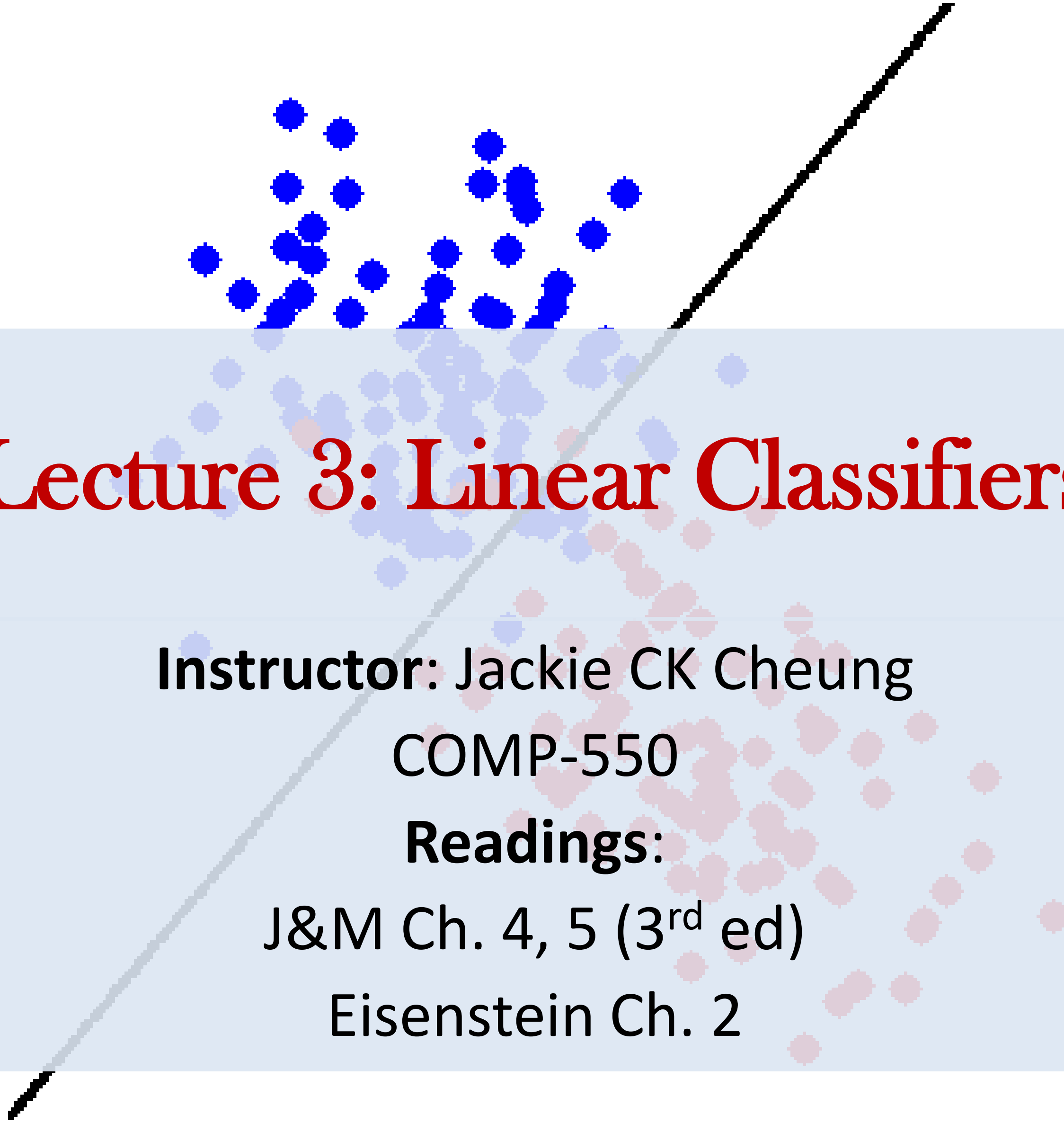


Lecture 3: Linear Classifiers

Instructor: Jackie CK Cheung

COMP-550

Readings:

J&M Ch. 4, 5 (3rd ed)

Eisenstein Ch. 2

Classification

Map input x to output y:

$$y = f(x)$$

Classification: y is a discrete outcome

- Genre of the document (news text, novel, …?)
- Overall topic of the document
- Spam vs. non-spam
- Identity, gender, native language, etc. of author
- Positive vs. negative movie review
- Other examples?

Steps in Building a Text Classifier

1. Define problem and collect data set
2. Extract features from documents
3. **Train a classifier on a training set [today]**
4. Apply classifier on test data

Feature Extraction

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

document label

classifier

document

Represent document $\vec{x}$ as a list of features

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

$$\begin{array}{cccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \ldots \\ 1.0, & 0.0, & 1.0, & 1.0, & 0.0, & 0.0, & 0.0, & 1.0 \ldots \end{array}$$

Think Abstractly

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

document label

classifier

document

What are possible choices for the form of f?

