Feature Extraction and Classification

COMP-599

Sept 19, 2016

Good-Turing Smoothing Defined

Let N be total number of observed word-tokens, w_c be a word that occurs c times in the training corpus.

$N = \sum_i f_i \times i$

$P(UNK) = f_1 / N$ ← Note that this is for **all** OOV words

Then: $c^* = \frac{(c+1) f_{c+1}}{f_c}$

$P(w_c) = c^* / N$ ← Note that this is for **one** word that occurs c times

Example:

Let N be 100,000.

Word frequency	# word-types
1	f_1 = 3,993
2	f_2 = 1,292
3	f_3 = 664
…	…

P(UNK) = 3993 / 100000
= 0.03993
(for all unknown words)

c_1^* = 2 * 1292 / 3993
= 0.647

c_2^* = 3 * 664 / 1292
= 1.542

Exercises

Suppose we have the following counts:

Word	ship	pass	camp	frock	soccer	mother	tops
Freq	8	7	3	2	1	1	1

Give the MLE and Good-Turing estimates for the probabilities of:

- any unknown word
- *soccer*
- *camp*

Outline

Machine learning basics

- Supervised vs. unsupervised methods
- Classification vs. regression

Document classification

- Feature extraction—N-grams again!
- Common classification methods

Machine Learning for NLP

Language modelling: our first example of statistical modelling in NLP

It is important to cover some basic terminology and distinctions in machine learning.

Common research paradigm:

- Find interesting NLP problem from language data or need
- Formulate NLP problem as machine learning problem
- Solve problem by using machine learning techniques

This Class

Will be a review if you have already taken a machine learning course.

Will go by very quickly if you haven't. Focus on:

- basic terminology and distinctions between different kinds of methods
- names of popular techniques and an intuitive understanding of how they work

You can read up on any technique that you find interesting in further detail.

Supervised vs. Unsupervised Learning

How much information do we give to the machine learning model?

Supervised – model has access to some input data, and their corresponding output data (e.g., a label)

- Learn a function $y = f(x)$, given examples of (x, y) pairs

input

output

model

Unsupervised – model only has the input data

- Given only examples of x, find some interesting patterns in the data

Supervised Learning

1. Given examples, predict the **part of speech** (POS) of a word
    - *run* is a verb (or a noun)
    - *ran* is a verb
    - *cat* is a noun
    - *the* is a determiner
2. Predict whether an e-mail is spam or non-spam (given examples of spam and non-spam e-mails)

What Does Learning Mean?

Determining what the function f(x) should be, given the data.

- i.e., find parameters to the model θ that minimize some kind of **loss** or **error** function
- For example, the model should minimize the number of incorrectly classified pairs in the training set.

Unsupervised Learning

Find hidden structure in the data without any labels. Think of this as **clustering**.

1. Grammar induction
    - *the* and *a* seem to appear in similar contexts
    - *very* and *hope* don't appear in similar contexts
    - Cluster *the* and *a* into the same POS, *very* and *hope* into different ones
2. Learning word relatedness
    - *cat* and *dog* are related words with similarity score 0.81
    - *good* and *bad* are related words with similarity score 0.56

What Does Learning Mean?

Coming up with a good characterization of the data.

- In non-probabilistic models, define some similarity and/or clustering algorithm that make sense
- In probabilistic models, find the parameters to the model θ that maximize the probability of the training corpus $P(X;\theta)$

Semi-Supervised Learning

You have the outputs for some of the inputs, but not all of them

> e.g., I have the POS tags for some of the sentences in my corpus, but not for most of them.

Learning means to find a model that fits both the cases where we have the output label, and the cases where we don't.

Grey Area 1: Specify Rules

Examples:

- e.g., label the first word of each sentence as a determiner
- a noun may only follow a determiner
- anything ending in *–ed* is a verb

Often combined with further clustering or other training, because there tend to be many exceptions to rules.

Variously called **semi-supervised**, **distantly supervised**, **minimally supervised**, or simply **unsupervised**.

