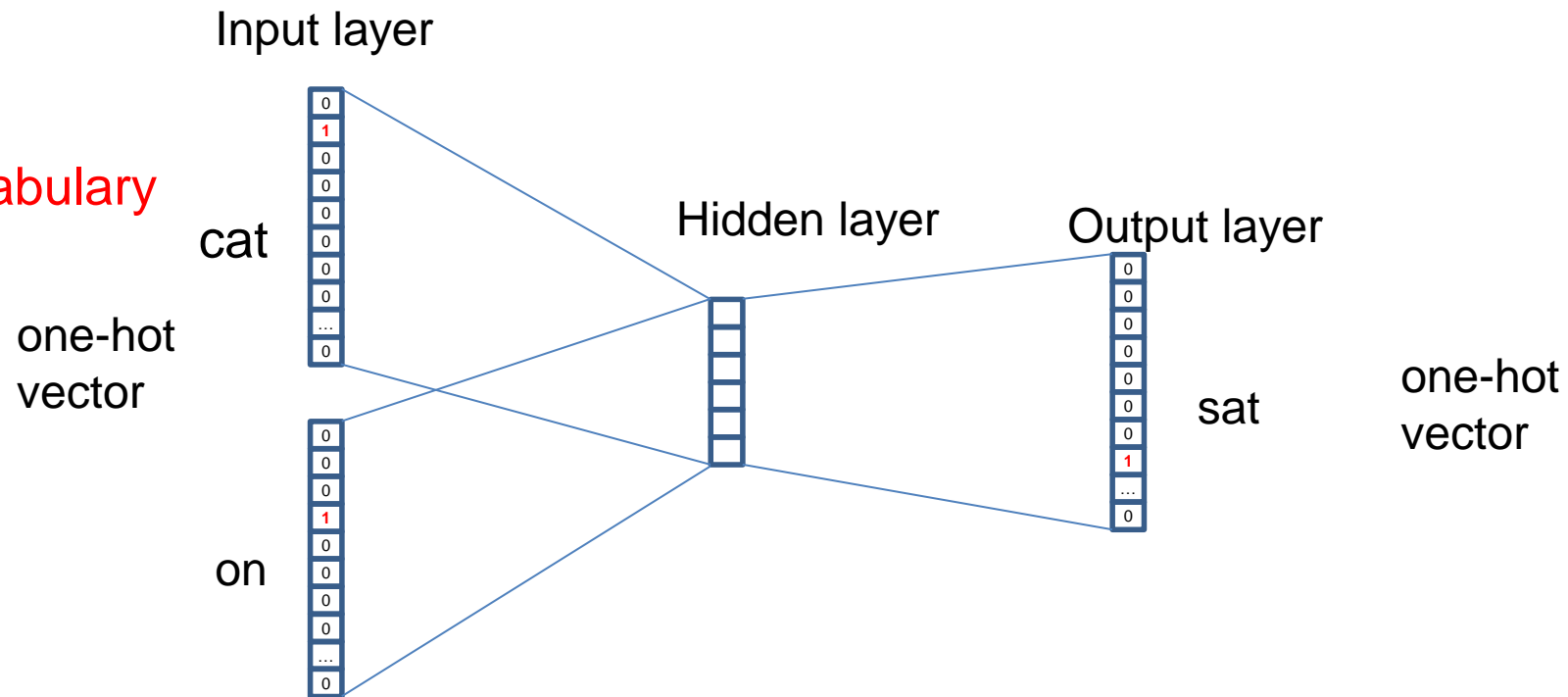
$$p_i = \frac{e^{u_i}}{\sum_j e^{u_j}}$$

