Learning from Data
Lecture 2: Decision Tree, Vector Space Classification, K-nearest neighbor

Malvina Nissim
m.nissim@rug.nl
room 1311.421

21 November 2016

$$c_{map} = \arg\max_{c \in C} p(i|c) \cdot p(c)$$

Issues

- independence of features
- zeros and smoothing (Laplace or add-one smoothing)
- underflow

Issues

- independence of features
  features are often not independent, but we assume they are, otherwise calculations get too complicated. This is why it is called *naive*. Because we do, we can estimate conditional probability of each feature j separately, for a total of n features.

  $$c_{map} = \arg\max_{c \in C} \prod_{j=1}^{n} p(i_j|c) \cdot p(c)$$

  (a *Bayesian Network* doesn't include the independence assumption.)
- zeros and smoothing (Laplace or add-one smoothing)
- underflow

Issues

- independence of features
- zeros and smoothing (Laplace or add-one smoothing)
  it can happen that some values are zero. To prevent this problem in the calculations, the value 1 is added to all observed counts
- underflow

Issues

- independence of features
- zeros and smoothing (Laplace or add-one smoothing)
- underflow
  posterior probabilities are usually very very small, especially with lots of features (think of a bag-of-words approach in text classification). This is called the *underflow* problem
  For this reason, most implementations of a NB classifier are like this:

$$c_{map} = \arg\max_{c \in C} [\sum_{j=1}^{n} \log p(i_j|c) + \log p(c)]$$

  log(xy) = log(x) + log(y)

  because the logarithm function is monotonic, the decision of MAP stays the same.

Decision Trees

(slides by Barbara Rosario)

Decision Trees

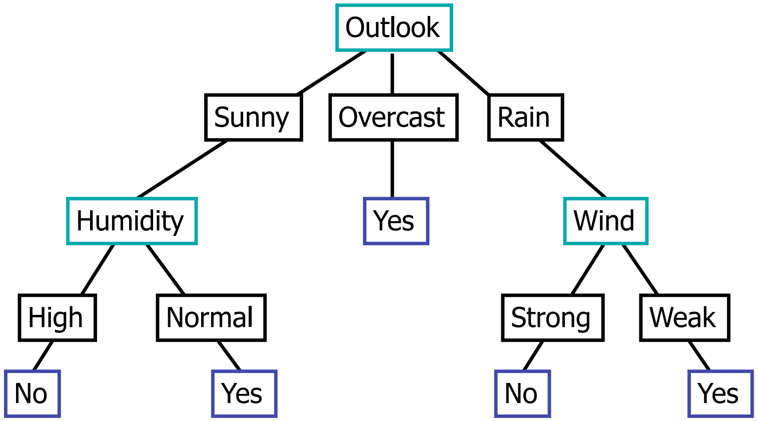
- *Decision tree* is a classifier in the form of a tree structure, where each node is either:
  - *Decision node* - specifies some test to be carried out on a single attribute-value, with one branch and sub-tree for each possible outcome of the test
  - *Leaf node* - indicates the value of the target attribute (class) of examples
- A decision tree can be used to classify an example by starting at the root of the tree and moving through it until a leaf node, which provides the classification of the instance.

Training Examples

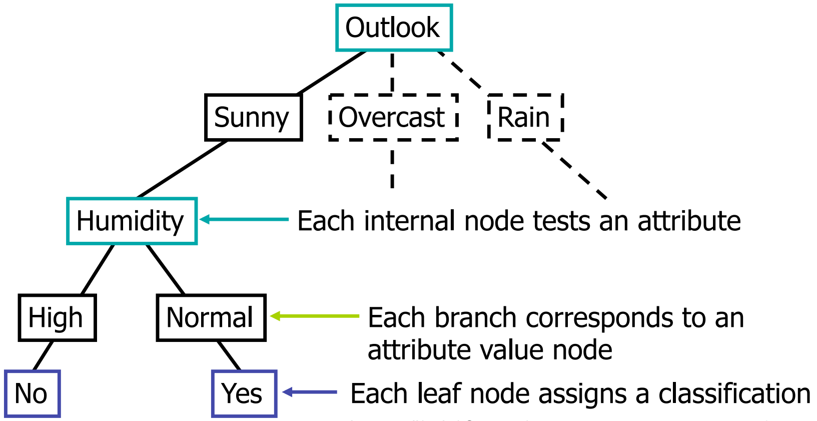
Goal: learn when we can play Tennis and when we cannot

Day	Outlook	Temp.	Humidity	Wind	**Play Tennis**
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Decision Tree for PlayTennis

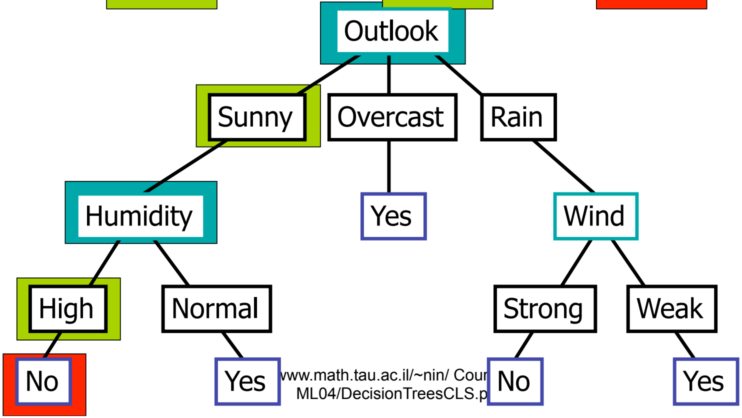


Decision Tree for PlayTennis


Outlook
Sunny
Overcast
Rain
Humidity
High
Normal
No
Yes
Each internal node tests an attribute
Each branch corresponds to an attribute value node
Each leaf node assigns a classification

Decision Tree for PlayTennis

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	? No



Building Decision Trees

- Once we have a decision tree, it is straightforward to use it to assign labels to new input values.
- How we can build a decision tree that models a given training set?

Building Decision Trees

- The central focus of the decision tree growing algorithm is selecting which attribute to test at each node in the tree. The goal is to select the attribute that is most useful for classifying examples.
- Top-down, greedy search through the space of possible decision trees.
  - That is, it picks the best attribute and never looks back to reconsider earlier choices.