Grey Area 2: Give Seed Set

Similar to above: give a set of seeds for the categories to be learned, then perform further training to propagate the labels

e.g., learn a sentiment lexicon

- Positive seeds: *good, awesome, magnificent, great*
- Negative seeds: *bad, horrible, awful, dreadful*

- Label words that are similar to positive seeds as positive; words that are similar to negative seeds as negative

Language Modelling

Predict the next word given some context

Mary had a little _____

• *lamb*	GOOD
• *accident*	GOOD?
• *very*	BAD
• *up*	BAD

Is this a supervised or unsupervised machine learning problem?

- (You're not allowed to answer if you've taken a machine learning course before.)

Another Dimension

Within supervised learning, another distinction is between **classification** and **regression**.

$$y = f(x)$$

- **Regression**: y is a continuous outcome

  e.g., similarity score of 3.5
- **Classification**: y is a discrete outcome

  e.g., spam vs. non-spam, verb vs. noun vs. adjective, etc.

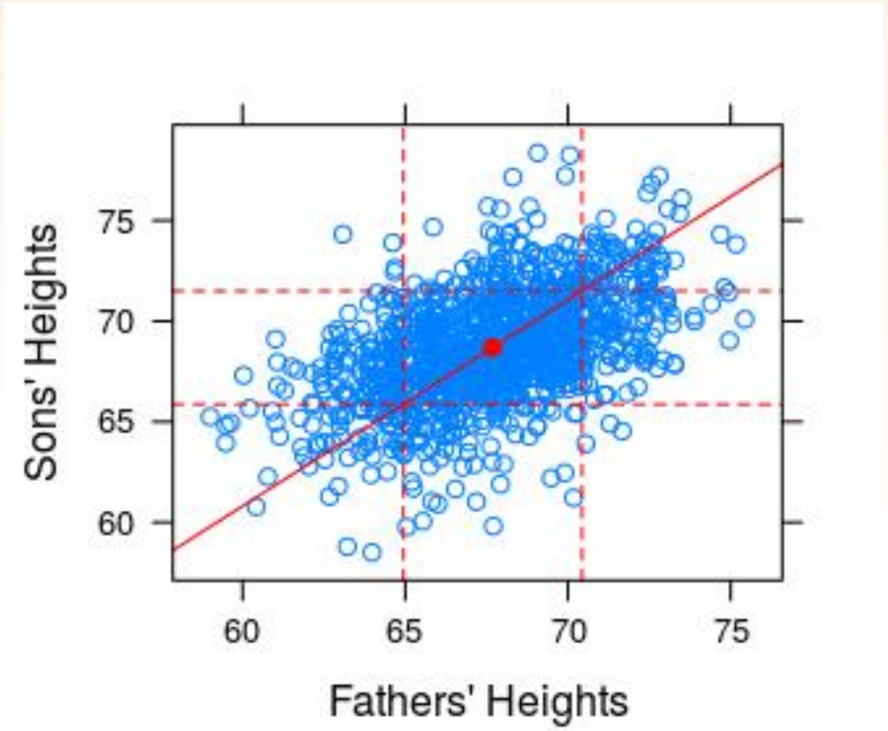
Linear Regression

The function is linear:

$$y = a_1x_1 + a_2x_2 + \ldots + a_nx_n + b$$

Line of best fit:

- Galton plotted son's height to father's height



Classification

Most NLP work involving text end up being classification problems.

Linguistic units of interest are often discrete:

• words:	*apple, banana, orange*
• POS tags:	NOUN, VERB, ADJECTIVE
• semantic categories	AGENT, PATIENT, EXPERIENCER
• discourse relations	EXPLANATION, CAUSE, ELABORATION

Document Classification

Determine some discrete property of a document

- 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

1. Define problem and collect data set
2. Extract features from documents
3. Train a classifier on a training set
4. Actually, train multiple classifiers using a training set; do model selection by tuning hyperparameters on a development set
5. Use your final model to do classification on the test set

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.