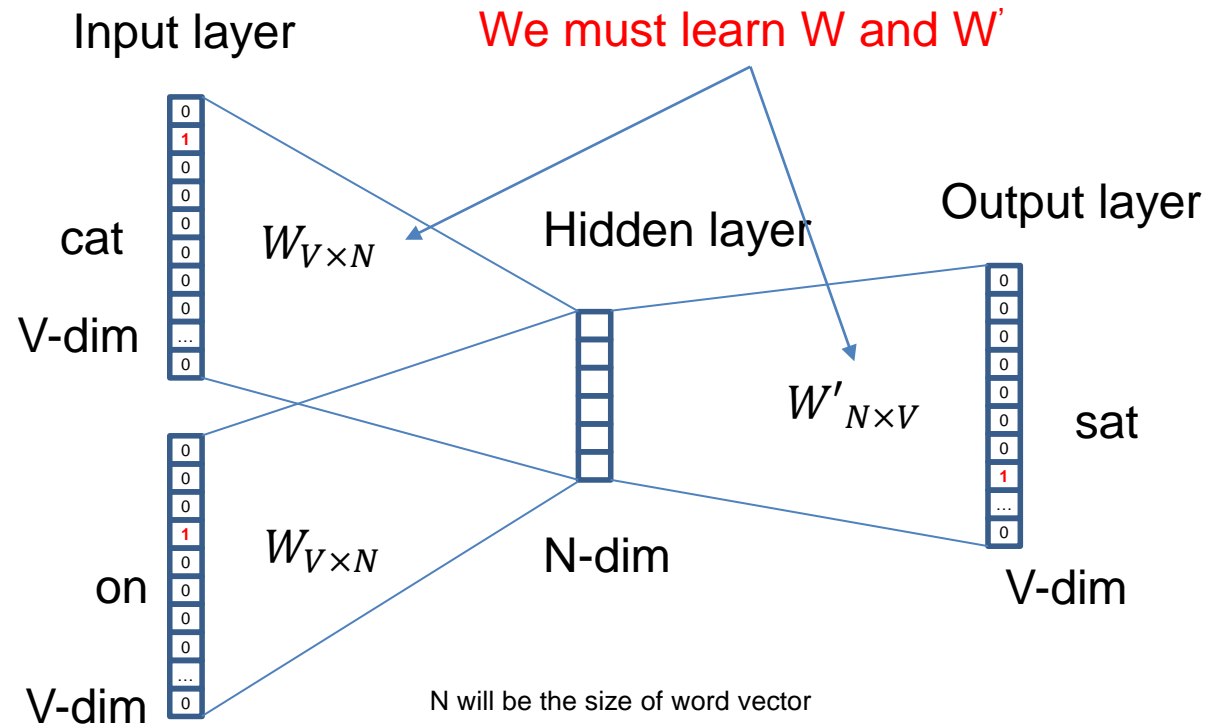


- E.g. “The cat **sat** on floor”
  - Window size = 2



Index of cat in vocabulary

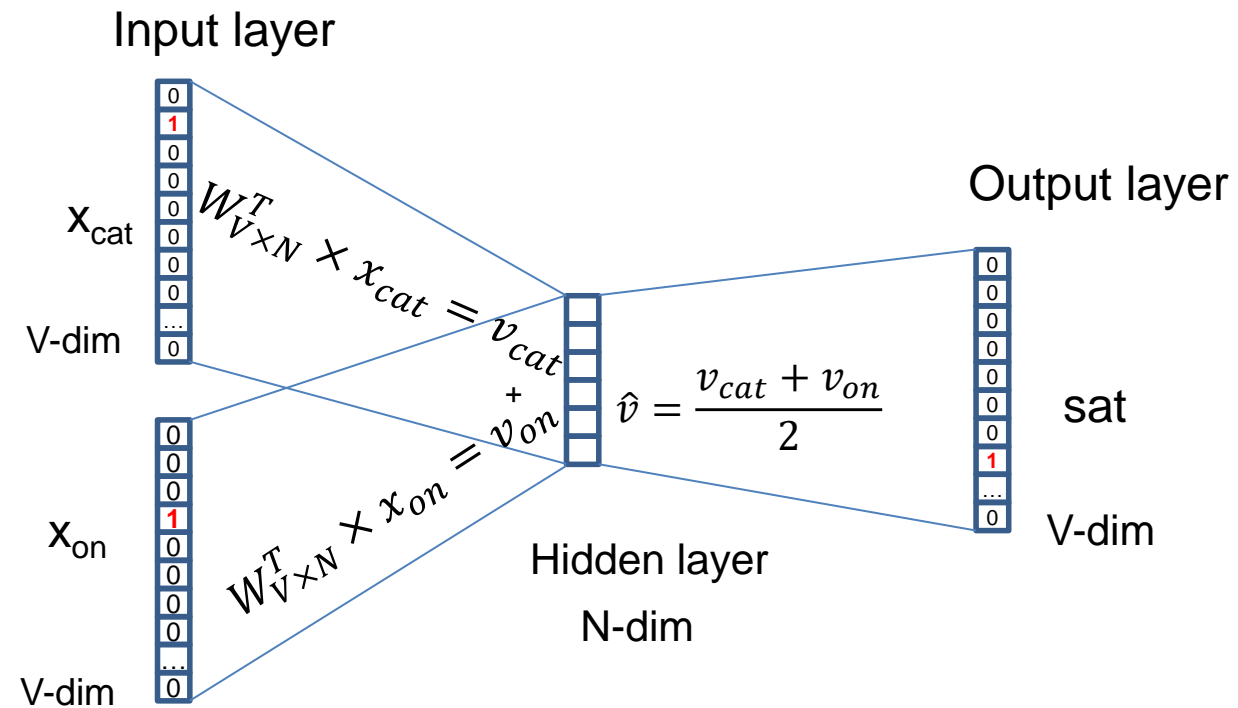




$$W_{V \times N}^T \times x_{cat} = v_{cat}$$

0.1	<b>2.4</b>	1.6	1.8	0.5	0.9	...	...	...	3.2
0.5	<b>2.6</b>	1.4	2.9	1.5	3.6	...	...	...	6.1
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
0.6	<b>1.8</b>	2.7	1.9	2.4	2.0	...	...	...	1.2

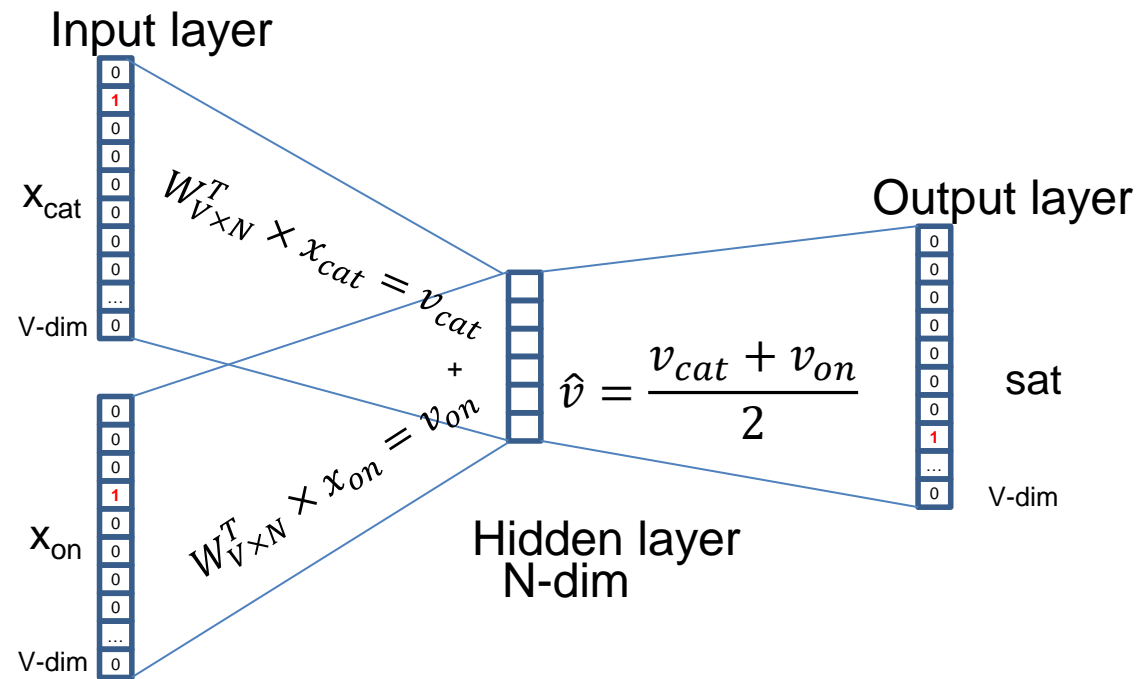
$$\times \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} = \begin{bmatrix} 2.4 \\ 2.6 \\ \dots \\ \dots \\ 1.8 \end{bmatrix}$$

