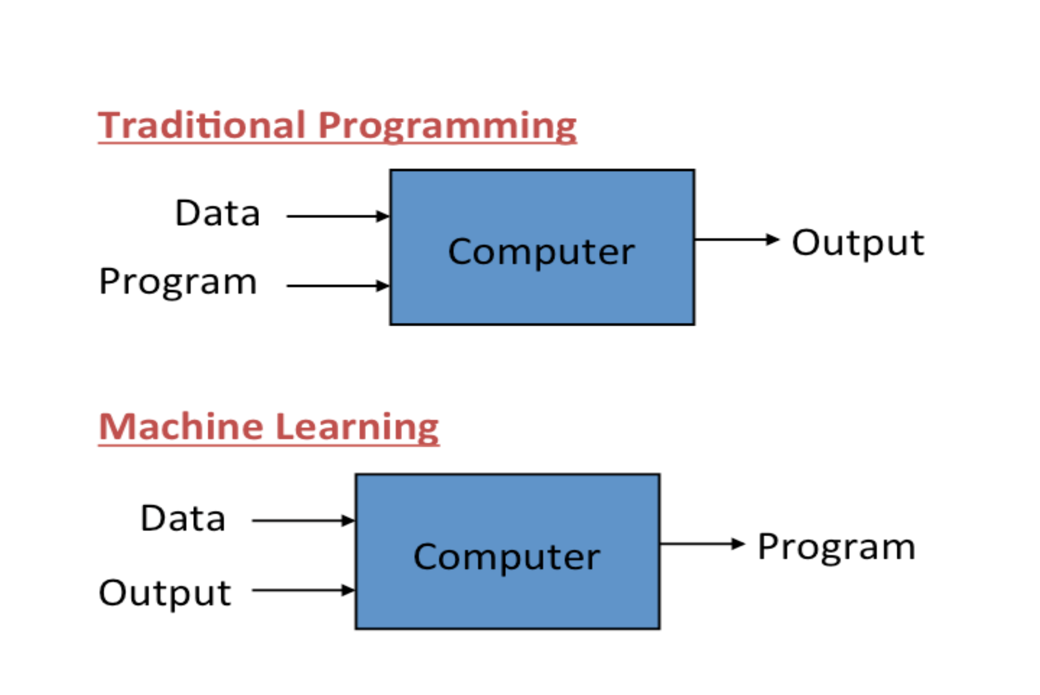


Introduction to Machine Learning

Last modified on 2022/04/30 by f.maire@qut.edu.au

Based on material by Stuart Russel, Peter Norvig

Outline

- The main machine learning tasks
- Good generalization
- Naïve Bayes classifier
- k-Nearest Neighbour classifier
- Classification errors (training, validation and test)

Reference
Part 5 of AIMA textbook (4th Ed)
Chapter 19 “Learning from Examples”

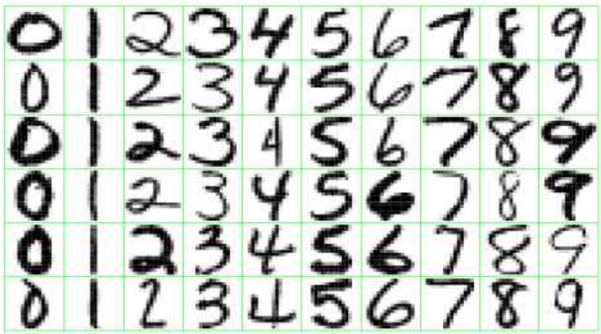
Machine Learning Tasks

Examples of Statistical Learning Problems

- Predict whether someone will have a heart attack on the basis of demographic, diet and clinical measurements
- Customize an email spam detection system

	george	you	hp	free	!	edu	remove
spam	0.00	2.26	0.02	0.52	0.51	0.01	0.28
email	1.27	1.27	0.90	0.07	0.11	0.29	0.01

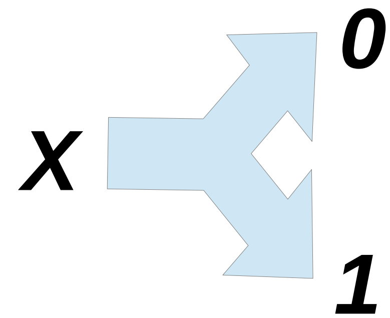
- Identify the numbers
  in a handwritten postcode



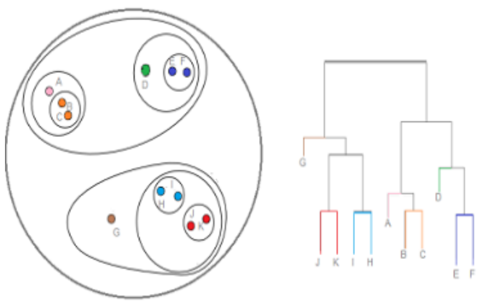
- Identify emergency landing places for UAV's

Machine Learning Tasks

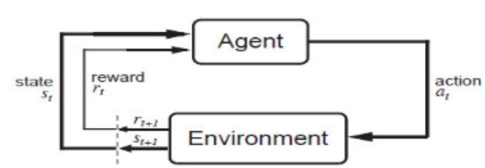
- Classification (label prediction)
- Regression (function approximation)



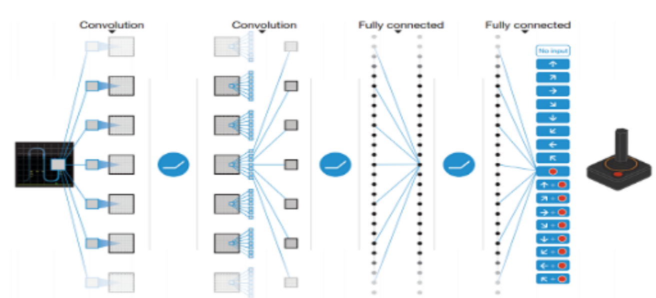
- Clustering
- Probability density estimation



- Policy learning (learning from experience)

