3F3: Signal and Pattern Processing

Lecture 3: Classification

Zoubin Ghahramani
zoubin@eng.cam.ac.uk

**Department of Engineering
University of Cambridge**

Lent Term

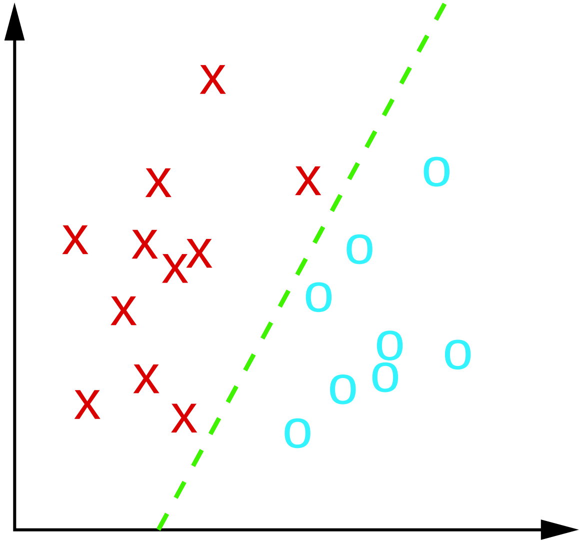
Classification

We will represent data by vectors in some vector space.
Let $\mathbf{x}$ denote a data point with elements $\mathbf{x} = (x_1, x_2, \ldots, x_D)$
The elements of $\mathbf{x}$, e.g. x_d, represent measured (observed) features of the data point; D denotes the number of measured features of each point.

The data set $\mathcal{D}$ consists of N pairs of data points and corresponding discrete class labels:

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}) \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$$

where $y^{(n)} \in \{1, \ldots, C\}$ and C is the number of classes.
The goal is to classify new inputs correctly (i.e. to *generalize*).



Examples:

- spam vs non-spam
- normal vs disease
- 0 vs 1 vs 2 vs 3 ... vs 9

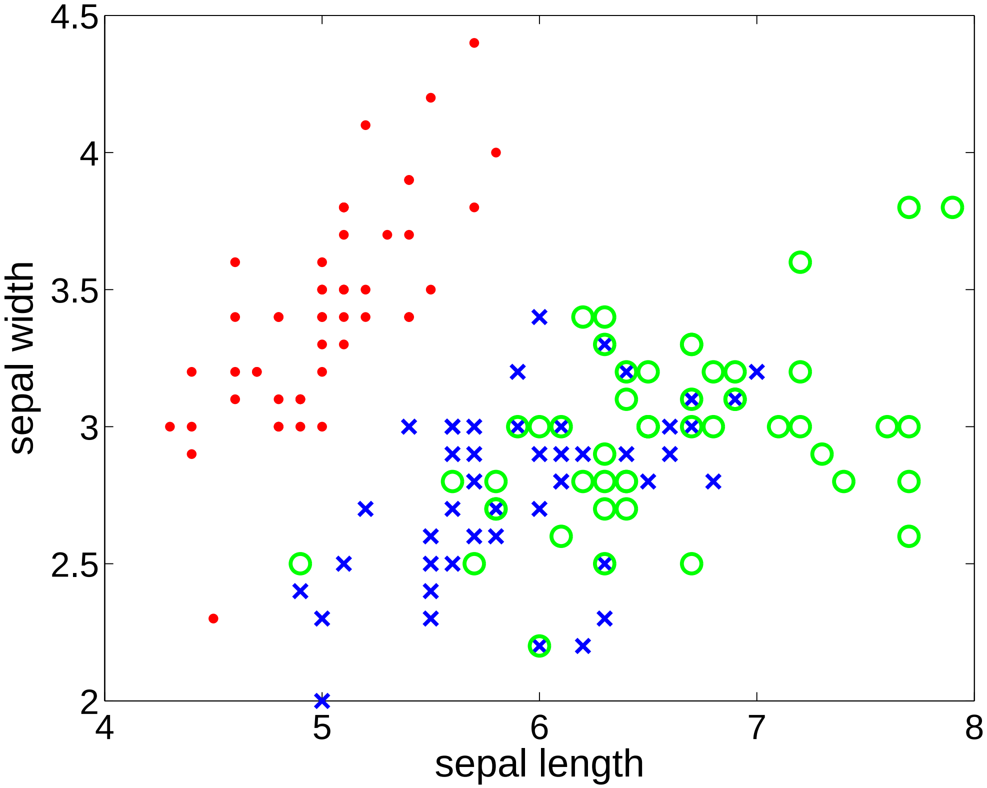
Classification: Example Iris Dataset

3 classes, 4 numeric attributes, 150 instances
A data set with 150 points and 3 classes. Each point is a random sample of measurements of flowers from one of three iris species—setosa, versicolor, and virginica—collected by Anderson (1935). Used by Fisher (1936) for linear discriminant function technique.

The measurements are sepal length, sepal width, petal length, and petal width in cm.

Data:

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
...
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
...
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
```



Linear Classification

Data set $\mathcal{D}$:

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}) \dots, (\mathbf{x}^{(N)}, y^{(N)})\}$$

Assume $y^{(n)} \in \{0, 1\}$ (i.e. two classes) and $\mathbf{x}^{(n)} \in \Re^D$, and let $\tilde{\mathbf{x}}^{(n)} = \left(\begin{array}{c} \mathbf{x}^{(n)} \\ 1 \end{array} \right)$.

Linear Classification (Deterministic)

$$y^{(n)} = H\left(\beta^\top \tilde{\mathbf{x}}^{(n)}\right)$$

where $H(z) = \left\{ \begin{array}{ll} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{array} \right.$ is known as the Heaviside, threshold, or step function.

Linear Classification

Data set $\mathcal{D}$:

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}) \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$$

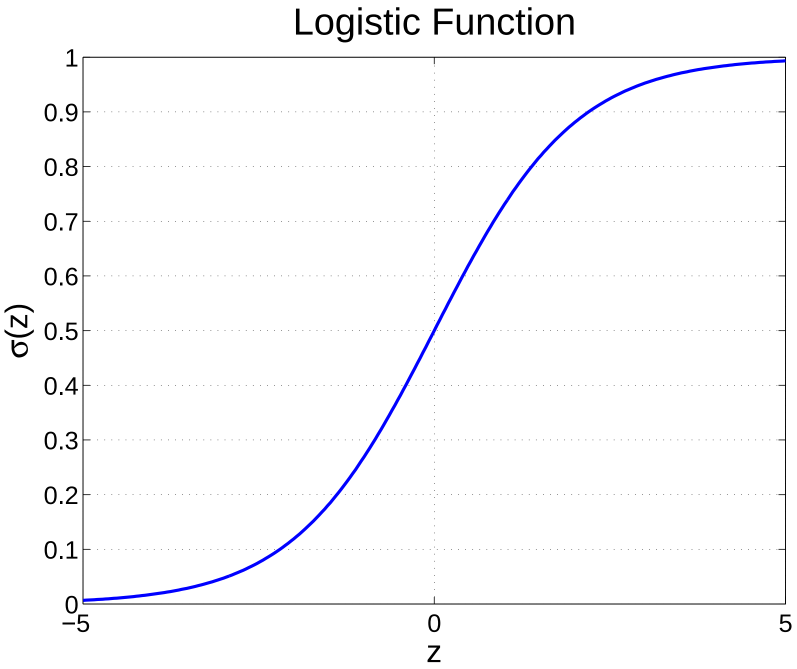
Assume $y^{(n)} \in \{0, 1\}$ (i.e. two classes) and $\mathbf{x}^{(n)} \in \Re^D$, and let $\tilde{\mathbf{x}}^{(n)} = \begin{pmatrix} \mathbf{x}^{(n)} \\ 1 \end{pmatrix}$.

Linear Classification (Probabilistic)

$$P(y^{(n)} = 1 | \tilde{\mathbf{x}}^{(n)}, \beta) = \sigma\left(\beta^\top \tilde{\mathbf{x}}^{(n)}\right)$$

where $\sigma(\cdot)$ is a monotonic increasing function $\sigma : \Re \to [0, 1]$.

For example if $\sigma(z)$ is $\sigma(z) = \frac{1}{1+\exp(-z)}$, the sigmoid or logistic function, this is called logistic regression. Compare this to...



Linear Regression

$$y^{(n)} = \beta^\top \tilde{\mathbf{x}}^{(n)} + \epsilon_n$$

Logistic Classification[1]

Data set: $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}) \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$.

$$\begin{aligned} P(y^{(n)} = 1|\tilde{\mathbf{x}}^{(n)}, \beta) &= \sigma\left(\beta^\top \tilde{\mathbf{x}}^{(n)}\right) \\ P(y^{(n)} = 0|\tilde{\mathbf{x}}^{(n)}, \beta) &= 1 - P(y^{(n)} = 1|\tilde{\mathbf{x}}^{(n)}, \beta) = \sigma\left(-\beta^\top \tilde{\mathbf{x}}^{(n)}\right) \end{aligned}$$

Since $\sigma(z) = \frac{1}{1+\exp(-z)}$, note that $1 - \sigma(z) = \sigma(-z)$.

