Machine Learning: Pattern Recognition
Lecture 12: Combining Models

University of Amsterdam

October 22, 2012
1. **Introduction**
   - Bias-Variance Decomposition

2. **Bagging and Boosting**
   - Bagging
   - Boosting

3. **Tree-based models**
   - Classification trees
   - Random Forests

4. **Conditional Mixture Models**
   - Mixture vs conditional mixture
1 Introduction
   - Bias-Variance Decomposition

2 Bagging and Boosting
   - Bagging
   - Boosting

3 Tree-based models
   - Classification trees
   - Random Forests

4 Conditional Mixture Models
   - Mixture vs conditional mixture
Combining models

Input → Classifier → Output

What are committees?

- **Traditional approach:** train a classifier to predict a class
- **Committee:** Combine the output of multiple classifiers
  - For example, average the outputs
  - Alternatively, create a “meta-classifier”
What are committees?

- Traditional approach: train a classifier to predict a class
- Committee: Combine the output of multiple classifiers
  - For example, average the outputs
  - Alternatively, create a “meta-classifier”
Combining models

What are committees?

- Traditional approach: train a classifier to predict a class
- **Committee**: Combine the output of multiple classifiers
  - For example, average the outputs
  - Alternatively, create a “meta-classifier”
What are committees?

- Traditional approach: train a classifier to predict a class
- Committee: Combine the output of multiple classifiers
  - For example, average the outputs
  - Alternatively, create a “meta-classifier”
Combining models

What are committees?

- Traditional approach: train a classifier to predict a class
- Committee: Combine the output of multiple classifiers
  - For example, average the outputs
  - Alternatively, create a “meta-classifier”
What are committees?

- Traditional approach: train a classifier to predict a class
- Committee: Combine the output of multiple classifiers
  - For example, average the outputs
  - Alternatively, create a “meta-classifier”
Consider $M$ regression models $y_m(x), 1 \leq m \leq M$ predicting $h(x)$. Each individual prediction error is

$$
\epsilon_m(x) = h(x) - y_m(x),
$$

with an averaging committee:

$$
y(x) = \frac{1}{M} \sum_{m=1}^{M} y_m(x)
$$
Why committees?

The expected sum-squared errors are:

<table>
<thead>
<tr>
<th>Individual model (average)</th>
<th>Committee</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_{AV} = \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_x[\epsilon_m^2(x)]$</td>
<td>$E_{COM} = \mathbb{E}<em>x \left[ \left( \frac{1}{M} \sum</em>{m=1}^{M} \epsilon_m(x) \right)^2 \right]$</td>
</tr>
</tbody>
</table>

so that, if the errors are uncorrelated, we get

$$E_{COM} = \frac{1}{M^2} \mathbb{E}_x \left[ \sum_{m=1}^{M} \epsilon_m^2(x) + 2 \sum_{m \neq n} \epsilon_m(x)\epsilon_n(x) \right] = \frac{1}{M} E_{AV}$$
Why committees?

In theory, committees can vastly reduce the expected error of individual classifiers

- Make the expected error arbitrarily small by increasing $M$
- In practice, the classifiers are highly correlated
  - The error reduction is then small
- But: it can be shown that

$$E_{AV} \geq E_{COM}$$

- We can improve the performance of committees by decreasing the correlation between the classifiers
Bias-Variance Decomposition

Consider multiple training data sets $D = \{(x_n, t_n)\}$ of fixed size. Taking the expected squared loss of a model, we can decompose:

$$
\mathbb{E}_D[(y_D(x) - t)^2] = \left(\mathbb{E}_D[y_D(x)] - t\right)^2 + \mathbb{E}_D[(y_D(x) - \mathbb{E}_D[y_D(x)])^2]
$$

**Interpretation:**
- The bias captures how well the model *can* perform. Flexible models will have low bias.
- The variance captures how much the end model depends on the specific dataset. Flexible models will have high variance.
Bias-Variance decomposition

Consider multiple training data sets $D = \{(x_n, t_n)\}$ of fixed size.

