An introduction to key concepts in Machine Learning
Part 2

Alberto Paccanaro
EMAp – FGV

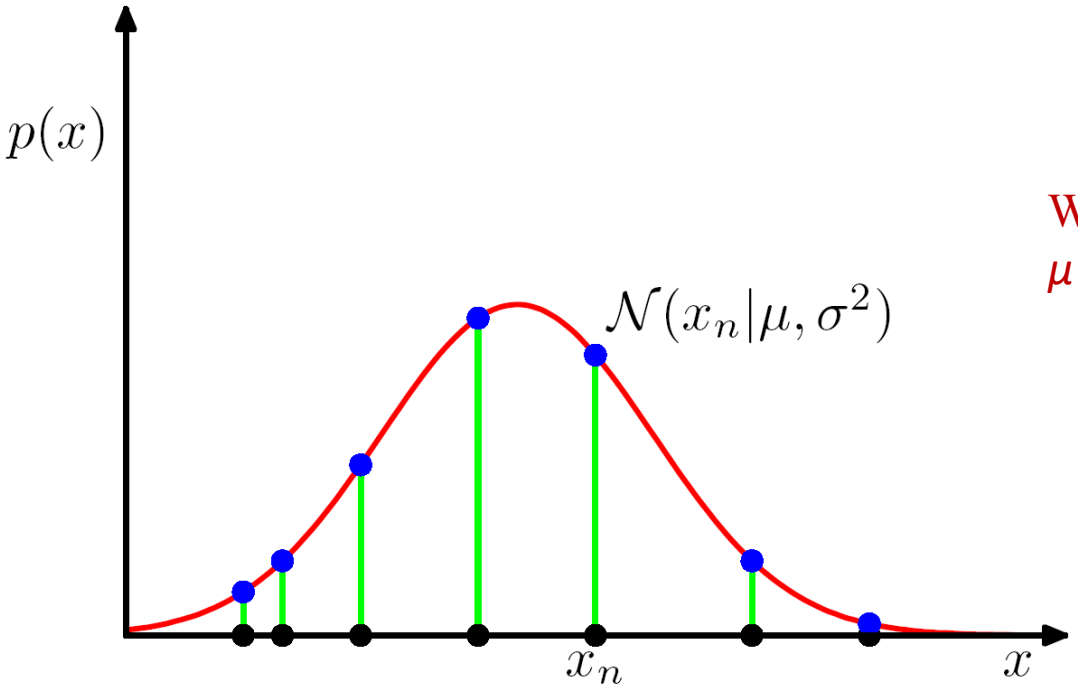
www.paccanarolab.org

Material and images in these slides are from (or adapted from):
C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006

Probability of a dataset (Gaussian distributed)

Dataset of observations: $\mathbf{x} = (x_1, \ldots, x_N)$ drawn independently from a Gaussian distribution
(i.i.d. = *independent and identically distributed)*

$$p(\mathbf{x}|\mu, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}\left(x_n|\mu, \sigma^2\right)$$



When viewed as a function of μ and σ^2, this is the likelihood function for the Gaussian

Which values for μ and σ² maximize the likelihood function?

Simple parametric method !

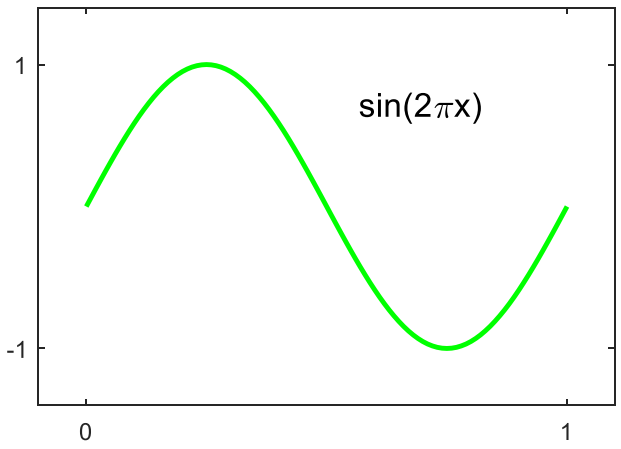
$$\ln p\left(\mathbf{x}|\mu,\sigma^2\right) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)$$

