7. Boosting and Bagging

Bagging
Bagging

• So far: Boosting as an **ensemble** learning method, i.e.: a combination of (weak) learners

• A different way to combine classifiers is known as **bagging ("bootstrap aggregating")**

• **Idea**: sample $M$ “bootstrap” data sets (sub sets) with replacement from the training set and train different models

• Overall classifier is then the **average** over all models:

$$\bar{y}(x) = \frac{1}{M} \sum_{m=1}^{M} y_m(x)$$
Bagging

Bagging reduces the expected error. E.g. in regression:

\[ y_m(x) = h(x) + \epsilon_m(x) \]

prediction  ground truth  error

- Expected error:  \[ E_x[(y_m(x) - h(x))^2] \]

- Average error over all (weak) learners:

\[ E_{AV} = \frac{1}{M} \sum_{m=1}^{M} E_x[(y_m(x) - h(x))^2] \]

- Average error of committee:

\[ E_{COM} = E_x[(\bar{y}(x) - h(x))^2] \]
Bagging

Bagging reduces the expected error. E.g. in regression:

\[ y_m(x) = h(x) + \epsilon_m(x) \]

- **Expected error:** \( E_x[(y_m(x) - h(x))^2] \)
- **Average error over all weak learners (indep.):**
  \[
  E_{AV} = \frac{1}{M} \sum_{m=1}^{M} E_x[(y_m(x) - h(x))^2]
  \]
- **In contrast: average error of committee:**
  \[
  E_{COM} = E_x \left[ \left( \frac{1}{M} \sum_{m=1}^{M} y_m(x) - h(x) \right)^2 \right]
  \]
Bagging

Bagging reduces the expected error. E.g. in regression:

\[ y_m(x) = h(x) + \epsilon_m(x) \]

- Expected error: \[ E_x[(y_m(x) - h(x))^2] \]
- Average error over all (weak) learners:

\[ E_{AV} = \frac{1}{M} \sum_{m=1}^{M} E_x[(y_m(x) - h(x))^2] \]

- Average error of committee if learners are uncorrelated:

\[ E_{COM} = \frac{1}{M} E_{AV} \]
Random Forests

Given: training set of size $N \{x_1, \ldots, x_N\}$ $x_i \in \mathbb{R}^d$

1. Randomly sample $n \leq N$ elements from training set with replacement (repetitions likely)
2. Randomly select a subset of $p$ features ($p<d$)
3. Pick from those the feature that produces the best split of the data
4. Perform the split and go back to 2.
5. If maximum tree depth is reached:
6. If number of trees $M$ is reached then stop.
7. Else: go to 1. building a new tree.
Random Forests

- Each bag is a subset of the entire training data
- Repetitions are very likely

**Note:** in this figure, repetitions are not shown
Performance of Random Forests

The error rate depends on two main aspects:

- **the correlation** between any two trees:
  - high correlation $\rightarrow$ high error rate
- **the strength** of each tree (low error per tree)
  - higher strength $\rightarrow$ lower overall error rate

These values are mainly influenced by $p$:

- If $p$ is low: correlation and strength are low
- If $p$ is high: correlation and strength are high

There is usually an “optimal range” of $p$
Splitting Criterion

- **Aim**: split such that both data sub-sets contain samples that are as **pure** as possible.
- **Possible impurity values**:
  - Misclassification error: let $\pi$ be the prob of class 1 (binary classification), i.e. $\pi = P(y = 1|\Omega)$, then use $\min(\pi, 1 - \pi)$.
  - Gini index: $2\pi(1 - \pi)$.
  - Deviance: $-\pi \log \pi - (1 - \pi) \log(1 - \pi)$.
  - For regression trees, we can use the mean-squared error.
Properties of Random Forests

- They reduce the **variance** of the classification estimate, by training several trees on randomly sampled subsets of the data ("**bagging**")
- They tend to give **uncorrelated** trees by randomly sampling the features (splits)
- They can **not overfit**! One can use as many trees as required
- Only restriction is memory
- Random Forests have very good accuracy and are widely used, e.g. for body pose recognition
Advantages of Random Forests

• One of the best classifiers in general
• Runs very **efficiently** on large data sets
• Can handle thousands of **feature dimensions**
• Can provide **importance** of variables
• Generates an **unbiased** estimate of the error
• Can deal with missing data
• Implicitly generates **proximities** of pairs of data samples, useful e.g. for clustering
• Can be extended to unlabeled data
Out-of-Bag (OOB) Error

• All samples that are not used to train a tree are called the out-of-bag data
• These samples can be used to evaluate the overall random forest without an additional validation set
Out-of-Bag (OOB) Error

