Overview:

- **Supervised classification**
  - perceptron, support vector machine, loss functions, kernels, random forests, neural networks and deep learning

- **Supervised regression**
  - ridge regression, lasso regression, SVM regression

- **Unsupervised learning (Frank Wood)**
  - graphical models, sequential Monte Carlo, PCA, Gaussian Mixture Models, probabilistic PCA, hidden Markov models

**Recommended book**

- **Pattern Recognition and Machine Learning**


  - Excellent on classification and regression
Textbooks 2

• **Elements of Statistical Learning**

Hastie, Tibshirani, Friedman, *Springer, 2009, second edition*
  - Good explanation of algorithms
  - pdf available online

One more book for background reading …

• **Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)**

Ian Witten & Eibe Frank, *Morgan Kaufmann, 2005.*
  - Very readable and practical guide
Web resources

• On line book:

**Information Theory, Inference, and Learning Algorithms.**

David J. C. MacKay, CUP, 2003

• Covers some of the course material though at an advanced level

• Further reading (www addresses) and the lecture notes are on

http://www.robots.ox.ac.uk/~az/lectures/ml

Introduction: What is Machine Learning?

Algorithms that can improve their performance using training data

• Typically the algorithm has a (large) number of parameters whose values are learnt from the data

• Can be applied in situations where it is very challenging (= impossible) to define rules by hand, e.g.:
  • Face detection
  • Speech recognition
  • Stock prediction
Example 1: hand-written digit recognition

![Images of handwritten digits (0-9)](image)

Images are 28 x 28 pixels

Represent input image as a vector \( x \in \mathbb{R}^{784} \)

Learn a classifier \( f(x) \) such that,
\[
f : x \rightarrow \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}
\]

How to proceed …

As a supervised classification problem

Start with training data, e.g. 6000 examples of each digit

- Can achieve testing error of 0.4%
- One of first commercial and widely used ML systems (for zip codes & checks)
Example 2: Face detection

- Again, a supervised classification problem
- Need to classify an image window into three classes:
  - non-face
  - frontal-face
  - profile-face

Classifier is learnt from labelled data

Training data for frontal faces
- 5000 faces
  - All near frontal
  - Age, race, gender, lighting
- $10^8$ non faces
- faces are normalized
  - scale, translation
Example 3: Spam detection

- This is a classification problem
- Task is to classify email into spam/non-spam
- Data $x_i$ is word count, e.g. of viagra, outperform, “you may be surprised to be contacted” ...
- Requires a learning system as “enemy” keeps innovating

Example 4: Stock price prediction

- Task is to predict stock price at future date
- This is a regression task, as the output is continuous
Example 5: Computational biology

Regression task: given sequence predict 3D structure

Protein: 1IMT

Web examples: Machine translation

Use of aligned text

What is the anticipated cost of collecting fees under the new proposal?

En vertu des nouvelles propositions, quel est le coût prévu de perception des droits?

E.g. Google translate
What is the anticipated cost of collecting fees under the new proposal?

Web examples: Recommender systems

People who bought Hastie …
Three canonical learning problems

1. **Regression - supervised**
   - estimate parameters, e.g. of weight vs height

2. **Classification - supervised**
   - estimate class, e.g. handwritten digit classification

3. **Unsupervised learning** – model the data
   - clustering
   - dimensionality reduction
Supervised Learning: Overview

Functions $\mathcal{F}$

\[ f : \mathcal{X} \rightarrow \mathcal{Y} \]

Training data

\[ \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\} \]

LEARNING

\[
\begin{align*}
\text{find } \hat{f} \in \mathcal{F} \\
\text{s.t. } y_i \approx \hat{f}(x_i)
\end{align*}
\]

Learning machine

PREDICTION

\[ y = \hat{f}(x) \]

New data

Classification

• Suppose we are given a training set of N observations

\[(x_1, \ldots, x_N) \text{ and } (y_1, \ldots, y_N), x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}\]

• Classification problem is to estimate $f(x)$ from this data such that

\[ f(x_i) = y_i \]
K Nearest Neighbour (K-NN) Classifier

**Algorithm**
- For each test point, $x$, to be classified, find the K nearest samples in the training data
- Classify the point, $x$, according to the majority vote of their class labels