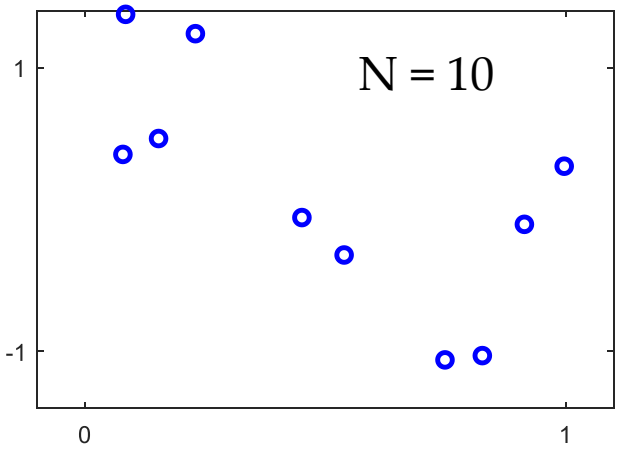
Maximizing with respect to μ and σ² we get:

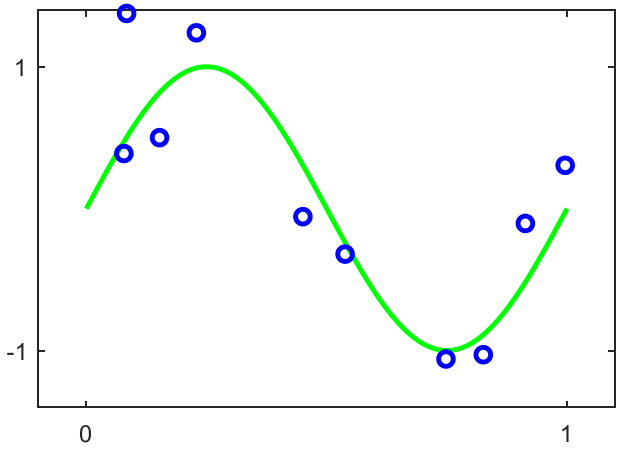
$$\mu_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N} x_n \qquad \sigma^2_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu_{\mathrm{ML}})^2$$

So the optimal values are: **sample mean** and **sample variance**!

Polynomial curve fitting, again…







We have seen it in terms of error minimization.

Now we view it from a probabilistic perspective.

We want to answer questions like:

if $x = 0.5738$, $y = ?$

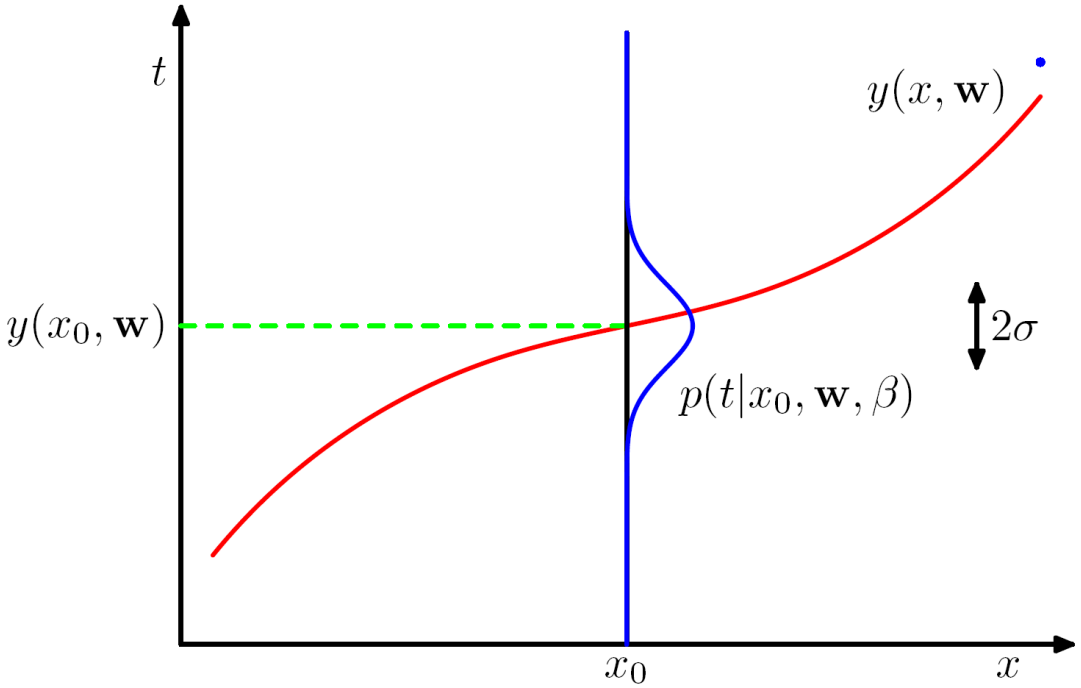
$$\mathbf{x} \equiv (x_1, \ldots, x_N)^{\mathrm{T}}$$
$$\mathbf{t} \equiv (t_1, \ldots, t_N)^{\mathrm{T}}$$

