Lecture 4: Classification. Logistic Regression. Naive Bayes

- Classification tasks
- Error functions for classification
- Generative vs. discriminative learning
- Naive Bayes
- If we have time: Logistic regression

Recall: Classification problems

- Given a data set $\langle \mathbf{x}_i, y_i \rangle$, where y_i *are discrete*, find a hypothesis which "best fits" the data
- If $y_i \in \{0, 1\}$, this is *binary classification* (very useful special case)
- If y_i can take more than two values, we have *multi-class classification*
- Multi-class versions of most binary classification algorithms can be developed

Example: Text classification

- A very important practical problem, occurring in many different applications: information retrieval, spam filtering, news filtering, building web directories etc.
- A simplified problem description:
  - Given: a collection of documents, classified as “interesting” or “not interesting” by people
  - Goal: learn a classifier that can look at the text of a new document and provide a label for it, without human intervention
- How do we represent the data (documents)?

A simple data representation

- Consider all the possible "significant" words that can occur in the documents (words in the English dictionary, proper names, abbreviations)
- Typically, words that appear in all documents (called *stopwords*) are not considered (prepositions, common verbs like "to be", "to do"...)
- In another preprocessing step, words are mapped to their root (process called *stemming*)

  E.g. learn, learned, learning are all represented by the root "learn"
- For each root, introduce a corresponding *binary feature*, specifying whether the word is present or not in the document.

Example

“Machine learning is fun” $\Longrightarrow$

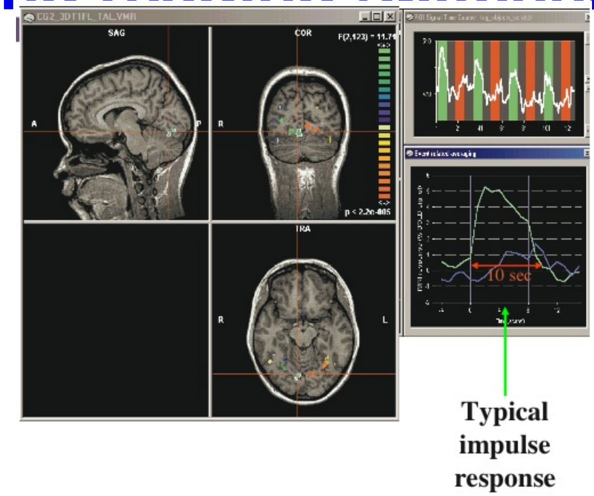
a	0
aardvark	0
⋮	⋮
fun	1
funel	0
⋮	⋮
learn	1
⋮	⋮
machine	1
⋮	⋮
zebra	0

What is special about this task?

- Lots of features! ≈ 100000 for any reasonable domain
- The feature vector is very sparse (a lot of 0 entries)
- It is difficult to get labeled data!

  This process is done by people, hence is very time consuming and tedious

Example: Mind reading (Mitchell et al., 2008)

