KNN and Naive Bayes

Le Ou-Yang

Shenzhen University

Outline

- K-Nearest Neighbors
- Measure of Similarity & Dissimilarity
- Naive Bayes

Types of classifiers

- We can divide the large variety of classification approaches into roughly three main types

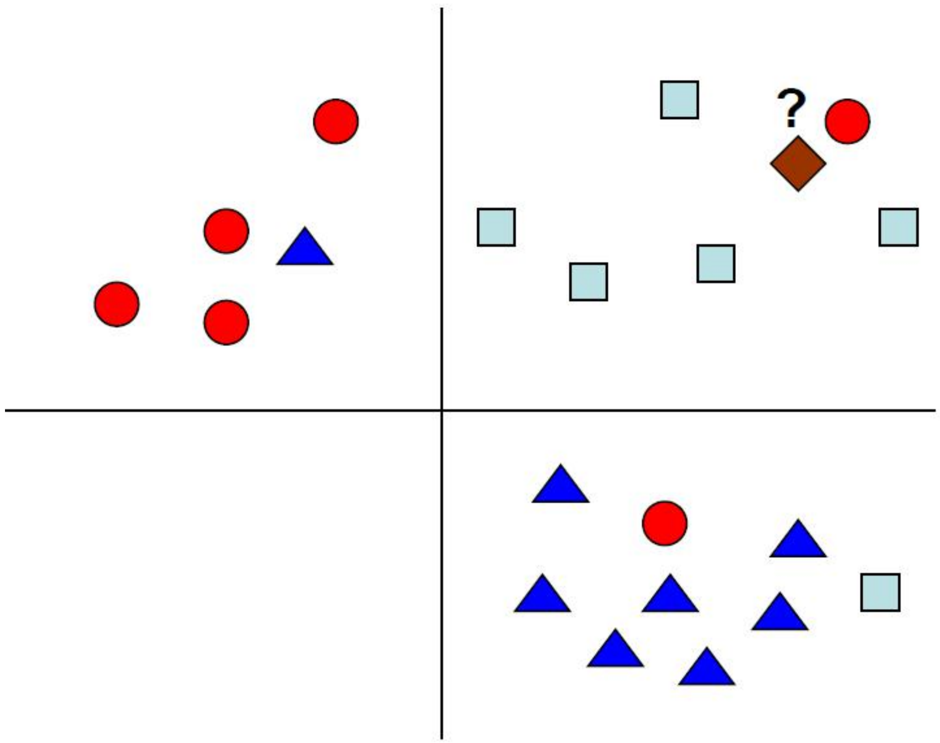
1. Instance based classifiers
   - Use observation directly (no models)
   - e.g. K-nearest neighbors

2. Generative
   - Build a generative statistical model
   - Linear discriminant analysis (LDA), QDA and naive Bayes

3. Discriminative
   - Directly estimate a decision rule/boundary
   - Logistic regression, decision tree, k-nearest neighbors, support vector machines (SVM), neural networks

Instance based classifiers

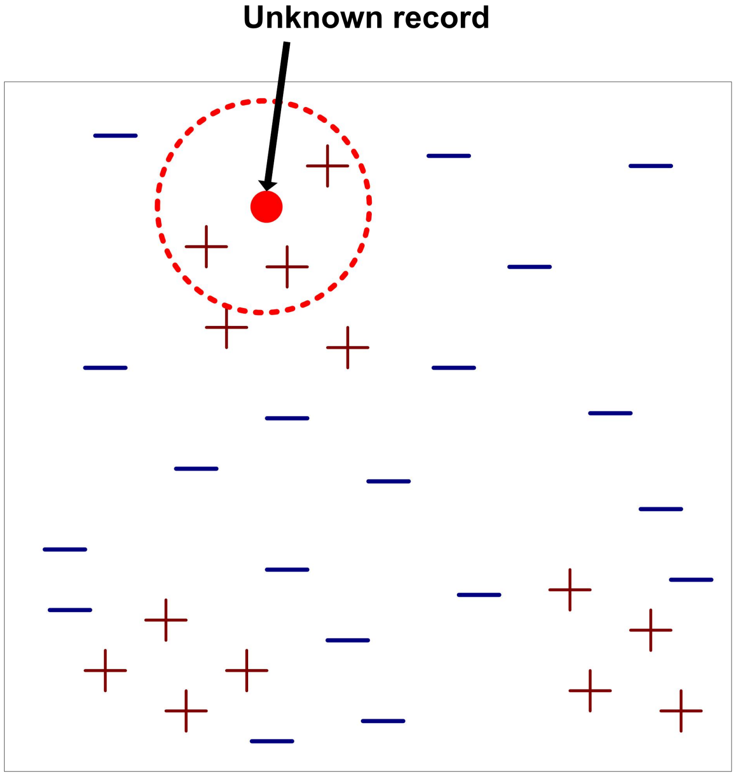
- Use observation directly (no models)
- e.g. K-nearest neighbors



K-Nearest Neighbors (KNN)

- Requires the definition a distance function or similarity measures between samples, and the value of k (the number of nearest neighbors to retrieve)

Select the class based on
the majority vote in
the k closest points



Outline

- K-Nearest Neighbors
- Measure of Similarity & Dissimilarity
- Naive Bayes

Measure of Similarity & Dissimilarity

- Similarity and dissimilarity/distance are important and fundermental as they are used by many machine learning techniques

- In some cases, the initial data set is not needed once these similarities or dissimilarities/distances have been computed.