Express our uncertainty over the value of the target variable using a probability distribution.

Given the value of x, the corresponding t has a Gaussian distribution with a mean equal to the value $y(x,\mathbf{w})$

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}\left(t|y(x, \mathbf{w}), \beta^{-1}\right)$$

$\beta = \sigma^{-1}$



Use the training data $\{\mathbf{x}, \mathbf{t}\}$ to determine the values of the unknown parameters $\mathbf{w}$ and β by **maximum likelihood**

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n | y(x_n, \mathbf{w}), \beta^{-1}\right)$$

Let's work on the $\mathbf{w}$ first:

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = -\frac{\beta}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi)$$

Maximizing the likelihood is equivalent, so far as determining w is concerned, to minimizing the *sum-of-squares error function.*

The sum-of-squares error function has arisen as a consequence of maximizing likelihood under the assumption of a Gaussian noise distribution.

- So we obtain **w** (closed form, as we did before).
- We can also work out a value for β

$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}_{\mathrm{ML}}) - t_n\}^2$$

Predictions for new values of *x*, in terms of the *predictive distribution:*

$$p(t|x, \mathbf{w}_{\mathrm{ML}}, \beta_{\mathrm{ML}}) = \mathcal{N}\left(t|y(x, \mathbf{w}_{\mathrm{ML}}), \beta_{\mathrm{ML}}^{-1}\right)$$

So far we use ML.
What happens if we plug in Bayes theorem?

We can determine **w** by finding the most probable value of **w** given the data (*Maximum Posterior* or *MAP*).

$$p(\mathbf{w}|\mathbf{x}, \mathbf{t}, \alpha, \beta) \propto p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha)$$

- Using a Gaussian prior over **w,** maximum of the posterior is given by the minimum of:

$$\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

Maximizing the posterior distribution is equivalent to minimizing the *regularized* sum-of-squares error function

Decision Theory

Input **x**, target output **t**

Determination of $p(\mathbf{x}, \mathbf{t})$ from a set of training data is an example of *inference*

In a practice we often must make a specific prediction for **t**

Decision theory to tell us how to make optimal decisions given the appropriate probabilities.