This defines a Bernoulli distribution over y at each point $\mathbf{x}$. (cf $\theta^y(1-\theta)^{(1-y)}$)

The likelihood for β is:

$$P(\mathbf{y}|\mathbf{X}, \beta) = \prod_n P(y^{(n)}|\tilde{\mathbf{x}}^{(n)}, \beta) = \prod_n \sigma(\beta^\top \tilde{\mathbf{x}}^{(n)})^{y^{(n)}}(1 - \sigma(\beta^\top \tilde{\mathbf{x}}^{(n)}))^{(1-y^{(n)})}$$

Given the data, this is just some function of β which we can optimize...

[1] This is usually called "Logistic Regression" but it's really solving a classification problem, so I'm going to correct this misnomer.

Logistic Classification

Maximum Likelihood learning:

$$\begin{aligned}\mathcal{L}(\beta) &= \ln P(\mathbf{y}|\mathbf{X},\beta) = \sum_n \ln P(y^{(n)}|\tilde{\mathbf{x}}^{(n)},\beta) \\ &= \sum_n \left[y^{(n)} \ln \sigma(\beta^\top \tilde{\mathbf{x}}^{(n)}) + (1-y^{(n)}) \ln \sigma(-\beta^\top \tilde{\mathbf{x}}^{(n)}) \right]\end{aligned}$$

Taking derivatives, using $\frac{\partial \ln \sigma(z)}{\partial z} = \sigma(-z)$, and defining $z_n \overset{\mathrm{def}}{=} \beta^\top \tilde{\mathbf{x}}^{(n)}$, we get:

$$\begin{aligned}\frac{\partial \mathcal{L}(\beta)}{\partial \beta} &= \sum_n \left[y^{(n)}\sigma(-z_n)\tilde{\mathbf{x}}^{(n)} - (1-y^{(n)})\sigma(z_n)\tilde{\mathbf{x}}^{(n)} \right] \\ &= \sum_n \left[y^{(n)}\sigma(-z_n)\tilde{\mathbf{x}}^{(n)} + y^{(n)}\sigma(z_n)\tilde{\mathbf{x}}^{(n)} - \sigma(z_n)\tilde{\mathbf{x}}^{(n)} \right] \\ &= \sum_n (y^{(n)} - \sigma(z_n))\tilde{\mathbf{x}}^{(n)}\end{aligned}$$

Intuition?

Logistic Classification

The gradient:

$$\frac{\partial \mathcal{L}(\beta)}{\partial \beta} = \sum_n (y^{(n)} - \sigma(z_n))\tilde{\mathbf{x}}^{(n)}$$

A learning rule (steepest gradient ascent in the likelihood):

$$\beta^{[t+1]} = \beta^{[t]} + \eta \frac{\partial \mathcal{L}(\beta^{[t]})}{\partial \beta^{[t]}}$$

where η is a learning rate

Which gives:

$$\beta^{[t+1]} = \beta^{[t]} + \eta \sum_n (y^{(n)} - \sigma(z_n))\tilde{\mathbf{x}}^{(n)}$$

This is a batch learning rule, since all N points are considered at once.

An online learning rule, can consider one data point at a time...

$$\beta^{[t+1]} = \beta^{[t]} + \eta(y^{(t)} - \sigma(z_t))\tilde{\mathbf{x}}^{(t)}$$