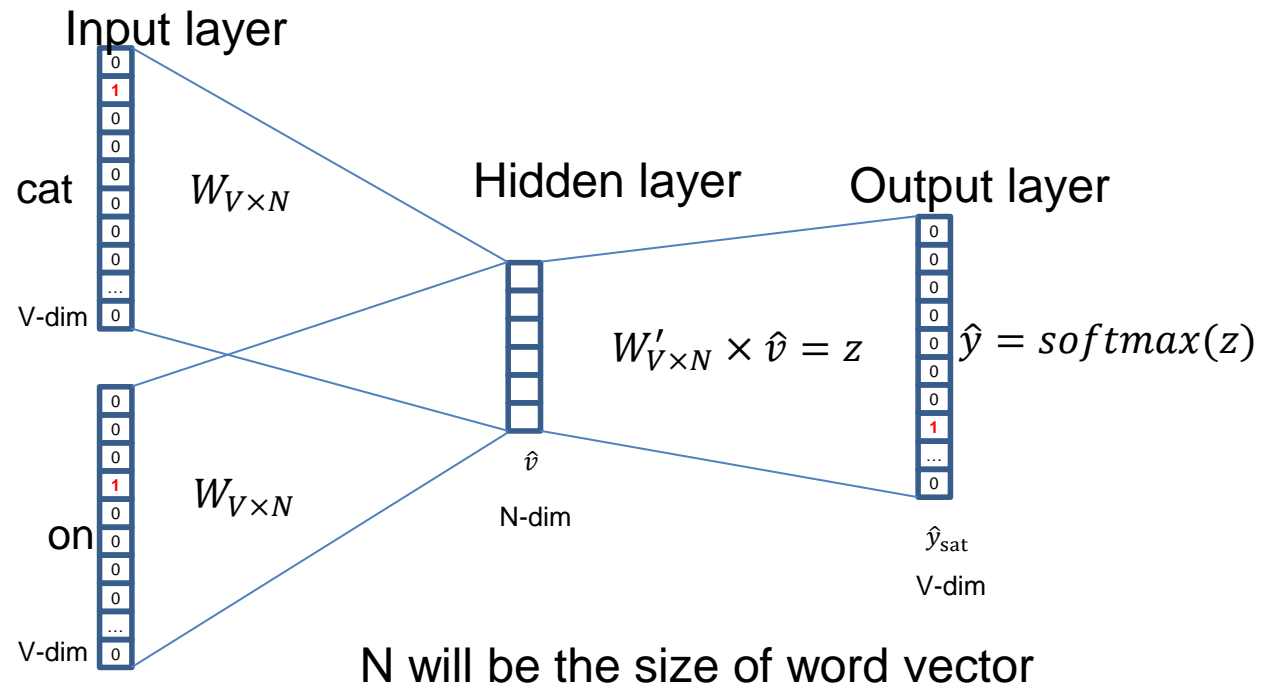


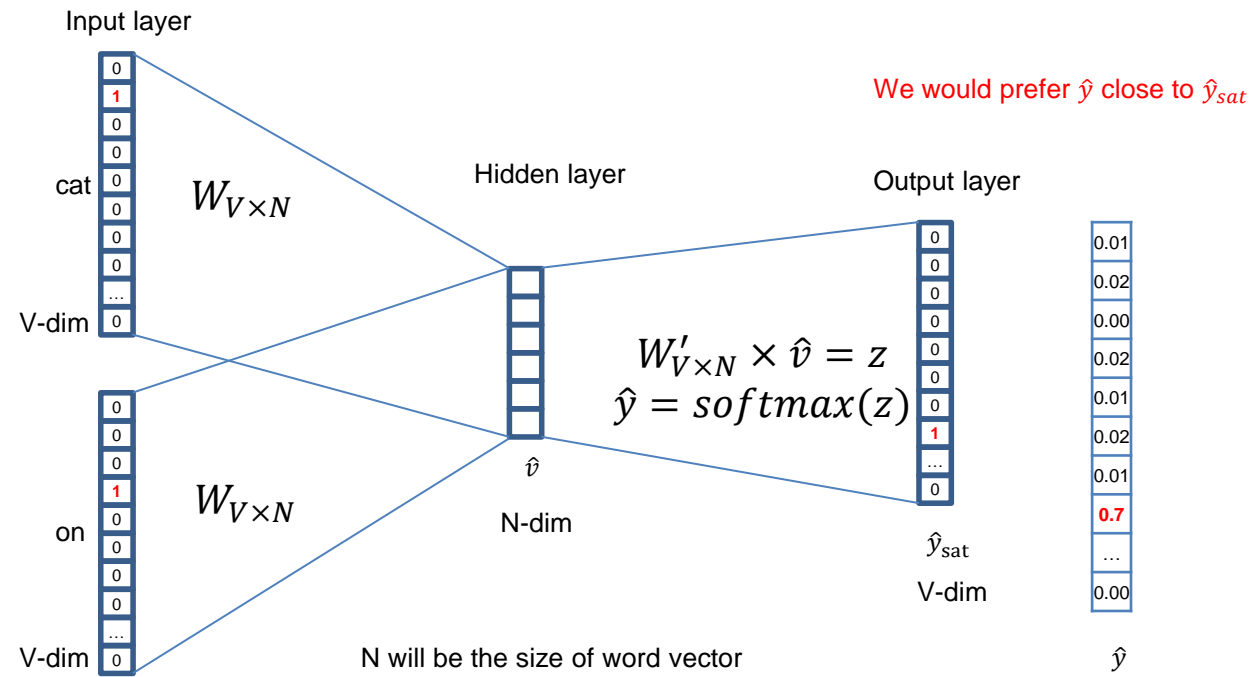
$$W_{V \times N}^T \times x_{on} = v_{on}$$

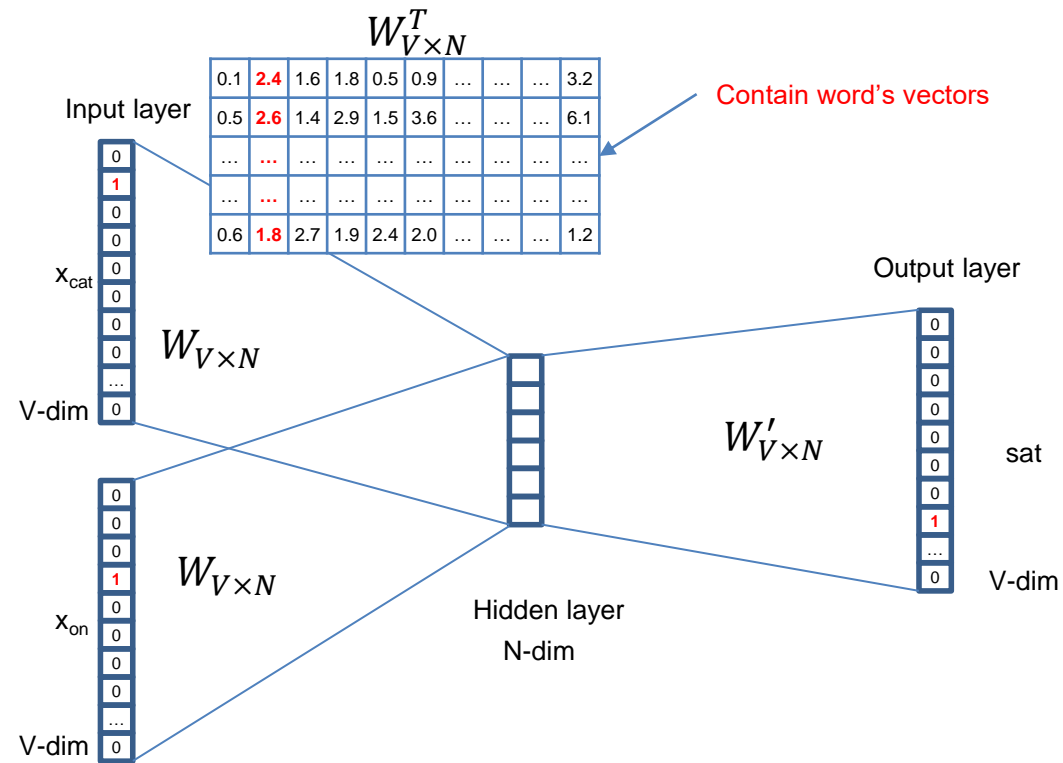
0.1	2.4	1.6	<b>1.8</b>	0.5	0.9	...	...	...	3.2
0.5	2.6	1.4	<b>2.9</b>	1.5	3.6	...	...	...	6.1
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
0.6	1.8	2.7	<b>1.9</b>	2.4	2.0	...	...	...	1.2

$$\times \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mathbf{1} \\ 0 \\ 0 \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mathbf{1.8} \\ 0 \\ 0 \\ 0 \\ 0 \\ \dots \\ \mathbf{1.9} \end{bmatrix}$$









We can consider either  $W$  (context) or  $W'$  (center) as the word's representation.  
Or even take the average.