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

Feature Extraction and Classification

We can use these feature vectors to train a classifier

Training set:

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8 …	y
1.0,	0.0,	1.0,	1.0,	0.0,	0.0,	1.0,	1.0 …	1
1.0,	0.0,	0.0,	1.0,	1.0,	0.0,	0.0,	1.0 …	1
0.0,	1.0,	1.0,	1.0,	0.0,	1.0,	0.0,	1.0 …	0
0.0,	1.0,	1.0,	1.0,	0.0,	1.0,	0.0,	0.0 …	0
0.0,	0.0,	0.0,	0.0,	1.0,	0.0,	0.0,	1.0 …	1
0.0,	0.0,	1.0,	1.0,	0.0,	1.0,	0.0,	0.0 …	1
				…				
1.0,	1.0,	1.0,	1.0,	0.0,	1.0,	1.0,	1.0 …	0

Testing:

1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0 … ?

N-grams

- Very popular
- Called “bag-of-words” (if unigrams), or “bag-of-n-grams”

Versions:

- Presence or absence of an N-gram (1 or 0)
- Count of N-gram
- Proportion of the total document
- Scaled versions of the counts (e.g., discount common words like *the*, and give higher weight to uncommon words like *penguin*)

POS Tags

Sequences of POS tags are also popular as features – crudely captures syntactic patterns in text

Need to **preprocess** the documents for their POS tags

Most common tag set in English:

https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

Classification Models

Some popular methods:

- Naïve Bayes
- Support vector machines
- Logistic regression
- Artificial neural networks (multilayer perceptrons)

Naïve Bayes

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 MLE probability that this student gets an A?

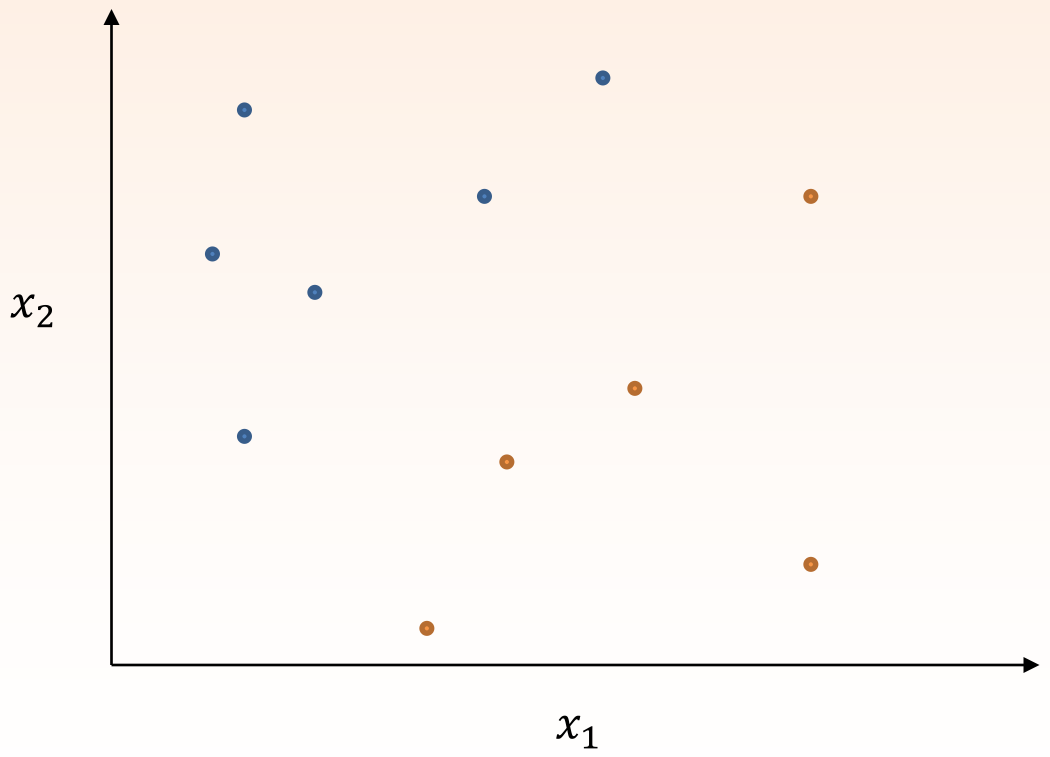
- Doesn't review notes, does assignments, asks questions