Similarity and Dissimilarity

- Similarity
  - Numerical measure of how **alike** two data objects are.
  - Is higher when objects are more alike.
  - Often falls in the range [0,1]
- Dissimilarity
  - Numerical measure of how **different** are two data objects
  - Lower when objects are more alike
  - Minimum dissimilarity is often 0 (e.g. same objects)
  - Upper limit varies

Similarity/Dissimilarity for Simple Attributes

- Similarity/Dissimilarity between p and q. p and q are the attribute **values** for two data objects (use *single feature* value for illustration)
- Object 1: p (e.g. p=male, p=young, or p=23)
- Object 2: q (e.g. q=female, p=old, or p=40)

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$
Ordinal	$d = \frac{\lvert p-q \rvert}{n-1}$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - \frac{\lvert p-q \rvert}{n-1}$
Interval or Ratio	$d = \lvert p - q \rvert$	$s = -d$, $s = \frac{1}{1+d}$ or $s = 1 - \frac{d - min\_d}{max\_d - min\_d}$

Similarity and dissimilarity for simple attributes

Common Properties of a Similarity

- Similarities have some well-known properties.
- Let us denote by $s(p, q)$ the similarity between two data objects (points) p and q.

1. Self-Similarity

s(p, q) = 1 (or maximum similarity) only if $p = q$.

2. Symmetry

$s(p, q) = s(q, p)$ for all p and q.

Similarity does not necessarily preserve the triangle inequality, like distance.

Similarity Between **Binary Vectors**

could be n-dimensional vectors

- Consider two objects, p and q, having only binary attributes

$$p = 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$$
$$q = 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

- Compute similarities using the following quantities
M_{01} = the number of attributes where p was 0 and q was 1
M_{10} = the number of attributes where p was 1 and q was 0
M_{00} = the number of attributes where p was 0 and q was 0
M_{11} = the number of attributes where p was 1 and q was 1

- **Simple Matching Coefficient (SMC)**
SMC = number of matches / number of attributes
= $(M_{11} + M_{00}) / (M_{01} + M_{10} + M_{11} + M_{00})$

- **Jaccard Coefficient (J)**
J = number of 11 matches / number of not-both-zero attributes values
= $M_{11} / (M_{01} + M_{10} + M_{11})$

SMC versus Jaccard: Example

$p =$ 1 0 0 0 0 0 0 0 0 0
$q =$ 0 0 0 0 0 0 1 0 0 1

$M_{01} = 2$ (the number of attributes where p was 0 and q was 1)
$M_{10} = 1$ (the number of attributes where p was 1 and q was 0)
$M_{00} = 7$ (the number of attributes where p was 0 and q was 0)
$M_{11} = 0$ (the number of attributes where p was 1 and q was 1)

SMC $= (M_{11} + M_{00})/(M_{01} + M_{10} + M_{11} + M_{00}) = (0+7) / (2+1+0+7) = 0.7$

J $= (M_{11}) / (M_{01} + M_{10} + M_{11}) = 0 / (2 + 1 + 0) = 0$

In what cases, SMC or Jaccard similarity is useful?

Cosine Similarity

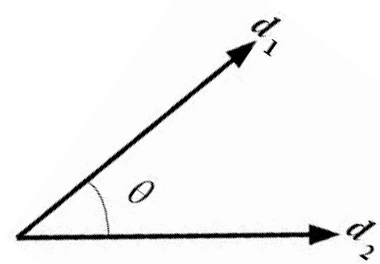
- If d_1 and d_2 are two vectors (e.g. document vectors), then

$$\cos(d_1, d_2) = (d_1 \bullet d_2) / ||d_1||\ ||d_2||$$

where $\bullet$ indicates vector dot product and $||\ d\ ||$ is the length of vector d.

- It is a measure of the *cosine* of the angle between the two vectors.

- Example:



d_1 = **3 2 0 5 0 0 0 2 0 0**

d_2 = **1 0 0 0 0 0 0 1 0 2**

$d_1 \bullet d_2 = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5$

$||d_1|| = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)^{\mathbf{0.5}} = (42)^{\mathbf{0.5}} = 6.4807$

$||d_2|| = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)^{\mathbf{0.5}} = (6)^{\mathbf{0.5}} = 2.4495$

$$\cos(d_1, d_2) = 5/(6.4807*2.4495) = 0.3150$$

Questions: Does the cosine similarity depend on the number of shared 0 values (0-0 matches) between two vectors?

Euclidean Distance

- Euclidean Distance between two n-dimensional vectors (objects) $\boldsymbol{p}$ and $\boldsymbol{q}$

$$\boldsymbol{dist} = \sqrt{\sum_{\boldsymbol{k}=1}^{\boldsymbol{n}} (\boldsymbol{p_k} - \boldsymbol{q_k})^2}$$

- where $\boldsymbol{p}=\{p_1, p_2, \ldots, p_k, \ldots, p_n\}$,
- $\boldsymbol{q}=\{q_1, q_2, \ldots, q_k, \ldots, q_n\}$.
- n is the number of dimensions (attributes) and p_k and q_k are the k^{th} attributes of data objects $\boldsymbol{p}$ and $\boldsymbol{q}$, respectively.
- Feature normalization is usually necessary if scales are different.

Scaling issues

- Attributes may have to be scaled or normalized to prevent distance measures from being dominated by one of the attributes
- Example:
  - F1: height of a person may vary from 1.2m to 2.4m
  - F2: weight of a person may vary from 35kg to 442kg
  - F3: Annual income of a person may vary from 10K to 50,000K

$p = (p_1\, p_2\, p_3) = (1.64, 48, 6000)$

$q = (q_1\, q_2\, q_3) = (1.82, 75, 10000)$

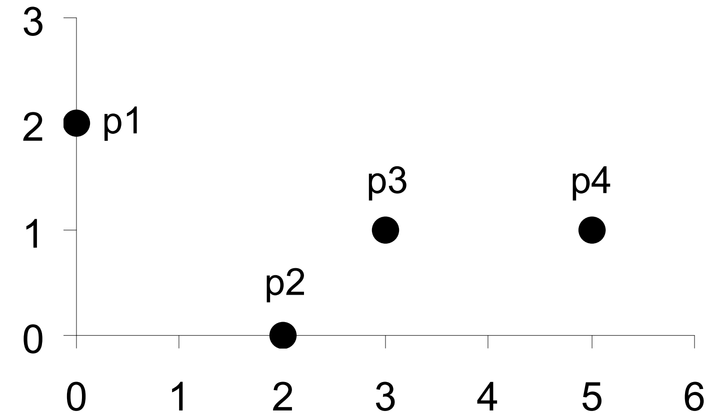
F3 dominates the calculation of Euclidean