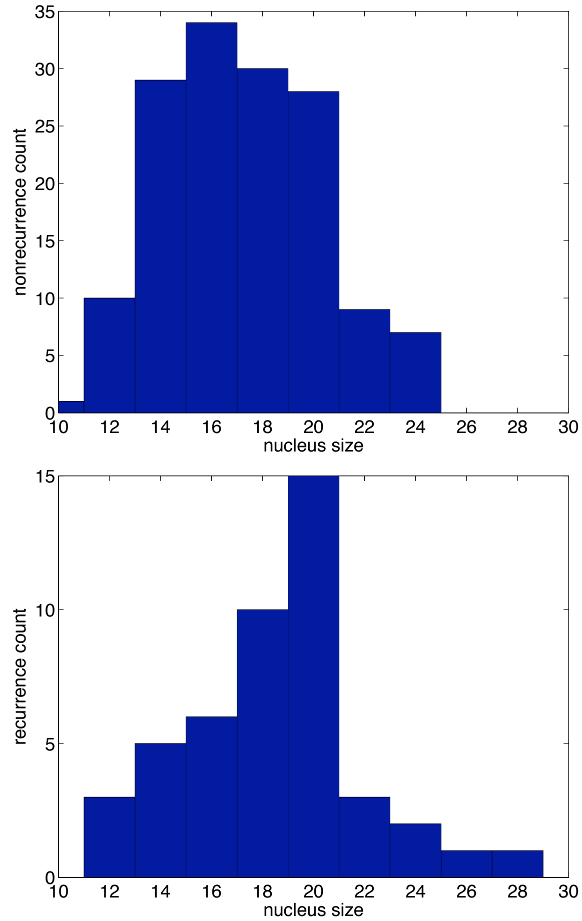


- Given MRI scans, identify what the person is thinking about
  - Roughly 15,000 voxels/image (1mm resolution)
  - 2 images/sec.
- E.g., people words vs. animal words

Classifier learning algorithms

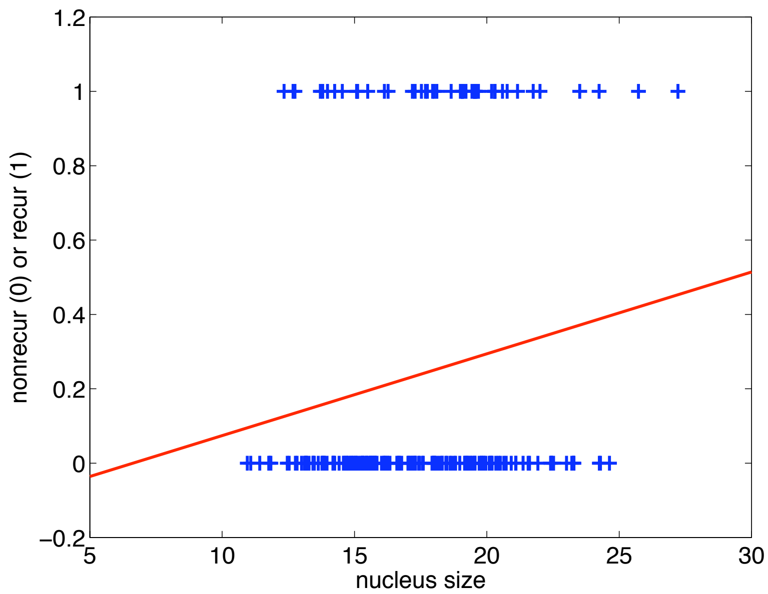
- What is a good error function for classification?
- What hypothesis classes can we use?
- What algorithms are useful for searching those hypotheses classes?

Classification problem example: Given “nucleus size” predict non/recurrrence



Solution by linear regression

- Univariate real input: nucleus size
- Output coding: non-recurrence = 0, recurrence = 1
- Sum squared error minimized



Linear regression for classification

- The predictor shows an increasing trend towards recurrence with larger nucleus size, as expected.
- Output *cannot be directly interpreted* as a class prediction.
- Thresholding output (e.g., at 0.5) could be used to predict 0 or 1. (In this case, prediction would be 0 except for extremely large nucleus size.)
- Output could be interpreted as probability. (Except that probabilities above 1 and below 0 may be output.)

We'd like a way of learning that is more suited for the problem

Probabilistic view

- Suppose we have two possible classes: $y \in \{0, 1\}$.
- What is the probability of a given input $\mathbf{x}$ to have class $y = 1$?
- Bayes Rule:

$$
\begin{aligned}
P(y=1|\mathbf{x}) &= \frac{P(\mathbf{x}, y=1)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|y=1)P(y=1)}{P(\mathbf{x}|y=1)P(y=1) + P(\mathbf{x}|y=0)P(y=0)} \\
&= \frac{1}{1+\exp(-a)} = \sigma(a)
\end{aligned}
$$

where

$$a = \ln \frac{P(\mathbf{x}|y=1)P(y=1)}{P(\mathbf{x}|y=0)P(y=0)}$$

- σ is the sigmoid function (also called "squashing") function
- a is the log-odds of the data being class 1 vs. class 0

