Preliminaries

- The ability of biological brains to sense, perceive, analyse and recognise patterns can only be described as stunning.
- They also have the ability to learn from new examples with or without being taught.
- Mankind's understanding of how biological brains operate exactly is embarrassingly limited.
- However, there do exist numerous *practical* techniques that give machines the *appearance* of being intelligent.
- This is the domain of statistical pattern recognition and machine learning.

Preliminaries

- Instead of attempting to mimic the complex workings of a biological brain, this course
  - aims at explaining mathematically well-founded techniques for analysing patterns and learning from them, and is therefore
  - a *mathematically involved* introduction into the field of pattern recognition and machine learning.
- It will prepare you for further study/research in
  - Pattern Recognition
  - Machine Learning
  - Computer Vision
  - Big Data Analytics

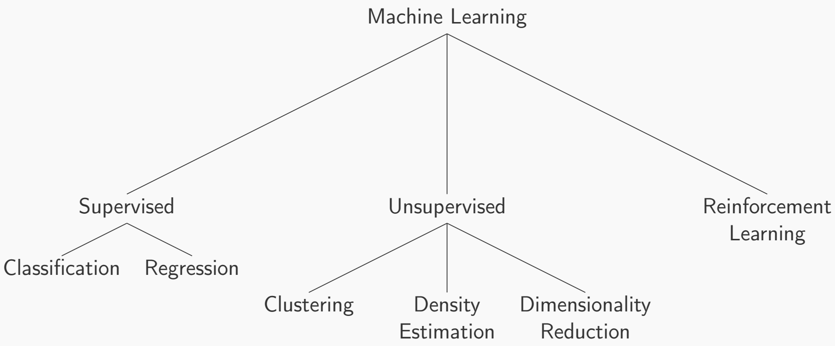
  and others areas attempting to solve Artificial Intelligence (AI) type problems.

Introduction

Machine Learning and Pattern Recognition are different names for essentialy the same thing.

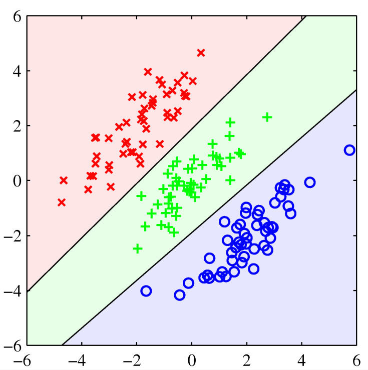
- Pattern Recognition arose out of Engineering.
- Machine Learning arose out of Computer Science.
- Both are concerned with automatic discovery of regularities in data.
- Regularity implies order. Learning implies exploiting order in order to make predictions.

Machine Learning

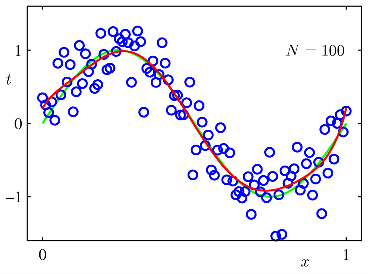


Supervised Learning

- **Classification**: Assign **x** to *discrete* categories.
  - Examples: Digit recognition, face recognition, *etc.*.
- **Regression**: Find *continuous* values for **x**.
  - Examples: Price prediction, profit prediction.



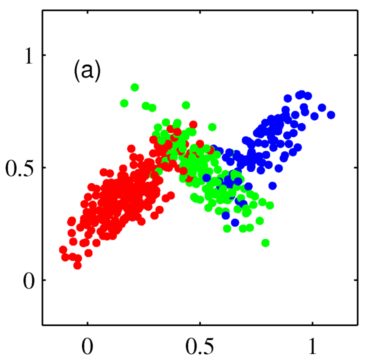
Classification



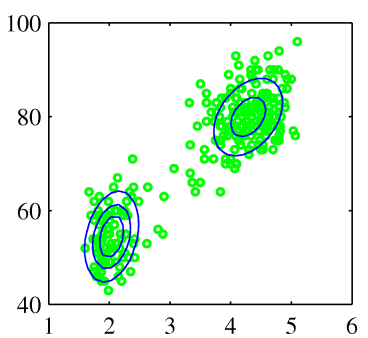
Regression

Unsupervised Learning

- **Clustering**: Discover groups of similar examples.
- **Density Estimation**: Determine probability distribution of data.
- **Dimensionality Reduction**: Map data to a lower dimensional space.