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2} = \sqrt{(1.65 - 1.82)^2 + (48 - 75)^2 + (6000 - 10000)^2}$$

Euclidean Distance in 2D

- Example:

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1



	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Euclidean Distance Matrix

Minkowski Distance

- Minkowski Distance is a generalization of Euclidean Distance

$$dist = \left(\sum_{k=1}^{n} | p_k - q_k |^r \right)^{\frac{1}{r}}$$

Where r is a parameter, n is the number of dimensions (attributes) and p_k and q_k are the k-th attributes (components) of data objects p and q respectively.

Minkowski Distance: Special Cases

$$dist = \left(\sum_{k=1}^{n} | p_k - q_k |^r \right)^{\frac{1}{r}}$$

(applied to any vectors)

- $r = 1$:

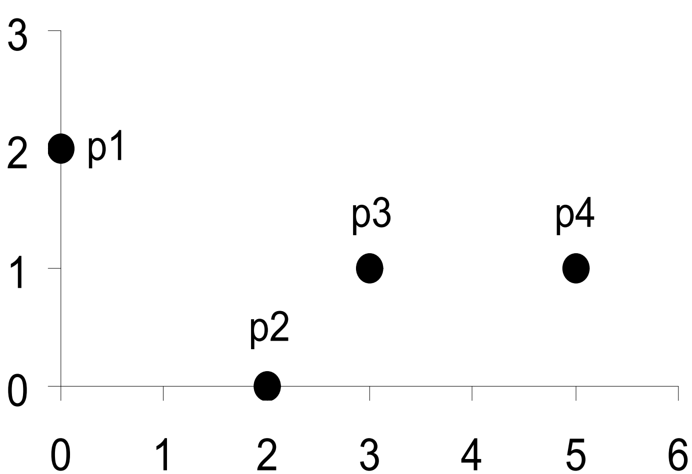
  City block (Manhattan, taxicab, **L_1 norm**) distance.

  – A common example of this is the **Hamming distance**, which is just the number of bits that are different between two binary vectors (**Hamming distance** is only applied to binary vectors)

- $r = 2$:

  Euclidean distance (**L_2 norm**)

point	x	y
p1	0	2
p2	2	0



L_1 norm: dist (p1,p2)=|0-2|+|2-0| =4

L_2 norm:

	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Minkowski Distance: Special Cases

$$dist = \left(\sum_{k=1}^{n} | p_k - q_k |^r \right)^{\frac{1}{r}}$$

- $r = 1$:

  City block (Manhattan, taxicab, **L_1 norm**) distance.
- $r = 2$:

  Euclidean distance (**L_2 norm**)
- $r \to \infty$:

  "supremum" (**L_{max} norm**, L_∞ norm) distance.

  – The **maximum difference** between any component of the two vectors: $\max(|p_1 - q_1|, \ldots, |p_n - q_n|)$

Do not confuse parameter *r* with dimensionality *n*, i.e., all these distances are defined for all the dimensions.

Minkowski Distance

$$dist = \left(\sum_{k=1}^{n} | p_k - q_k |^r \right)^{\frac{1}{r}}$$

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

Distance Matrix

L1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

City block

L2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Euclidean

L_∞	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0

Supremum

An Example

Distance between P1 and P3

- **r=1, L_1 norm, City block distance**

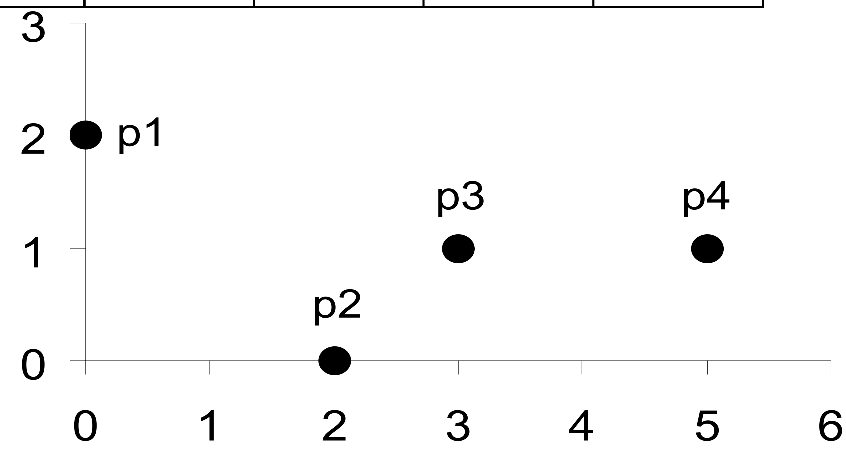
|0-3|+|2-1|=4

- **r=2, L_2 norm, Euclidean distance**

$$\sqrt{(0-3)^2 + (2-1)^2} = \sqrt{10} = 3.162$$

- **$r \to \infty$, L_∞ norm, supremum distance**

Max(|0-3|,|2-1|) =Max (3,1) =3



Common Properties of a Distance

Distances, such as the Euclidean distance, have some well known properties. Let us denote by ***d*(*p*, *q*)** is the distance (dissimilarity) between points (data objects) ***p*** and **q**.

1. **Positive Definiteness**

   $d(p, q) \geq 0$ for all p and q
   $d(p, q) = 0$ if only if $p = q$.

2. **Symmetry**

   $d(p, q) = d(q, p)$ for all p and q

3. **Triangle Inequality**

   $\mathrm{d}(p, r) \leq d(p, q) + d(q, r)$ for all points p, q, and r.