# Skipgram

Given the center word, predict (or, generate) the context words

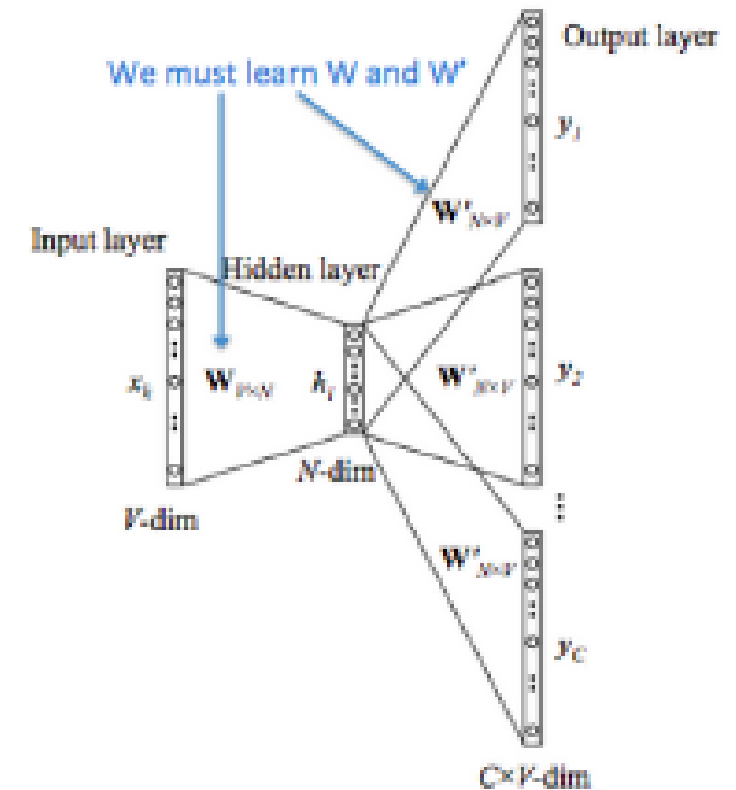
$W$ :  $N \times |V|$ , input matrix, word representation as **center** word

$W'$ :  $|V| \times N$ , output matrix, word representation as **context** word

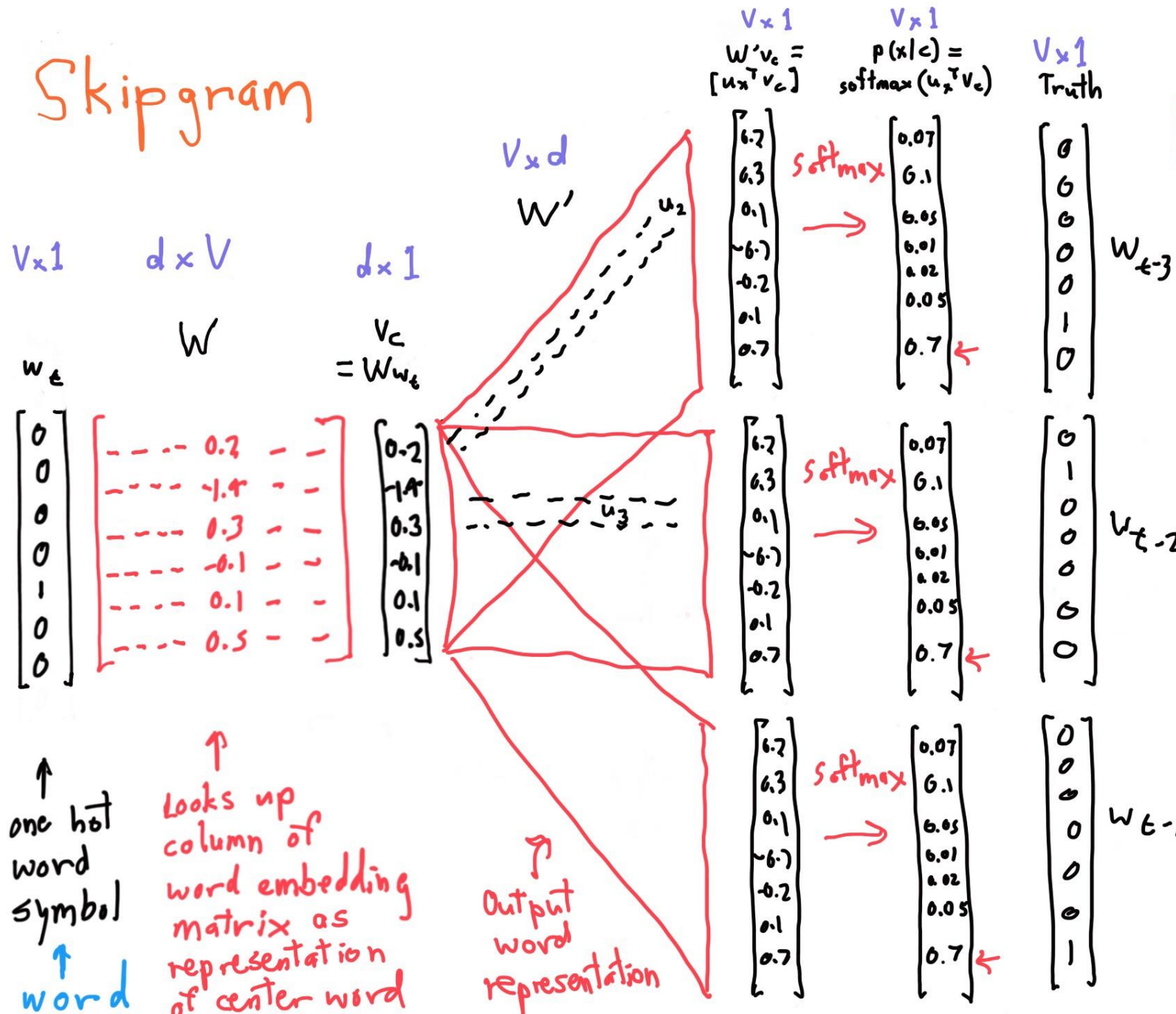
$y^{(j)}$  one hot vector for context words

1. Get *one hot vector* of the center word  $x$
2. Get the *embedding* of the center word  
 $v_c = W x$
3. Generate a *score vector* for *each context word*  
 $z = W' v_c$
5. Turn the *score vector into probabilities*  
 $\hat{y} = \text{softmax}(z)$

We want this to be close to 1 for the context words



# Skipgram



**Softmax**

$$p_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

↑  
one hot word symbol  
↑  
word

↑  
Looks up column of word embedding matrix as representation of center word

↑  
Output word representation

↑  
Actual context words

# Skipgram

- For each word  $t = 1 \dots T$ , predict surrounding words in a window of “radius”  $m$  of every word.
- **Objective function:** Maximize the probability of any context word given the current center word:

likelihood

$$J'(\theta) = \prod_{t=1}^T \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} p(w_{t+j} | w_t; \theta)$$

Negative  
Log Likelihood

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log p(w_{t+j} | w_t; \theta)$$

where  $\theta$  represents all variables we will optimize

- The basic skipgram utilizes the softmax function:

$$p(c|w) = \frac{\exp(v'_c{}^T v_w)}{\sum_{i=1}^T \exp(v'_i{}^T v_w)}$$

- Where:
  - T – # of words in the corpus.
  - $v_w$  - input vector of  $w$ .
  - $v'_w$  - output vector of  $w$ .