*Example: K = 3*

- Applicable to multi-class case

---

**K = 1**

**Voronoi diagram:**
- partitions the space into regions
- boundaries are equal distance from training points

**Classification boundary:**
- non-linear
A sampling assumption: training and test data

- Assume that the training examples are drawn independently from the set of all possible examples.
- This makes it very unlikely that a strong regularity in the training data will be absent in the test data.
- Measure classification error as \[ \text{error} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[y_i \neq f(x_i)] \] The “risk”

loss function

\[ K = 1 \]

error = 0.0  
error = 0.15
Generalization

- The real aim of supervised learning is to do well on test data that is not known during learning.
- Choosing the values for the parameters that minimize the loss function on the training data is not necessarily the best policy.
- We want the learning machine to model the true regularities in the data and to ignore the noise in the data.
K = 1

Training data

error = 0.0

Testing data

error = 0.15

K = 3

Training data

error = 0.0760

Testing data

error = 0.1340
$K = 7$

<table>
<thead>
<tr>
<th>Training data</th>
<th>Testing data</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Graph" /></td>
<td><img src="image2.png" alt="Graph" /></td>
</tr>
<tr>
<td>Error = 0.1320</td>
<td>Error = 0.1110</td>
</tr>
</tbody>
</table>

$K = 21$

<table>
<thead>
<tr>
<th>Training data</th>
<th>Testing data</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Graph" /></td>
<td><img src="image4.png" alt="Graph" /></td>
</tr>
<tr>
<td>Error = 0.1120</td>
<td>Error = 0.0920</td>
</tr>
</tbody>
</table>
Properties and training

As K increases:
- Classification boundary becomes smoother
- Training error can increase

Choose (learn) K by cross-validation
- Split training data into training and validation
- Hold out validation data and measure error on this

Example: hand written digit recognition

- MNIST data set
- Distance = raw pixel distance between images
- 60K training examples
- 10K testing examples
- K-NN gives 5% classification error

\[ D(A, B) = \sum_{ij} \sqrt{(a_{ij} - b_{ij})^2} \]
Summary

Advantages:

- K-NN is a simple but effective classification procedure
- Applies to multi-class classification
- Decision surfaces are non-linear
- Quality of predictions automatically improves with more training data
- Only a single parameter, K; easily tuned by cross-validation

![Graph showing decision surfaces]

Summary

Disadvantages:

- What does nearest mean? Need to specify a distance metric.
- Computational cost: must store and search through the entire training set at test time. Can alleviate this problem by thinning, and use of efficient data structures like KD trees.

![Graph showing data distribution]
• Suppose we are given a training set of N observations

\((x_1, \ldots, x_N)\) and \((y_1, \ldots, y_N)\), \(x_i, y_i \in \mathbb{R}\)

• Regression problem is to estimate \(y(x)\) from this data

### K-NN Regression

**Algorithm**

• For each test point, \(x\), find the \(K\) nearest samples \(x_i\) in the training data and their values \(y_i\)
  
  • Output is mean of their values

\[
 f(x) = \frac{1}{K} \sum_{i=1}^{K} y_i
\]

• Again, need to choose (learn) \(K\)
Regression example: polynomial curve fitting

- The green curve is the true function (which is not a polynomial)

- The data points are uniform in x but have noise in y.

- We will use a loss function that measures the squared error in the prediction of \( y(x) \) from \( x \). The loss for the red polynomial is the sum of the squared vertical errors.

\[
E(w) = \frac{1}{2} \sum_{i=1}^{N} [y(x_i, w) - t_i]^2
\]

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

Some fits to the data: which is best?
Over-fitting

- test data: a different sample from the same true function

Root-Mean-Square (RMS) Error: $E_{\text{RMS}} = \sqrt{\frac{2E(w^*)}{N}}$

- training error goes to zero, but test error increases with M

Trading off goodness of fit against model complexity

- If the model has as many degrees of freedom as the data, it can fit the training data perfectly

- But the objective in ML is generalization

- Can expect a model to generalize well if it explains the training data surprisingly well given the complexity of the model.
### Polynomial Coefficients