Modelling for binary classification

$$P(y = 1|\mathbf{x}) = \sigma(\ln \frac{P(\mathbf{x}|y = 1)P(y = 1)}{P(\mathbf{x}|y = 0)P(y = 0)})$$

- One approach is to model $P(y)$ and $P(\mathbf{x}|y)$, then use the approach above for classification
- This is called *generative learning*, because we can actually use the model to generate (i.e. fantasize) data
- Another idea is to model directly $P(y|\mathbf{x})$
- This is called *discriminative learning*, because we only care about discriminating (i.e. separating) examples of the two classes.

Implementing the idea for document classification

- We can compute $P(y)$ by counting the number of interesting and uninteresting documents we have
- How do we compute $P(\mathbf{x}|y)$?
- Assuming about 100000 words, and not too many documents, this is hopeless!

  Most possible combinations of words will not appear in the data at all...
- Hence, we need to make some extra assumptions.

Naive Bayes assumption

- Suppose the features x_i are discrete
- Assume the x_i *are conditionally independent given* y.
- In other words, assume that:

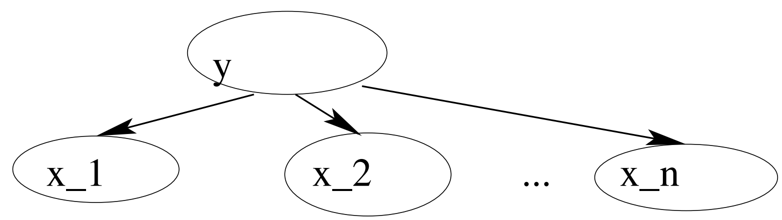
$$P(x_i|y) = P(x_i|y, x_j), \forall i, j$$

- Then we have:

$$\begin{aligned} P(x_1, x_2, \ldots x_m|y) &= P(x_1|y)P(x_2|y, x_1) \cdots P(x_m|y, x_1, \ldots x_{m-1}) \\ &= P(x_1|y)P(x_2|y) \ldots P(x_m|y) \end{aligned}$$

- For binary features, instead of $O(2^n)$ numbers to describe a model, we only need $O(n)$!

A graphical representation of the naive Bayesian model



- The nodes corresponding to x_i are parameterized by $P(x_i|y)$.
- The node corresponding to y is parameterized by $P(y)$

Naive Bayes for binary features

- The parameters of the model are $\theta_{i,1} = P(x_i = 1|y = 1)$, $\theta_{i,0} = P(x_i = 1|y = 0)$, $\theta_1 = P(y = 1)$
- What is the decision surface?

$$\frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})} = \frac{P(y=1)\prod_{i=1}^{n} P(x_i|y=1)}{P(y=0)\prod_{i=1}^{n} P(x_i|y=0)}$$

- Using the log trick, we get:

$$\log\frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})} = \log\frac{P(y=1)}{P(y=0)} + \sum_{i=1}^{n}\log\frac{P(x_i|y=1)}{P(x_i|y=0)}$$

- Note that in the equation above, the x_i would be 1 or 0

Decision boundary of naive Bayes with binary features

$$
\begin{aligned}
\text{Let: } w_0 &= \log \frac{P(y=1)}{P(y=0)} \\
w_{i,1} &= \log \frac{P(x_i=1|y=1)}{P(x_i=1|y=0)} \\
w_{i,0} &= \log \frac{P(x_i=0|y=1)}{P(x_i=0|y=0)}
\end{aligned}
$$

We can re-write the decision boundary as:

$$
\log \frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})} = w_0 + \sum_{i=1}^{n} (w_{i,1}x_i + w_{i,0}(1-x_i)) = w_0 + \sum_{i=1}^{n} w_{i,0} + \sum_{i=1}^{n} (w_{i,1} - w_{i,0}) x_i
$$