A distance satisfying all the above three properties is a metric.

Correlation

- In statistics, the **Pearson correlation coefficient** (typically denoted by *r*) is a measure of the correlation (linear dependence) between two variables *X* and *Y.*
- The values of r are between +1 and −1 inclusive.
- It is widely used in the sciences as a measure of the strength of linear dependence between two variables

Formula - Pearson's correlation coefficient

- Pearson's correlation coefficient between two variables is defined as the covariance of the two variables divided by the product of their standard deviations:

$$\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y},$$

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}.$$

Easy to compute

Example: Visually Evaluating Correlation

Scatter plots showing the correlation from –1 to 1.

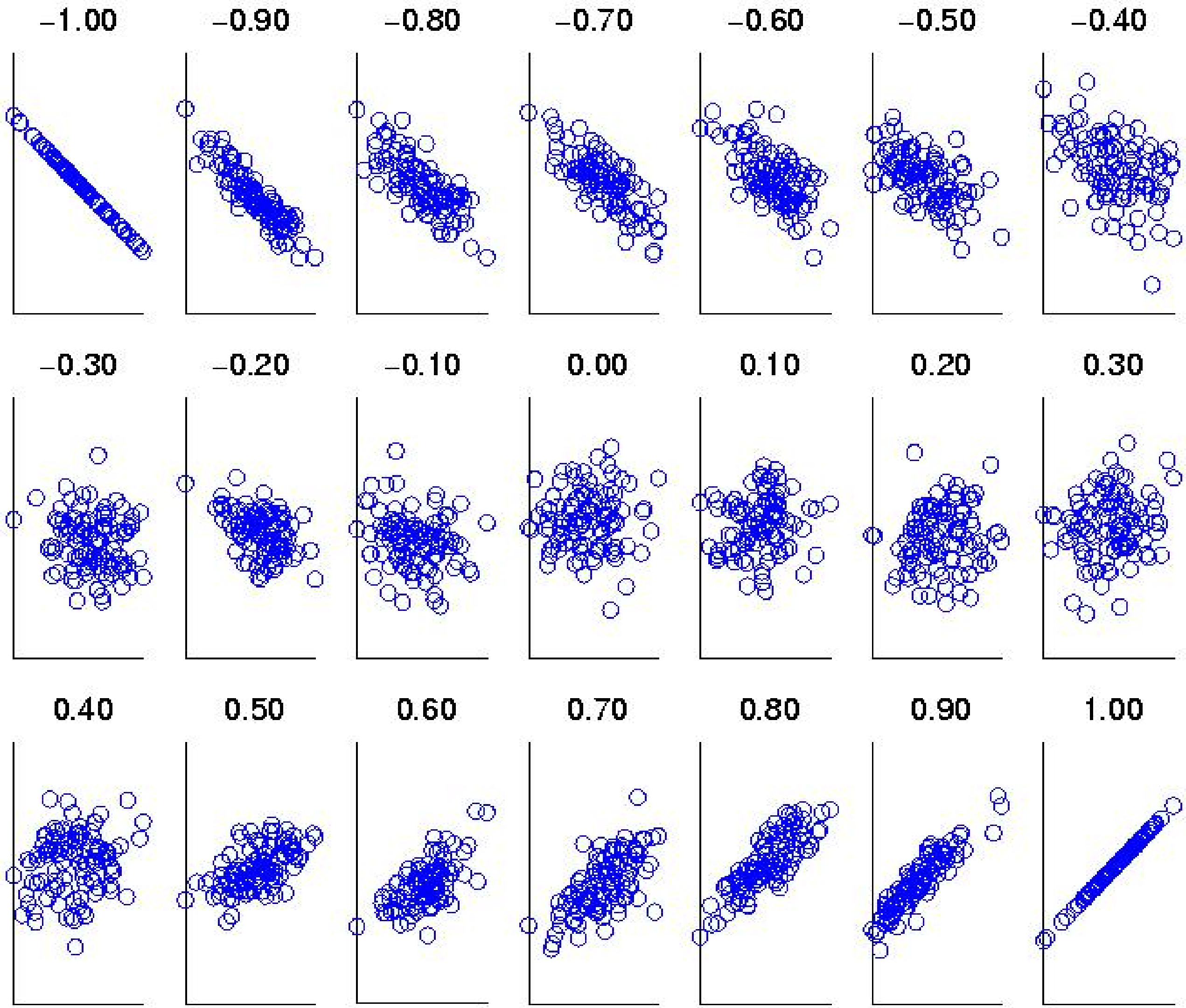


Figure 5.11. Scatter plots illustrating correlations from -1 to 1.

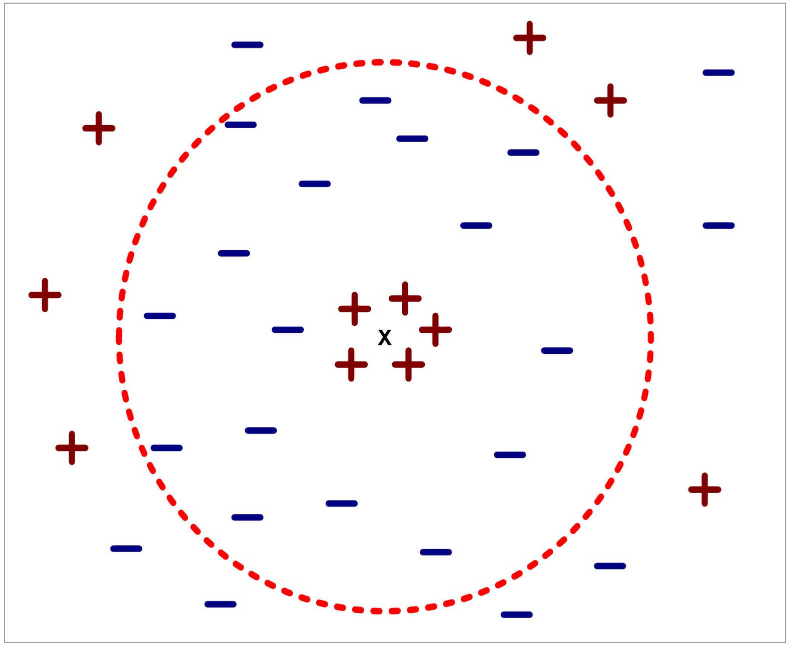
Example of Correlation

- (Perfect Correlation)
  - Correlation is always in the range -1 and 1.
    A correlation of value 1 (-1) means that p and q have a perfect positive (negative) linear relationship, i.e.,
    y = a* **x** + b, where a and b are constants.
  - The follow two sets of **x** and **y** indicate two cases of correlation -1 and +1, respectively