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

Support Vector Machines

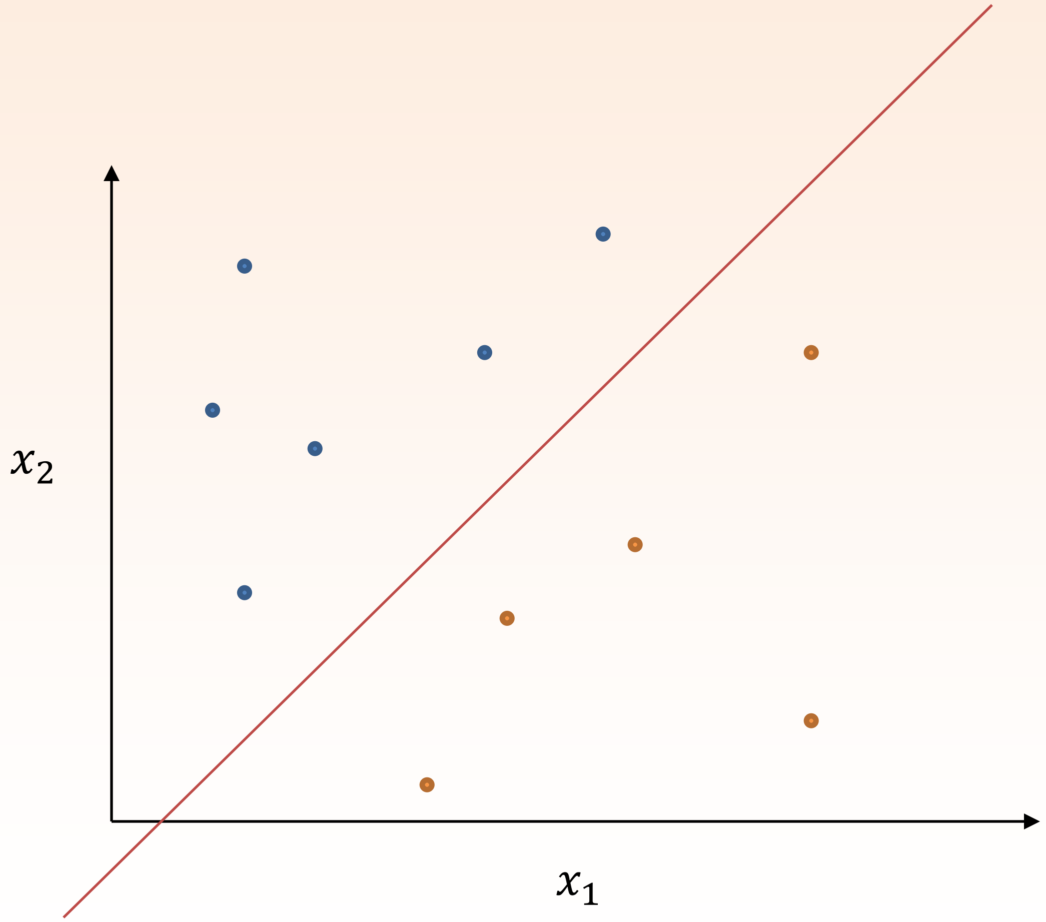
Let's visualize $\vec{x}$ as points in a high dimensional space.