This is a *linear decision boundary!*

Learning the parameters of a naive Bayes classifier

- Use *maximum likelihood*!
- The likelihood in this case is:

$$L(\theta_1, \theta_{i,1}, \theta_{i,0}) = \prod_{j=1}^{m} P(\mathbf{x}_j, y_j) = \prod_{j=1}^{m} P(y_j) \prod_{i=1}^{n} P(x_{j,i}|y_j)$$

- First, use the log trick:

$$\log L(\theta_1, \theta_{i,1}, \theta_{i,0}) = \sum_{j=1}^{m} \left(\log P(y_j) + \sum_{i=1}^{n} \log P(x_{j,i}|y_j) \right)$$

- Like above, observe that each term in the sum depends on the values of y_j, $\mathbf{x}_j$ that appear in the jth instance

Maximum likelihood parameter estimation for naive Bayes

$$\begin{aligned}
\log L(\theta_1, \theta_{i,1}, \theta_{i,0}) &= \sum_{j=1}^{m} [y_j \log \theta_1 + (1 - y_j) \log(1 - \theta_1) \\
&+ \sum_{i=1}^{n} y_j \left(x_{ji} \log \theta_{i,1} + (1 - x_{ji}) \log(1 - \theta_{i,1}) \right) \\
&+ \sum_{i=1}^{n} (1 - y_j) \left(x_{ji} \log \theta_{i,0} + (1 - x_{ji}) \log(1 - \theta_{i,0}) \right)]
\end{aligned}$$

To estimate θ_1, we take the derivative of $\log L$ wrt θ_1 and set it to 0:

$$\frac{\partial L}{\partial \theta_1} = \sum_{j=1}^{m} \left(\frac{y_j}{\theta_1} + \frac{1 - y_j}{1 - \theta_1}(-1) \right) = 0$$

Maximum likelihood parameters estimation for naive Bayes

By solving for θ_1, we get:

$$\theta_1 = \frac{1}{m}\sum_{j=1}^{m} y_j = \frac{\text{number of examples of class 1}}{\text{total number of examples}}$$

Using a similar derivation, we get:

$$\theta_{i,1} = \frac{\text{number of instances for which } x_{j,i} = 1 \text{ and } y_j = 1}{\text{number of instances for which } y_j = 1}$$

$$\theta_{i,0} = \frac{\text{number of instances for which } x_{j,i} = 1 \text{ and } y_j = 0}{\text{number of instances for which } y_j = 0}$$

Text classification revisited

- Consider again the text classification example, where the features x_i correspond to words
- Using the approach above, we can compute probabilities for all the words which appear in the document collection
- But what about words that do not appear?

  They would be assigned zero probability!
- As a result, the probability estimates for documents containing such words would be $0/0$ for both classes, and hence no decision can be made

Laplace smoothing

- Instead of the maximum likelihood estimate:

$$\theta_{i,1} = \frac{\text{number of instances for which } x_{j,i} = 1 \text{ and } y_j = 1}{\text{number of instances for which } y_j = 1}$$

use:

$$\theta_{i,1} = \frac{(\text{number of instances for which } x_{j,i} = 1 \text{ and } y_j = 1) + 1}{(\text{number of instances for which } y_j = 1) + 2}$$

- Hence, if a word does not appear at all in the documents, it will be assigned prior probability 0.5.
- If a word appears in a lot of documents, this estimate is only slightly different from max. likelihood.
- This is an example of *Bayesian prior* for Naive Bayes (more on this later)

Example: 20 newsgroups

Given 1000 training documents from each group, learn to classify new documents according to which newsgroup they came from

comp.graphics	misc.forsale
comp.os.ms-windows.misc	rec.autos
comp.sys.ibm.pc.hardware	rec.motorcycles
comp.sys.mac.hardware	rec.sport.baseball
comp.windows.x	rec.sport.hockey
alt.atheism	sci.space
soc.religion.christian	sci.crypt
talk.religion.misc	sci.electronics
talk.politics.mideast	sci.med
talk.politics.misc	talk.politics.guns