Decision Theory
CLASSIFICATION

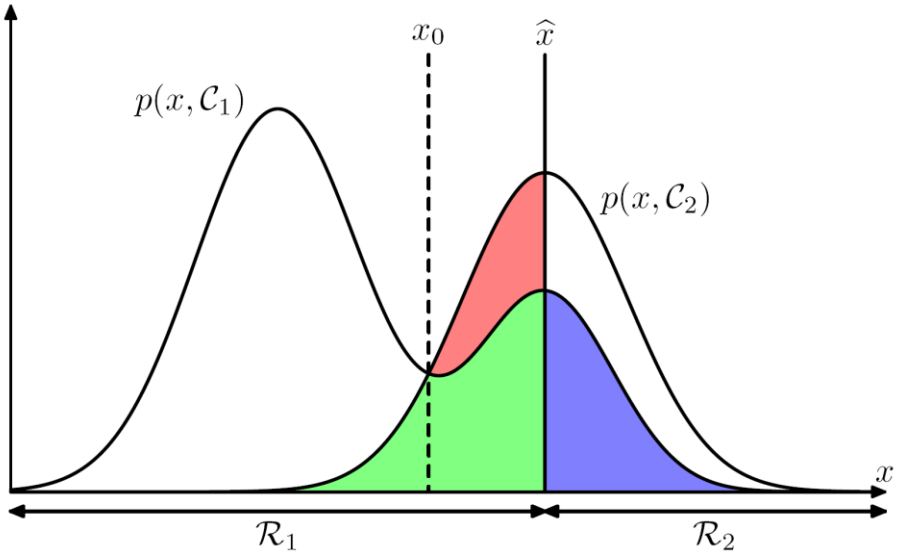
X-ray image of a patient (**x**), and we wish to determine whether the patient has cancer or not (C_1 or C_2).

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$$

Note: all these quantities can be obtained from $p(\mathbf{x}, C_k)$

Intuition: to minimize the chance of assigning **x** to the wrong class, choose the class having the higher posterior probability. Let's show it …

A mistake occurs when an input vector belonging to class C_1 is assigned to class C_2 or vice versa.



$$
\begin{aligned}
p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\
&= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2)\, \mathrm{d}\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1)\, \mathrm{d}\mathbf{x}
\end{aligned}
$$

$p(\mathbf{x}, C_k) = p(C_k \mid \mathbf{x})\, p(\mathbf{x})$, hence…

the minimum probability of making a mistake is obtained if each value of **x** is assigned to the class for which the posterior probability $p(C_k \mid \mathbf{x})$ is largest

Minimizing the expected loss

Cost function: measure of loss incurred in taking decisions.

Our goal: minimize the total loss.

L loss matrix, where $L_{k,j}$ loss that we incur when we classify something of class k *(true class)* as class j *(predicted class)*.

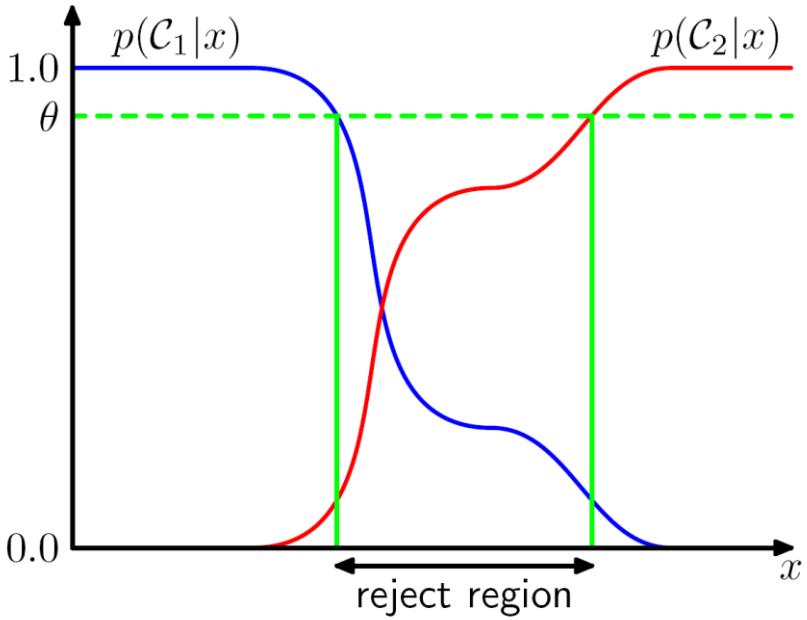
	cancer	normal
cancer	0	1000
normal	1	0

To minimize the expected loss assign each new datapoint **x** to the class j for which the quantity:

$$\sum_k L_{kj} p(\mathcal{C}_k|\mathbf{x})$$

is minimum

Reject option



There might be regions where we are very uncertain about class membership. It might be appropriate to avoid making decisions.