Some popular approaches:

- Naïve Bayes
- Logistic regression
- Support vector machines
- Artificial neural networks – nonlinear, for next class

Training

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

Say we select an architecture (e.g., Naïve Bayes). f can now be described in terms of parameters θ:

$$y = f(\vec{x}; \theta)$$

Training the model specifically means to select parameters θ^* according to some objective function (e.g., minimize error on training set; maximize likelihood of training data).

Naïve Bayes

A probabilistic classifier that uses **Baye's rule**

$$P(y|\vec{x}) = P(y)P(\vec{x}|y) \, / \, P(\vec{x})$$

Naïve Bayes is a **generative** model

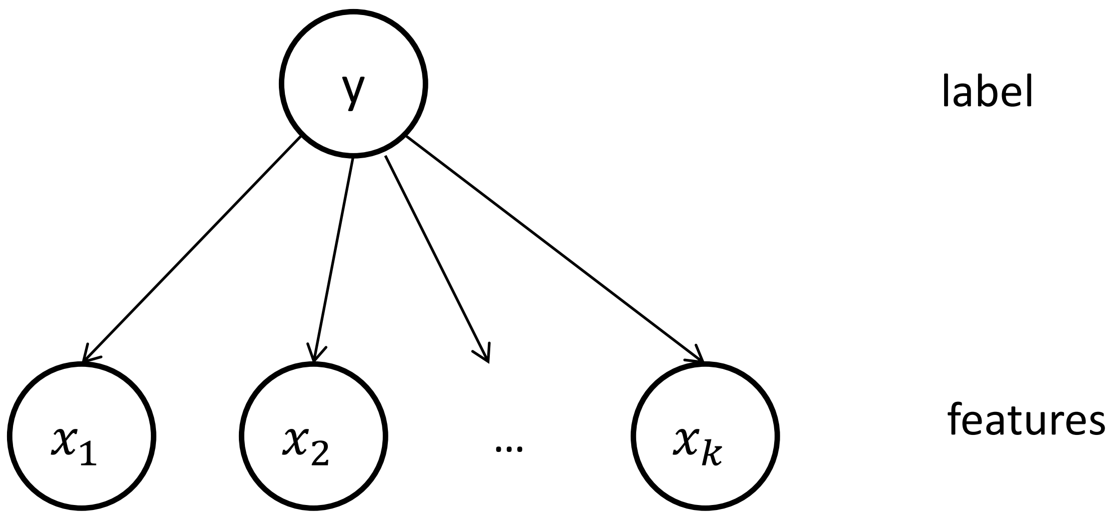
- Probabilistic account of the data $P(\vec{x}, y)$
- Naïve Bayes assumes the dataset is generated in the following way:

For each sample:

1. Generate label by $P(y)$
2. Generate feature vector $\vec{x}$ by generating each feature **independently**, conditioned on y
    - $P(x_i|y)$

Naïve Bayes Graphically

Assumption about how data is generated, as a probabilistic graphical model:



$$P(\vec{x}, \mathrm{y}) = P(\mathrm{y}) \prod_i P(x_i | y)$$

Note how the independence between features is expressed!

Naïve Bayes Model Parameters

The parameters to the model, θ, consist of:

- Parameters of prior class distribution $P(\mathrm{y})$
- Parameters of each feature's distribution conditioned on class $P(x_i|y)$

With discrete data, we assume that the distributions $P(\mathrm{y})$ and $P(x_i|y)$ are **categorical distributions**

Reminder: Categorical Distribution

A categorical **random variable** follows this distribution if it can take one of *k* outcomes, each with a certain probability

- The probabilities of the outcomes must sum to 1

Examples:

- Coin flip (*k* = 2; Bernoulli distribution)
- Die roll (*k* = 6)
- Distribution of class labels (e.g., spam vs non-spam, *k* = number of classes)
- Generating unigrams! (*k* = size of vocabulary)

Training a Naïve Bayes Classifier

Objective: pick θ such as to maximize the **likelihood** of the training corpus, D:

$$L^{NB}(\theta) = \prod_{(\vec{x},y)\in D} P(\vec{x}, y; \theta)$$

$$= \prod_{(\vec{x},y)\in D} P(\mathrm{y}) \prod_i P(x_i|y)$$

Can show that this boils down to computing relative frequencies:

$P(Y = y)$ should be set to proportion of samples that with class y

$P(X_i = x|Y = y)$ should be set to proportion of samples with feature value x among samples of class y

Inference in Naïve Bayes

After training, we would like to classify a new instance (e.g., is a new document spam)

- i.e., want $P(y|\vec{x})$

Easy to get from $P(\vec{x}, y)$:

$$
\begin{aligned}
P(y|\vec{x}) &= P(\vec{x}, y)/P(\vec{x}) \\
&= P(y) \prod_i P(x_i|y) \;/\; P(\vec{x})
\end{aligned}
$$

To calculate denominator $P(\vec{x})$, **marginalize** over random variable y by summing up numerator for all possible classes (all possible values of y).