- All samples that are not used to train a tree are called the **out-of-bag data**
- This is done by evaluating each tree with its own out-of-bag data
Variable Importance

Idea: rate variables (features) according to their potential to change the tree structure

Method:
1. compute tree impurity $\iota_m$ (sum of node impurities of leaf nodes per tree) for each tree $m = 1, \ldots, M$
2. for all features $j = 1, \ldots, d$: permute the $j$th feature value in the out-of-bag data
3. compute tree impurity of the permuted data $\iota_{jm}$
4. compute the difference of tree impurity:

$$\delta_{mj} = \iota_{mj} - \iota_m$$
Variable Importance

Idea: rate variables (features) according to their potential to change the tree structure

Method:
1. compute **tree impurity** $\ell_m$ (sum of node impurities of leaf nodes per tree) for each tree $m=1,\ldots,M$
2. for all features $j=1,\ldots,d$: **permute** the $j$th feature value in the out-of-bag data
3. compute tree impurity of the **permuted** data $\ell_{jm}$
4. compute the **difference** of tree impurity
5. variable importance is: $\frac{\bar{\delta}_j}{\sqrt{\frac{1}{M} \sum_m (\delta_{mj} - \bar{\delta}_m)^2}}$
Summary

- Boosting uses **weak** classifiers and turns them into a **strong** one (arbitrarily small training error!)
- AdaBoost minimizes the **exponential** loss
- To be more robust against outliers, we can use **LogitBoost**
- Face detection can be done with Boosting
- Bagging reduces the overall committee error
- Random Forests are an example of bagging with a very good performance
8. Sequential Data
Bayes Filter (Rep.)

We can describe the overall process using a **Dynamic Bayes Network**:

\[
\begin{align*}
    & p(z_t \mid x_{0:t}, u_{1:t}, z_{1:t}) = p(z_t \mid x_t) \quad \text{(measurement)} \\
    & p(x_t \mid x_{0:t-1}, u_{1:t}, z_{1:t}) = p(x_t \mid x_{t-1}, u_t) \quad \text{(state)}
\end{align*}
\]
Bayes Filter Without Actions

Removing the action variables we obtain:

\[ p(z_t \mid x_{0:t}, z_{1:t}) = p(z_t \mid x_t) \quad \text{(measurement)} \]
\[ p(x_t \mid x_{0:t-1}, z_{1:t}) = p(x_t \mid x_{t-1}) \quad \text{(state)} \]
A Model for Sequential Data

- Observations in sequential data should not be modeled as independent variables such as:

\[ Z_1, Z_2, Z_3, Z_4, Z_5, \ldots \]

- Examples: weather forecast, speech, handwritten text, etc.

- The observation at time \( t \) depends on the observation(s) of (an) earlier time step(s):

\[ Z_1 \rightarrow Z_2 \rightarrow Z_3 \rightarrow Z_4 \rightarrow Z_5 \rightarrow \ldots \]
A Model for Sequential Data

• The joint distribution is therefore (d-sep):

\[ p(z_1 \ldots z_n) = p(z_1) \prod_{i=2}^{n} p(z_i \mid z_{i-1}) \]

• **However:** often data depends on several earlier observations (not just one)
A Model for Sequential Data

\[ p(z_1 \ldots z_n) = p(z_1)p(z_2 | z_1) \prod_{i=3}^{n} p(z_i | z_{i-1}, z_{i-2}) \]

- **Problem**: number of stored parameters grows exponentially with the **order** of the Markov chain

- **Question**: can we model dependency of all previous observations with a limited number of parameters?
A Model for Sequential Data

Idea: Introduce hidden (unobserved) variables:

\[ \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5 \ldots \]

\[ \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, \mathbf{z}_5 \]
A Model for Sequential Data

Idea: Introduce **hidden** (unobserved) variables:

![Diagram of a model with hidden variables](image)

Now we have: \( \text{dsep}(x_n, \{x_1, \ldots, x_{n-2}\}, x_{n-1}) \)

\[
\iff p(x_n \mid x_1, \ldots, x_{n-2}, x_{n-1}) = p(x_n \mid x_{n-1})
\]

But:

\( \neg \text{dsep}(z_n, \{z_1, \ldots, z_{n-2}\}, z_{n-1}) \)

\[
\iff p(z_n \mid z_1, \ldots, z_{n-2}, z_{n-1}) \neq p(z_n \mid z_{n-1})
\]

And: number of parameters is \( nK(K-1) + \text{const.} \).
Example

- Place recognition for mobile robots
- 3 different states: corridor, room, doorway
- Problem: misclassifications
- Idea: use information from previous time step