Building Decision Trees

- Splitting criterion
  - Finding the features and the values to split on
    - for example, why test first “cts” and not “vs”?
    - Why test on “cts < 2” and not “cts < 5” ?
  - Split that gives us the *maximum information gain* (or the *maximum reduction of uncertainty*)
- Stopping criterion
  - When all the elements at one node have the same class, no need to split further
- In practice, one first builds a large tree and then one prunes it back (to avoid overfitting)

- See *Foundations of Statistical Natural Language Processing*, Manning and Schuetze for a good introduction what did we say was the **best split**?

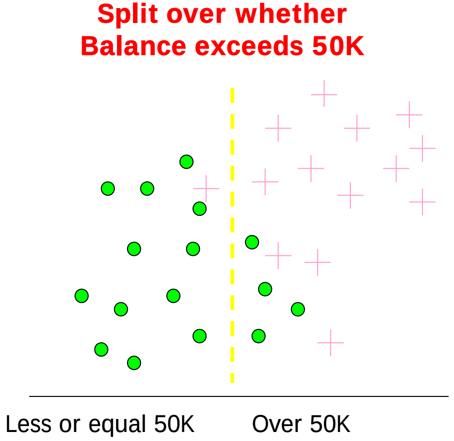
what did we say was the **best split**?

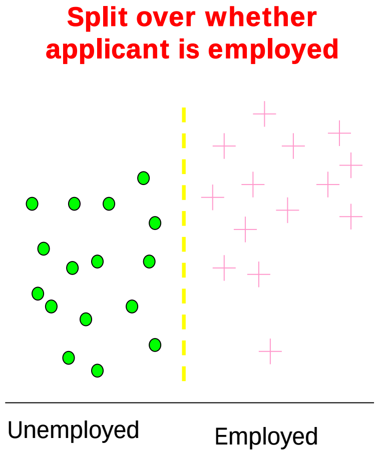
- Information Gain is the **mutual information** between input attribute A and target variable Y
- Information Gain is the **expected reduction in entropy** of target variable Y for data sample S, due to sorting on variable A

- (feature) “cheerful” is found in
    - 70% of POSITIVE tweets
    - 25% of NEGATIVE tweets
- (feature) “emotion” is found in
    - 40% of POSITIVE tweets
    - 30% of NEGATIVE tweets

Information Gain

Which test is more informative?

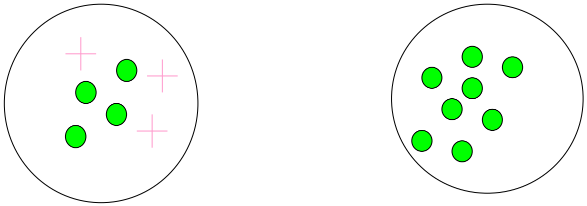




Information Gain

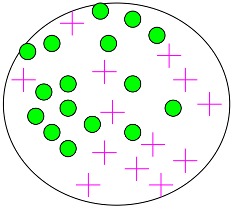
Impurity/Entropy (informal)

– Measures the level of **impurity** in a group of examples

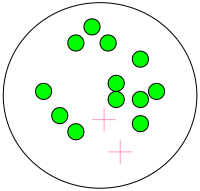


Impurity

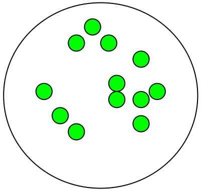
Very impure group



Less impure

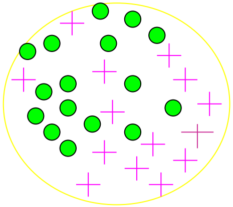


Minimum impurity



Entropy: a common way to measure impurity

- Entropy = $\sum_i - p_i \log_2 p_i$



p_i is the probability of class i

Compute it as the proportion of class i in the set.

16/30 are green circles; 14/30 are pink crosses
$\log_2(16/30)$ = -.9; $\log_2(14/30)$ = -1.1
Entropy = -(16/30)(-.9) –(14/30)(-1.1) = .99

- Entropy comes from information theory. The higher the entropy the more the information content.

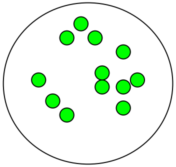
What does that mean for learning from examples?

2-Class Cases:

- What is the entropy of a group in which all examples belong to the same class?
  - entropy = - 1 $\log_2 1$ = 0

  not a good training set for learning

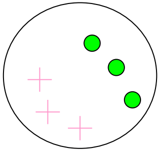
Minimum impurity



- What is the entropy of a group with 50% in either class?
  - entropy = -0.5 $\log_2 0.5$ – 0.5 $\log_2 0.5$ =1

  good training set for learning

Maximum impurity



Information Gain

- We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.

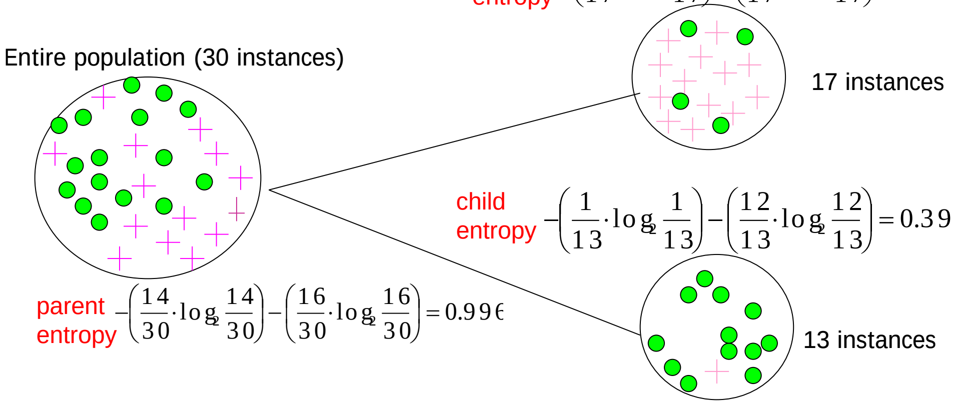
- Information gain tells us how important a given attribute of the feature vectors is.

- We will use it to decide the ordering of attributes in the nodes of a decision tree.

Calculating Information Gain

Information Gain = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$



child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Information Gain= 0.996 - 0.615 = 0.38 for this split

Simple Example

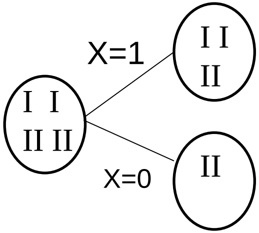
Training Set: 3 features and 2 classes

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

How would you distinguish class I from class II?

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Split on attribute X



If X is the best attribute,
this node would be further split.