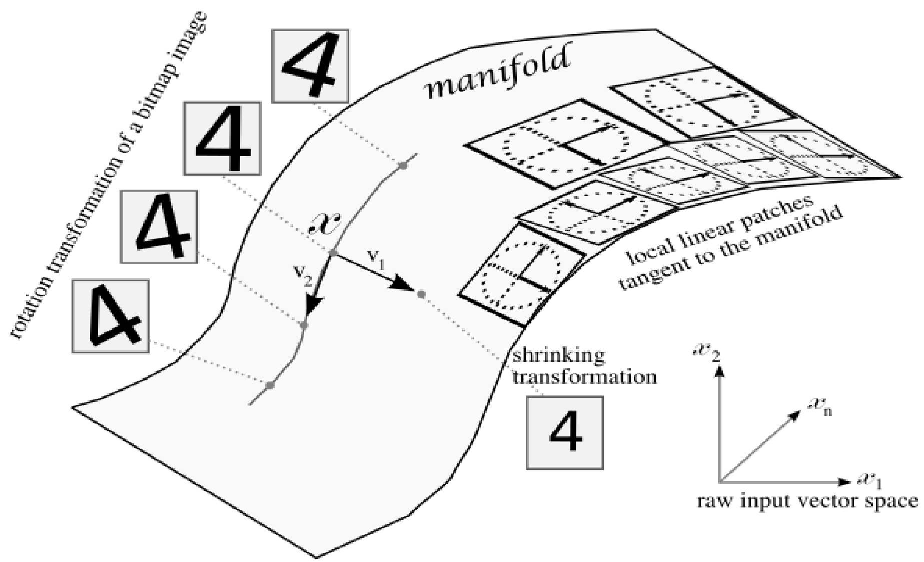


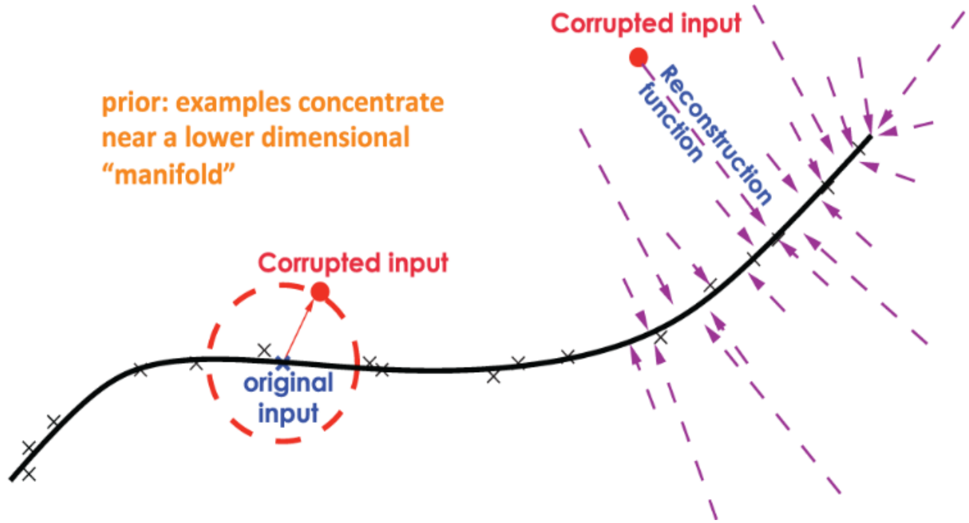
AlphaGo

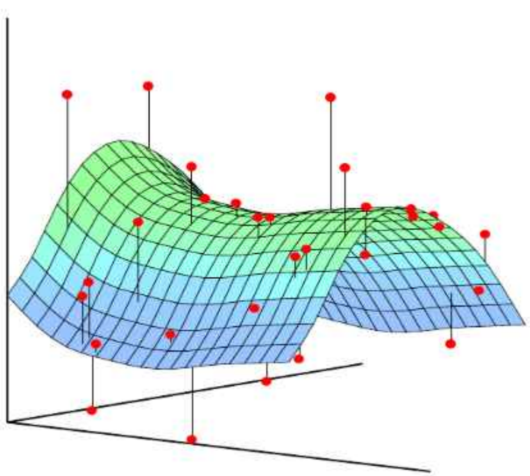


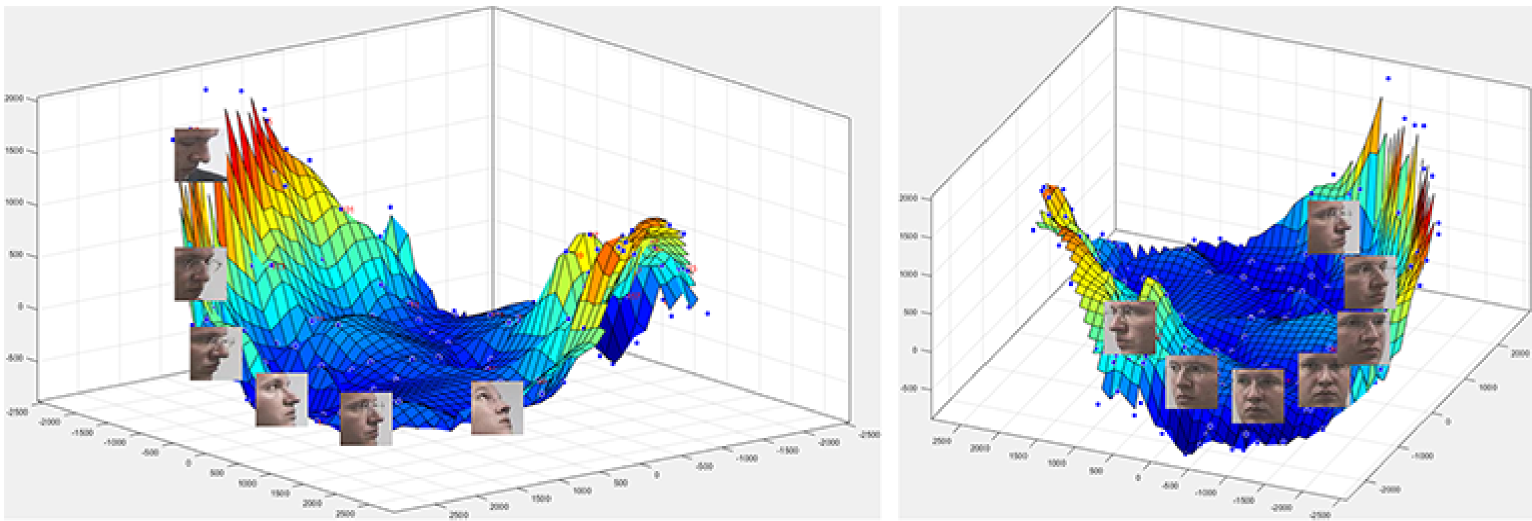
Manifold Learning

- Noise removal
- Classification
- Autoencoders
- Embeddings









(a)

(b)

Unsupervised Learning

- No outcome variable, just a set of predictors (features) measured on a set of samples
- Objective is more fuzzy — find groups of samples that behave similarly, find features that behave similarly, find linear combinations of features with the most variation
- Difficult to know how well your are doing!
- Different from supervised learning, but can be useful as a pre-processing step for supervised learning