Taking the expected squared loss of a model, we can decompose:

$$
E_D[(y_D(x) - t)^2] = \left( E_D[y_D(x)] - t \right)^2 + E_D[(y_D(x) - E_D[y_D(x)])^2] 
$$

### Bias

The bias captures how well the model can perform.

- **Flexible models will have low bias.**

### Variance

The variance captures how much the end model depends on the specific dataset.

- **Flexible models will have high variance.**
Bias-Variance decomposition:

- Gives us insight into how a particular model generalises
  - High bias-low variance models do not learn from the data
  - Low bias-high variance models overfit on the training data
  - Optimal model flexibility (e.g., regularisation): good bias–variance trade-off.
- Has little practical value: single training dataset
- Provides insight into why committees are useful

Optimal ensemble learning

For best ensemble performance, we want the base learners to be as accurate as possible and as diverse as possible.
Bias-Variance Decomposition:

- Gives us insight into how a particular model generalises
  - High bias-low variance models do not learn from the data
  - Low bias-high variance models overfit on the training data
  - Optimal model flexibility (e.g., regularisation): good bias–variance trade-off.
- Has little practical value: single training dataset
- Provides insight into why committees are useful

Optimal ensemble learning
For best ensemble performance, we want the base learners to be as accurate as possible and as diverse as possible
Bias-variance: an example
Bias-variance: an example
1. Introduction
   - Bias-Variance Decomposition

2. Bagging and Boosting
   - Bagging
   - Boosting

3. Tree-based models
   - Classification trees
   - Random Forests

4. Conditional Mixture Models
   - Mixture vs conditional mixture
Making committees

Where do we get the base learners?

1. Single type of classifiers:
   - Homogeneous learners

2. Multiple types of classifiers:
   - Heterogeneous learners

Diversity in homogeneous learners?

- Subsample the training data
- Add randomness to the learning algorithm
- Manipulate attributes or outputs
Making committees

Where do we get the base learners?

1. Single type of classifiers:
   
   Homogeneous learners

2. Multiple types of classifiers:
   
   Heterogeneous learners

Diversity in homogeneous learners?

- Subsample the training data
- Add randomness to the learning algorithm
- Manipulate attributes or outputs
Bagging: Bootstrap Aggregation

We rarely have infinite training datasets...

- Nor do we have many
- Using bootstrapping, we can create new datasets
- The correlation between datasets is then known and kept small

**Bootstrap aggregation:**
Simply average the outcomes of classifiers trained on different bootstrap datasets
In this example:

- A polynomial was fitted to 10 noisy training points (red)
- 1000 polynomials were fitted to bootstrap sets from the same 10 datapoints and averaged (blue line)
Bagging: an example

In this example:
- A polynomial was fitted to 10 noisy training points (red)
- 1000 polynomials were fitted to bootstrap sets from the same 10 datapoints and averaged (blue line)
Bagging

- Improves results with high-variance models
- No independent datasets (⇒ small improvements)
- Cannot help with high bias models
Bagging and Boosting

Introduction

Tree-based models

Conditional Mixture Models

Summary

Bagging
**Boosting**

**Weak learner** Learner that performs better than random

**Strong learner** Learner with accuracy $1 - \epsilon$, where $\epsilon$ is arbitrarily small

[Shapire 1990]: Weak learners in the same class as strong learners

**Boosting**

A technique to combine weak learners to form a strong learner
Boosting

Weak learner Learner that performs better than random

Strong learner Learner with accuracy $1 - \epsilon$, where $\epsilon$ is arbitrarily small

[Shapire 1990]: Weak learners in the same class as strong learners

Boosting

A technique to combine weak learners to form a strong learner
Adaptive boosting (Adaboost)

Adaptive boosting:
- Assign each training datapoint a weight
- Iterate:
  - Train a classifier based on the weighted training data
  - Assign this classifier a weight based on how well it performs
  - Update the datapoints' weights based on how many classifiers classify it correctly
Adaptive boosting: the algorithm