Word	Input	Output
<i>King</i>	[0.2,0.9,0.1]	[0.5,0.4,0.5]
<i>Queen</i>	[0.2,0.8,0.2]	[0.4,0.5,0.5]
<i>Apple</i>	[0.9,0.5,0.8]	[0.3,0.9,0.1]
<i>Orange</i>	[0.9,0.4,0.9]	[0.1,0.7,0.2]



These representations are *very good* at encoding **similarity** and **dimensions of similarity**!

- Analogies testing dimensions of similarity can be solved quite well just by doing vector subtraction in the embedding space

Syntactically

$$- X_{apple} - X_{apples} \approx X_{car} - X_{cars} \approx X_{family} - X_{families}$$

– Similarly for verb and adjective morphological forms

Semantically (Semeval 2012 task 2)

$$- X_{shirt} - X_{clothing} \approx X_{chair} - X_{furniture}$$

$$- X_{king} - X_{man} \approx X_{queen} - X_{woman}$$

## Test for linear relationships, examined by Mikolov et al.

a:b :: c:?



$$d = \arg \max_x \frac{(w_b - w_a + w_c)^T w_x}{\|w_b - w_a + w_c\|}$$

man:woman :: king:?

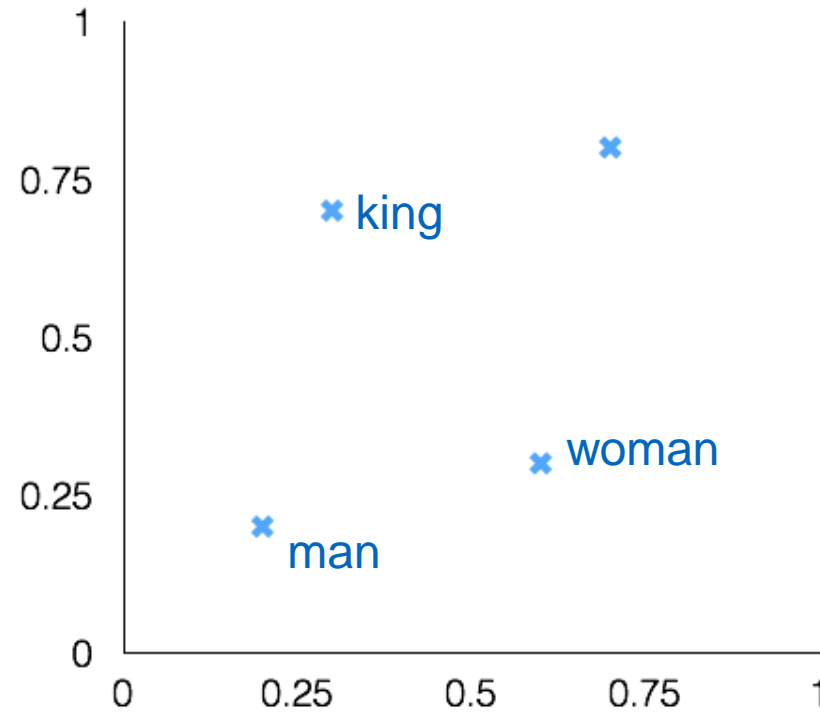
+ king [ 0.30 0.70 ]

- man [ 0.20 0.20 ]

+ woman [ 0.60 0.30 ]

---

queen [ 0.70 0.80 ]



# NAÏVE BAYES CLASSIFIER

---



# Bayes Classifier

- A probabilistic framework for solving classification problems
- $A$ ,  $C$  random variables
- **Joint** probability:  $\Pr(A=a, C=c)$
- **Conditional** probability:  $\Pr(C=c | A=a)$
- Relationship between joint and conditional probability distributions  
$$\Pr(C, A) = \Pr(C|A) P(A) = P(A|C)P(C)$$
- **Bayes Theorem:**

$$P(C|A) = \frac{P(A|C)P(C)}{P(A)}$$

# Bayesian Classifiers

- How to classify the new record  $X = (\text{'Yes'}, \text{'Single'}, 80\text{K})$

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Find the class with the highest probability given the vector values.

Maximum A posteriori Probability estimate:

- Find the value  $c$  for class  $C$  that maximizes  $P(C=c | X)$

How do we estimate  $P(C|X)$  for the different values of  $C$ ?

- We want to estimate  $P(C=\text{Yes} | X)$
- and  $P(C=\text{No} | X)$

# Bayesian Classifiers

- In order for probabilities to be well defined:
  - Consider each attribute and the class label as **random variables**
  - Probabilities are determined from the data

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

## Evade C

Event space: {Yes, No}

$$P(C) = (0.3, 0.7)$$

## Refund $A_1$

Event space: {Yes, No}

$$P(A_1) = (0.3, 0.7)$$

## Marital Status $A_2$

Event space: {Single, Married, Divorced}

$$P(A_2) = (0.4, 0.4, 0.2)$$

## Taxable Income $A_3$

?

Event space: R

$$P(A_3) \sim Normal( \mu, \sigma^2 )$$

$\mu = 104$ :sample mean,  $\sigma^2 = 1874$ :sample var

# Bayesian Classifiers

- Approach:
  - compute the posterior probability  $P(C | A_1, A_2, \dots, A_n)$  using the Bayes theorem

$$P(C|A_1, A_2, \dots, A_n) = \frac{P(A_1, A_2, \dots, A_n|C)P(C)}{P(A_1, A_2, \dots, A_n)}$$

- Maximizing

$$P(C | A_1, A_2, \dots, A_n)$$

is equivalent to maximizing

$$P(A_1, A_2, \dots, A_n|C) P(C)$$

- The value  $P(A_1, \dots, A_n)$  is the same for all values of  $C$ .
- How to estimate  $P(A_1, A_2, \dots, A_n | C)$ ?

# Naïve Bayes Classifier

- Assume **conditional independence** among attributes  $A_i$  when class  $C$  is given:

- $P(A_1, A_2, \dots, A_n | C) = P(A_1 | C) P(A_2 | C) \cdots P(A_n | C)$

- We can estimate  $P(A_i | C)$  from the data.

- New point  $X = (A_1 = \alpha_1, \dots, A_n = \alpha_n)$  is classified to class  $c$  if

$$P(C = c | X) = P(C = c) \prod_i P(A_i = \alpha_i | c)$$

is maximum over all possible values of  $C$ .

# Example

- Record

$X = (\text{Refund} = \text{Yes}, \text{Status} = \text{Single}, \text{Income} = 80\text{K})$

- For the class  $C = \text{'Evade'}$ , we want to compute:

$P(C = \text{Yes}|X)$  and  $P(C = \text{No}| X)$

- We compute:

- $P(C = \text{Yes}|X) = P(C = \text{Yes}) * P(\text{Refund} = \text{Yes} | C = \text{Yes})$

- \*  $P(\text{Status} = \text{Single} | C = \text{Yes})$

- \*  $P(\text{Income} = 80\text{K} | C = \text{Yes})$

- $P(C = \text{No}|X) = P(C = \text{No}) * P(\text{Refund} = \text{Yes} | C = \text{No})$

- \*  $P(\text{Status} = \text{Single} | C = \text{No})$

- \*  $P(\text{Income} = 80\text{K} | C = \text{No})$

# How to Estimate Probabilities from Data?

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Class Prior Probability:

$$P(C = c) = \frac{N_c}{N}$$

$N_c$ : Number of records with class  $c$

$N$  = Number of records

$$P(C = \text{No}) = 7/10$$

$$P(C = \text{Yes}) = 3/10$$

# How to Estimate Probabilities from Data?

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Discrete attributes:

$$P(A_i = a | C = c) = \frac{N_{a,c}}{N_c}$$

$N_{a,c}$ : number of instances having attribute  $A_i = a$  and belong to class  $c$