Clustering



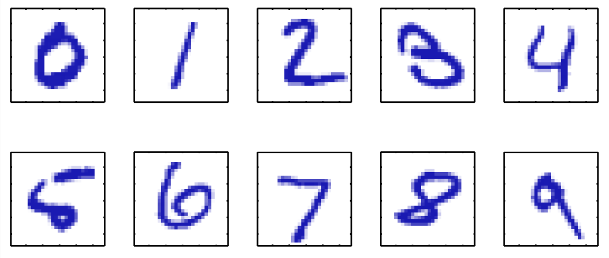
Density Estimation

Reinforcement Learning

- Find actions that maximise a reward. Example: chess playing program competing against a copy of itself.
- Active area of ML research.
- We will not be covering reinforcement learning in this course.

Classical Algorithms vs. Machine Learning

Problem: Given an image $\mathbf{x}$ of a digit, classify it between $0, 1, \ldots, 9$.



Non-trivial due to high variability in hand-writing.

Classical Algorithms vs. Machine Learning

Classical Approach: Make hand-crafted rules or heuristics for distinguishing digits based on shapes of strokes.
Problems:

- Need lots of rules.
- Exceptions to rules and so on.
- Almost always gives poor results.

Classical Algorithms vs. Machine Learning

ML Approach:

- Collect a large *training set* $\mathbf{x}_1, \ldots, \mathbf{x}_N$ of hand-written digits with known labels $t_1, \ldots, t_N$.
- Learn/tune the parameters of an *adaptive* model.
  - The model can adapt so as to reproduce correct labels for all the training set images.

Classical Algorithms vs. Machine Learning

- Every sample $\mathbf{x}$ is mapped to $f(\mathbf{x})$.
- ML determines the mapping f during the *training phase*. Also called the *learning phase*.
- Trained model f is then used to label a new *test image* $\mathbf{x}_{\text{test}}$ as $f(\mathbf{x}_{\text{test}})$.

Terminology

- *Generalization*: ability to correctly label *new* examples.
  - Very important because training data can only cover a tiny fraction of all possible examples in practical applications.
- *Pre-processing*: Transform data into a new space where solving the problem becomes
  - easier, and
  - faster.

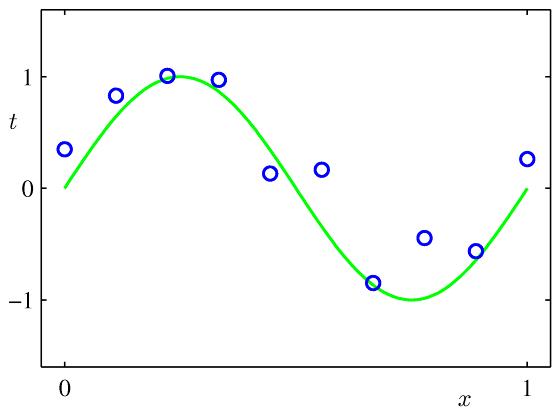
  Also called *feature extraction*. The extracted features should
  - be quickly computable, and
  - preserve useful discriminatory information.

Essential Topics for ML

1. Probability theory – deals with uncertainty.
2. Decision theory – uses probabilistic representation of uncertainty to make optimal predictions.
3. Information theory

Example: Polynomial Curve Fitting

Problem: Given N observations of input x_i with corresponding observations of output t_i, find function $f(x)$ that predicts t for a new value of x.



First, let's generate some data.

```
N=10;
x=0:1/(N-1):1;
t=sin(2*pi*x);
plot(x,t,'o');
```

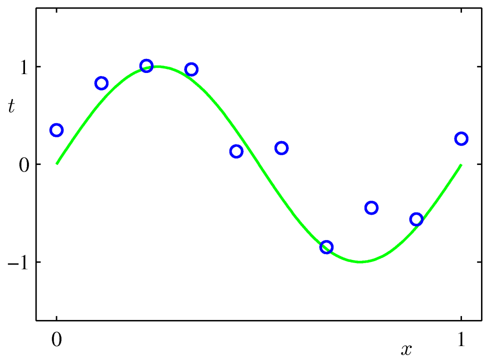
Notice that the data is generated through the function $\sin(2\pi x)$.
Real-world observations are always 'noisy'.
Let's add some noise to the data

```
n=randn(1,N)*0.3;
t=t+n;
plot(x,t,'o');
```

Real-world Data

Real-world data has 2 important properties

1. underlying regularity,
2. individual observations are corrupted by noise.



Learning corresponds to discovering the underlying regularity of data (the $\sin(\cdot)$ function in our example).