Naïve Bayes in Summary

Bayes' rule:

$$P(y|\vec{x}) = P(y)P(\vec{x}|y) \;/\; P(\vec{x})$$

Assume that all the features are independent:

$$P(y|\vec{x}) = P(y)\prod_i P(x_i|y) \;/\; P(\vec{x})$$

Training the model means estimating the parameters $P(y)$ and $P(x_i|y)$.

- e.g., P(SPAM) = 0.24, P(NON-SPAM) = 0.76

  P(*money at home*|SPAM) = 0.07

  P(*money at home*|NON-SPAM) = 0.0024

Exercise: Train a NB Classifier

Table of whether a student will get an A or not based on their habits (nominal data, Bernoulli distributions):

Reviews notes	Does assignments	Asks questions	Grade
Y	N	Y	A
Y	Y	N	A
N	Y	N	A
Y	N	N	non-A
N	Y	Y	non-A

What is the probability that this student gets an A?

- Doesn't review notes, does assignments, asks questions

$$P(y|\vec{x}) = P(y) \prod_i P(x_i|y) \,/\, P(\vec{x})$$

Type/Token Distinction

What if a word appears more than once in a document? Frequency matters!

Type the identity of a word (i.e., count unique words)

Token an instance of a word (i.e., each occurrence is separate)

In text classification, we usually deal with tokens, and assume that there is a categorical distribution that is used to generate all of the tokens seen in a sample, conditioned on class y.

yo buy my stuff yo class: spam

P(spam)P(yo|spam)P(my|spam)P(stuff|spam)P(yo|spam)

Generative vs. Discriminative

Generative models learn a distribution for all of the random variables involved: *joint* distribution, $P(\vec{x}, y)$

But for text classification, we really only care about the *conditional* distribution $P(y|\vec{x})$!

Discriminative models directly parameterize and learn $P(y|\vec{x})$

- May be easier than learning the joint!
- Can flexibly design many different features
- Model can only do classification!

Logistic Regression

Linear regression:

$$y = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n + b$$

Intuition: Linear regression gives as continuous values in [-∞, ∞] —let's squish the values to be in [0, 1]!

Function that does this: logit function

$$P(y|\vec{x}) = \frac{1}{Z} e^{a_1 x_1 + a_2 x_2 + \ldots + a_n x_n + b}$$

This Z is a normalizing constant to ensure this is a probability distribution.

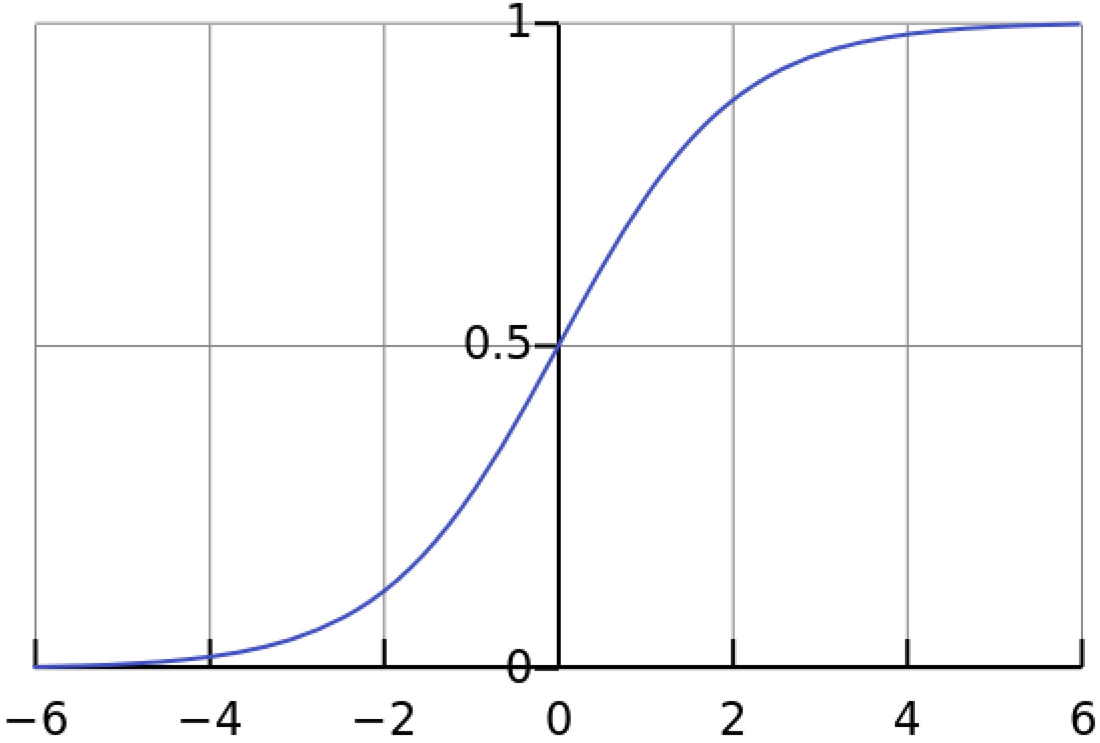
(a.k.a., maximum entropy or MaxEnt classifier)