<table>
<thead>
<tr>
<th></th>
<th>$M = 0$</th>
<th>$M = 1$</th>
<th>$M = 3$</th>
<th>$M = 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w^*_0$</td>
<td>0.19</td>
<td>0.82</td>
<td>0.31</td>
<td>0.35</td>
</tr>
<tr>
<td>$w^*_1$</td>
<td>-1.27</td>
<td>7.99</td>
<td>232.37</td>
<td></td>
</tr>
<tr>
<td>$w^*_2$</td>
<td>-25.43</td>
<td>-5321.83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w^*_3$</td>
<td>17.37</td>
<td>48568.31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w^*_4$</td>
<td></td>
<td>-231639.30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w^*_5$</td>
<td></td>
<td>640042.26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w^*_6$</td>
<td></td>
<td>-1061800.52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w^*_7$</td>
<td></td>
<td>1042400.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w^*_8$</td>
<td></td>
<td>-557682.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w^*_9$</td>
<td></td>
<td>125201.43</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### How to prevent overfitting?

- Add more data than the model “complexity”

- For 9th order polynomial:
How to prevent over fitting? II

- Regularization: penalize large coefficient values

\[
\tilde{E}(w) = \frac{1}{2} \sum_{i=1}^{N} \left\{ y(x_i, w) - t_i \right\}^2 + \frac{\lambda}{2} \| w \|^2
\]

“ridge” regression

In practice use validation data to choose \( \lambda \) (not test)

- cf with KNN classification as \( K \) increases
- we will return to regularization for regression later

### Polynomial Coefficients

<table>
<thead>
<tr>
<th>( w_i )</th>
<th>( \ln \lambda = -\infty )</th>
<th>( \ln \lambda = -18 )</th>
<th>( \ln \lambda = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_0 )</td>
<td>0.35</td>
<td>0.35</td>
<td>0.13</td>
</tr>
<tr>
<td>( w_1 )</td>
<td>232.37</td>
<td>4.74</td>
<td>-0.05</td>
</tr>
<tr>
<td>( w_2 )</td>
<td>-5321.83</td>
<td>-0.77</td>
<td>-0.06</td>
</tr>
<tr>
<td>( w_3 )</td>
<td>48568.31</td>
<td>-31.97</td>
<td>-0.05</td>
</tr>
<tr>
<td>( w_4 )</td>
<td>-231639.30</td>
<td>-3.89</td>
<td>-0.03</td>
</tr>
<tr>
<td>( w_5 )</td>
<td>640042.26</td>
<td>55.28</td>
<td>-0.02</td>
</tr>
<tr>
<td>( w_6 )</td>
<td>-1061800.52</td>
<td>41.32</td>
<td>-0.01</td>
</tr>
<tr>
<td>( w_7 )</td>
<td>1042400.18</td>
<td>-45.95</td>
<td>-0.00</td>
</tr>
<tr>
<td>( w_8 )</td>
<td>-557682.99</td>
<td>-91.53</td>
<td>0.00</td>
</tr>
<tr>
<td>( w_9 )</td>
<td>125201.43</td>
<td>72.68</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Summary: How to set parameters?

Use a validation set:

Divide the total dataset into three subsets:
- **Training data** is used for learning the parameters of the model.
- **Validation data** is not used for learning but is used for deciding what type of model and what amount of regularization works best.
- **Test data** is used to get a final, unbiased estimate of how well the learning machine works. We expect this estimate to be worse than on the validation data.

We could then re-divide the total dataset to get another unbiased estimate of the true error rate.

Generalization Problem in Classification

- **Underfitting**
- **Overfitting**

- Again, need to control the complexity of the (discriminant) function
What comes next?

• Learning by optimizing a cost function:

\[ \hat{E}(w) = \frac{1}{2} \sum_{i=1}^{N} \{y(x_i, w) - t_i\}^2 + \frac{\lambda}{2} \|w\|^2 \]

loss function  regularization

• In general  Minimize with respect to \( f \in \mathcal{F} \)

\[ \sum_{i=1}^{N} l(f(x_i), y_i) + \lambda R(f) \]

• choose loss function for: classification, regression, clustering …

• choose regularization function

Background reading

• Bishop, chapter 1

• Hastie et al, chapter 2

• Witten & Frank, chapter 1 for example applications

• More on web page:  
  http://www.robots.ox.ac.uk/~az/lectures/ml