$N_c$ : number of instances of class  $c$



# How to Estimate Probabilities from Data?

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Discrete attributes:

$$P(A_i = a | C = c) = \frac{N_{a,c}}{N_c}$$

$N_{a,c}$ : number of instances having attribute  $A_i = a$  and belong to class  $c$

$N_c$ : number of instances of class  $c$

$$P(\text{Refund} = \text{Yes} | \text{No}) = 3/7$$

# How to Estimate Probabilities from Data?

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Discrete attributes:

$$P(A_i = a | C = c) = \frac{N_{a,c}}{N_c}$$

$N_{a,c}$ : number of instances having attribute  $A_i = a$  and belong to class  $c$

$N_c$ : number of instances of class  $c$

$$P(\text{Refund} = \text{Yes} | \text{Yes}) = 0$$

# How to Estimate Probabilities from Data?

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Discrete attributes:

$$P(A_i = a | C = c) = \frac{N_{a,c}}{N_c}$$

$N_{a,c}$ : number of instances having attribute  $A_i = a$  and belong to class  $c$

$N_c$ : number of instances of class  $c$

$$P(\text{Status}=\text{Single}|\text{No}) = 2/7$$

# How to Estimate Probabilities from Data?

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Discrete attributes:

$$P(A_i = a | C = c) = \frac{N_{a,c}}{N_c}$$

$N_{a,c}$ : number of instances having attribute  $A_i = a$  and belong to class  $c$

$N_c$ : number of instances of class  $c$

$$P(\text{Status}=\text{Single}|\text{Yes}) = 2/3$$

# How to Estimate Probabilities from Data?

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i = a | c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(a-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

- One for each  $(A_i, c_j)$  pair
- For **Class=No**
  - sample mean  $\mu = 110$
  - sample variance  $\sigma^2 = 2975$
- For **Income = 80**

$$P(\text{Income} = 80 | \text{No}) = \frac{1}{\sqrt{2\pi(54.54)}} e^{-\frac{(80-110)^2}{2(2975)}} = 0.0062$$

# How to Estimate Probabilities from Data?

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i = a | c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(a-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

- One for each  $(A_i, c_j)$  pair
- For **Class=Yes**
  - sample mean  $\mu = 90$
  - sample variance  $\sigma^2 = 2975$
- For **Income = 80**

$$P(\text{Income} = 80 | \text{Yes}) = \frac{1}{\sqrt{2\pi(5)}} e^{-\frac{(80-90)^2}{2(25)}} = 0.01$$



# Example of Naïve Bayes Classifier

- Creating a Naïve Bayes Classifier, essentially means to compute **counts**:

Total number of records:  $N = 10$

## Class No:

Number of records: 7

### Attribute Refund:

Yes: 3

No: 4

### Attribute Marital Status:

Single: 2

Divorced: 1

Married: 4

### Attribute Income:

mean: 110

variance: 2975

## Class Yes:

Number of records: 3

### Attribute Refund:

Yes: 0

No: 3

### Attribute Marital Status:

Single: 2

Divorced: 1

Married: 0

### Attribute Income:

mean: 90

variance: 25

naive Bayes Classifier:

$$P(\text{Refund}=\text{Yes} | \text{No}) = 3/7$$

$$P(\text{Refund}=\text{No} | \text{No}) = 4/7$$

$$P(\text{Refund}=\text{Yes} | \text{Yes}) = 0$$

$$P(\text{Refund}=\text{No} | \text{Yes}) = 1$$

$$P(\text{Marital Status}=\text{Single} | \text{No}) = 2/7$$

$$P(\text{Marital Status}=\text{Divorced} | \text{No}) = 1/7$$

$$P(\text{Marital Status}=\text{Married} | \text{No}) = 4/7$$

$$P(\text{Marital Status}=\text{Single} | \text{Yes}) = 2/7$$

$$P(\text{Marital Status}=\text{Divorced} | \text{Yes}) = 1/7$$

$$P(\text{Marital Status}=\text{Married} | \text{Yes}) = 0$$

For taxable income:

If class=No: sample mean=110

sample variance=2975

If class=Yes: sample mean=90

sample variance=25



# Example of Naïve Bayes Classifier

Given a Test Record:

$X = (\text{Refund} = \text{Yes}, \text{Status} = \text{Single}, \text{Income} = 80\text{K})$

naive Bayes Classifier:

$$P(\text{Refund}=\text{Yes} | \text{No}) = 3/7$$

$$P(\text{Refund}=\text{No} | \text{No}) = 4/7$$

$$P(\text{Refund}=\text{Yes} | \text{Yes}) = 0$$

$$P(\text{Refund}=\text{No} | \text{Yes}) = 1$$

$$P(\text{Marital Status}=\text{Single} | \text{No}) = 2/7$$

$$P(\text{Marital Status}=\text{Divorced} | \text{No}) = 1/7$$

$$P(\text{Marital Status}=\text{Married} | \text{No}) = 4/7$$

$$P(\text{Marital Status}=\text{Single} | \text{Yes}) = 2/7$$

$$P(\text{Marital Status}=\text{Divorced} | \text{Yes}) = 1/7$$

$$P(\text{Marital Status}=\text{Married} | \text{Yes}) = 0$$

For taxable income:

If class=No: sample mean=110  
sample variance=2975

If class=Yes: sample mean=90  
sample variance=25

- $P(X|\text{Class}=\text{No}) = P(\text{Refund}=\text{Yes}|\text{Class}=\text{No})$   
 $\times P(\text{Married} | \text{Class}=\text{No})$   
 $\times P(\text{Income}=120\text{K} | \text{Class}=\text{No})$   
 $= 3/7 * 2/7 * 0.0062 = 0.00075$
  - $P(X|\text{Class}=\text{Yes}) = P(\text{Refund}=\text{No} | \text{Class}=\text{Yes})$   
 $\times P(\text{Married} | \text{Class}=\text{Yes})$   
 $\times P(\text{Income}=120\text{K} | \text{Class}=\text{Yes})$   
 $= 0 * 2/3 * 0.01 = 0$
  - $P(\text{No}) = 0.3, P(\text{Yes}) = 0.7$
- Since  $P(X|\text{No})P(\text{No}) > P(X|\text{Yes})P(\text{Yes})$   
Therefore  $P(\text{No}|X) > P(\text{Yes}|X)$   
 $\Rightarrow \text{Class} = \text{No}$

# Naïve Bayes Classifier

- If one of the conditional probabilities is **zero**, then the entire expression becomes zero
- Laplace Smoothing:

$$P(A_i = a | C = c) = \frac{N_{ac} + 1}{N_c + N_i}$$

- $N_i$ : number of attribute **values** for attribute  $A_i$

# Example of Naïve Bayes Classifier

- Creating a Naïve Bayes Classifier, essentially means to compute **counts**:

With Laplace Smoothing

naive Bayes Classifier:

Total number of records:  $N = 10$

**Class No:**

Number of records: 7

**Attribute Refund:**

Yes: 3

No: 4

**Attribute Marital Status:**

Single: 2

Divorced: 1

Married: 4

**Attribute Income:**

mean: 110

variance: 2975

**Class Yes:**

Number of records: 3

**Attribute Refund:**

Yes: 0

No: 3

**Attribute Marital Status:**

Single: 2

Divorced: 1

Married: 0

**Attribute Income:**

mean: 90

variance: 25

$P(\text{Refund}=\text{Yes}|\text{No}) = 4/9$

$P(\text{Refund}=\text{No}|\text{No}) = 5/9$

$P(\text{Refund}=\text{Yes}|\text{Yes}) = 1/5$

$P(\text{Refund}=\text{No}|\text{Yes}) = 4/5$

$P(\text{Marital Status}=\text{Single}|\text{No}) = 3/10$

$P(\text{Marital Status}=\text{Divorced}|\text{No}) = 2/10$

$P(\text{Marital Status}=\text{Married}|\text{No}) = 5/10$

$P(\text{Marital Status}=\text{Single}|\text{Yes}) = 3/6$

$P(\text{Marital Status}=\text{Divorced}|\text{Yes}) = 2/6$

$P(\text{Marital Status}=\text{Married}|\text{Yes}) = 1/6$

For taxable income:

If class=No: sample mean=110

sample variance=2975

If class=Yes: sample mean=90

sample variance=25

# Example of Naïve Bayes Classifier

Given a Test Record:

With Laplace Smoothing

$X = (\text{Refund} = \text{Yes}, \text{Status} = \text{Single}, \text{Income} = 80\text{K})$

naive Bayes Classifier:

$P(\text{Refund}=\text{Yes} | \text{No}) = 4/9$   
 $P(\text{Refund}=\text{No} | \text{No}) = 5/9$   
 $P(\text{Refund}=\text{Yes} | \text{Yes}) = 1/5$   
 $P(\text{Refund}=\text{No} | \text{Yes}) = 4/5$

$P(\text{Marital Status}=\text{Single} | \text{No}) = 3/10$   
 $P(\text{Marital Status}=\text{Divorced} | \text{No}) = 2/10$   
 $P(\text{Marital Status}=\text{Married} | \text{No}) = 5/10$   
 $P(\text{Marital Status}=\text{Single} | \text{Yes}) = 3/6$   
 $P(\text{Marital Status}=\text{Divorced} | \text{Yes}) = 2/6$   
 $P(\text{Marital Status}=\text{Married} | \text{Yes}) = 1/6$

For taxable income:

If class=No: sample mean=110  
sample variance=2975

If class=Yes: sample mean=90  
sample variance=25

- $P(X|\text{Class}=\text{No}) = P(\text{Refund}=\text{No}|\text{Class}=\text{No})$   
 $\times P(\text{Married} | \text{Class}=\text{No})$   
 $\times P(\text{Income}=120\text{K} | \text{Class}=\text{No})$   
 $= 4/9 \times 3/10 \times 0.0062 = 0.00082$
  - $P(X|\text{Class}=\text{Yes}) = P(\text{Refund}=\text{No} | \text{Class}=\text{Yes})$   
 $\times P(\text{Married} | \text{Class}=\text{Yes})$   
 $\times P(\text{Income}=120\text{K} | \text{Class}=\text{Yes})$   
 $= 1/5 \times 3/6 \times 0.01 = 0.001$
  - $P(\text{No}) = 0.7, P(\text{Yes}) = 0.3$
  - $P(X|\text{No})P(\text{No}) = 0.0005$
  - $P(X|\text{Yes})P(\text{Yes}) = 0.0003$
- => Class = No**

# Implementation details

- Computing the conditional probabilities involves multiplication of many very small numbers
  - Numbers get very close to zero, and there is a danger of numeric instability
- We can deal with this by computing the **logarithm** of the conditional probability

$$\begin{aligned}\log P(C|A) &\sim \log P(A|C) + \log P(C) \\ &= \sum_i \log P(A_i|C) + \log P(C)\end{aligned}$$

# Naïve Bayes for Text Classification

- Naïve Bayes is commonly used for **text classification**
- For a document with **k** terms  $d = (t_1, \dots, t_k)$

$$P(c|d) = P(c)P(d|c) = P(c) \prod_{t_i \in d} P(t_i|c)$$

Fraction of documents in c

- $P(t_i|c)$  = Fraction of terms from **all documents** in c that are  $t_i$ .

$$P(t_i|c) = \frac{N_{ic} + 1}{N_c + T}$$

Number of times  $t_i$  appears in all documents in c

Laplace Smoothing

Number of unique words (vocabulary size)

Total number of terms in all documents in c

- Easy to implement and works relatively well
- **Limitation:** Hard to incorporate **additional features** (beyond words).
  - E.g., number of adjectives used.

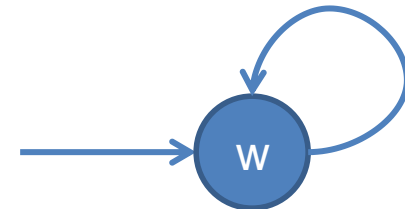
# Multinomial document model

- Probability of document  $d = (t_1, \dots, t_k)$  in class  $c$ :

$$P(d|c) = P(c) \prod_{t_i \in d} P(t_i|c)$$

- This formula assumes a **multinomial distribution** for the document generation:
  - If we have probabilities  $p_1, \dots, p_T$  for events  $t_1, \dots, t_T$  the probability of a subset of these is

$$P(d) = \frac{N}{N_{t_1}! N_{t_2}! \dots N_{t_T}!} p_1^{N_{t_1}} p_2^{N_{t_2}} \dots p_T^{N_{t_T}}$$



- Equivalently: There is an **automaton** spitting words from the above distribution

```

TRAINMULTINOMIALNB(C, D)
1  V ← EXTRACTVOCABULARY(D)
2  N ← COUNTDOCS(D)
3  for each c ∈ C
4  do Nc ← COUNTDOCSINCLASS(D, c)
5     prior[c] ← Nc/N
6     textc ← CONCATENATETEXTOFALLDOCSINCLASS(D, c)
7     for each t ∈ V
8     do Tct ← COUNTTOKENSOFTERM(textc, t)
9     for each t ∈ V
10    do condprob[t][c] ←  $\frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$ 
11  return V, prior, condprob

```

```

APPLYMULTINOMIALNB(C, V, prior, condprob, d)
1  W ← EXTRACTTOKENSFROMDOC(V, d)
2  for each c ∈ C
3  do score[c] ← log prior[c]
4     for each t ∈ W
5     do score[c] += log condprob[t][c]
6  return arg maxc∈C score[c]

```

► Figure 13.2 Naive Bayes algorithm (multinomial model): Training and testing.