1. Set $w_n^{(1)} = \frac{1}{N}$
2. For $m = 1, \ldots, M$
   1. Fit $y_m(x)$ by minimising
      $$E_m = \sum_{n \in M_m} w_n^{(m)}$$
   2. Evaluate
      $$\epsilon_m = \frac{\sum_{n \in M_m} w_n^{(m)}}{\sum_n w_n^{(m)}}$$
      set $\alpha_m = \log \frac{1-\epsilon_m}{\epsilon_m}$
3. Update the weights
   $$w_n^{(m+1)} = \begin{cases} w_n^{(m)} & \text{if } y_m(x_n) = t_n \\ w_n^{(m)} \exp \alpha_m & \text{Otherwise} \end{cases}$$
Adaptive boosting: the algorithm

1. Set $w_n^{(1)} = \frac{1}{N}$

2. For $m = 1, \ldots, M$
   - Fit $y_m(x)$ by minimising
     \[ E_m = \sum_{n \in M_m} w_n^{(m)} \]
   - Evaluate
     \[ \epsilon_m = \frac{\sum_{n \in M} w_n^{(m)}}{\sum_n w_n^{(m)}} \]
     set $\alpha_m = \log \frac{1-\epsilon_m}{\epsilon_m}$

3. Update the weights

\[
 w_n^{(m+1)} = \begin{cases} 
 w_n^{(m)} & \text{if } y_m(x_n) = t_n \\
 w_n^{(m)} \exp \alpha_m & \text{Otherwise}
\end{cases}
\]
Adaptive boosting: the algorithm

1. Set $w_n^{(1)} = \frac{1}{N}$

2. For $m = 1, \ldots, M$
   1. Fit $y_m(x)$ by minimising
      $$E_m = \sum_{n \in \mathcal{M}_m} w_n^{(m)}$$
   2. Evaluate
      $$\epsilon_m = \frac{\sum_{n \in \mathcal{M}_m} w_n^{(m)}}{\sum_n w_n^{(m)}},$$
      set $\alpha_m = \log \frac{1-\epsilon_m}{\epsilon_m}$
   3. Update the weights
      $$w_n^{(m+1)} = \begin{cases} w_n^{(m)} & \text{if } y_m(x_n) = t_n \\ w_n^{(m)} \exp \alpha_m & \text{Otherwise} \end{cases}$$
Committee $M = 0$
Boosting Example

Classifier 1
Committee $M = 1$
Committee $M = 2$
Boosting Example

Classifier 3
Boosting Example

Committee $M = 3$
Committee $M = 4$
Introduction

Bagging and Boosting

Tree-based models

Conditional Mixture Models

Summary

Boosting Example

Classifier 5

Classifier 5
Committee $M = 5$
Committee $M = 6$
Introduction

Bagging and Boosting

Tree-based models

Conditional Mixture Models

Summary

Boosting Example

Classifier 7
Committee $M = 7$
Committee $M = 10$
Committee $M = 20$
Committee $M = 30$
About Adaboost

Adaboost can be interpreted as minimising

\[ E = \sum_{n=1}^{N} \exp \left( -\frac{t_n}{2} \sum_{m=1}^{M} \alpha_m y_m(x_n) \right) \]

As a consequence:

1. It strongly penalises misclassifications, not robust to outliers!
2. It does not generalise to more than 2 classes
3. Choosing a different error function
   - Allows multiclass classification and even regression (e.g. Gradient Boosting)
   - Makes robust classifiers possible
A nice application of boosting:

- Very simple features (HAAR wavelets)
  - Use integral images to compute these very fast
- Use cascading for speedup
Viola-Jones face detector

A nice application of boosting:
- Very simple features (HAAR wavelets)
  - Use integral images to compute these very fast
- Use *cascading* for speedup
Viola-Jones face detector

A nice application of boosting:
- Very simple features (HAAR wavelets)
  - Use integral images to compute these very fast
- Use *cascading* for speedup
Viola-Jones face detector
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Bagging and Boosting</th>
<th>Tree-based models</th>
<th>Conditional Mixture Models</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Introduction</td>
<td>- Bagging</td>
<td>- Classification trees</td>
<td>- Mixture vs conditional mixture</td>
<td></td>
</tr>
<tr>
<td>- Bias-Variance Decomposition</td>
<td>- Boosting</td>
<td>- Random Forests</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Binary Tree classifier

\[ x_1 > \theta_1 \]
\[ x_2 \leq \theta_2 \]
\[ x_1 \leq \theta_4 \]
\[ x_2 > \theta_3 \]