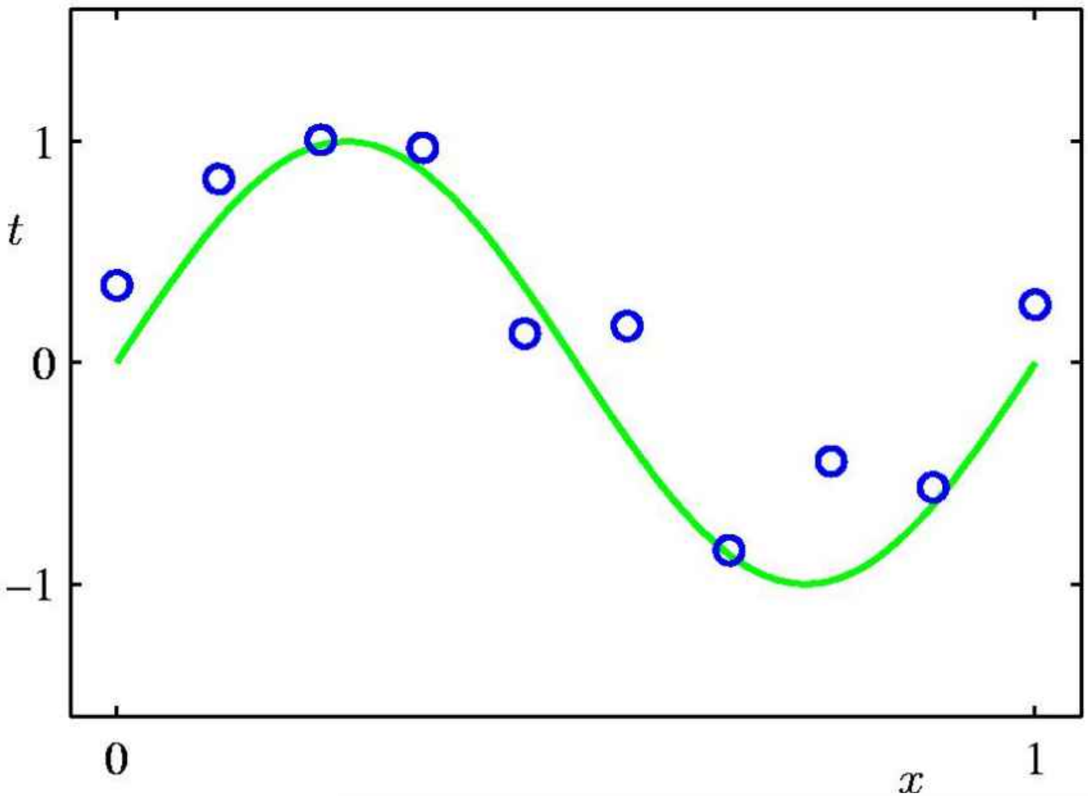
Supervised Learning

- Outcome measurement Y (also called dependent variable, response, target)
- Vector of p predictor measurements X (also called inputs, regressors, covariates, features, independent variables)
- In the regression problem, Y is quantitative (e.g price, blood pressure)
- In the classification problem, Y takes values in a finite, unordered set (survived/died, digit 0-9, safe for landing)
- We have training data $(x1, y1), \ldots, (xN, yN)$. These are observations (examples, instances) of these measurements

ML Objectives

- Accurately predict the labels of **new** test cases
- Understand which inputs affect the outcome, and how
- Assess the quality of predictions and inferences

Polynomial Curve Fitting

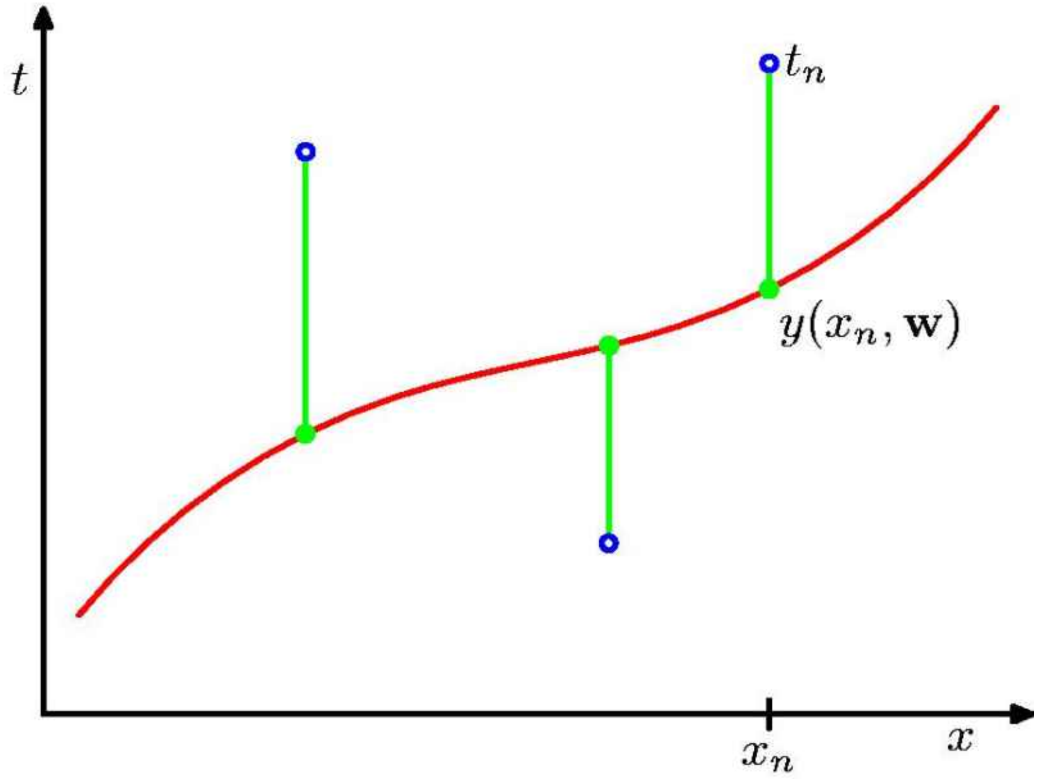


Let's look at **generalization**!
Good generalization is
what we care about

Blue dots are our data points
They are noisy measurements
of hidden/unknown green data

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

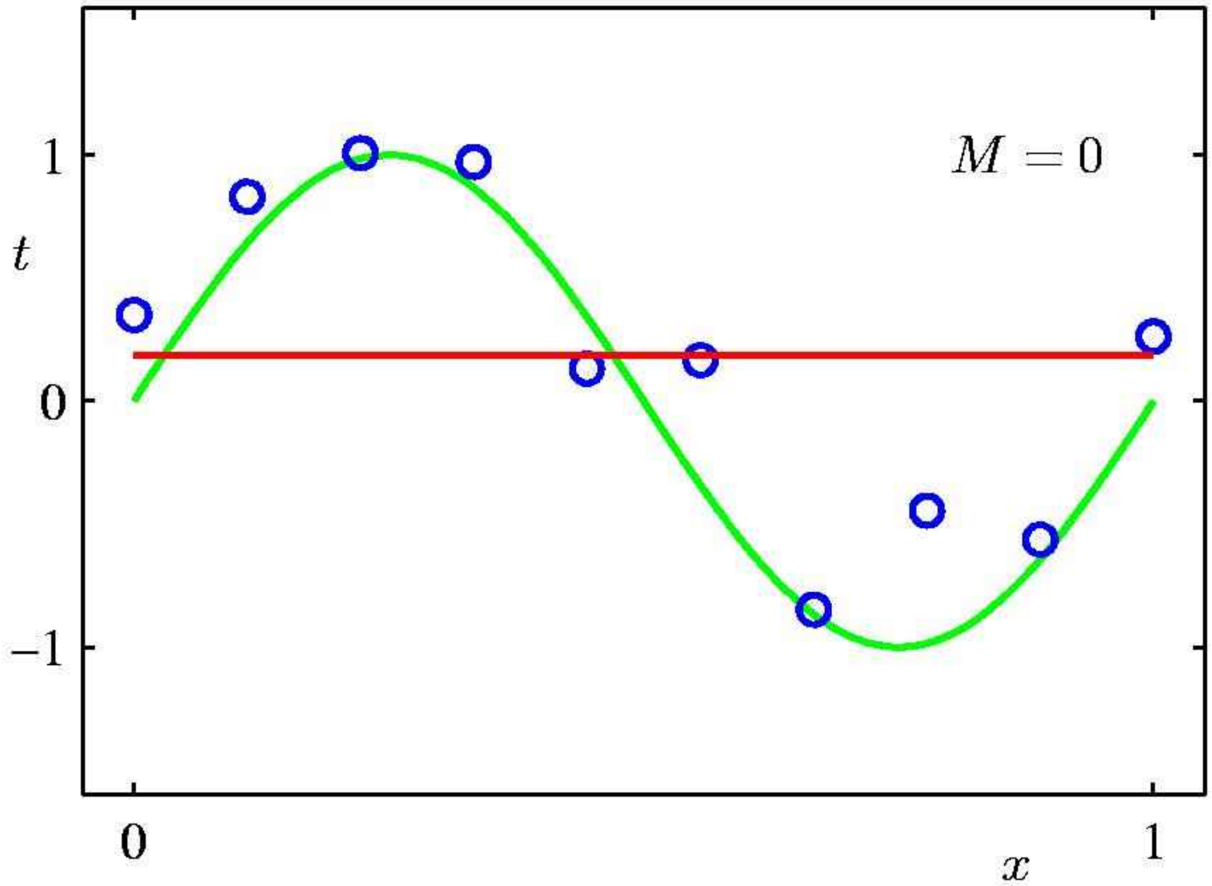
Sum-of-Squares Error Function



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

E(w) measures the discrepancy between the model *y(x,w)* and the dataset *{(x1, y1), . . . , (xN, yN)}*