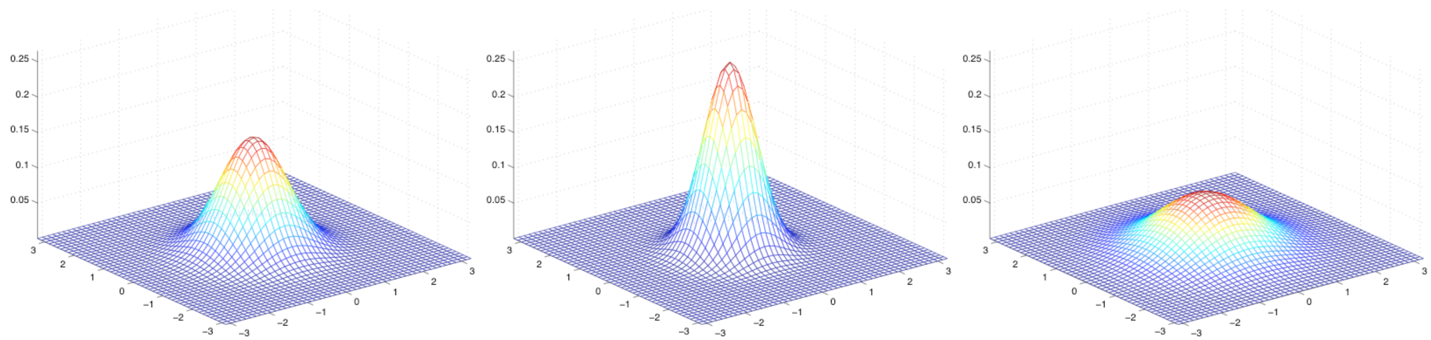
Naive Bayes: 89% classification accuracy - comparable to other state-of-art methods

Gaussian Naive Bayes

- This is a generative model for continuous inputs (also known as Gaussian Discriminant Analysis)
- $P(y)$ is still assumed to be binomial
- $P(\mathbf{x}|y)$ is assumed to be a multivariate Gaussian (normal distribution), with mean $\mu \in \mathbb{R}^n$ and covariance $\Sigma \in \mathbb{R}^n \times \mathbb{R}^n$.