N.B.: Don't be confused by name—this method is most often used to solve classification problems.

Logistic Function

y-axis: $P(y|\vec{x}) = \frac{1}{Z} e^{a_1 x_1 + a_2 x_2 + \ldots + a_n x_n + b}$

x-axis: $a_1 x_1 + a_2 x_2 + \ldots + a_n x_n + b$



Features Can Be Anything!

We don't have to care about generating the data, so can go wild in designing features!

- Does the document start with a capitalized letter?
- What is the length of the document in words? In sentences?
  - Actually, would usually scale and/or bin this
- How many sentiment-bearing words are there?

In practice, the features depend on both the document and the proposed class:

- Does the document contain the word *money* with the proposed class being *spam*?

Parameters in Logistic Regression

$$P(y|\vec{x};\theta) = \frac{1}{Z} e^{a_1 x_1 + a_2 x_2 + \ldots + a_n x_n + b}$$

where, $\theta = \{a_1, a_2, \ldots, a_n, b\}$

Learning means to maximize the **conditional likelihood** of the training corpus

$$L^{LR}(\theta) = \prod_{(\vec{x},y)\in D} P(y|\vec{x};\theta)$$

or more usually, the log conditional likelihood

$$\log L^{LR}(\theta) = \sum_{(\vec{x},y)\in D} \log P(y|\vec{x};\theta)$$

Optimizing the Objective

We want to maximize

$$\log L^{LR}(\theta) = \sum_{(\vec{x},y)\in D} \log P(y|\vec{x};\theta)$$

$$= \sum_{(\vec{x},y)\in D} \log(\frac{1}{Z} e^{a_1x_1 + a_2x_2 + \ldots + a_nx_n + b})$$

$$= \sum_{(\vec{x},y)\in D} (\sum_i a_ix_i - \log Z)$$

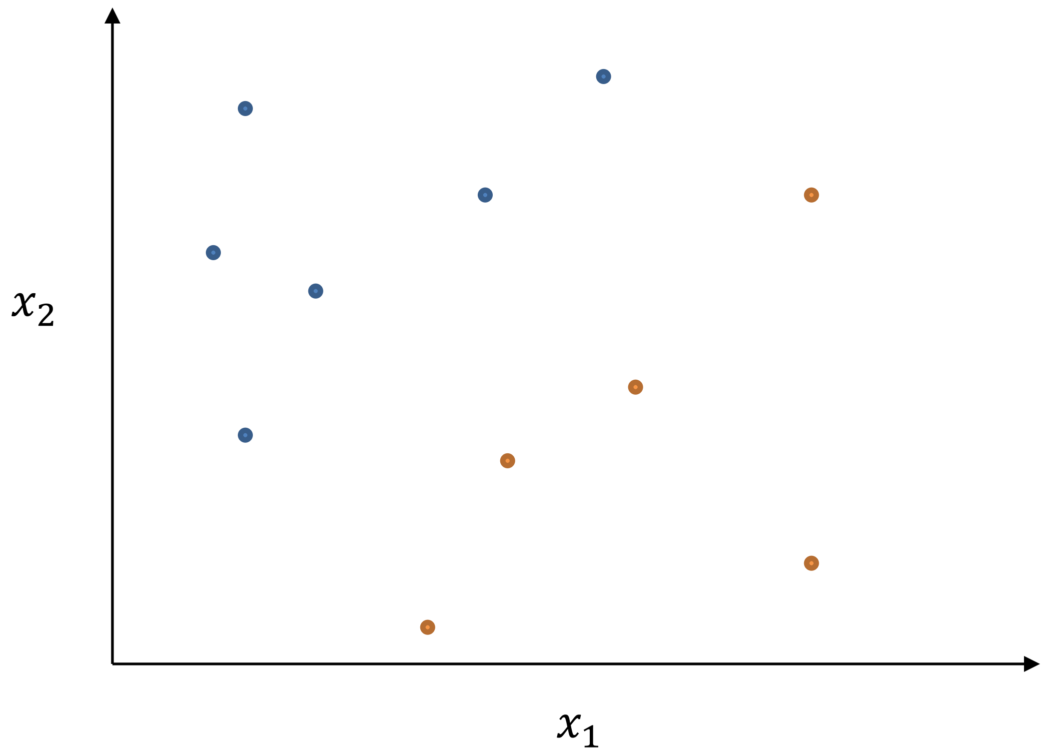
This can be optimized by **gradient descent**