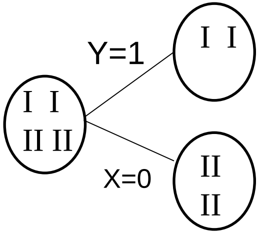
$$E_{child1} = -(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$$
$$= .5284 + .39$$
$$= .9184$$

$$E_{child2} = 0$$

$$E_{parent} = 1$$
$$\text{GAIN} = 1 - (3/4)(.9184) - (1/4)(0) = .3112$$

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Split on attribute Y



$E_{child1} = 0$

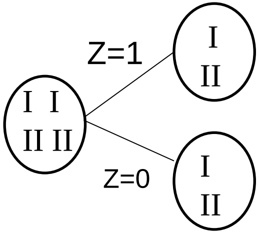
$E_{child2} = 0$

$E_{parent} = 1$

GAIN = 1 –(1/2) 0 – (1/2)0 = 1; BEST ONE

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Split on attribute Z



$E_{child1} = 1$

$E_{child2} = 1$

$E_{parent} = 1$

GAIN = 1 – (1/2)(1) – (1/2)(1) = 0   ie. NO GAIN; WORST

Building Decision Trees

- Once we have a decision tree, it is straightforward to use it to assign labels to new input values.
- How we can build a decision tree that models a given training set?

Pruning

- grow decision tree to its entirety
- trim the nodes of the decision tree in a bottom-up fashion
- if generalisation error improves after trimming, replace sub-tree by a leaf node
- class label of leaf node is determined from majority class of instances in the sub-tree

Pruning

- grow decision tree to its entirety
- trim the nodes of the decision tree in a bottom-up fashion
- if generalisation error improves after trimming, replace sub-tree by a leaf node
- class label of leaf node is determined from majority class of instances in the sub-tree

For the experiments you can manipulate the following parameters:

- maximum number of leaves
- minimum number of samples per leaf
- (features)

Decision trees in scikit-learn

http://scikit-learn.org/stable/modules/tree.html

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Not all features are equally important

(some slides by Manning et al (2009), Intro to IR)

Feature selection methods

we can think of feature selection as a way to replace a complex classifier which uses *all* features with a simpler one which only uses a *subset* of those.
$\rightarrow$ **how** do we choose this subset?

- frequency
- mutual information
- (. . .)

Mutual information

- Compute the feature utility $A(t, c)$ as the expected mutual information (MI) of term t and class c.

- MI tells us "how much information" the term contains about the class and vice versa.

- For example, if a term's occurrence is independent of the class (same proportion of docs within/without class contain the term), then MI is 0.

Mutual Information – worked

intuitively, we need four values:

- N_{11}: presence (1) of feature (term) t in documents of class c (11)
- N_{01}: absence (0) of feature (term) t in documents of class c (01)
- N_{10}: presence (1) of feature (term) t in documents of class $\neg c$ (10)
- N_{00}: absence (0) of feature (term) t in documents of class $\neg c$ (00)

Mutual Information – worked

intuitively, we need four values:

- N_{11}: presence (1) of feature (term) t in documents of class c (11)
- N_{01}: absence (0) of feature (term) t in documents of class c (01)
- N_{10}: presence (1) of feature (term) t in documents of class $\neg c$ (10)
- N_{00}: absence (0) of feature (term) t in documents of class $\neg c$ (00)

	$e_c = 1$	$e_c = 0$
$e_t = 1$	N_{11}	N_{10}
$e_t = 0$	N_{01}	N_{00}

$$N = N_{11} + N_{01} + N_{10} + N_{00}$$

Computing MI

$$MI(t,c) = \sum_{e_t \in 0,1} \sum_{e_c \in 0,1} p(e_t, e_c) log \frac{p(e_t, e_c)}{p(e_t)p(e_c)}$$

based on maximum likelihood estimates, we use the following formula:

$$MI(t,c) = \frac{N_{11}}{N} log_2 \frac{NN_{11}}{N_1 N_1} + \frac{N_{01}}{N} log_2 \frac{NN_{01}}{N_0 N_1} + \frac{N_{10}}{N} log_2 \frac{NN_{10}}{N_1 N_0} + \frac{N_{00}}{N} log_2 \frac{NN_{00}}{N_0 N_0}$$

$$MI(t,c) = \frac{N_{11}}{N} log_2 \frac{NN_{11}}{N_1 N_1} + \frac{N_{01}}{N} log_2 \frac{NN_{01}}{N_0 N_1} + \frac{N_{10}}{N} log_2 \frac{NN_{10}}{N_1 N_0} + \frac{N_{00}}{N} log_2 \frac{NN_{00}}{N_0 N_0}$$

MI example for *poultry*/EXPORT in Reuters

	$e_c = e_{poultry} = 1$	$e_c = e_{poultry} = 0$
$e_t = e_{\text{EXPORT}} = 1$	$N_{11} = 49$	$N_{10} = 27{,}652$
$e_t = e_{\text{EXPORT}} = 0$	$N_{01} = 141$	$N_{00} = 774{,}106$

Plug these values into formula:

$$\begin{aligned} I(U;C) &= \frac{49}{801{,}948} \log_2 \frac{801{,}948 \cdot 49}{(49+27{,}652)(49+141)} \\ &+ \frac{141}{801{,}948} \log_2 \frac{801{,}948 \cdot 141}{(141+774{,}106)(49+141)} \\ &+ \frac{27{,}652}{801{,}948} \log_2 \frac{801{,}948 \cdot 27{,}652}{(49+27{,}652)(27{,}652+774{,}106)} \\ &+ \frac{774{,}106}{801{,}948} \log_2 \frac{801{,}948 \cdot 774{,}106}{(141+774{,}106)(27{,}652+774{,}106)} \\ &\approx 0.000105 \end{aligned}$$

MI feature selection on Reuters

Class: *coffee*

term	MI
COFFEE	0.0111
BAGS	0.0042
GROWERS	0.0025
KG	0.0019
COLOMBIA	0.0018
BRAZIL	0.0016
EXPORT	0.0014
EXPORTERS	0.0013
EXPORTS	0.0013
CROP	0.0012

Class: *sports*

term	MI
SOCCER	0.0681
CUP	0.0515
MATCH	0.0441
MATCHES	0.0408
PLAYED	0.0388
LEAGUE	0.0386
BEAT	0.0301
GAME	0.0299
GAMES	0.0284
TEAM	0.0264

Vector space model

the representation of a set of documents as vectors in a common space

(parts of some following slides are based on slides by Yannick Parmentier)