# Example

News titles for **Politics** and **Sports**

**Politics**

documents

“Obama meets Merkel”  
“Obama elected again”  
“Merkel visits Greece again”

$$P(p) = 0.5$$

terms

Vocabulary  
size: 14

obama:2, meets:1, merkel:2,  
elected:1, again:2, visits:1,  
greece:1

Total terms: 10

**Sports**

“OSFP European basketball champion”  
“Miami NBA basketball champion”  
“Greece basketball coach?”

$$P(s) = 0.5$$

OSFP:1, european:1, basketball:3,  
champion:2, miami:1, nba:1,  
greece:1, coach:1

Total terms: 11

New title: **X = “Obama likes basketball”**

$$\begin{aligned} P(\text{Politics}|X) &\sim P(p) \cdot P(\text{obama}|p) \cdot P(\text{likes}|p) \cdot P(\text{basketball}|p) \\ &= 0.5 * 3/(10+14) * 1/(10+14) * 1/(10+14) = 0.000108 \end{aligned}$$

$$\begin{aligned} P(\text{Sports}|X) &\sim P(s) \cdot P(\text{obama}|s) \cdot P(\text{likes}|s) \cdot P(\text{basketball}|s) \\ &= 0.5 * 1/(11+14) * 1/(11+14) * 4/(11+14) = 0.000128 \end{aligned}$$

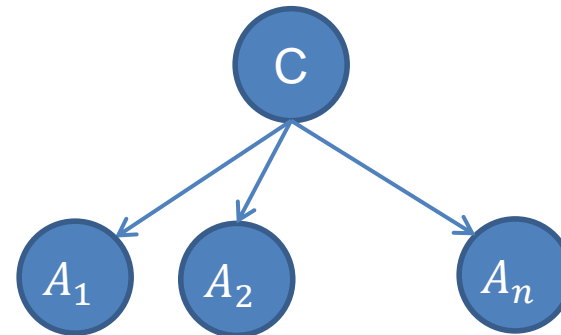
# Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
  - Use other techniques such as Bayesian Belief Networks (BBN)
- Naïve Bayes can produce a probability estimate, but it is usually a very biased one
  - Logistic Regression is better for obtaining probabilities.

# Generative vs Discriminative models

- Naïve Bayes is a type of a **generative model**
  - Generative process:
    - First pick the category of the record
    - Then given the category, generate the attribute values from the distribution of the category

- Conditional independence given C



- We use the training data to learn the distribution of the values in a class

# Generative vs Discriminative models

- Logistic Regression and SVM are **discriminative models**
  - The goal is to find the boundary that discriminates between the two classes from the training data
- In order to classify the language of a document, you can
  - Either learn the two languages and find which is more likely to have generated the words you see
  - Or learn what differentiates the two languages.

# SUPERVISED LEARNING

---

# Learning

- **Supervised Learning**: learn a model from the data using **labeled data**.
  - **Classification** and **Regression** are the prototypical examples of supervised learning tasks. Other are possible (e.g., ranking)
- **Unsupervised Learning**: learn a model – extract structure from **unlabeled data**.
  - **Clustering** and **Association Rules** are prototypical examples of unsupervised learning tasks.
- **Semi-supervised Learning**: learn a model for the data using both **labeled and unlabeled data**.

# Supervised Learning Steps

- **Model** the problem
  - What is you are trying to predict? What kind of optimization function do you need? Do you need classes or probabilities?
- **Extract Features**
  - How do you find the right features that help to discriminate between the classes?
- **Obtain training data**
  - Obtain a collection of labeled data. Make sure it is large enough, accurate and representative. Ensure that classes are well represented.
- **Decide on the technique**
  - What is the right technique for your problem?
- **Apply** in practice
  - Can the model be trained for very large data? How do you test how you do in practice? How do you improve?

# Modeling the problem

- Sometimes it is not obvious. Consider the following three problems
  - Detecting if an email is spam
  - Categorizing the queries in a search engine
  - Ranking the results of a web search
  - Predicting the reply to a question.



# Feature extraction

- Feature extraction, or **feature engineering** is the most tedious but also the most important step
  - How do you separate the players of the Greek national team from those of the Swedish national team?
- One line of thought: throw features to the classifier and the classifier will figure out which ones are important
  - **More features**, means that you need **more training data**
- Another line of thought: **Feature Selection**: Select carefully the features using various functions and techniques
  - Computationally intensive
- Deep Neural Networks
  - They use raw data for classification
  - They learn a representation from the data

# Training data

- An overlooked problem: How do you get **labeled data** for training your model?
  - E.g., how do you get training data for ranking?
  - Chicken and egg problem
- Usually requires a lot of manual effort and domain expertise and carefully planned labeling
  - Results are not always of high quality (lack of expertise)
  - And they are not sufficient (low coverage of the space)
- Recent trends:
  - Find a **source** that generates the labeled data for you, or use the data themselves for the prediction task
  - **Crowd-sourcing** techniques

# Dealing with small amounts of labeled data

- **Semi-supervised learning** techniques have been developed for this purpose.
- **Self-training**: Train a classifier on the data, and then feed back the high-confidence output of the classifier as input
- **Co-training**: train two “independent” classifiers and feed the output of one classifier as input to the other.
- **Regularization**: Treat learning as an optimization problem where you define relationships between the objects you want to classify, and you exploit these relationships
  - Example: Image restoration

# Technique

- The choice of technique depends on the problem requirements (do we need a probability estimate?) and the problem specifics (does independence assumption hold? do we think classes are linearly separable?)
- For many cases finding the right technique may be trial and error
- For many cases the exact technique does not matter.

# Big Data Trumps Better Algorithms

- If you have enough data then the algorithms are not so important
- The web has made this possible.
  - Especially for text-related tasks
  - Search engine uses the **collective human intelligence**

Google lecture: [Theorizing from the Data](#)

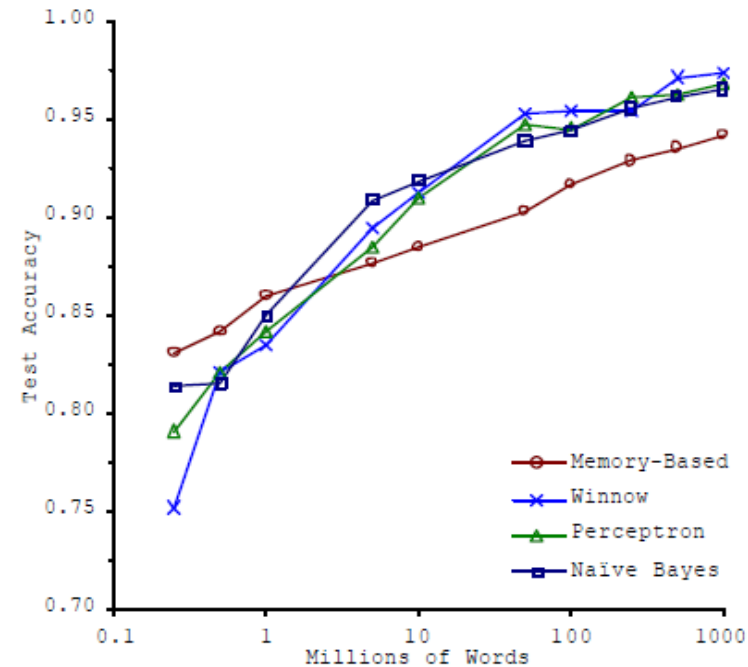


Figure 1. Learning Curves for Confusion Set Disambiguation

# Apply-Test

- How do you **scale** to very large datasets?
  - Distributed computing – **map-reduce** implementations of machine learning algorithms (Mahout, over Hadoop)
- How do you test something that is running online?
  - You cannot get labeled data in this case
  - **A/B testing**
- How do you deal with changes in data?
  - **Active learning**