More on this in a future class

Support Vector Machines

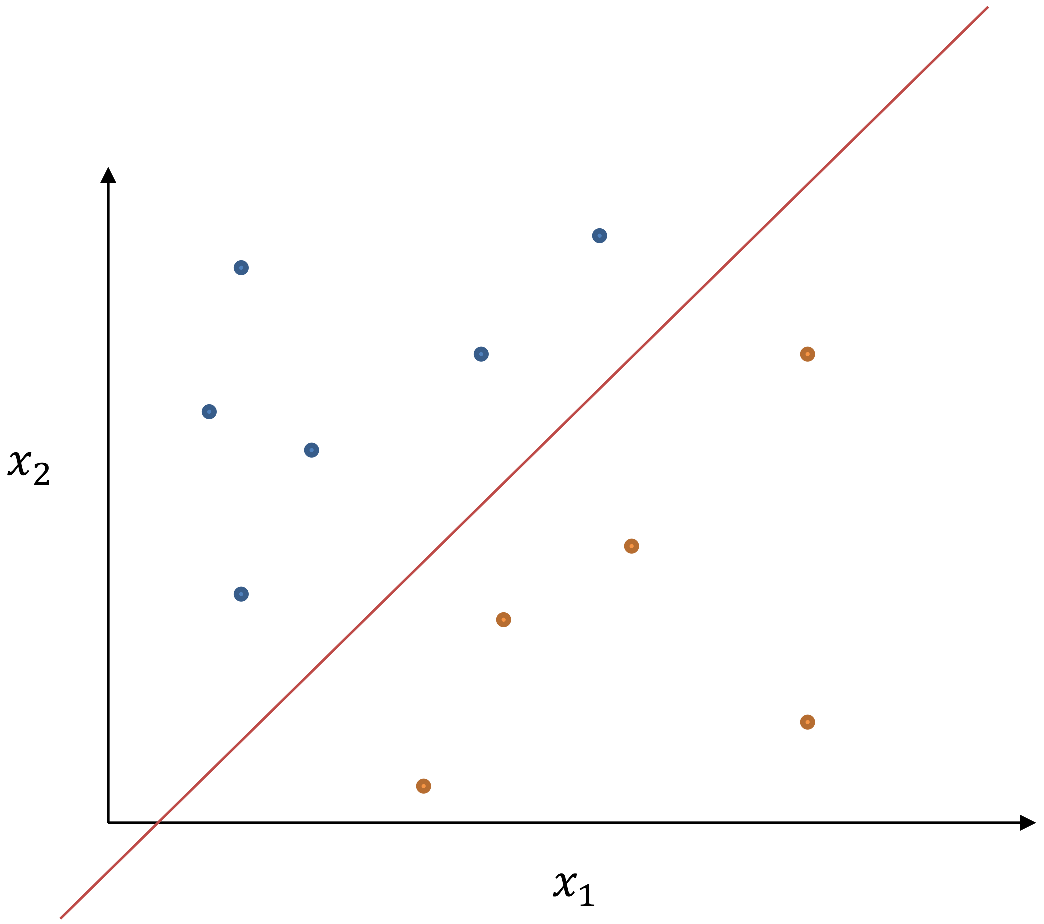
Let's visualize $\vec{x}$ as points in a high dimensional space.

e.g., if we have two features, each sample is a point in a 2D scatter plot. Label y using colour.