Vector space model

- Each term t of the dictionary is considered as a *dimension*
- A document d can be represented by the weight of each vocabulary term:
$$\vec{V}(d) = (w(t_1, d), w(t_2, d), \ldots, w(t_n, d))$$
- Note that also raw frequency could be used (that's what we are doing)

Vector space model

- Each term t of the dictionary is considered as a *dimension*
- A document d can be represented by the weight of each vocabulary term:
$$\vec{V}(d) \ = \ (w(t_1, d), w(t_2, d), \ldots, w(t_n, d))$$
- Note that also raw frequency could be used (that's what we are doing)

representation of three documents using term raw frequencies: d_1, d_2, d_3

	d_1	d_2	d_3
affection	115	58	20
jealous	10	7	11
gossip	2	0	6

Vector space model

- Each term t of the dictionary is considered as a *dimension*
- A document d can be represented by the weight of each vocabulary term:
$$\vec{V}(d) = (w(t_1, d), w(t_2, d), \ldots, w(t_n, d))$$
- Note that also raw frequency could be used (that's what we are doing)

representation of three documents using term raw frequencies: d_1, d_2, d_3

	d_1	d_2	d_3
affection	115	58	20
jealous	10	7	11
gossip	2	0	6

- **Question**: how do we compute the **similarity between documents**?

Vector normalization and similarity

- Similarity between vectors
  $\rightarrow$ inner product $\vec{V}(d_1) \cdot \vec{V}(d_2)$

Vector normalization and similarity

- Similarity between vectors
  $\rightarrow$ inner product $\vec{V}(d_1) \cdot \vec{V}(d_2)$
- What about the length of a vector?
  Longer documents will be represented with longer vectors, but that does not mean they are more important
- Euclidian normalization (vector length normalization):

$$\vec{v}(d) = \frac{\vec{V}(d)}{\|\vec{V}(d)\|} \qquad \text{where } \|\vec{V}(d)\| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Vector normalization and similarity

- Similarity between vectors
  $\rightarrow$ inner product $\vec{V}(d_1) \cdot \vec{V}(d_2)$
- Similarity given by the *cosine* measure between normalized vectors:

$$sim(d_1, d_2) \; = \; \vec{v}(d_1) \cdot \vec{v}(d_2)$$

Example (Manning et al, 09)

- $sim(d1, d2) = \vec{v}(d1) \cdot \vec{v}(d2)$
  let's backtrack this:
- $\vec{v}(d1) \cdot \vec{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

  $$\vec{v}(d_i) = \frac{\vec{V}(d_i)}{\|\vec{V}(d)\|}$$
- Euclidean length:

  $$\|\vec{V}(d)\| = \sqrt{\sum_{i=1}^{n} \vec{V}_i^2(d)}$$

Example (Manning et al, 09)

- $sim(d1, d2) = \vec{v}(d1) \cdot \vec{v}(d2)$
- $\vec{v}(d1) \cdot \vec{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\vec{v}(d_i) = \frac{\vec{V}(d_i)}{\|\vec{V}(d)\|}$$

- Euclidean length:

$$\|\vec{V}(d)\| = \sqrt{\sum_{i=1}^{n} \vec{V}_i^2(d)}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

Example (Manning et al, 09)

- $sim(d1, d2) = \vec{v}(d1) \cdot \vec{v}(d2)$
- $\vec{v}(d1) \cdot \vec{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\vec{v}(d_i) = \frac{\vec{V}(d_i)}{\|\vec{V}(d)\|}$$

- Euclidean length:

$$\|\vec{V}(d)\| = \sqrt{\sum_{i=1}^{n} \vec{V}_i^2(d)}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

$$\|\vec{V}(d1)\| = \sqrt{115^2 + 10^2 + 2^2}$$

Example (Manning et al, 09)

- $sim(d1, d2) = \vec{v}(d1) \cdot \vec{v}(d2)$
- $\vec{v}(d1) \cdot \vec{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\vec{v}(d_i) = \frac{\vec{V}(d_i)}{\|\vec{V}(d)\|}$$

- Euclidean length:

$$\|\vec{V}(d)\| = \sqrt{\sum_{i=1}^{n} \vec{V}_i^2(d)}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

$$\|\vec{V}(d1)\| = \sqrt{115^2 + 10^2 + 2^2}$$

$$\vec{v}(d1_1) = \frac{115}{\sqrt{115^2+10^2+2^2}} = 0.996$$

$$\vec{v}(d1_2) = \frac{10}{\sqrt{115^2+10^2+2^2}} = 0.087$$

$$\vec{v}(d1_3) = \frac{2}{\sqrt{115^2+10^2+2^2}} = 0.017$$

Example (Manning et al, 09)

- $sim(d1, d2) = \vec{v}(d1) \cdot \vec{v}(d2)$
- $\vec{v}(d1) \cdot \vec{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\vec{v}(d_i) = \frac{\vec{V}(d_i)}{\|\vec{V}(d)\|}$$

- Euclidean length:

$$\|\vec{V}(d)\| = \sqrt{\sum_{i=1}^{n} \vec{V}_i^2(d)}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

$$\|\vec{V}(d2)\| = \sqrt{58^2 + 7^2 + 0}$$

$$\vec{v}(d2_1) = \frac{58}{\sqrt{58^2+7^2+0}} = 0.993$$

$$\vec{v}(d2_2) = \frac{7}{\sqrt{58^2+7^2+0}} = 0.120$$

$$\vec{v}(d2_3) = 0$$

Example (Manning et al, 09)

- $sim(d1, d2) = \vec{v}(d1) \cdot \vec{v}(d2)$
- $\vec{v}(d1) \cdot \vec{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\vec{v}(d_i) = \frac{\vec{V}(d_i)}{\|\vec{V}(d)\|}$$

- Euclidean length:

$$\|\vec{V}(d)\| = \sqrt{\sum_{i=1}^{n} \vec{V}_i^2(d)}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

$$\|\vec{V}(d3)\| = \sqrt{20^2 + 11^2 + 6^2}$$

$$\vec{v}(d3_1) = \frac{20}{\sqrt{20^2+11^2+6^2}} = 0.847$$

$$\vec{v}(d3_2) = \frac{11}{\sqrt{20^2+11^2+6^2}} = 0.466$$

$$\vec{v}(d3_3) = \frac{6}{\sqrt{20^2+11^2+6^2}} = 0.254$$

Example (Manning et al, 09)