Three approaches for classification

1. **Generative models:** Infer $p(\mathbf{x}|C_k)$ and $p(C_k)$ for each class. Use them to find $p(\mathbf{x})$. Use Bayes theorem to infer $p(C_k|\mathbf{x})$. Use decision theory to determine the class for a new $\mathbf{x}$.

   Here you model the distribution of inputs and outputs. By sampling from them it is possible to *generate* synthetic data points in the input space.

2. **Discriminative models:** Infer $p(C_k|\mathbf{x})$ directly. Use decision theory to determine the class for a new $\mathbf{x}$.

3. **Discriminant function:** learn a function $f(\mathbf{x})$, which maps each input $\mathbf{x}$ directly onto a class label. Probabilities play no role.

Considerations

- Approach 1) is the most demanding, you need to get the distribution over x and C_k. If x has high dimensionality we need a lot of data to obtain $p(\mathbf{x}|C_k)$.
- In 1), having $p(\mathbf{x})$ can be useful for outlier detection
- Approach 3) is the simplest, but we don't have access to $p(C_k|\mathbf{x})$.

Why is it important to have $p(C_k|\mathbf{x})$?

Why probabilities ?

A. **Minimizing risk** – including elements of the loss matrix is trivial. Include revisions of the loss matrix.

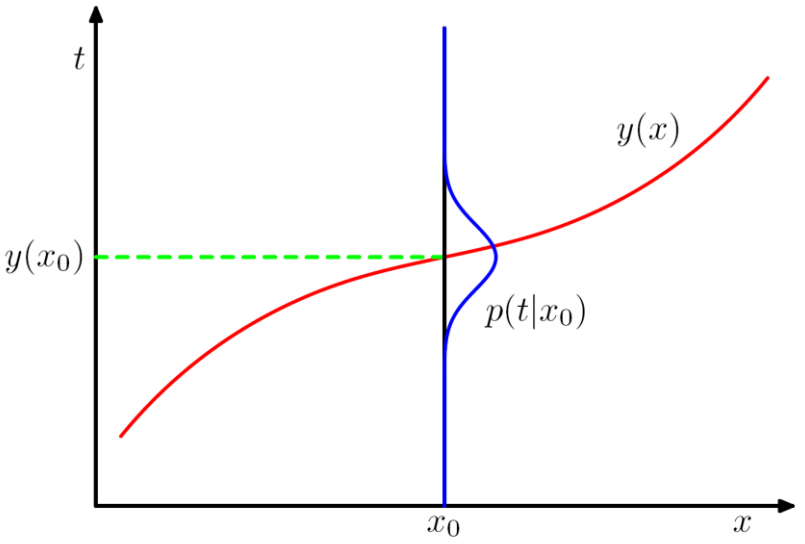
B. **Reject option**

C. **Compensating for class priors** – this allows us to learn from balanced datasets and then compensate for class imbalance in real world data.

D. **Combining models** – for complex applications, we can break the problem into smaller subproblems and then combine the posteriors.

Decision Theory
REGRESSION

The function $y(x)$, which minimizes the expected squared loss, is given by the mean of the conditional distribution $p(t|\mathbf{x})$.



As before, 3 possible approaches:

1. infer $p(\mathbf{x}, t)$, find $p(t|\mathbf{x})$, then find the conditional mean.
2. determine $p(t|\mathbf{x})$, and then find the conditional mean
3. learn a regression function $y(\mathbf{x})$ directly