Polynomial curve fitting

- We will fit the points (x, t) using a polynomial function

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

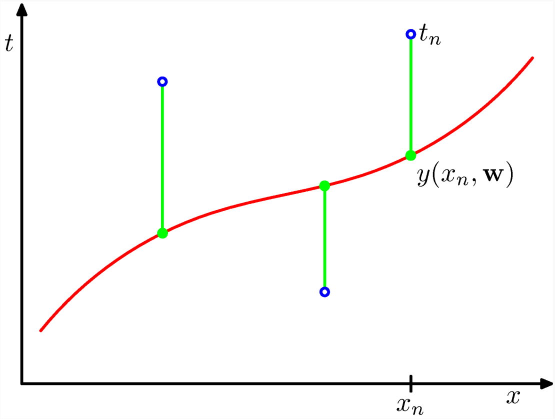
  where M is the *order* of the polynomial.
- Function $y(x, \mathbf{w})$ is a
  - non-linear function of the input x, but
  - a linear function of the parameters $\mathbf{w}$.
- So our model $y(x, \mathbf{w})$ is a *linear model*.

Polynomial curve fitting

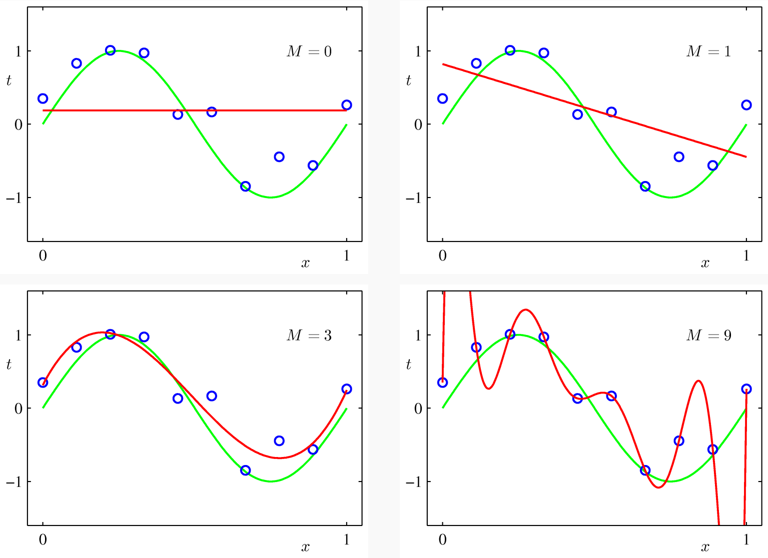
- Fitting corresponds to finding the optimal $\mathbf{w}$. We denote it as $\mathbf{w}^*$.
- Optimal $\mathbf{w}^*$ can be found by *minimising* an *error function*

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

- Why does minimising $E(\mathbf{w})$ make sense?
- Can $E(\mathbf{w})$ ever be negative?
- Can $E(\mathbf{w})$ ever be zero?



Geometric interpratation of the sum-of-squares error function.


$M = 0$
$M = 1$
$M = 3$
$M = 9$
t
x
1
0
−1

Over-fitting

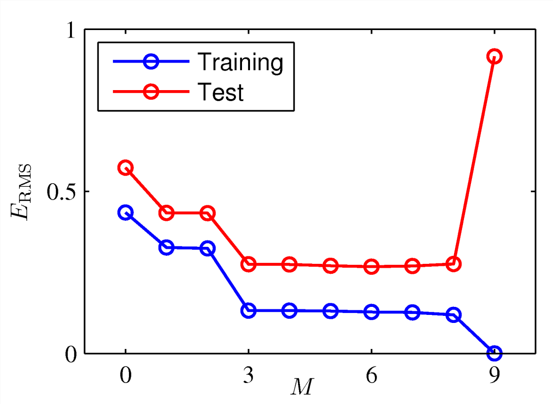
- Lower order polynomials can't capture the variation in data.
- Higher order leads to *over-fitting*.
  - Fitted polynomial passes *exactly* through each data point.
  - But it oscillates wildly in-between.
  - Gives a very poor representation of the real underlying function.
- Over-fitting is bad because it gives bad generalization.

Over-fitting

- To check generalization performance of a certain $\mathbf{w}^*$, compute $E(\mathbf{w}^*)$ on a *new* test set.
- Alternative performance measure: root-mean-square error (RMS)

$$E_{RMS} = \sqrt{\frac{2E(\mathbf{w}^*)}{N}}$$

- Mean ensures datasets of different sizes are treated equally. (How?)
- Square-root brings the *squared* error scale back to the scale of the target variable t.