x=(-3, 6, 0, 3, -6)	**x**= (3, 6, 0, 3, 6)
y = (1, -2, 0, -1, 2)	**y**=(1, 2, 0, 1, 2)
corr(**x**, **y**) = -1	corr(**x**, **y**) = 1
y=-1/3*x	**y=1/3*x**

K-Nearest Neighbors (KNN)

- 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



Nearest Neighbors (1-NN)

When to Consider

- Instance map to points in R^n
- Less than 20 attributes per instance
- Lots of training data

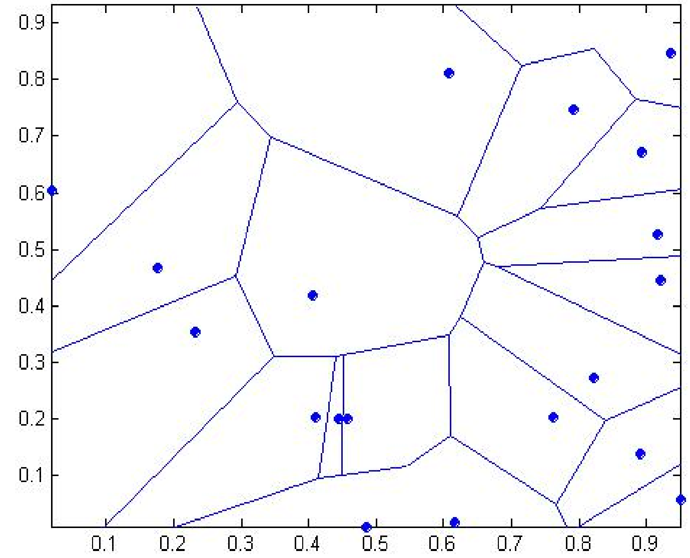
Advantages

- Training is very fast
- Learn complex target functions
- Do not lose information

Disadvantages

- Slow at query time
- Easily fooled by irrelevant attributes

1-NN decision boundary is a Voronoi Diagram



K-Nearest Neighbors (KNN)

- Distance measure
  - Most common: Euclidean
- Choosing k
  - Increasing k reduces variance, increases bias
- For high-dimensional space, problem that the nearest neighbor may not be very close at all!
- Memory-based technique. Must make a pass through the data for each classification. This can be prohibitive for large data sets.

Outline

- K-Nearest Neighbors
- Measure of Similarity & Dissimilarity
- Naive Bayes

Types of classifiers

- We can divide the large variety of classification approaches into roughly three main types

1. Instance based classifiers
   - Use observation directly (no models)
   - e.g. K-nearest neighbors

2. Generative
   - Build a generative statistical model
   - Linear discriminant analysis (LDA), QDA and Naive Bayes

3. Discriminative
   - Directly estimate a decision rule/boundary
   - Logistic regression, decision tree, k-nearest neighbors, support vector machines (SVM), neural networks

Bayes decision rule

- If we know the conditional probability P(X | y) we can determine the appropriate class by using Bayes rule:

$$P(y=i \mid X) = \frac{P(X \mid y=i)P(y=i)}{P(X)} \stackrel{def}{=} q_i(X)$$

But how do we determine p(X|y)?

Naive Bayes Classifier

- Naïve Bayes classifiers assume that given the class label (Y) the attributes are **conditionally independent** of each other:

$$p(X \mid y) = \prod_j p_j(x^j \mid y)$$

Product of probability terms

Specific model for attribute *j*

- Using this idea the full classification rule becomes:

$$\begin{aligned}\hat{y} &= \arg\max_v p(y = v \mid X) \\ &= \arg\max_v \frac{p(X \mid y = v)\, p(y = v)}{p(X)} \\ &= \arg\max_v \prod_j p_j(x^j \mid y = v)\, p(y = v)\end{aligned}$$

v are the classes we have

Conditional likelihood: Full version

$$L(X_i \mid y_i = 1, \Theta) = \prod_j p(x_i^j \mid y_i = 1, \theta_1^j)$$

Vector of binary attributes for sample *i*

The set of all parameters in the NB model

The specific parameters for attribute *j* in class 1

Note the following:

1. We assumes conditional independence between attributes **given** the class label
2. We learn a **different** set of parameters for the two classes (class 1 and class 2).

Learning parameters

$$L(X_i \mid y_i = 1, \Theta) = \prod_j p(x_i^j \mid y_i = 1, \theta_1^j)$$

- Let X_1 … X_{k1} be the set of input samples with label 'y=1'
- Assume all attributes are **binary**
- To determine the MLE parameters for $p(x^j = 1 \mid y = 1)$

  we simply count how many times the j'th entry of those samples in class 1 is 0 (termed n0) and how many times is 1 (n1). Then we set:

$$p(x^j = 1 \mid y = 1) = \frac{n1}{n0 + n1}$$

Final classification

- Once we computed all parameters for attributes in both classes we can easily decide on the label of a **new** sample X.