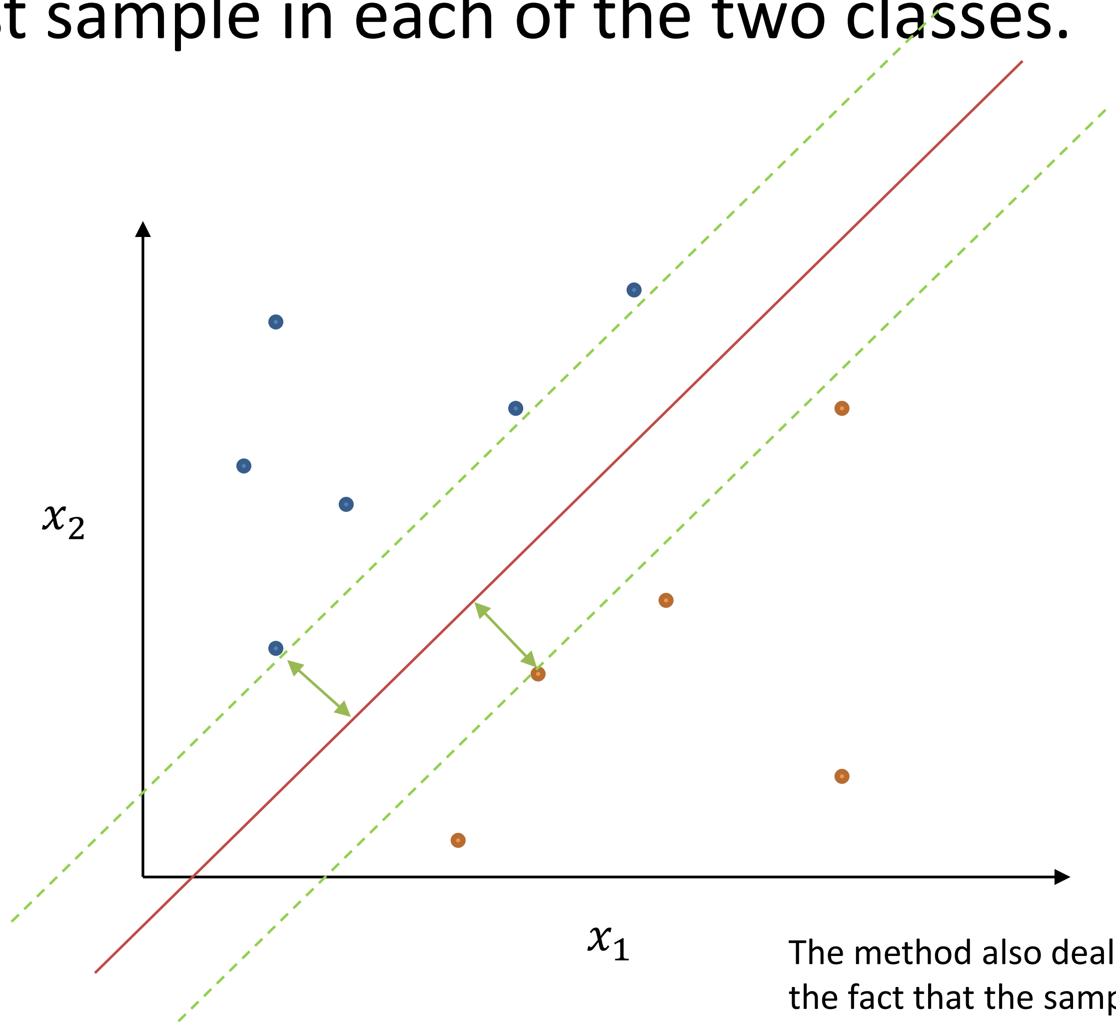
Support Vector Machines

A SVM learns a decision boundary as a line (or hyperplane when >2 features)