This is useful for real-time learning and adaptation applications.

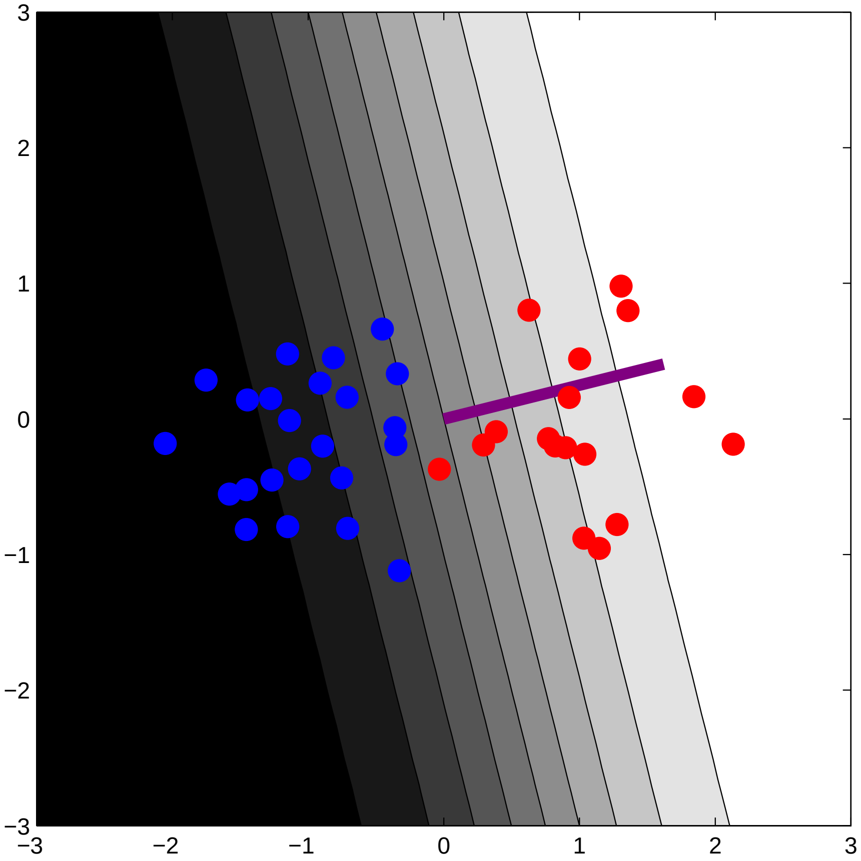
One can also use any standard optimizer (e.g. conjugate gradients, Newton's method).

Logistic Classification

Demo of the online logistic classification learning rule:

$$\beta^{[t+1]} = \beta^{[t]} + \eta(y^{(t)} - \sigma(z_t))\tilde{\mathbf{x}}^{(t)}$$

where $z_t = \beta^{[t]\top}\tilde{\mathbf{x}}^{(t)}$.



lrdemo

Maximum a Posteriori (MAP) and Bayesian Learning

MAP: Assume we define a prior on β, for example a Gaussian prior with variance λ:

$$P(\beta) = \frac{1}{\sqrt{(2\pi\lambda)^{D+1}}} \exp\{-\frac{1}{2\lambda}\beta^\top\beta\}$$

The instead of maximizing the likelihood we can maximize the posterior probability of β

$$\begin{aligned} \ln P(\beta|\mathcal{D}) &= \ln P(\mathcal{D}|\beta) + \ln P(\beta) + \text{const} \\ &= \mathcal{L}(\beta) - \frac{1}{2\lambda}\beta^\top\beta + \text{const'} \end{aligned}$$

The second term penalizes the length of β.

Why would we want to do this?

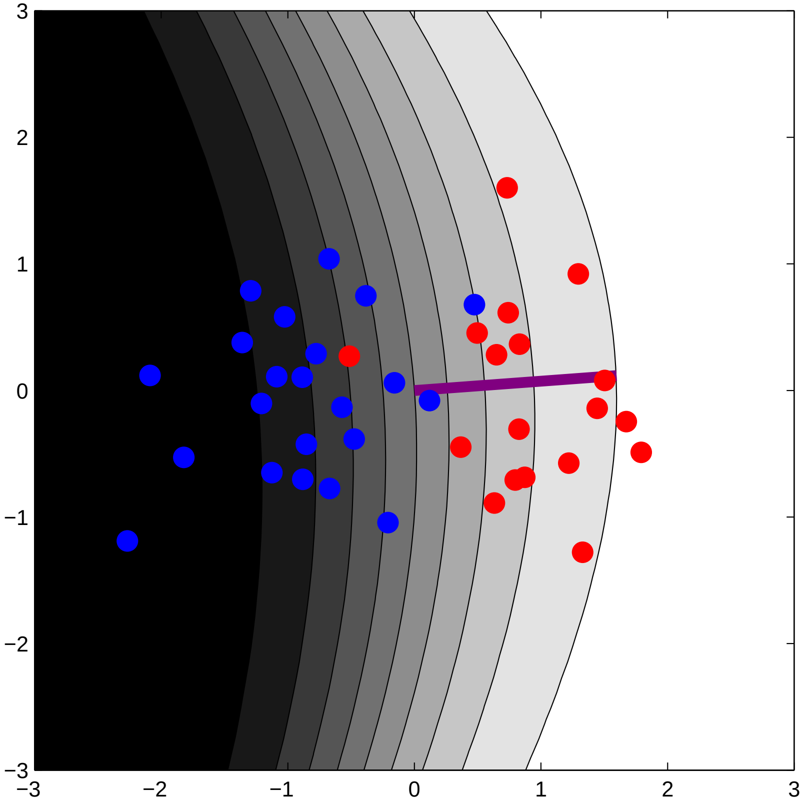
Bayesian Learning: We can also do Bayesian learning of β by trying to compute or approximate $P(\beta|\mathcal{D})$ rather than optimize it.

Nonlinear Logistic Classification

Easy: just map the inputs into a higher dimensional space by computing some nonlinear functions of the inputs. For example:

$$(x_1, x_2) \rightarrow (x_1, x_2, x_1 x_2, x_1^2, x_2^2)$$

Then do logistic classification using these new inputs.



nlrdemo

Multinomial Classification

How do we do multiple classes?

$$y \in \{1, \ldots, C\}$$

Answer: Use a softmax function which generalizes the logistic function:

Each class c has a vector β_c

$$P(y = c|\mathbf{x}, \beta) = \frac{\exp\{\beta_c^\top \mathbf{x}\}}{\sum_{c'=1}^{C} \exp\{\beta_{c'}^\top \mathbf{x}\}}$$

Some exercises

1. show how using the softmax function is equivalent to doing logistic classification when $C = 2$.

2. implement the online logistic classification rule in Matlab/Octave.

3. implement the nonlinear extension.

4. investigate what happens when the classes are well separated, and what role the prior has in MAP learning.