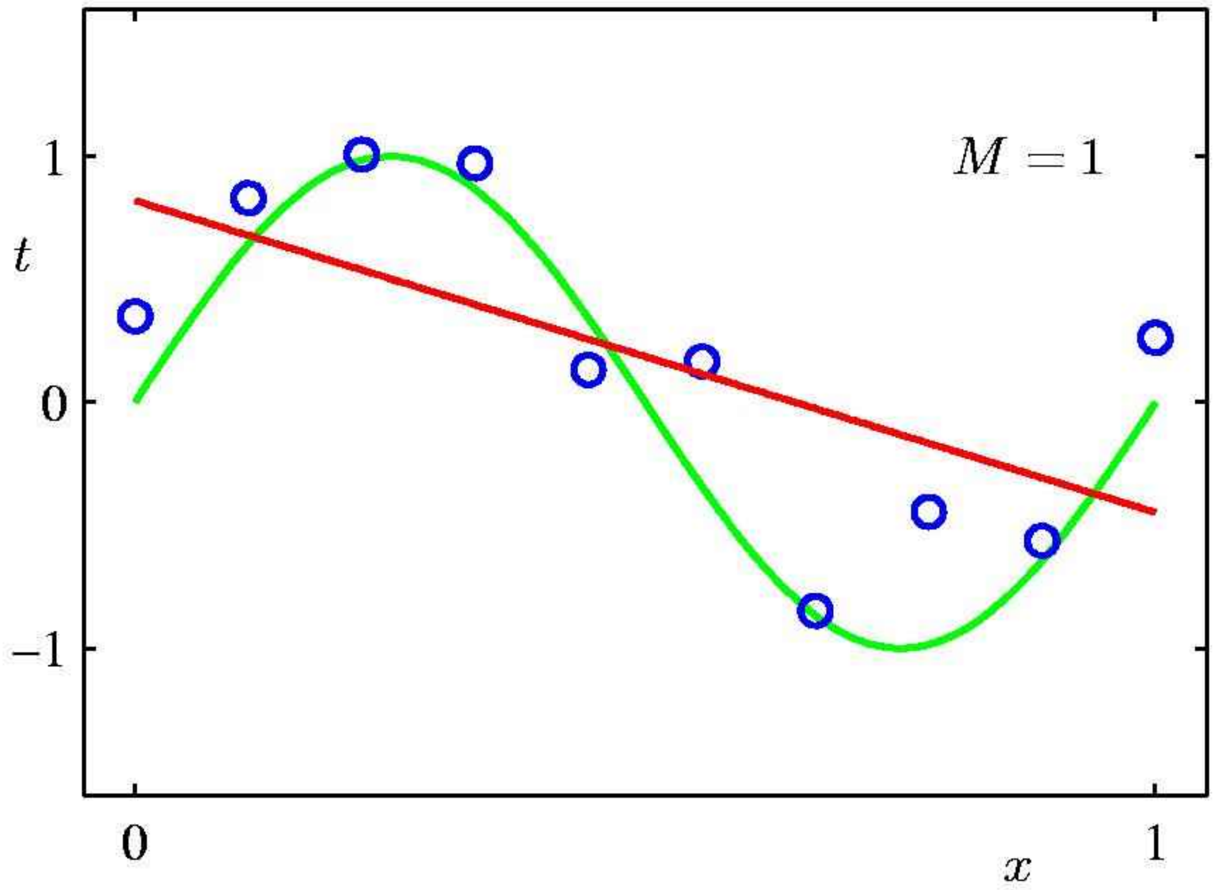
0th Order Polynomial



M is the degree of the polynomial

In red, the best fit for a polynomial model of degree *M=0*

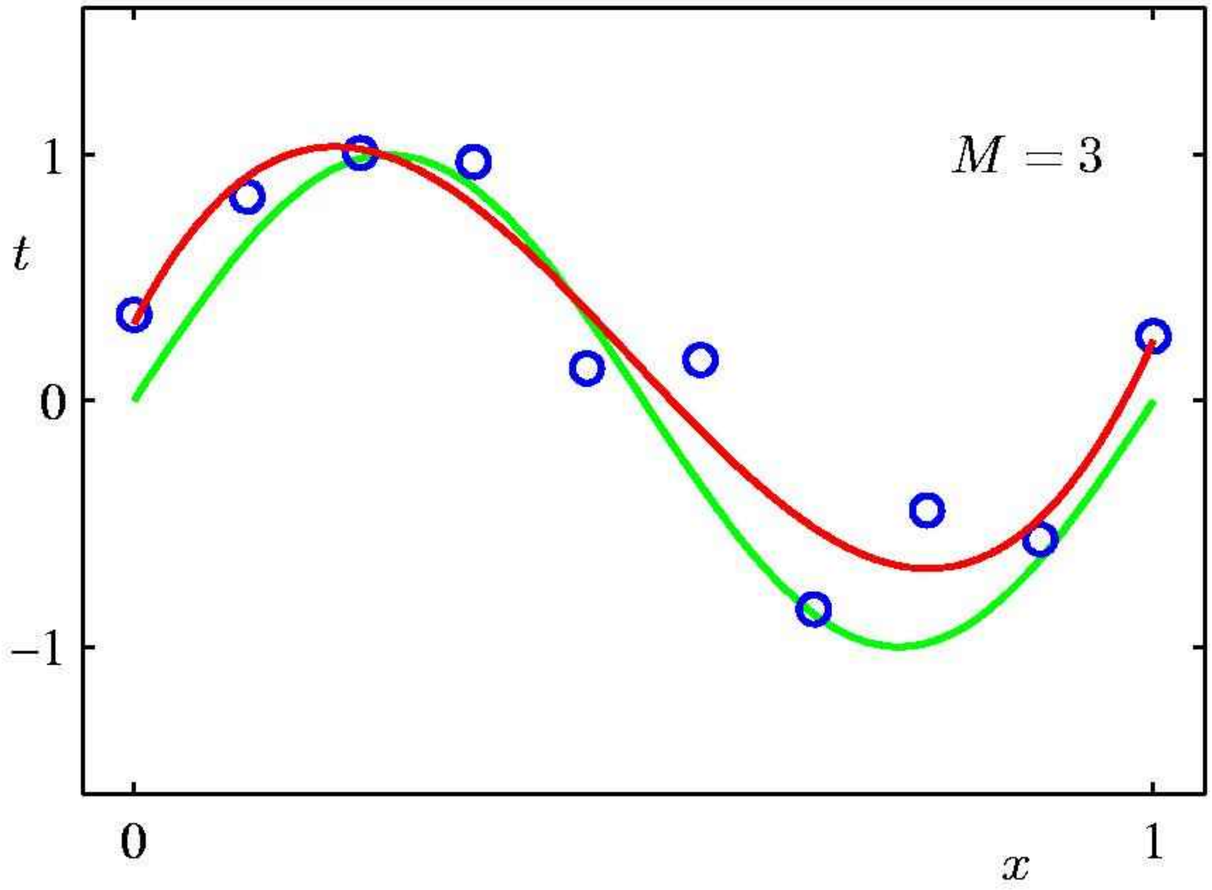
1^{st} Order Polynomial



As *M* increases, the capacity of the model increases.
Capacity ~ ability of the model to fit the data