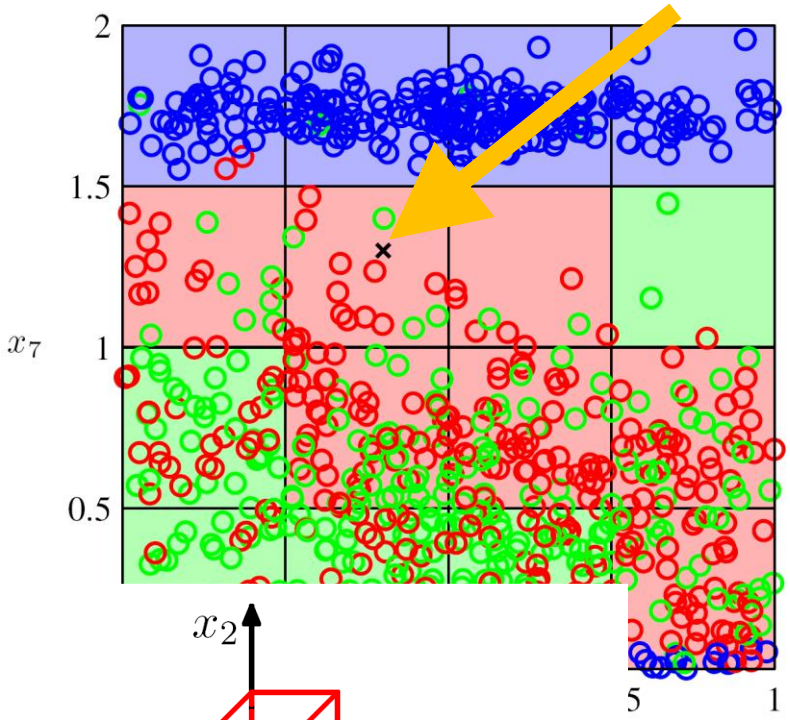
The K-nearest-neighbour algorithm
(a.k.a. K-nearest neighbour, KNN, K-NN, Knn, K-nn)

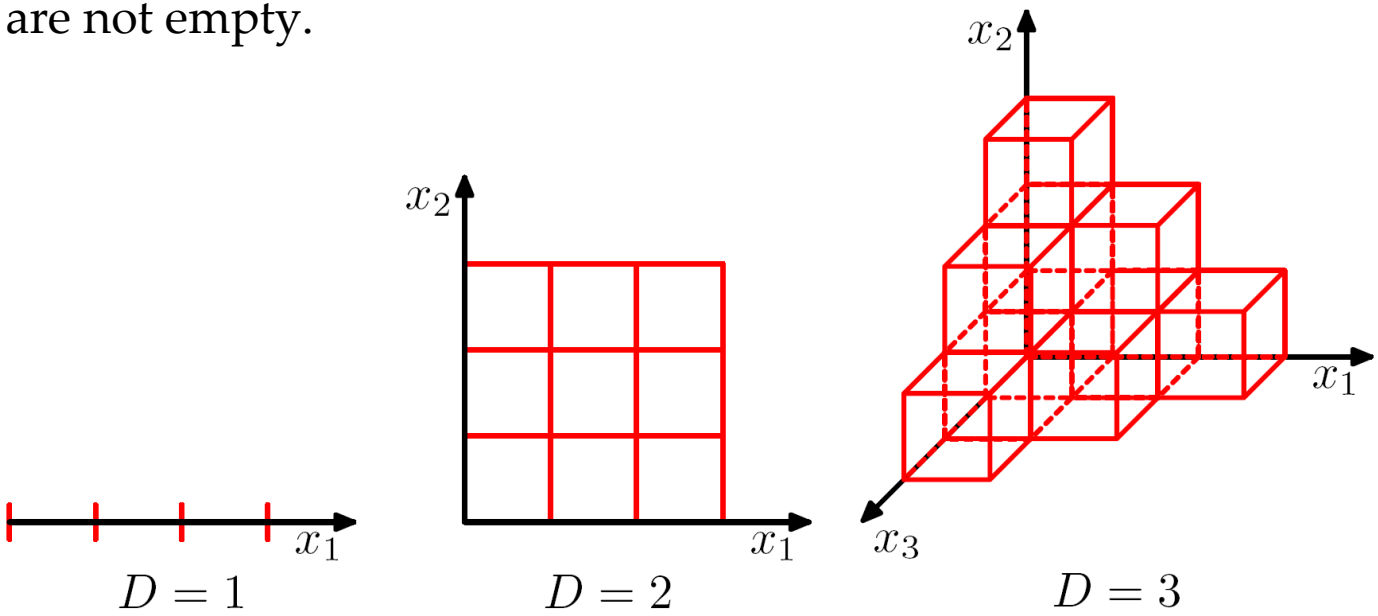
classification

Example: 12 dim points, 3 classes

Idea: determine the identity of the cross using nearby points from the training set.