- $sim(d1, d3) = \vec{v}(d1) \cdot \vec{v}(d3)$
- $\vec{v}(d1) \cdot \vec{v}(d3) = \sum_{i=1}^{n} d1_i d3_i$

$$\vec{v}(d1_1) = \frac{115}{\sqrt{115^2+10^2+2^2}} = 0.996$$
$$\vec{v}(d1_2) = \frac{10}{\sqrt{115^2+10^2+2^2}} = 0.087$$
$$\vec{v}(d1_3) = \frac{2}{\sqrt{115^2+10^2+2^2}} = 0.017$$

[i=1] $0.996 * 0.847 +$
[i=2] $0.087 * 0.466 +$
[i=3] $0.017 * 0.254 =$
$= 0.888$

$$\vec{v}(d3_1) = \frac{20}{\sqrt{20^2+11^2+6^2}} = 0.847$$
$$\vec{v}(d3_2) = \frac{11}{\sqrt{20^2+11^2+6^2}} = 0.466$$
$$\vec{v}(d3_3) = \frac{6}{\sqrt{20^2+11^2+6^2}} = 0.254$$

Example (Manning et al, 09)

- $sim(d1, d2) = \vec{v}(d1) \cdot \vec{v}(d2)$
- $\vec{v}(d1) \cdot \vec{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$

$$\vec{v}(d1_1) = \frac{115}{\sqrt{115^2+10^2+2^2}} = 0.996$$
$$\vec{v}(d1_2) = \frac{10}{\sqrt{115^2+10^2+2^2}} = 0.087$$
$$\vec{v}(d1_3) = \frac{2}{\sqrt{115^2+10^2+2^2}} = 0.017$$

[i=1] $0.996 * 0.993+$
[i=2] $0.087 * 0.120+$
[i=3] $0.017 * 0.000 =$
$= 0.999$

$$\vec{v}(d2_1) = \frac{58}{\sqrt{58^2+7^2+0}} = 0.993$$
$$\vec{v}(d2_2) = \frac{7}{\sqrt{58^2+7^2+0}} = 0.120$$
$$\vec{v}(d2_3) = 0$$

Example (Manning et al, 09)

summing up:

dictionary	$\vec{v}(d_1)$	$\vec{v}(d_2)$	$\vec{v}(d_3)$
affection	0.996	0.993	0.847
jealous	0.087	0.120	0.466
gossip	0.017	0	0.254

$$sim(d_1, d_2) = 0.999$$
$$sim(d_1, d_3) = 0.888$$

New documents

- each new document *n* is represented using vectors in the same way

	d_1	d_2	d_3	n
affection	115	58	20	0
jealous	10	7	11	1
gossip	2	0	6	1

- $sim(n, d) = \vec{v}(n) \cdot \vec{v}(d)$

New documents

- each new document *n* is represented using vectors in the same way

	d_1	d_2	d_3	n
affection	115	58	20	0
jealous	10	7	11	1
gossip	2	0	6	1
class	A	A	B	?

- $sim(n, d) = \vec{v}(n) \cdot \vec{v}(d)$

New documents

- each new document n is represented using vectors in the same way

	d_1	d_2	d_3	n
affection	115	58	20	0
jealous	10	7	11	1
gossip	2	0	6	1
class	A	A	B	B

- $sim(n, d) = \vec{v}(n) \cdot \vec{v}(d)$
- with $n = < jealous, gossip >$
  we obtain:

$$\begin{array}{lcl} \vec{v}(n) \cdot \vec{v}(d_1) & = & 0.074 \\ \vec{v}(n) \cdot \vec{v}(d_2) & = & 0.085 \\ \vec{v}(n) \cdot \vec{v}(d_3) & = & 0.509 \end{array}$$

Classifying new documents

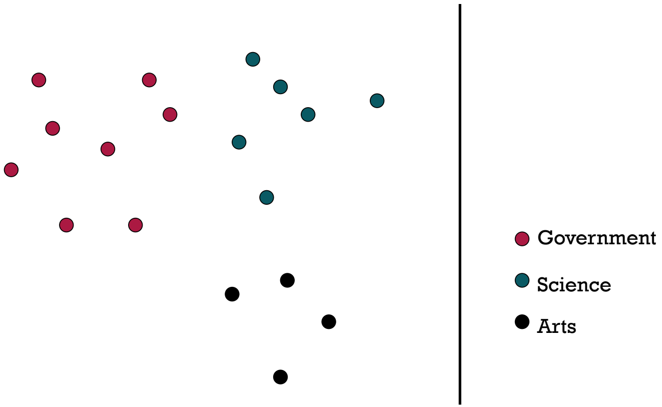
- Basic idea: similarity cosines between the new document's vector and each classified document's vector; finally selection of the top K scores
- NB: the decisions of many vector space classifiers are based on a notion of *distance*.

  There is a direct correspondence between cosine similarity and Euclidean distance for length-normalised vectors, so it rarely matters whether the relatedness of two documents is expressed in terms of similarity or distance

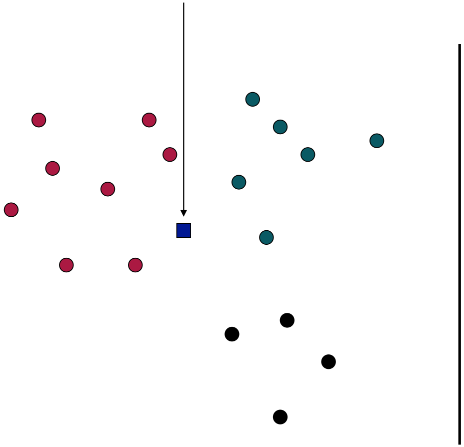
Classification Using Vector Spaces

- In vector space classification, training set corresponds to a labeled set of points (equivalently, vectors)
- Premise 1: Documents in the same class form a contiguous region of space
- Premise 2: Documents from different classes don't overlap (much)
- Learning a classifier: build surfaces to delineate classes in the space

Documents in a Vector Space



Test Document of what class?

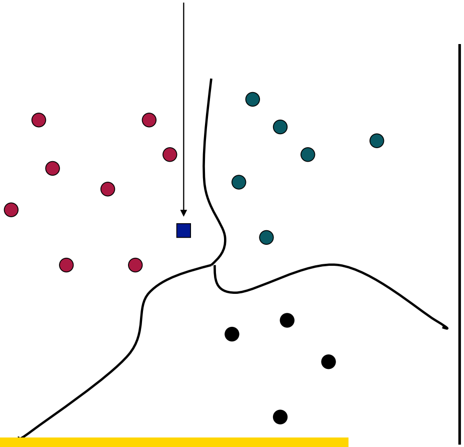


● Government

● Science

● Arts

Test Document = Government



Is this similarity hypothesis true in general?