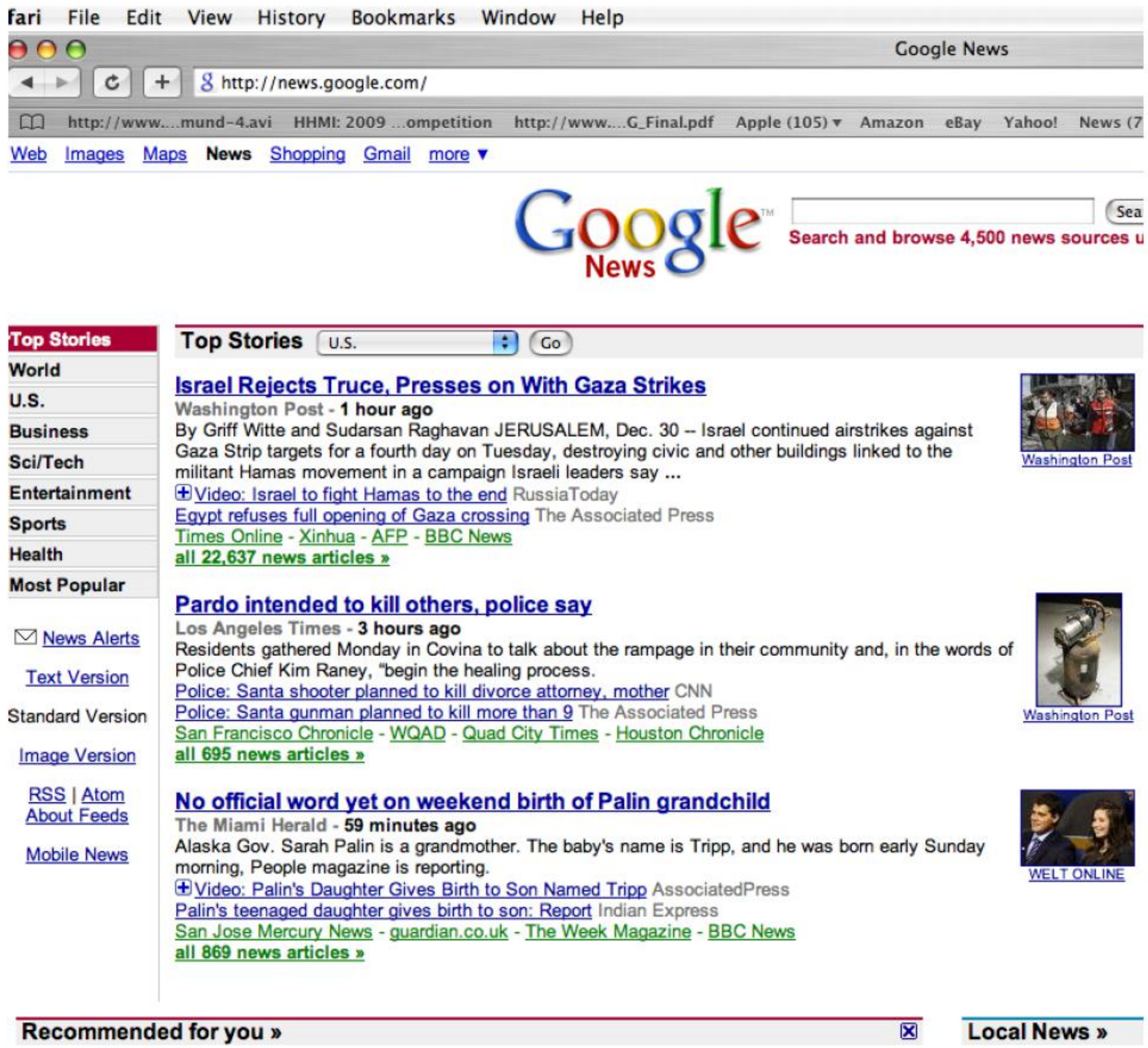
$$
\begin{aligned}
\hat{y} &= \arg\max_v \; p(y = v \mid X) \\
&= \arg\max_v \; \frac{p(X \mid y = v)\, p(y = v)}{p(X)} \\
&= \arg\max_v \prod_j p_j(x^j \mid y = v)\, p(y = v)
\end{aligned}
$$

Perform this computation for both class 1 and class 2 and select the class that leads to a higher probability as your decision

Prior on the prevalence of samples from each class

Example: Text classification

- Text classification is all around us



Example: Text classification

- What is the major topic of this article?

boston.com

HOME OBITUARIES SPORTS ENTERTAINMENT BUSINESS LIFESTYLE HEALTH TRAVEL CARS JOBS REAL ESTA

NEWS SHARI

The story behind Mitt Romney's loss in the presidential campaign to President Obama

By Michael Kranish
Globe Staff
DECEMBER 22, 2012 7:00 PM

...who believe that they are entitled to health care, to food, to housing, to you-name-it. That that's an entitlement. And the government should give it to them. And they will vote for this president no matter what.

Mother Jones VIDEO

A video from a May fund-raiser in Florida showed Romney characterizing nearly half of Americans as "victims" who want
sociated Press photo of Mother Jones video) Credit: Associated Press photo of Mother Jones video

Feature transformation

- How do we encode the set of features (words) in the document?
- What type of information do we wish to represent? What can we ignore?
- Most common encoding: '**Bag of Words**'
- Treat document as a collection of words and encode each document as a vector based on some dictionary
- The vector can either be binary (present / absent information for each word) or discrete (number of appearances)

- Google is a good example
- Other applications include job search adds, spam filtering and many more.

Feature transformation: Bag of Words

- In this example we will use a binary vector
- For document X_i we will use a vector of *m** indicator features $\{\phi^j(X_i)\}$ for whether a word appears in the document
  - $\phi^j(X_i) = 1$, if word *j* appears in document X_i;
    $\phi^j(X_i) = 0$ if it does not appear in the document
- $\Phi(X_i) = [\phi^1(X_i) \ldots \phi^m(X_i)]^T$ is the resulting feature vector for the entire dictionary for document X_i
- For notational simplicity we will replace each document X_i with a fixed length vector $\Phi_i = [\phi^1 \ldots \phi^m]^T$, where $\phi^j = \phi^j(X_i)$.