Margin

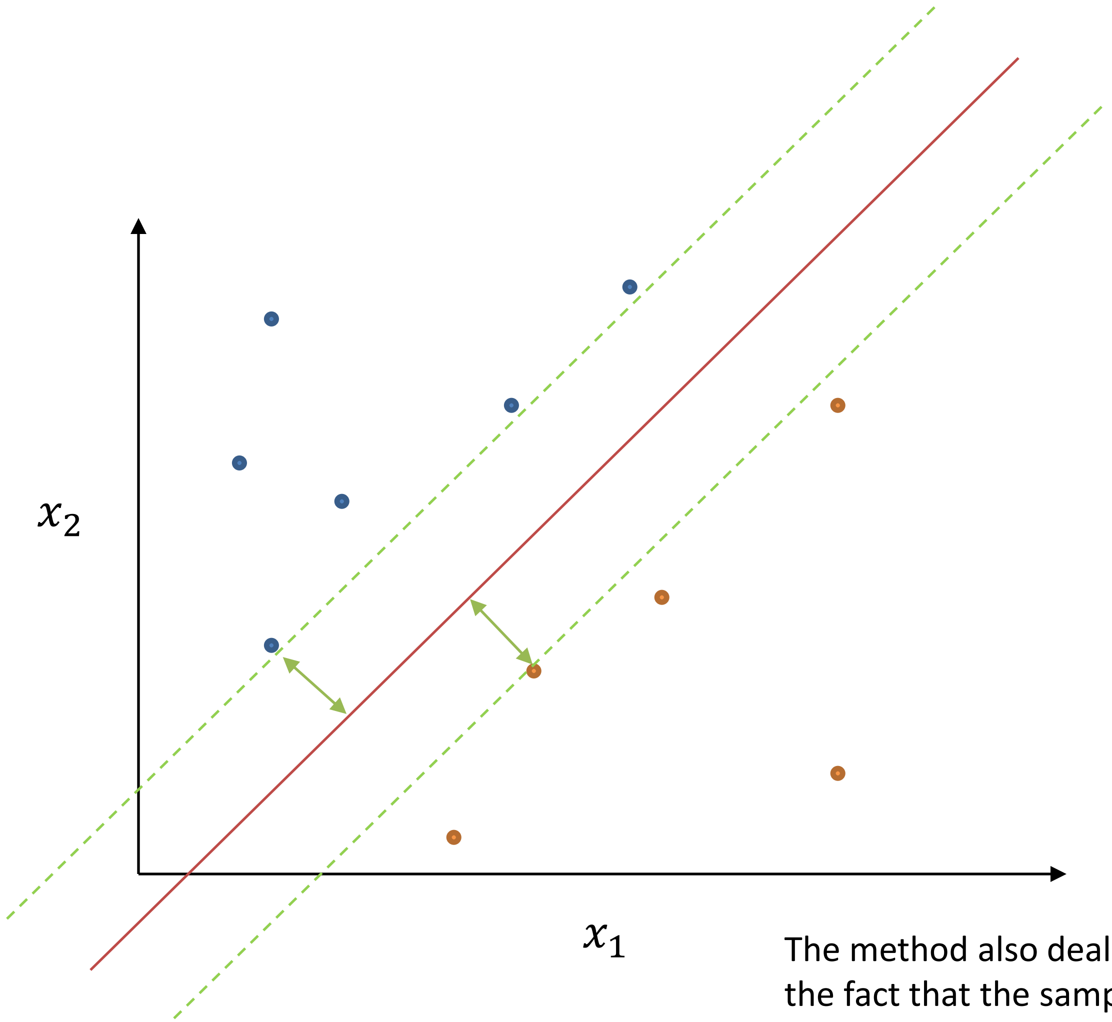
This hyperplane is chosen to maximize the margin to the nearest sample in each of the two classes.



The method also deals with the fact that the samples may not be linearly separable.

SVMs – Generative or Discriminative?

Are SVMs a generative or a discriminative model?



The method also deals with the fact that the samples may not be linearly separable.

How To Decide?

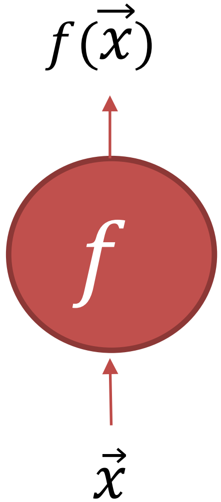
- Naïve Bayes, logistic regression, and SVMs can all work well in different tasks and settings.
- Usually, given little training data, Naïve Bayes are a good bet—strong independence assumptions.
- In practice, try them all and select between them on a development set!

Perceptron

Closely related to logistic regression (differences in training and output interpretation)