● Government

● Science

● Arts

Our focus: how to find good separators

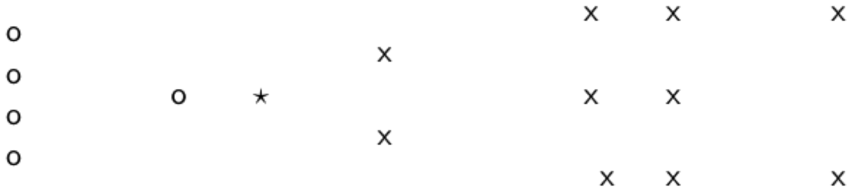
k-nearest neighbor

(some slides by Manning et al (2009), Intro to IR)

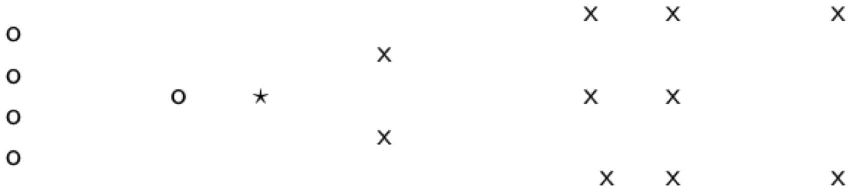
k Nearest Neighbor Classification

- kNN = *k* Nearest Neighbor

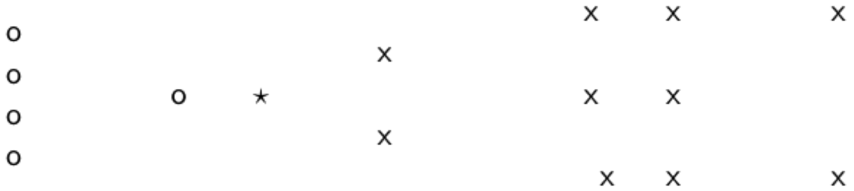
- To classify a document *d*:
- Define *k*-neighborhood as the *k* nearest neighbors of *d*
- Pick the majority class label in the *k*-neighborhood



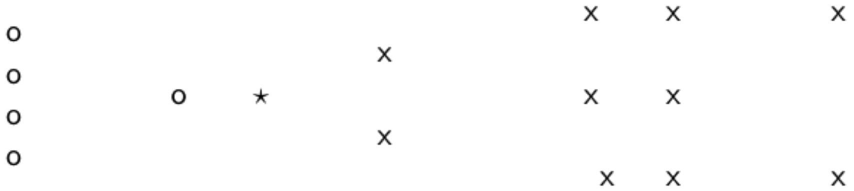
can you classify the *star*?
what is it most similar/close to?



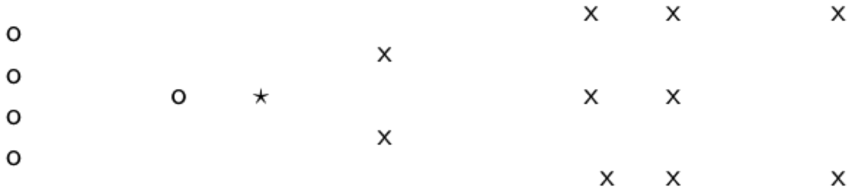
can you classify the *star*?
what is it most similar/close to ($K = 3$)?



can you classify the *star*?
what is it most similar/close to ($K = 5$)?



can you classify the *star*?
what is it most similar/close to ($K = 9$)?



can you classify the *star*?
what is it most similar/close to ($K = 15$)?

Nearest-Neighbor Learning

- Learning: just store the labeled training examples *D*
- Testing instance *x (under 1NN)*:
  - Compute similarity between *x* and all examples in *D*.
  - Assign *x* the category of the most similar example in *D*.
- Does not compute anything beyond storing the examples
- Also called:
  - Case-based learning
  - Memory-based learning
  - Lazy learning
- Rationale of kNN: contiguity hypothesis

k Nearest Neighbor

- Using only the closest example (1NN) subject to errors due to:
  - A single atypical example.
  - Noise (i.e., an error) in the category label of a single training example.
- More robust: find the *k* examples and return the majority category of these *k*
- *k* is typically odd to avoid ties; 3 and 5 are most common

Distance

- all instances correspond to points in an n-dimensional Euclidean space
- classification is done by comparing feature vectors of the different points

Crucial decisions:

A: how we calculate “distance” or “similarity”

B: how many points we take to measure A (K)

Distance

how to calculate "distance" or "similarity":

- an instance a is represented as $t_1(a), t_2(a), t_3(a), ... t_n(a)$
- the Euclidean distance d of two instances a and b is as follows

$$\textit{Euclidean distance} : d(a,b) = d(b,a) = \sqrt{\sum_{i=1}^{n} (t_i(a) - t_i(b))^2}$$

(also used: cosine similarity of tf.idf weighted vectors)

How to choose K

- experience/knowledge about a certain classification problem
- picking best K on development set or via cross-validation

kNN: Discussion

- No feature selection necessary
- No training necessary
- Scales well with large number of classes
  - Don’t need to train *n* classifiers for *n* classes
- Classes can influence each other
  - Small changes to one class can have ripple effect
- May be expensive at test time
- In most cases it’s more accurate than NB or Rocchio