*The size of the vector for English is usually ~10000 words

Example: Text classification

Assume we would like to classify documents as election related or not.

Dictionary

- Washington
- Congress

…

54. Romney
55. Obama
56. Nader

$$\phi^{54}=\phi^{54}(X_i) = 1$$
$$\phi^{55}=\phi^{55}(X_i) = 1$$
$$\phi^{56}=\phi^{56}(X_i) = 0$$

Example: Text classification

We would like to classify documents as election related or not.

- Given a collection of documents with their labels (usually termed ‘training data’) we learn the parameters for our model.
- For example, if we see the word ‘Obama’ in *n1* out of the *n* documents labeled as ‘election’ we set *p(‘obama’|’election’)=n1/n*
- Similarly we compute the priors (*p(‘election’)*) based on the proportion of the documents from both classes.

Example: Text classification

Assume we learned the following model

$P(\phi^{romney}=1 \ |E) = 0.8$, $P(\phi^{romney}=1 \mid S) = 0.1$ $P(S) = 0.5$

$P(\phi^{obama}=1|E) = 0.9$, $P(\phi^{obama}=1| \ S) = 0.05$ $P(E) = 0.5$

$P(\phi^{clinton}=1|E) = 0.9$, $P(\phi^{clinton}=1|S) = 0.05$

$P(\phi^{football}=1|E) = 0.1$, $P(\phi^{football}=1|S) = 0.7$

For a specific document we have the following feature vector

$\phi^{romney} = 1 \ \phi^{obama} = 1 \ \phi^{clinton} = 1 \ \phi^{football} = 0$

$P(y = E \mid 1,1,1,0) \propto 0.8*0.9*0.9*0.9*0.5 = 0.5832$

$P(y = S \mid 1,1,1,0) \propto 0.1*0.05*0.05*0.3*0.5 = 0.000075$

So the document is classified as ‘Election’

Naive Bayes classifiers for continuous values

- So far we assumed a binomial or discrete distribution for the data given the model ($p(X_i|y)$)
- However, in many cases the data contains continuous features:
  - Height, weight
  - Levels of genes in cells
  - Brain activity
- For these types of data we often use a Gaussian model
- In this model we assume that the observed input vector X is generated from the following distribution

$$X \sim N(\mu, \Sigma)$$

Gaussian Bayes Classification

- To determine the class when using the Gaussian assumption we need to compute p(X|y):

$$P(y=v \mid X) = \frac{p(X \mid y=v)P(y=v)}{p(X)}$$

$$P(X \mid y) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(X-\mu)^T \Sigma^{-1} (X-\mu)\right]$$

Once again, we need lots of data to compute the values of the mean μ and the covariance matrix Σ

Gaussian Bayes Classification

• Here we can also use the Naïve Bayes assumption: Attributes are independent given the class label
• In the Gaussian model this means that the covariance matrix becomes a **diagonal matrix** with zeros everywhere except for the diagonal
• Thus, we only need to learn the values for the variance term for each attribute: $x^j \sim N(\mu^j, \sigma^j)$

$$P(X \mid y = v) = \prod_j \frac{1}{(2\pi)^{1/2} \sigma_v^j} \exp\left[-\frac{\left(\mathbf{x}_j - \mu_v^j\right)^2}{2(\sigma_v^j)^2} \right]$$

Separate means and variance for each class

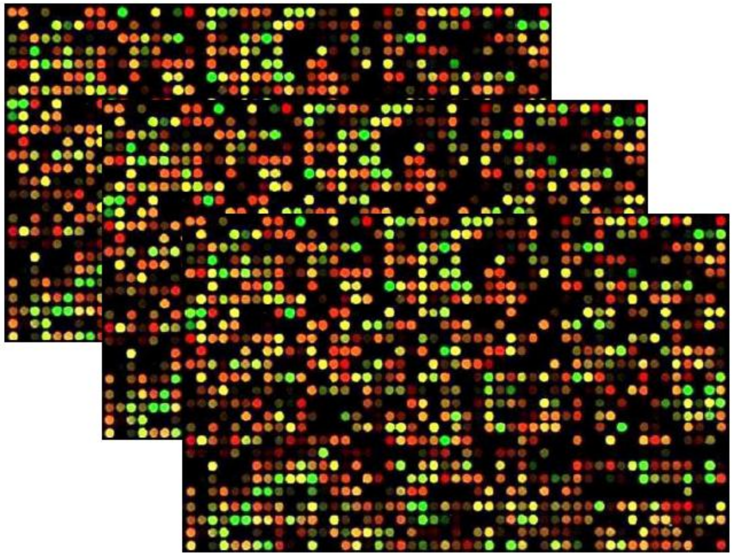
MLE for Gaussian Naive Bayes Classifier

- For each class we need to estimate one global value (prior) and two values for each feature (mean and variance)
- The prior is computed in the same way we did before (counting) which is the MLE estimate For each feature
- Let the numbers of input samples in class 1 be k1. The MLE for mean and variance is computed by setting:

$$\mu_1^j = \frac{1}{k1} \sum_{X_i | s.t. y_i = 1} x_i^j \qquad \sigma_1^{j\,2} = \frac{1}{k1} \sum_{X_i | s.t. y_i = 1} (x_i^j - \mu_1^j)^2$$

Example: Classifying gene expression data

- Measures the levels (up or down) of genes in our cells
- Differs between healthy and sick people and between different disease types
- Given measurement of patients with two different types of cancer we would like to generate a classifier to distinguish between them