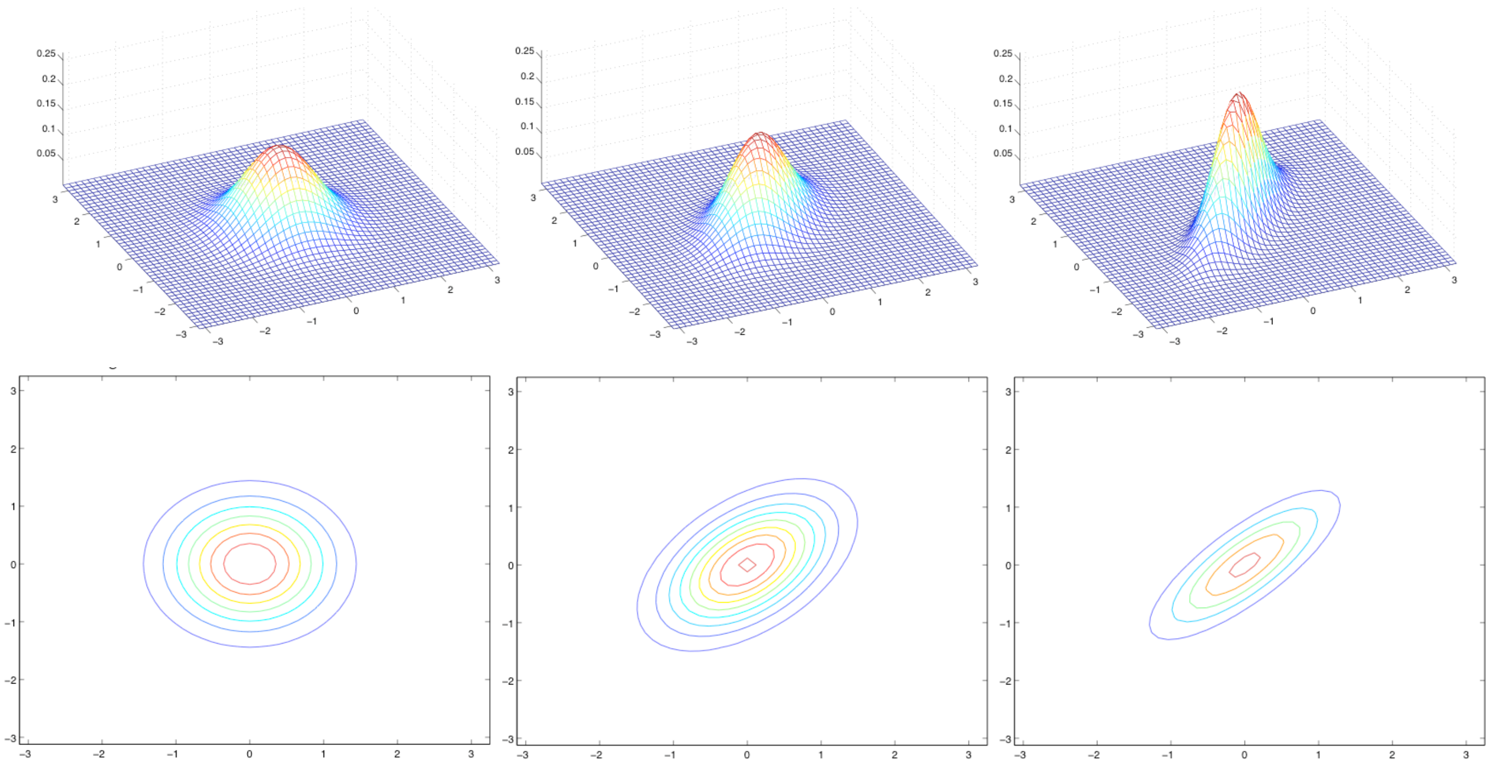
Example

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix} \qquad \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$



Example

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$



Gaussian Naive Bayes model

- The class label is modeled as $P(y) = \theta^y(1-\theta)^{1-y}$ (binomial, like before)
- The two classes are modeled as:

$$P(\mathbf{x}|y=0) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_{\mathbf{0}})^T\Sigma^{-1}(\mathbf{x}-\mu_{\mathbf{0}})\right)$$

$$P(\mathbf{x}|y=1) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_{\mathbf{1}})^T\Sigma^{-1}(\mathbf{x}-\mu_{\mathbf{1}})\right)$$

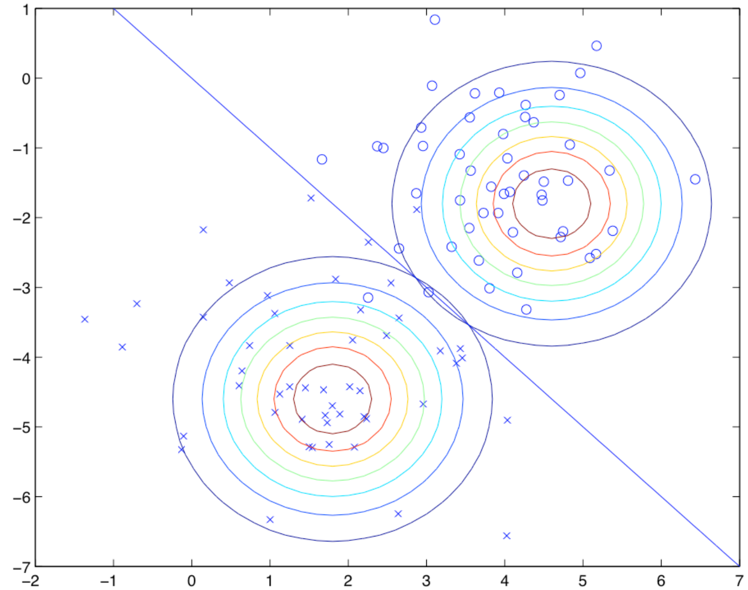
- The parameters to estimate are: θ, $\mu_{\mathbf{0}}$, $\mu_{\mathbf{1}}$, Σ
- Note that the covariance is considered the same!