KNN in scikit-learn

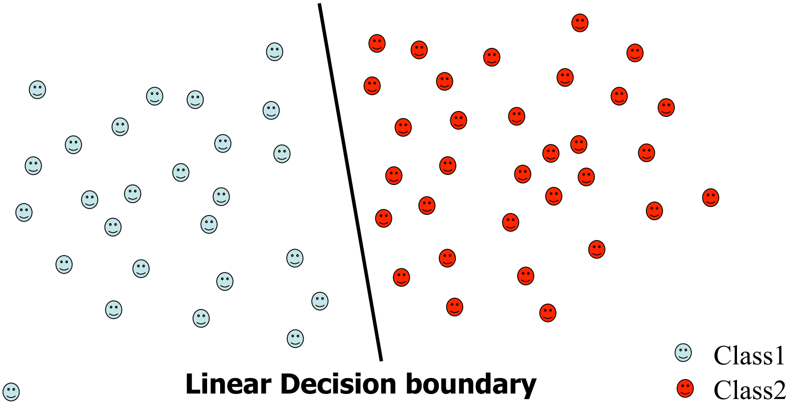
http://scikit-learn.org/stable/modules/neighbors.html

http://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification

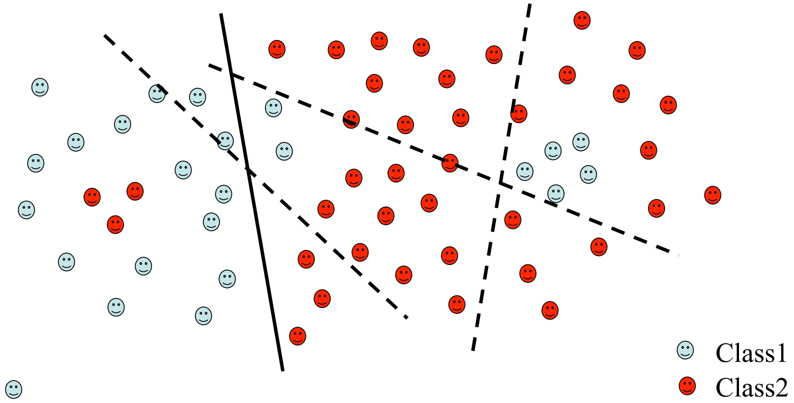
Linear versus Non Linear algorithms

- **Linearly separable data**: if all the data points can be correctly classified by a linear (hyperplanar) decision boundary

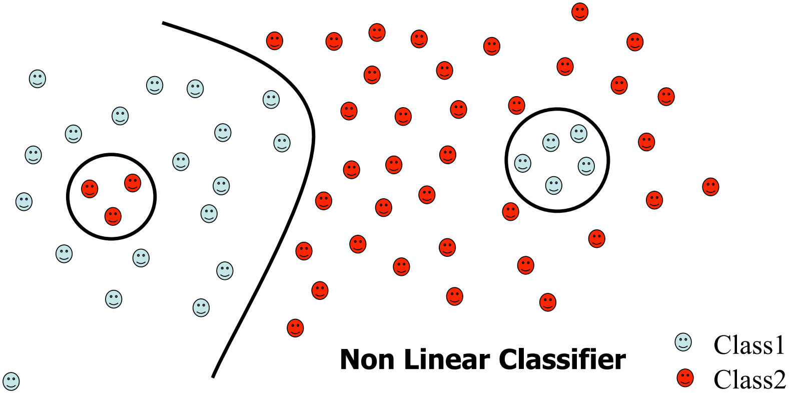
Linearly separable data



Non linearly separable data



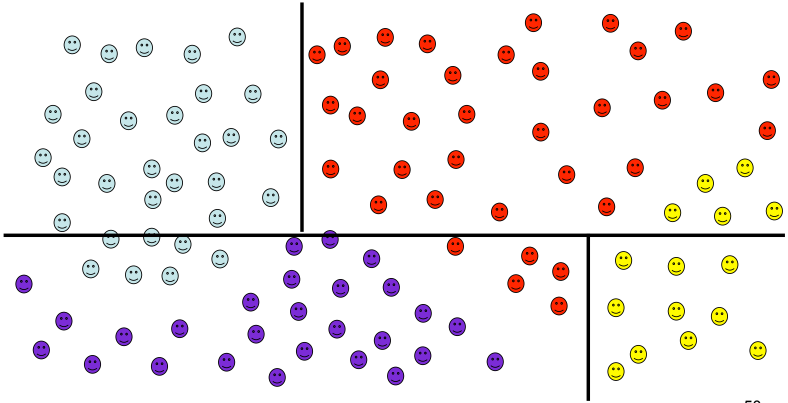
Non linearly separable data



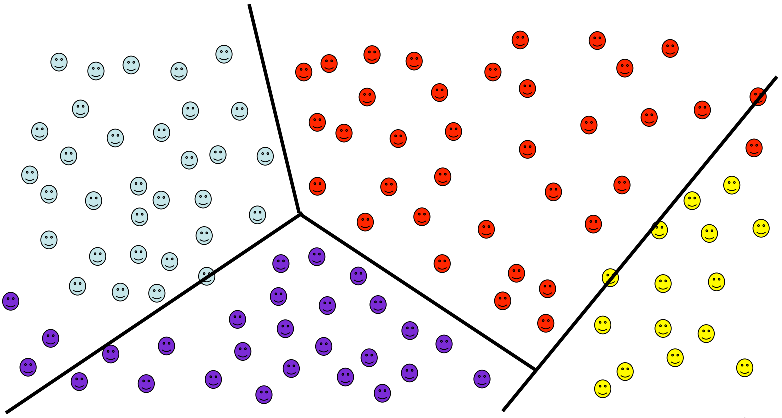
(Some) Algorithms for multi-class classification

- Linear
  - Parallel class separators: Decision Trees
  - Non parallel class separators: Naïve Bayes and Maximum Entropy
- Non Linear
  - K-nearest neighbors

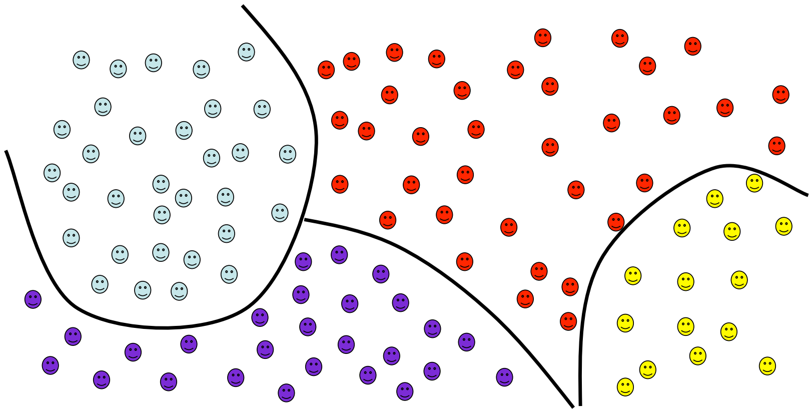
Linear, parallel class separators (ex: Decision Trees)



Linear, NON parallel class separators (ex: Naïve Bayes)



Non Linear (ex: *k* Nearest Neighbor)



bias vs variance

Bias vs. capacity – notions and terminology

- Consider asking a botanist: Is an object a tree?
  - Too much *capacity*, low *bias*
    - Botanist who memorizes
    - Will always say "no" to new object (e.g., different # of leaves)
  - Not enough capacity, high bias
    - Lazy botanist
    - Says "yes" if the object is green
  - You want the middle ground

(Example due to C. Burges)

kNN vs. Naive Bayes

- Bias/Variance tradeoff
  - Variance ≈ Capacity
- kNN has high variance and low bias.
  - Infinite memory
- NB has low variance and high bias.
  - Linear decision surface (hyperplane – see later)

Overfitting

- too lazy botanist → everthing green is a tree
- too picky botanist → nothing he hasn't seen yet is a tree

Overfitting

- too lazy botanist → everthing green is a tree
- too picky botanist → nothing he hasn't seen yet is a tree

→ a model is overfitting when it fits the data too tightly, and it's modelling noise or error instead of generalising well

→ how can we find this out?