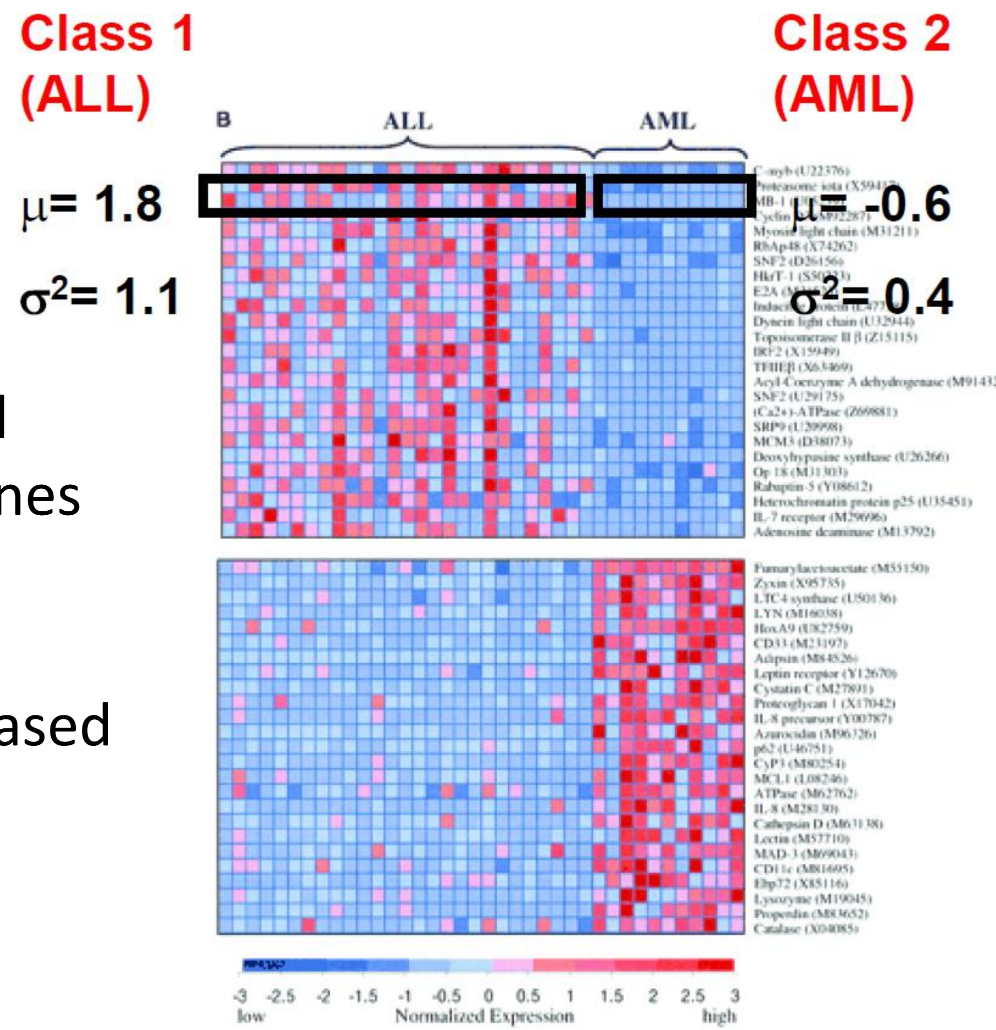


Example: Classifying cancer types

We select a subset of the genes.

We compute the mean and variance for each of the genes in each of the class.

Compute the class priors based on the input samples.



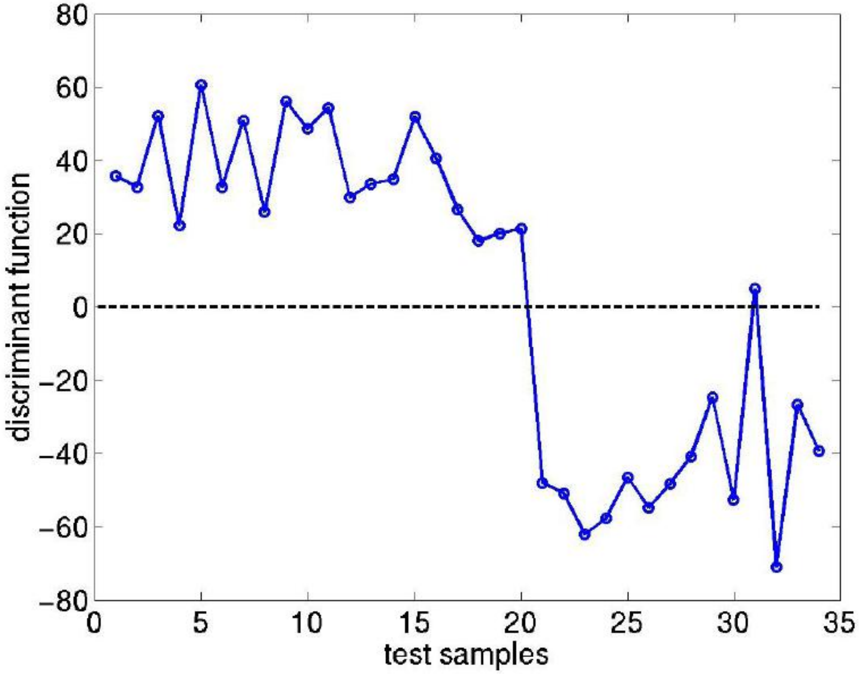
Classification accuracy

- The figure shows the value of the discriminate function

$$f(x) = \log \frac{p(y=1 \mid X)}{p(y=0 \mid X)}$$

across the test examples

- The only test error is also the decision with the lowest confidence



Possible problems with Naive Bayes classifiers: Assumptions

- In most cases, the assumption of conditional independence given the class label is violated

  - much more likely to find the word ‘Barack’ if we saw the word ‘Obama’ regardless of the class
- This is, unfortunately, a major shortcoming which makes these classifiers inferior in many real world applications (though not always)
- There are models that can improve upon this assumption without using the full conditional model (one such model are Bayesian networks which we will discuss later in this class).