Root-mean-square error on training and test set for various polynomial orders M.

Paradox?

- A polynomial of order M contains all polynomials of lower order.
- So higher order should *always* be better than lower order.
- **But**, it's not better. Why?
    - Because higher order polynomial starts fitting the noise instead of the underlying function.

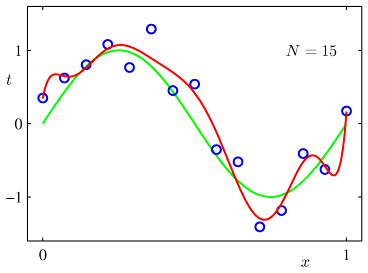
Over-fitting

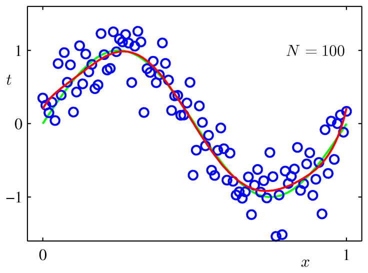
	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

- Typical magnitude of the polynomial coefficients is increasing dramatically as M increases.
- This is a sign of over-fitting.
- The polynomial is trying to fit the data points exactly by having larger coefficients.

Over-fitting

- Large $M \implies$ more flexibility $\implies$ more tuning to noise.
- **But**, if we have more data, then over-fitting is reduced.





- Fitted polynomials of order $M = 9$ with $N = 15$ and $N = 100$ data points. More data reduces the effect of over-fitting.
- Rough heuristic to avoid over-fitting: Number of data points should be greater than $k|\mathbf{w}|$ where k is some multiple like 5 or 10.

How to avoid over-fitting

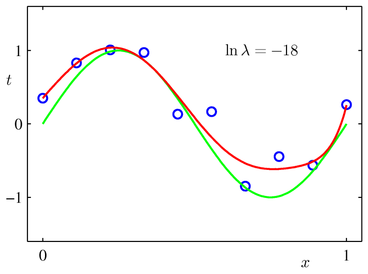
- Since large coefficients $\Longrightarrow$ over-fitting, *discourage large coefficents* in $\mathbf{w}$.

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

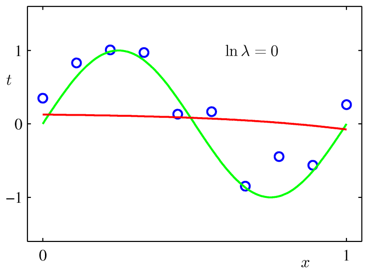
where $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w} = w_0^2 + w_1^2 + \cdots + w_M^2$ and λ controls the relative importance of the regularizer compared to the error term.

- Also called *regularization, shrinkage, weight-decay*.

For a polynomial of order 9



For $\lambda = e^{-18}$
No over-fitting

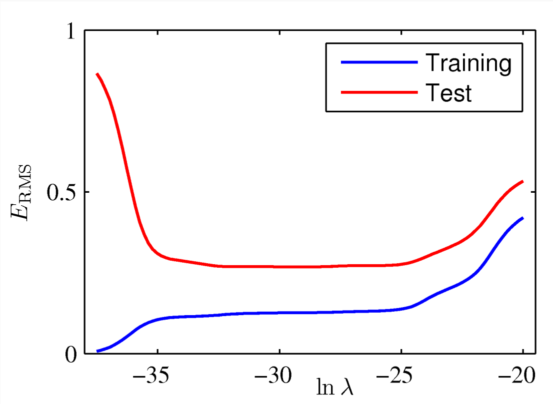


For $\lambda = 1$
Too much smoothing (no fitting)

Effect of regularization

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

- As λ increases, the typical magnitude of coefficients gets smaller.
- We go from over-fitting ($\lambda = 0$) to no over-fitting ($\lambda = e^{-18}$) to poor fitting ($\lambda = 1$).
- Since $M = 9$ is fixed, regularization controls the degree of over-fitting.



Graph of root-mean-square (RMS) error of fitting the $M = 9$ polynomial as λ is increased.

How to avoid over-fitting

- A more principled approach to control over-fitting is the *Bayesian approach* (to be covered later).
  - Determines the *effective* number of parameters automatically.
- We need the machinery of *probability* to understand the Bayesian approach.
- Probability theory also offers a more principled approach for our polynomial fitting example.
- Will be covered in the next lecture.