Overfitting

- too lazy botanist $\rightarrow$ everthing green is a tree
- too picky botanist $\rightarrow$ nothing he hasn't seen yet is a tree

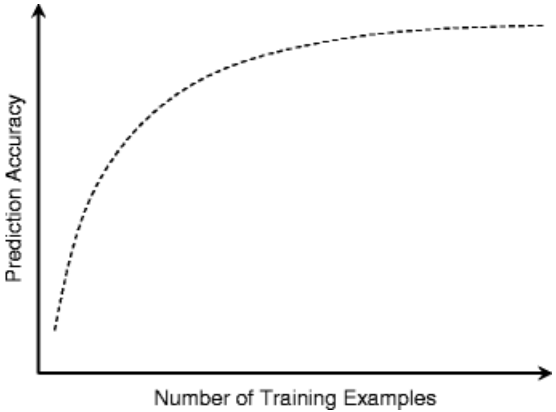
$\rightarrow$ a model is overfitting when it fits the data too tightly, and it's modelling noise or error instead of generalising well

$\rightarrow$ how can we find this out?
observe error: learning curve on training vs test data

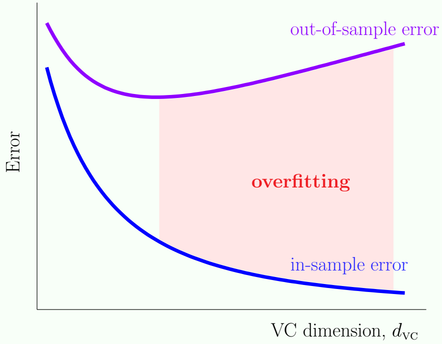
Learning Curves

what is a learning curve?

intuitively: we plot accuracy (or error rate) on one axis (y) and training size on the other (x)



Overfitting is Not Just Bad Generalization

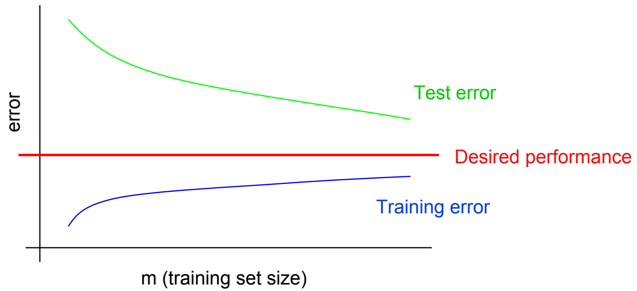


Overfitting:

Going for lower and lower E_{in} results in higher and higher E_{out}

More on bias vs. variance

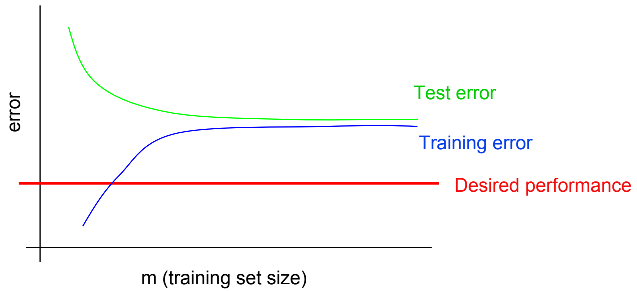
Typical learning curve for high variance:



• Test error still decreasing as m increases. Suggests larger training set will help.
• Large gap between training and test error.

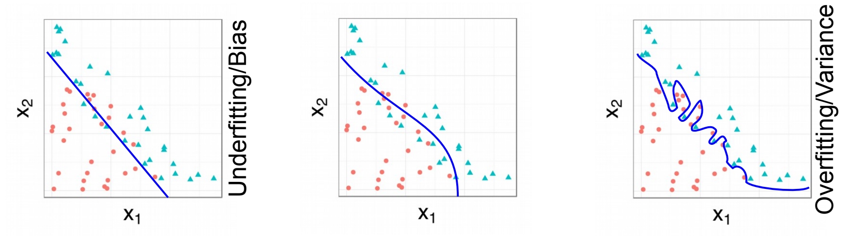
More on bias vs. variance

Typical learning curve for high bias:



- Even training error is unacceptably high.
- Small gap between training and test error.

Over- and Underfitting: Classification Example



Underfitting/Bias

- Error on training set is high
- *Simple* hypothesis fails to generalize to new examples

Overfitting/Variance

- Error on training set is low
- *Complex* hypothesis fails to generalize to new examples

Fixes

- high variance
  - get more training examples
  - reduce number of features
  - (prune tree)
  - regularization
    - penalty for model complexity
    - aim at reducing training error while keeping validation error constant (NB: cross-validation)
    - works well for lots of features where each contributes a little bit to predicting y
- high bias
  - get more features

Generative vs Discriminative

Assume this task: to determine the language that someone is speaking

- generative approach: learn each language and determine to which language the speech belongs to
- discriminative approach: determine the linguistic differences without learning any language

Generative vs Discriminative

Assume this task: to determine the language that someone is speaking

- generative approach: learn each language and determine to which language the speech belongs to
    - *generative* because it can simulate values of any variable in the model
    - example algorithm:
- discriminative approach: determine the linguistic differences without learning any language
    - directly estimate posterior probabilities
    - no attempt to model underlying probability distributions
    - example algorithm:

Generative vs Discriminative

Assume this task: to determine the language that someone is speaking

- generative approach: learn each language and determine to which language the speech belongs to
  - *generative* because it can simulate values of any variable in the model
  - example algorithm: Naive Bayes
- discriminative approach: determine the linguistic differences without learning any language
  - directly estimate posterior probabilities
  - no attempt to model underlying probability distributions
  - example algorithm: k-nearest neighbor