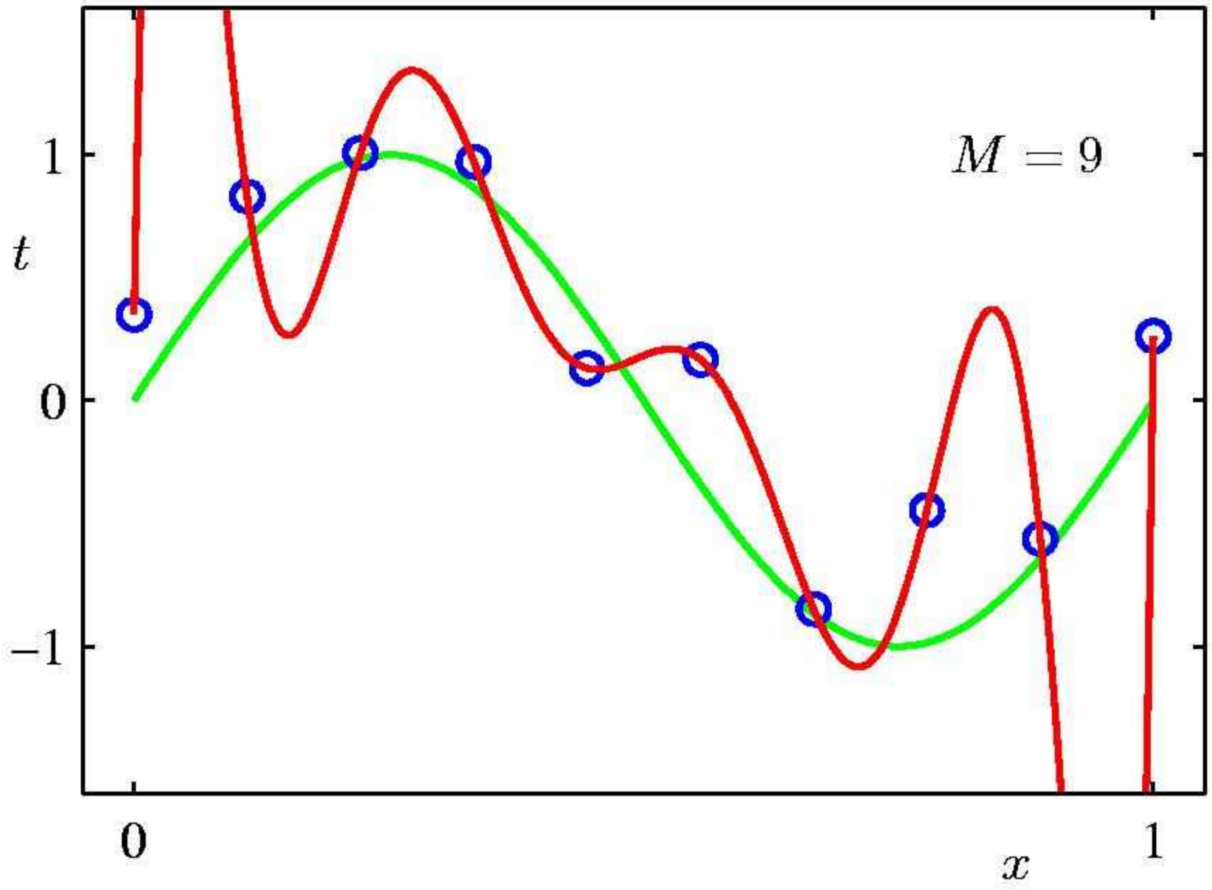
In red, the best fit for a polynomial of degree *M=1*

3rd Order Polynomial



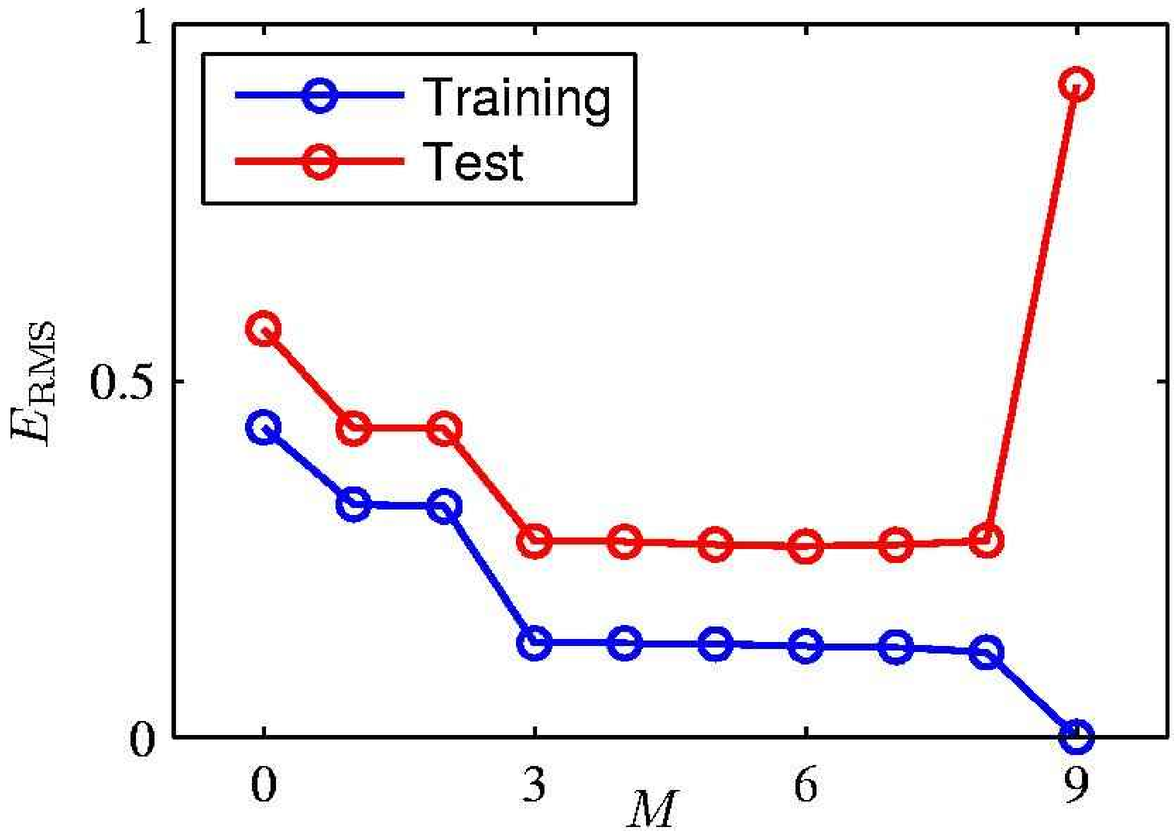
As *M* increases, the capacity of the model increases

9th Order Polynomial



At some point the capacity can become too large!

Over-fitting



Only the examples in the **Training Set** are used for selecting *w.* The examples of the **Test Set** are used for the evaluation of the generalization performance of the trained model.

Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^{\star})/N}$

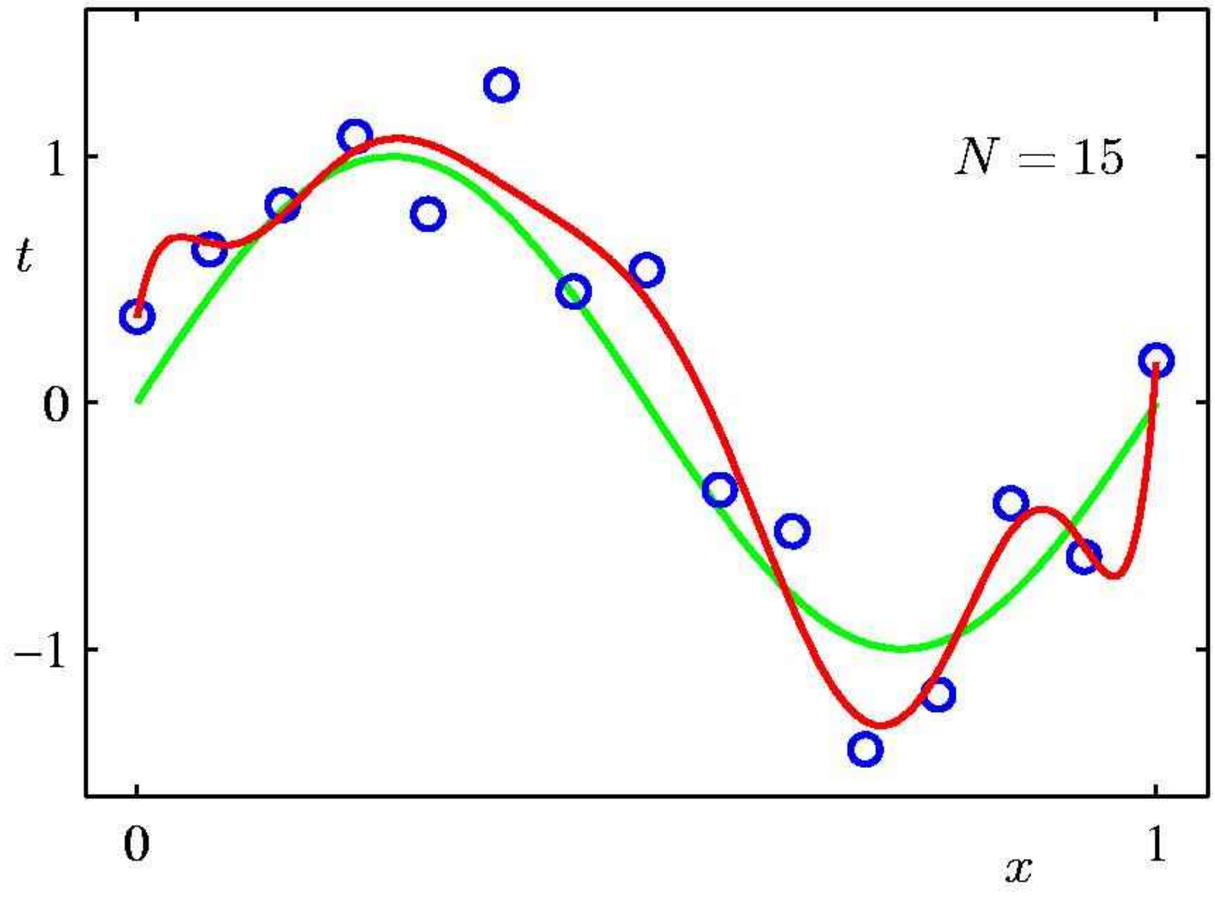
Polynomial Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Weight explosion as *M* increases

Data Set Size: $N = 15$

9^{th} Order Polynomial



The appropriate capacity of the model (*M*) depends on the size of the training set

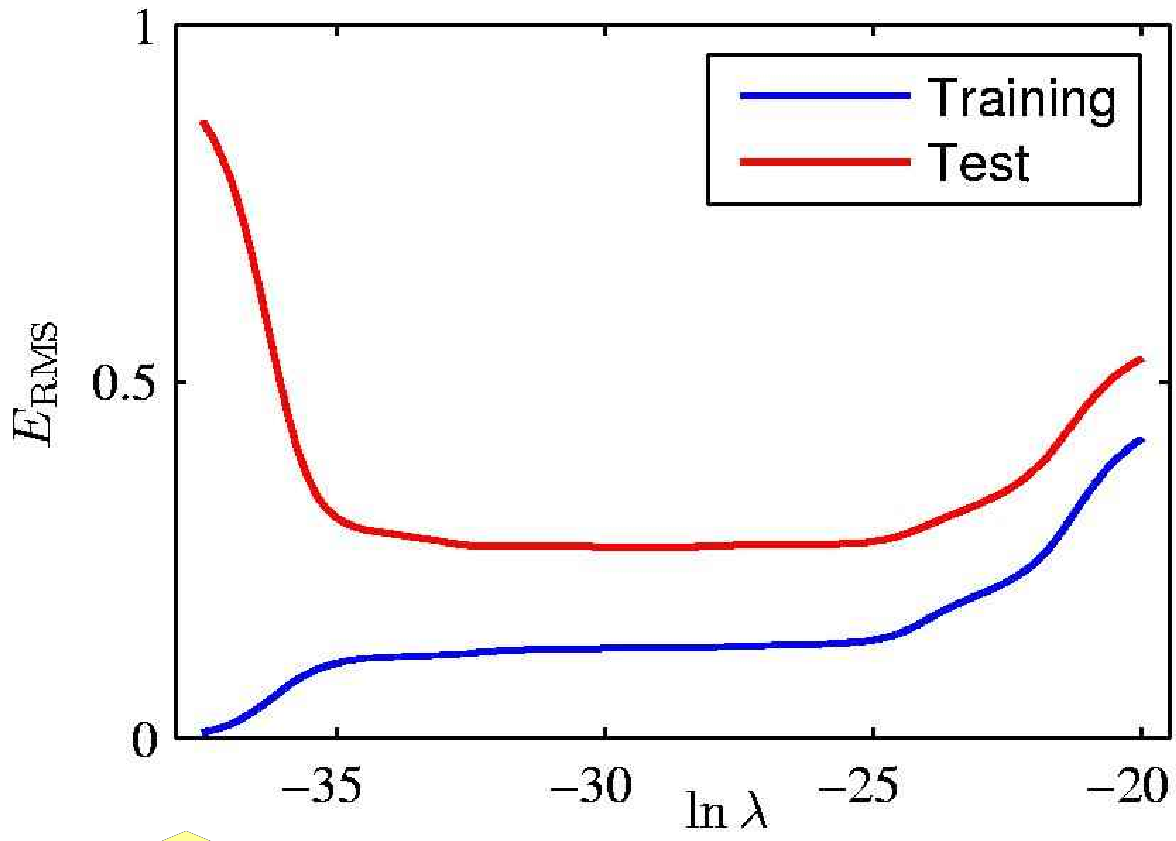
Regularization

Idea : Penalize large coefficient values

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

We can control the capacity
of the parameterized function
by imposing a penalty
for large weights