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



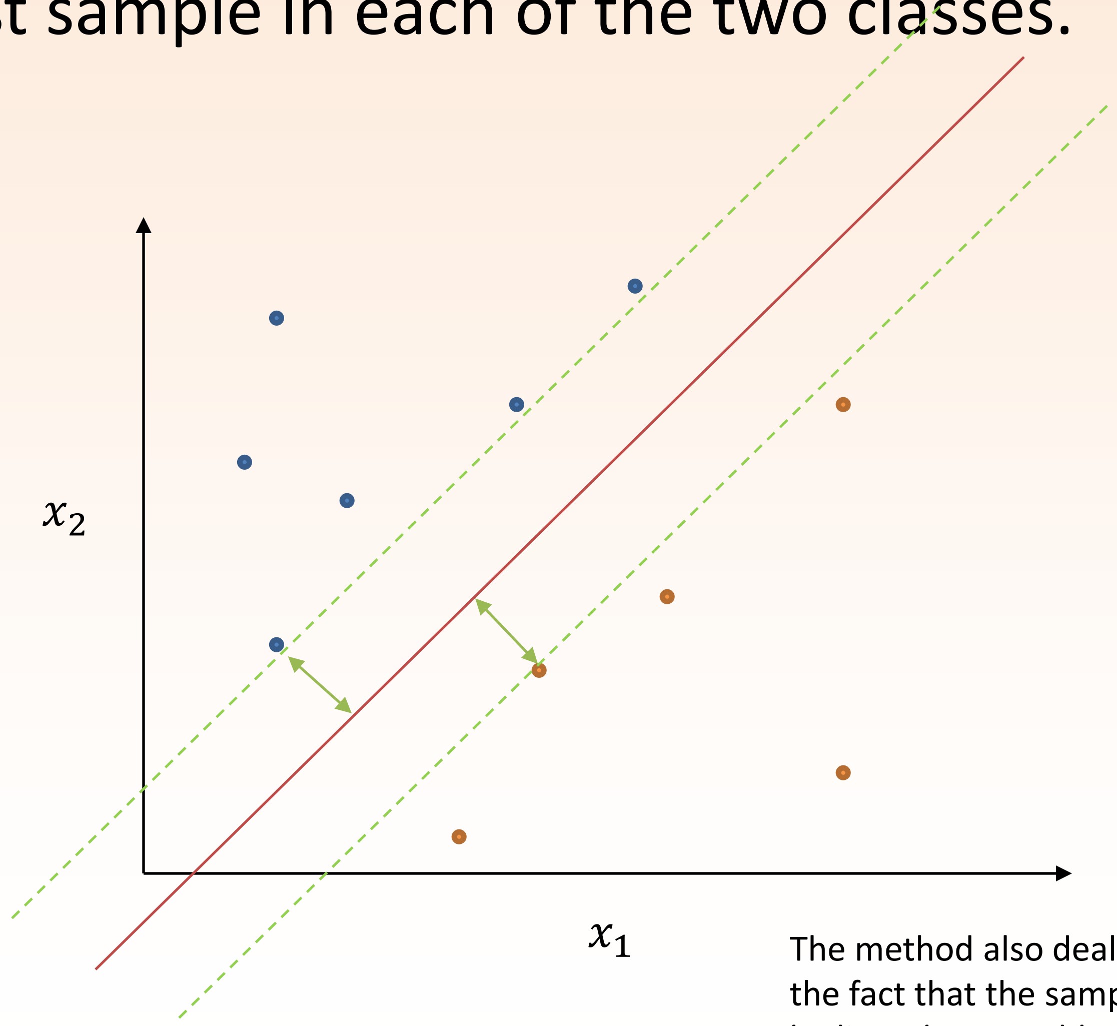
Support Vector Machines

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.

Logistic Regression

Linear regression:

$$y = a_1x_1 + a_2x_2 + \ldots + a_nx_n + b$$

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

Function that does this: logit function

$$P(y|\vec{x}) = \frac{1}{Z} e^{a_1x_1 + a_2x_2 + \ldots + a_nx_n + b}$$

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

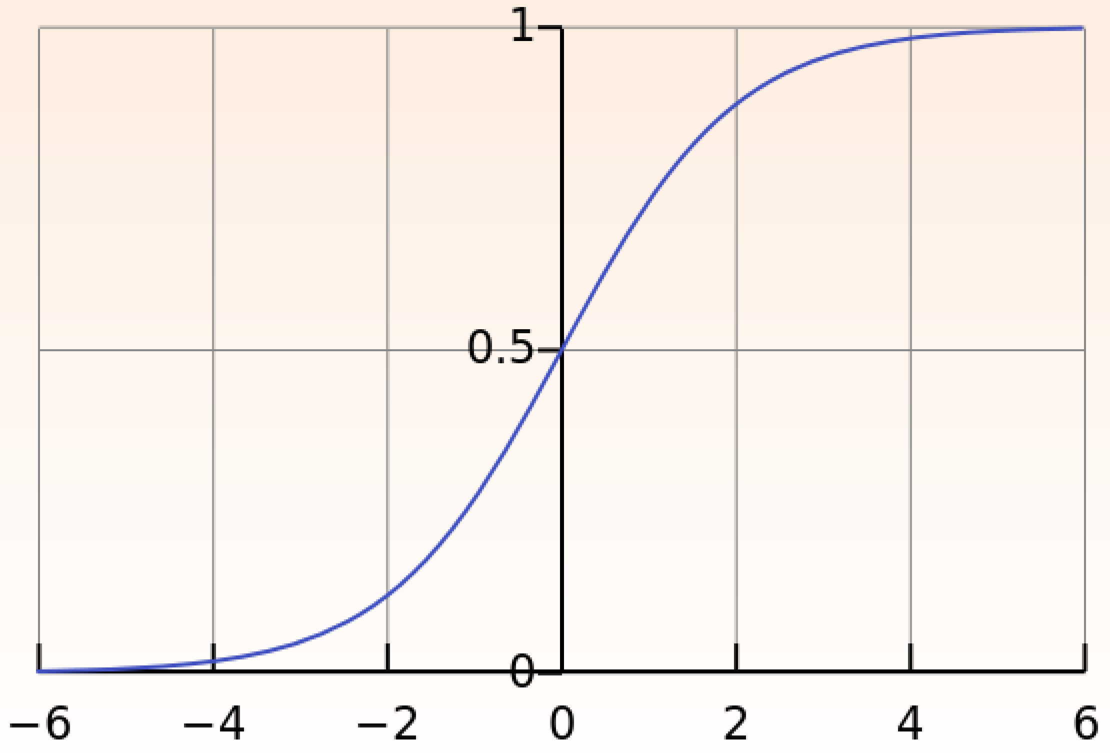
(a.k.a., maximum entropy or MaxEnt classifier)

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

Logistic Function

y-axis: $P(y|\vec{x})$

x-axis: $a_1x_1 \ + \ a_2x_2 \ + \ \ldots + \ a_nx_n \ + \ b$



How To Decide?

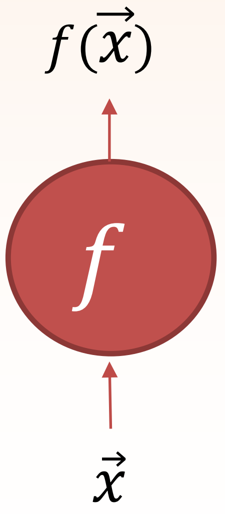
- Naïve Bayes, SVMs, and logistic regression can all work well in different tasks and settings.
- Usually, given little training data, Naïve Bayes are a good bet—strong independence assumptions.
- In practice, try them all and select between them on a development set!