- The number of cells grows exponentially with the dimensionality of the space.
- We need an exponentially large quantity of training data to ensure cells are not empty.





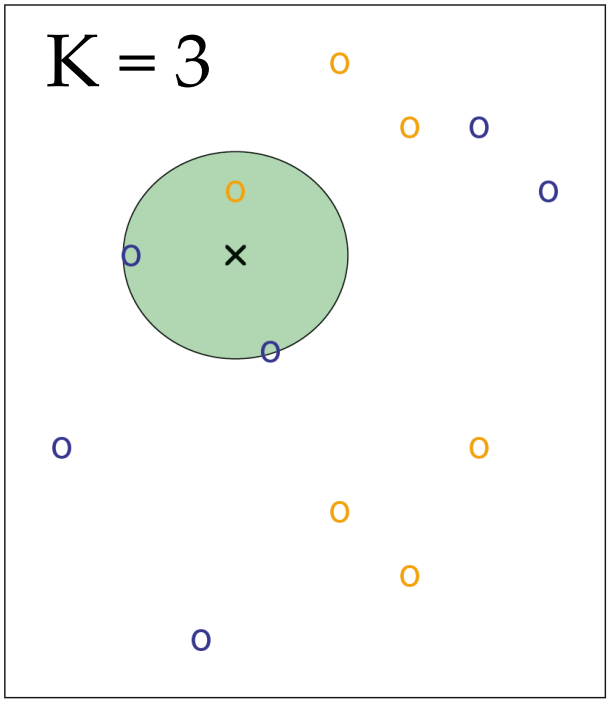
The KNN algorithm

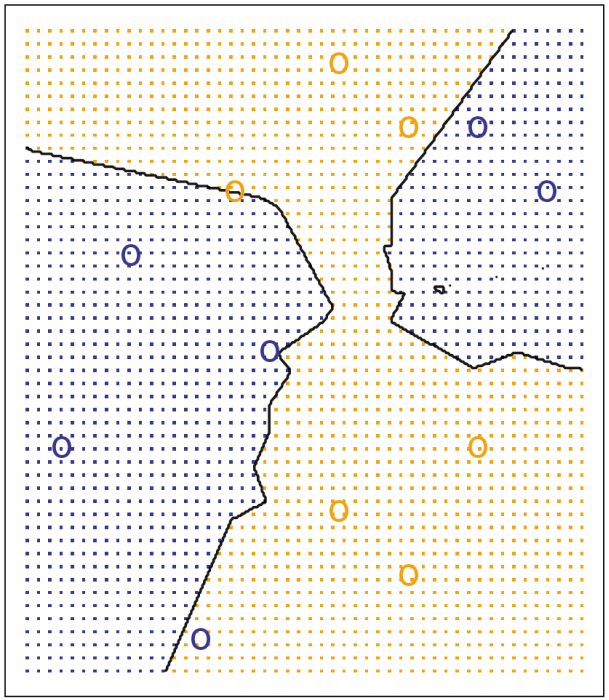
Dataset: N datapoints belonging to L classes

Choose: $K > 0$

To classify a new test observation $\mathbf{x}_0$,

- identify the neighbours K points in the training data that are closest to $\mathbf{x}_0$, represented by N_0.
- For each class j, calculate the fraction F_j of points in N_0 whose response values equal j
- Classify the test observation $\mathbf{x}_0$ to the class with the largest F_j





Questions

1. Is there an underlying probabilistic model? Can you explain in terms of probabilities what we are doing when we are applying this algorithm?

2. Which parameters controls the complexity of the model? Can the model overfit? How?