A B C D E
Tree-based models split the input space into regions

- Each region gets its own classifier
- The classifiers can be extremely simple (typically: constant)
Tree-based models

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpretable!</td>
<td>Final tree depends strongly on particular data</td>
</tr>
<tr>
<td>Simple and fast</td>
<td>Hard decisions, aligned with dimensions</td>
</tr>
<tr>
<td>If let to grow, will learn perfect</td>
<td>Finding best tree is intractable</td>
</tr>
<tr>
<td>classification on the training data</td>
<td></td>
</tr>
<tr>
<td>Pruning (using validation set) allows</td>
<td></td>
</tr>
<tr>
<td>proper generalisation</td>
<td></td>
</tr>
</tbody>
</table>

**Pros**

- Interpretable!
- Simple and fast
- If let to grow, will learn perfect classification on the training data
- Pruning (using validation set) allows proper generalisation

**Cons**

- Final tree depends strongly on particular data
- Hard decisions, aligned with dimensions
- Finding best tree is intractable
Random Forests

Combine trees with bagging and random feature selection

Procedure: for $N$ datapoints and $M$ features, pre-specify $m \ll M$

1. Repeat $K$ times:
   1. Get a bootstrap sample
   2. At each node in the tree:
      1. select $m$ features at random
      2. Find the optimal split based on these $m$ features and the training set
   3. Fully grow the tree (no pruning)

This is often considered one of the most powerful committee methods
$M = 2$
Random Forests

\[ M = 10 \]
Random Forests

\[ M = 50 \]
Random Forests

$M = 100$
$M = 100$ Final decision

$M = 100$, voting
1 Introduction
   - Bias-Variance Decomposition

2 Bagging and Boosting
   - Bagging
   - Boosting

3 Tree-based models
   - Classification trees
   - Random Forests

4 Conditional Mixture Models
   - Mixture vs conditional mixture
Conditional Mixture Models

Traditional mixture models

\[ p(x) = \sum_{k=1}^{K} \pi_k p(x|k) \]
Conditional mixture models

\[ p(x) = \sum_{k=1}^{K} \pi_k(x)p(x|k) \]
Mixture model
Mixture model

Logistic regression
Mixture model
Hierarchical Conditional Mixture Models

The distribution specified by each mixture element can be anything

- Including a Conditional Mixture Model

\[ p(x) = \sum_{k=1}^{K} \pi_k(x)p(x|k) \]

- If \( \pi \) were a constant, this would simplify to a normal mixture model (with \( \sum_k L_k \) elements)

- Since \( \pi(x) \) can be a complex function of \( x \), the HCM can model very complex distributions

- Notice the similarity with MDN!
Hierarchical Conditional Mixture Models

The distribution specified by each mixture element can be anything

- Including a Conditional Mixture Model

\[
p(x) = \sum_{k=1}^{K} \pi_k(x) \sum_{l=1}^{L_k} \pi_{kl}(x)p(x|l)
\]

- If \( \pi \) were a constant, this would simplify to a normal mixture model (with \( \sum_k L_k \) elements)

- Since \( \pi(x) \) can be a complex function of \( x \), the HCM can model very complex distributions

- Notice the similarity with MDN!
Mixture of density networks
1. Introduction
   - Bias-Variance Decomposition

2. Bagging and Boosting
   - Bagging
   - Boosting

3. Tree-based models
   - Classification trees
   - Random Forests

4. Conditional Mixture Models
   - Mixture vs conditional mixture
Wrap-up

To summarise:

- Combine models to improve their expressive power (cfr. Mixture of Gaussians)
- Combining independent models can dramatically improve performance
- Making different models responsible for different areas of the space combines simple models into very flexible models

Exercise session:

- Questions
- Mock exam

Lab session:

- No additional lab exercise, so you can prepare for the exam.