Perceptron

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

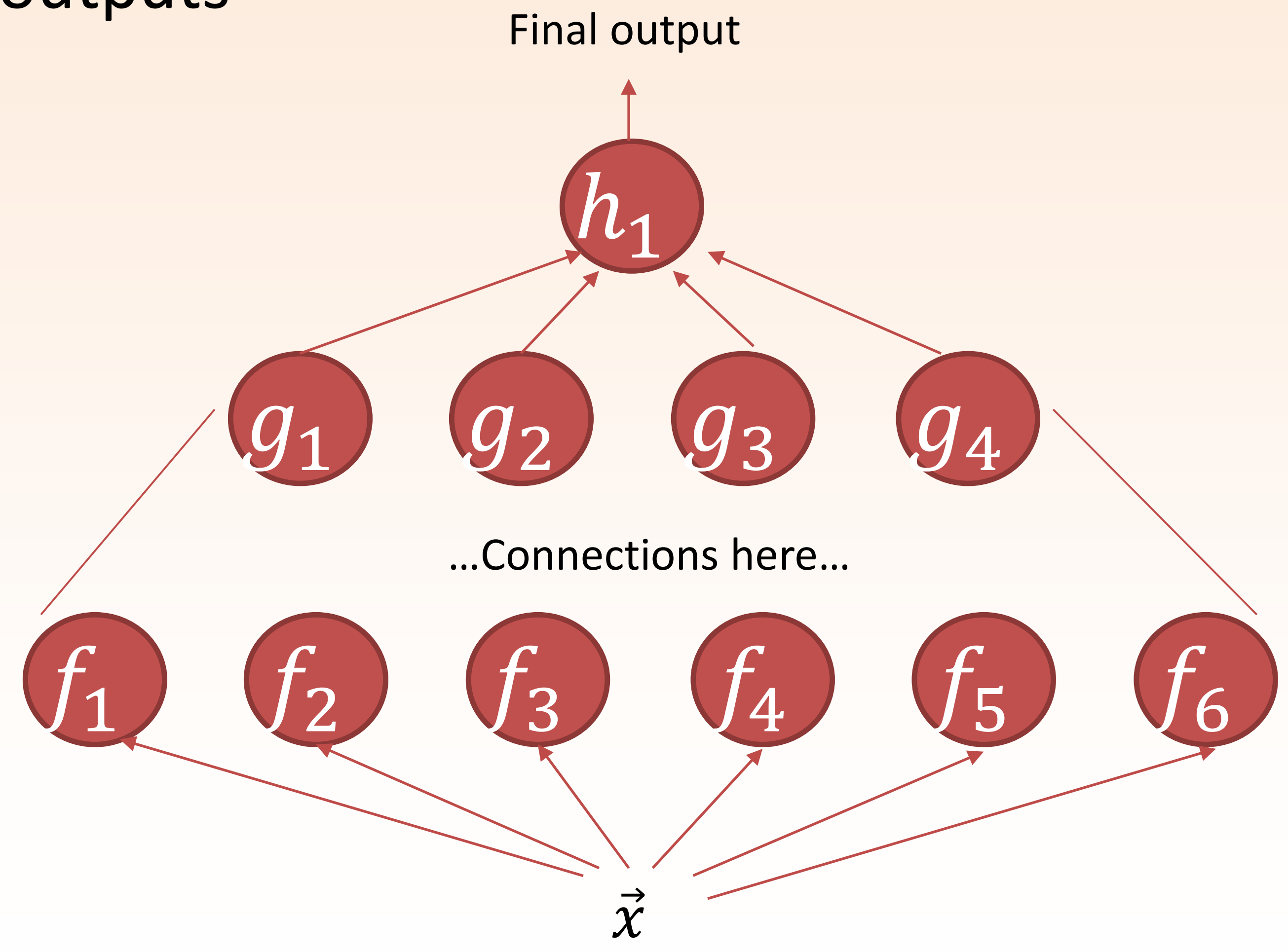
$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Let's visualize this graphically:



Stacked Perceptrons

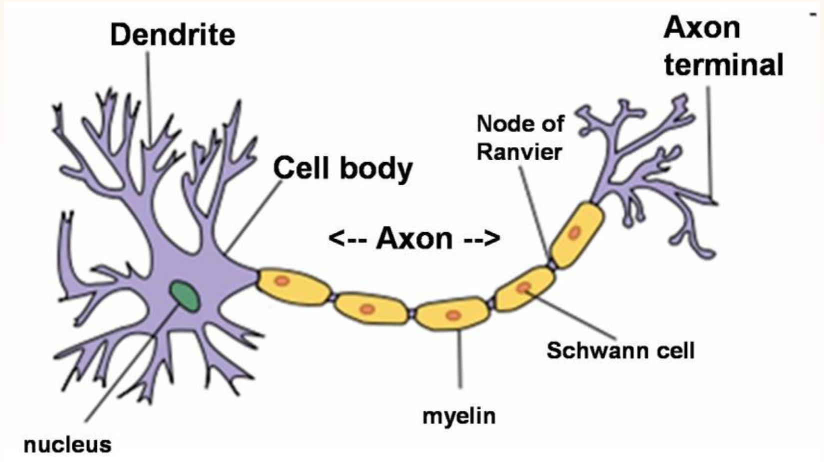
Let’s have multiple units, then stack and recombine their outputs



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

Supervised Classifiers in Python

scikits-learn has many simple classifiers implemented, with a common interface (See A1).

e.g., SVMs

```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
>>> clf.predict([[2., 2.]])
```

Confusion Matrix

It is often helpful to visualize the performance of a classifier using a confusion matrix:

Predicted class

Actual class

	C1	C2	C3	C4
C1	count	count	count	count
C2	count	count	count	count
C3	count	count	count	count
C4	count	count	count	count

Hopefully, most of the cases will fall into the diagonal entries!