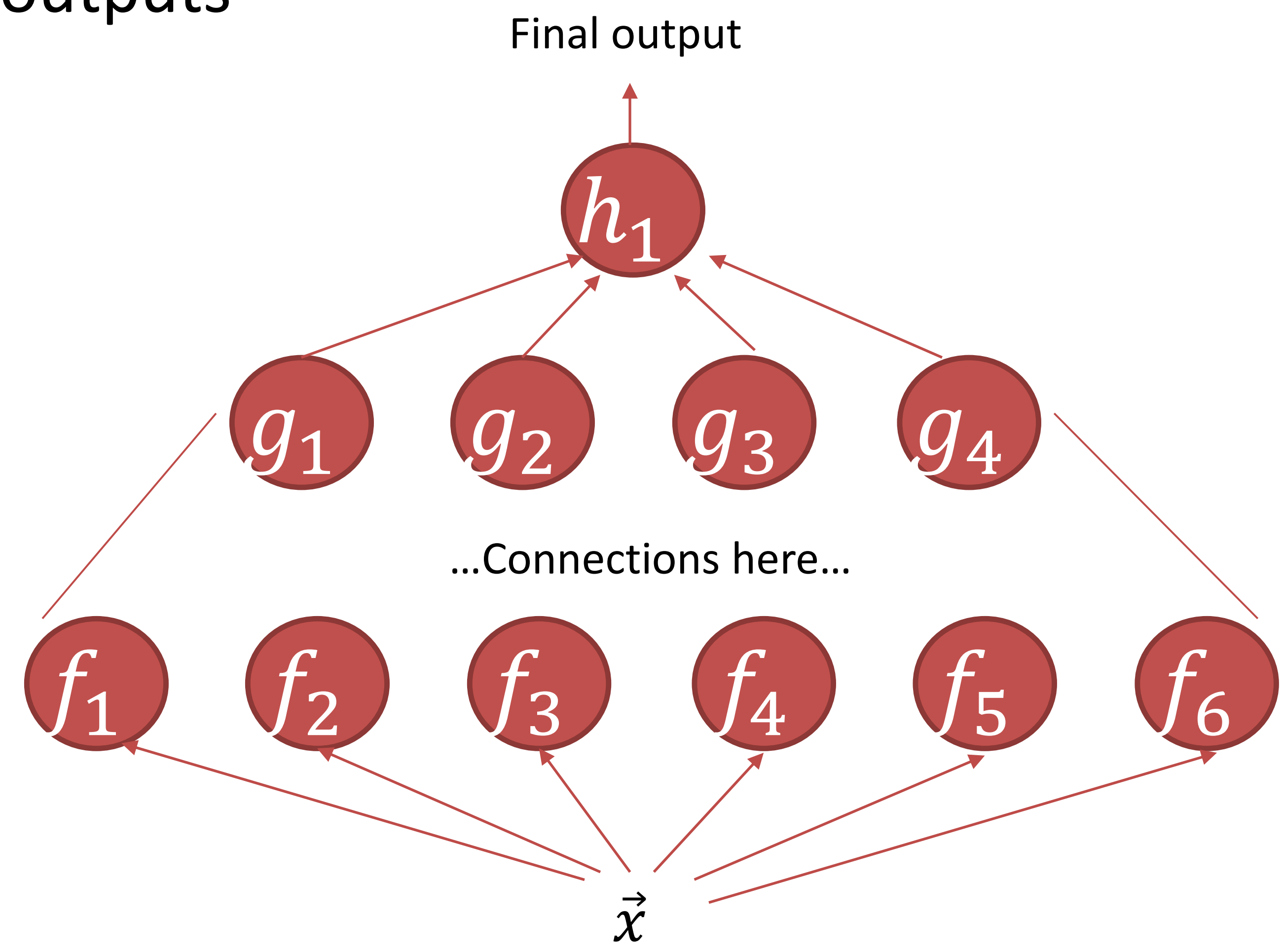
$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Let's visualize this graphically:



Stacked Perceptrons

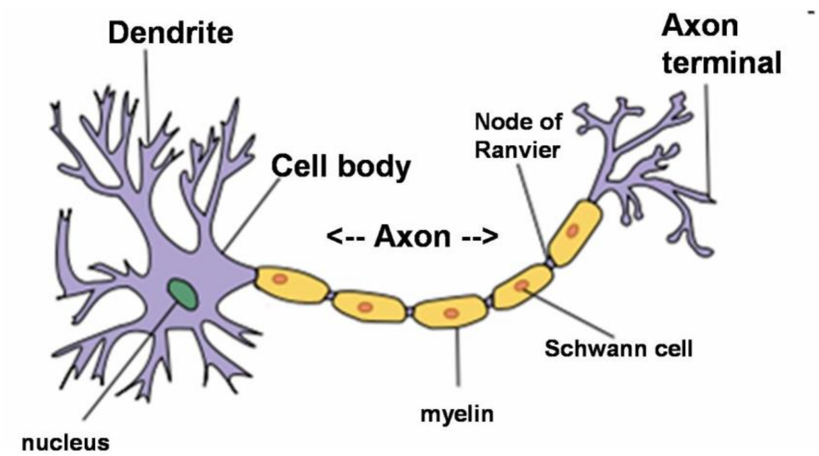
Let's have multiple units, then stack and recombine their outputs



Artificial Neural Networks

Above is an example of an **artificial neural network**:

- Each unit is a neuron with many inputs (dendrites) and one output (axon)
- The nucleus fires (sends an electric signal along the axon) given input from other neurons.
- Learning occurs at the synapses that connect neurons, either by amplifying or attenuating signals.



Artificial Neural Networks

Advantages:

- Can learn very complex functions
- Many possible different network structures possible
- Given enough training data, are currently achieving the best results in many NLP tasks

Disadvantages:

- Training can take a long time
- Often need a lot of training data to work well

Even More Classification Algorithms

Read up on them or ask me if you're interested:

- k-nearest neighbour
- decision trees
- transformation-based learning
- random forests