Regularization: E_{RMS} vs. $\ln \lambda$



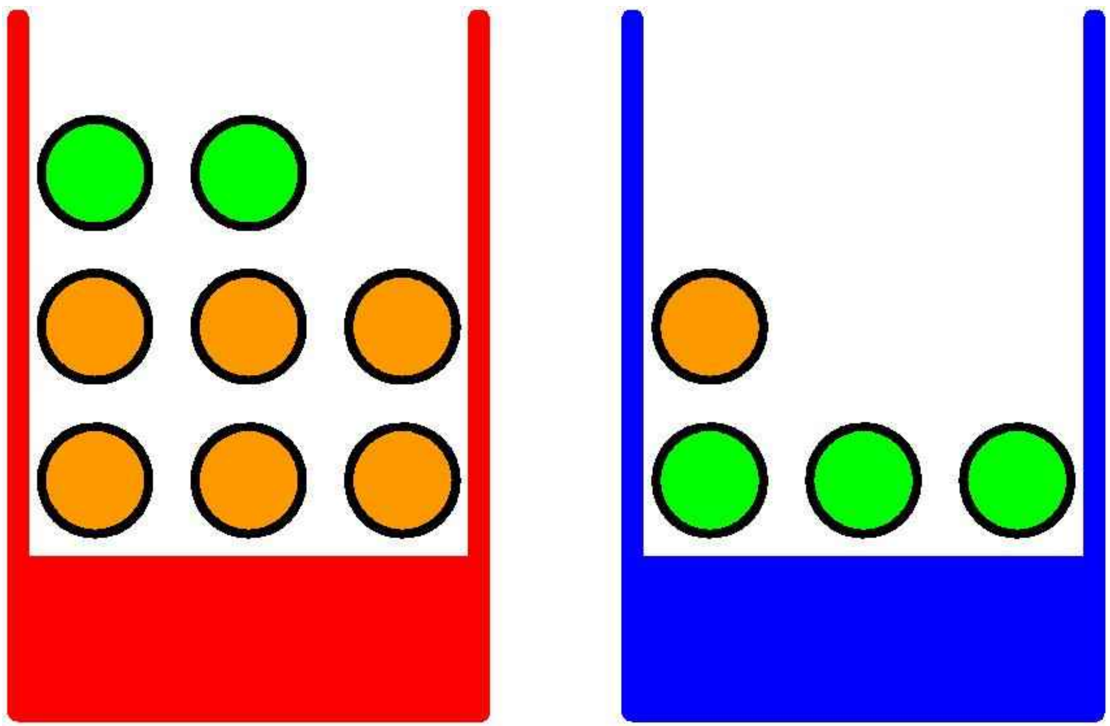
Small λ value
for the regularization
term in the loss

Large λ value
for the regularization
term in the loss

Probability Refresher

Probability Theory

Apples and Oranges



Concept review:
- ***Random variable***
- ***Conditional probability***

Random variable examples:
B : colour of the bag
F : type of fruit

Question:

$$P(B|F) \ ?$$

What is the probabiliy that the fruit was picked from the blue bag given that it is an orange?

2-step random process:
(1) select randomly a bag
(2) select randomly a fruit from that bag

The Rules of Probability

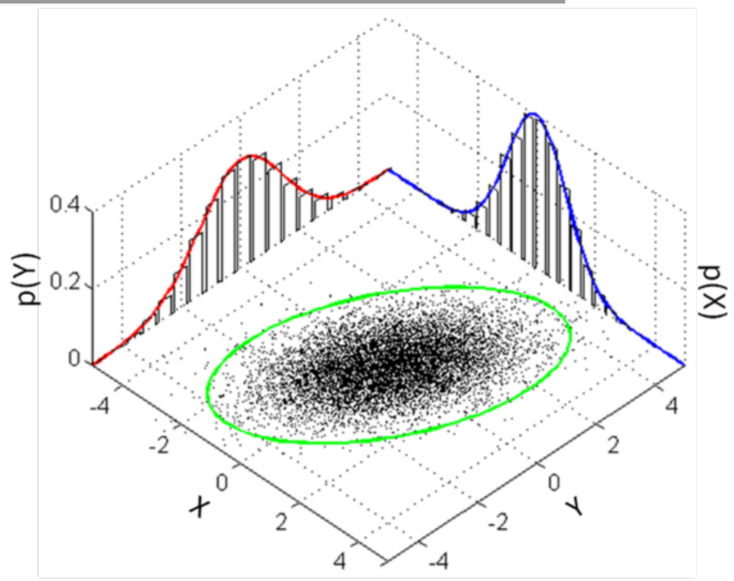
Sum Rule

$$p(X) = \sum_Y p(X, Y)$$

Product Rule

$$p(X, Y) = p(Y|X)p(X)$$

Y \ X	x_1	x_2	x_3	x_4	$p_y(Y)\downarrow$
y_1	4/32	2/32	1/32	1/32	8/32
y_2	2/32	4/32	1/32	1/32	8/32
y_3	2/32	2/32	2/32	2/32	8/32
y_4	8/32	0	0	0	8/32
$p_x(X)\rightarrow$	16/32	8/32	4/32	4/32	32/32



Bayes' Theorem

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

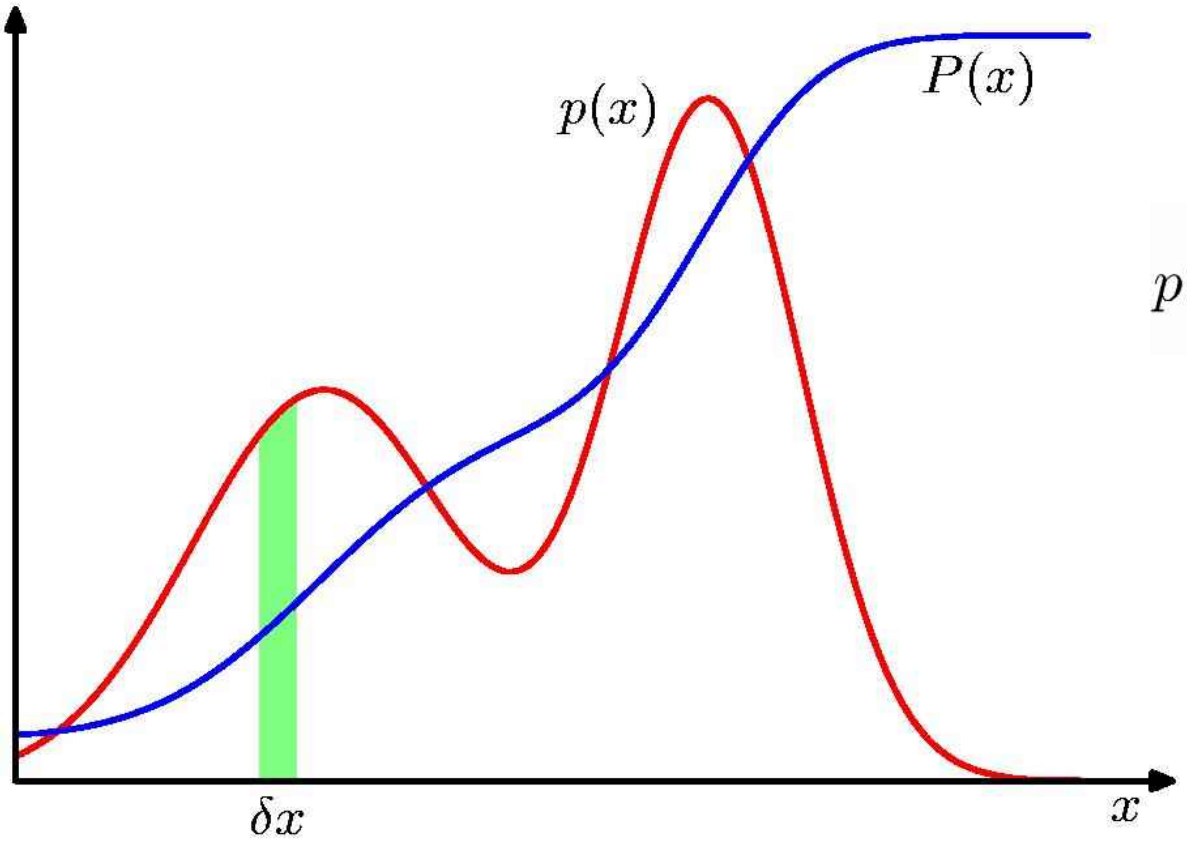
$$p(X) = \sum_{Y} p(X|Y)p(Y)$$

posterior   likelihood × prior

Y: class
X: measurements

Probability Densities

For non discrete random variables



$$p(x \in (a,b)) = \int_a^b p(x)\,\mathrm{d}x$$

$$P(z) = \int_{-\infty}^{z} p(x)\,\mathrm{d}x$$

$$p(x) \geqslant 0 \qquad \int_{-\infty}^{\infty} p(x)\,\mathrm{d}x = 1$$

Classification Learning

- Given a training sample, we want to predict $y \in Y$ for a new $x = (x_1, \ldots, x_n) \in X$
- Error minimised by $y = \operatorname*{argmax}_y P(y | x_1, \ldots, x_n)$
- Can estimate probabilities using observed frequencies

$$P(W) \simeq F(W)$$

$$P(W | Z) \simeq \frac{F(W, Z)}{F(Z)}$$

Estimating Probabilities

To estimate the probability of an attribute-value $A = v$ for a given class C we use

the *relative frequency* n_c / n

where n_c is the number of training instances that belong to the class C and have value v for the attribute A, and n is the number of training instances of the class C