Determining the parameters

- We can write down the likelihood function, like before
- We take the derivatives wrt the parameters and set them to 0
- The parameter θ is just the empirical frequency of class 1
- The means $\mu_{\mathbf{0}}$ and $\mu_{\mathbf{1}}$ are just the empirical means for examples of class 0 and 1 respectively
- The covariance matrix is the empirical estimate from the data:

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \mu_{y_i})(\mathbf{x}_i - \mu_{y_i})^T$$

Example

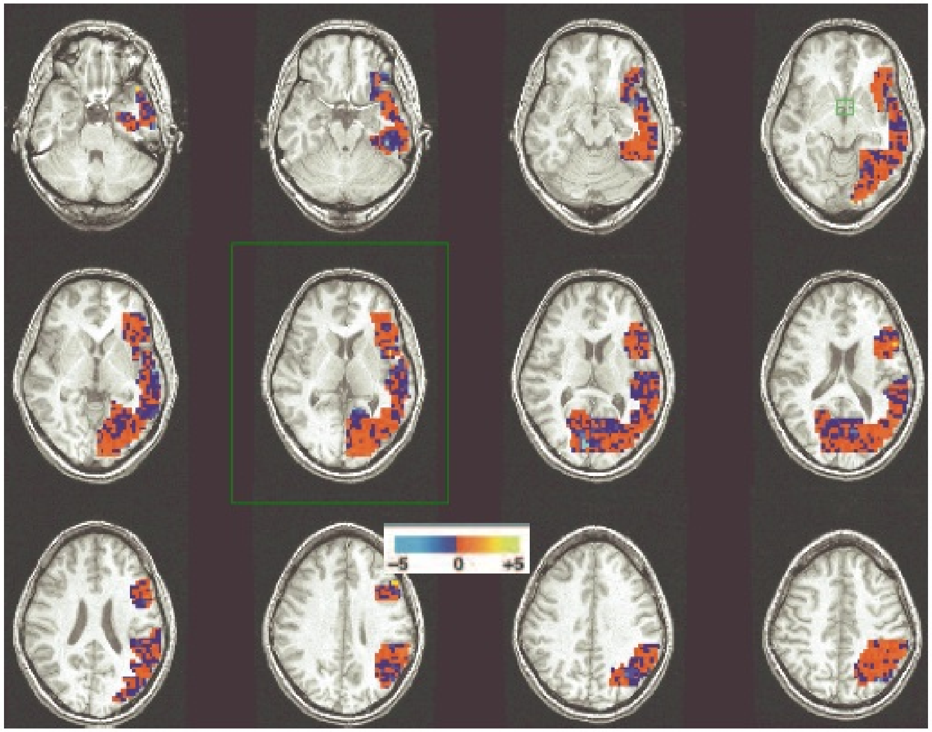


The line is the decision boundary between the classes (on the line, both have equal probability)

Other variations

- Covariance matrix can be different for the two classes, if we have enough data to estimate it
- Covariance matrix can be restricted to diagonal, or mostly diagonal with few off-diagonal elements, based on prior knowledge.
- The shape of the covariance is influenced both by assumptions about the domain and by the amount of data available.

Mind reading revisited: Word models



Mind reading revisited: Average class distributions