Naïve Bayes Classifier

Let each instance *x* of a training set *D* be described by a conjunction of *n* attribute values $<a_1,a_2,..,a_n>$ and let the target function, be such that it can take any value from a finite set *V*.

$$
\begin{aligned}
v_{\text{MAP}} &= \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, \ldots, a_n) \\
&= \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, \ldots, a_n | v_j) P(v_j)}{P(a_1, \ldots, a_n)} \\
&= \underset{v_j \in V}{\operatorname{argmax}} P(a_1, \ldots, a_n | v_j) P(v_j) \\
&= \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j)
\end{aligned}
$$

Naïve Bayes simplifying assumption is that attributes are conditionally independent!

Example

Consider the weather data and we have to classify the instance:

< Outlook = sunny, Temp = cool, Hum = high, Wind = strong >

The task is to predict the value (*yes* or *no*) of the concept PlayTennis. We apply the Naïve Bayes rule:

$$v_{MAP} = \underset{vj \in \{yes, no\}}{\operatorname{argmax}} P(v_j) \prod_i P(a_i | v_j)$$

$$= \underset{vj \in \{yes, no\}}{\operatorname{argmax}} P(v_j) P(Outlook = sunny | v_j) P(Temp = cool | v_j)$$

$$P(Hum = high | v_j) P(Wind = strong | v_j)$$

Example: Estimating Probabilities

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

$P(\text{yes}) = 9/14$

$P(\text{no}) = 5/14$

Outlook

$P(\text{sunny}|\text{yes}) = 2/9$ $P(\text{sunny}|\text{no}) = 3/5$

$P(\text{overcast}|\text{yes}) = 4/9$ $P(\text{overcast}|\text{no}) = 0$

$P(\text{rain}|\text{yes}) = 3/9$ $P(\text{rain}|\text{no}) = 2/5$

Temp

$P(\text{hot}|\text{yes}) = 2/9$ $P(\text{hot}|\text{no}) = 2/5$

$P(\text{mild}|\text{yes}) = 4/9$ $P(\text{mild}|\text{no}) = 2/5$

$P(\text{cool}|\text{yes}) = 3/9$ $P(\text{cool}|\text{no}) = 1/5$

Hum

$P(\text{high}|\text{yes}) = 3/9$ $P(\text{high}|\text{no}) = 4/5$

$P(\text{normal}|\text{yes}) = 6/9$ $P(\text{normal}|\text{no}) = 1/5$

Windy

$P(\text{true}|\text{yes}) = 3/9$ $P(\text{true}|\text{no}) = 3/5$

$P(\text{false}|\text{yes}) = 6/9$ $P(\text{false}|\text{no}) = 2/5$

Example

$$P(yes)P(sunny|yes)P(cool|yes)P(high|yes)P(strong|yes)=.0053$$
$$P(no)p(sunny|no)P(cool|no)P(high|no)P(strong|no)=.0206$$

Thus, the naïve Bayes classifier assigns the value *no* to PlayTennis!

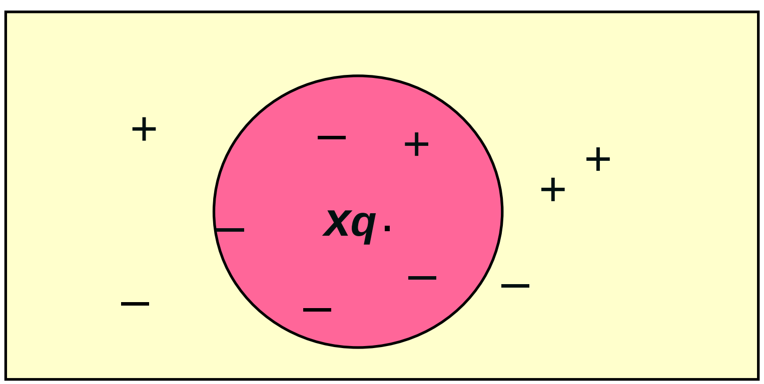
Nearest Neighbours Classifiers

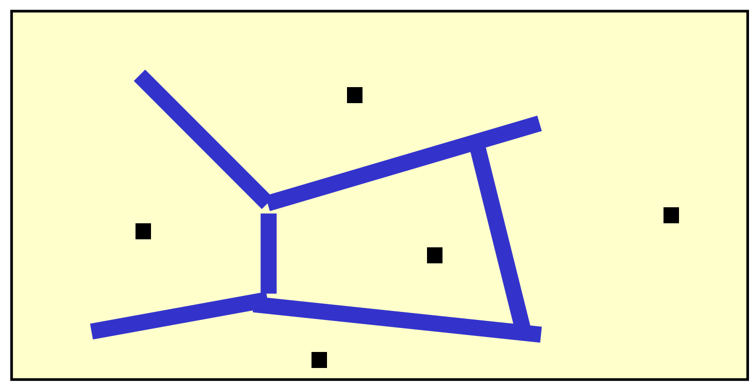
Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical approaches
  - *k*-nearest neighbour approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation

The *k*-Nearest Neighbour Algorithm

- All instances correspond to points in the n-D space.
- The nearest neighbor are defined in terms of Euclidean distance.
- The target function could be discrete- or real- valued.
- For discrete-valued, the *k*-NN returns the most common value among the k training examples nearest to *xq*.
- Vonoroi diagram: the decision surface induced by 1-NN for a typical set of training examples.





Discussion on the *k*-NN Algorithm

- The k-NN algorithm for continuous-valued target functions
  - Calculate the mean values of the *k* nearest neighbours
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the k neighbours according to their distance to the query point x_q
    - giving greater weight to closer neighbours

$$w = \frac{1}{d(x_q, x_i)^2}$$

  - Similarly, for real-valued target functions
- Robust to noisy data by averaging k-nearest neighbours
- Curse of dimensionality: distance between neighbours could be dominated by irrelevant attributes.
  - To overcome it, axes stretch or elimination of the least relevant attributes.

Training, Validation and Test Errors

What is the difference between a *training set*, *a validation set* and *a test set*?

- The **training set** is used to select the parameter vector of the classifier
- The **validation set** is used to select the **capacity/complexity** of the classifier. For example, the capacity can be the maximum depth of a decision tree. Shallow decision trees have less discriminative power then deeper ones. But deeper decision trees might overfit the data.
- Assume that the capacity of your classifier is an integer n. In order to select the capacity of your classifier, you can train (using the training set) a number of classifiers with different capacity n, and evaluate the generalization of the classifier using the validation set. You should pick the capacity n for which the generalization error is the lowest.
- So, why do we need the **test set**?!
- The validation set cannot be used as a test set, because it is "tainted". It has been used to select the capacity/architecture of the classifier. So, it cannot be said that it is "unseen data". This is why, statisticians require the use of a different set (the test set) to assess the generalization error of the trained classifier).