Lecture 7 – Boosting

Thomas Schön
Division of Systems and Control
Department of Information Technology
Uppsala University.

Email: thomas.schon@it.uu.se
Summary of Lecture 6 (I/III)

CART: Classification And Regression Trees

Partition the input space using **recursive binary splitting**

- **Classification:** Majority vote within the region.
- **Regression:** Mean of training data within the region.
Deep tree models tend to have **low bias** but **high variance**.

**Bagging** (=bootstrap aggregating) is a general **ensemble method** that can be used to reduce model variance (well suited for trees):

1. For \( b = 1 \) to \( B \) (*can run in parallel*)
   (a) Draw a bootstrap data set \( \tilde{T}^b \) from \( T \).
   (b) Train a model \( \tilde{y}^b(x) \) using the bootstrapped data \( \tilde{T}^b \).

2. Aggregate the \( B \) models by outputting,
   - **Regression**: \( \hat{y}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \tilde{y}^b(x) \).
   - **Classification**: \( \hat{y}_{bag}(x) = \text{MajorityVote}\{\tilde{y}^b(x)\}_{b=1}^{B} \).
Random forests = bagged trees, but with further variance reduction achieved by decorrelating the $B$ ensemble members.

Specifically:

For each split in each tree only a random subset of $q \leq p$ inputs are considered as splitting variables.
Contents – Lecture 7

1. Boosting – the general idea
2. AdaBoost – the first practical boosting method
3. Robust loss functions
4. Gradient boosting
Boosting

Even a simple (classification or regression) model can typically capture some aspects of the input-output relationship.

Can we then learn an ensemble of “weak models”, each describing some part of this relationship, and combine these into one “strong model”?

**Boosting:**

- **Sequentially** learns an ensemble of weak models.
- Combine these into one strong model.
- General strategy – can in principle be used to improve any supervised learning algorithm.
- One of the most successful machine learning ideas!
The models are built **sequentially** such that each model tries to **correct the mistakes** made by the previous one!
Binary classification

We will restrict our attention to binary classification.

- Class labels are \(-1\) and \(1\), i.e. \(y \in \{-1, 1\}\).
- We have access to some (weak) base classifier, e.g. a classification tree.

\[\text{Note. Using labels } -1 \text{ and } 1 \text{ is mathematically convenient as it allows us to express a majority vote between } B \text{ classifiers } \hat{y}^1(x), \ldots, \hat{y}^B(x) \text{ as}
\]
\[
\text{sign} \left( \sum_{b=1}^{B} \hat{y}^b(x) \right) = \begin{cases} 
+1 & \text{if more plus-votes than minus-votes}, \\
-1 & \text{if more minus-votes than plus-votes}. 
\end{cases}
\]
Boosting procedure (for classification)

Boosting procedure:

1. Assign weights $w_i^1 = 1/n$ to all data points.
2. For $b = 1$ to $B$
   (a) Train a weak classifier $\hat{y}^{(b)}(x)$ on the weighted training data $\{(x_i, y_i, w_i^b)\}_{i=1}^n$.
   (b) Update the weights $\{w_i^{b+1}\}_{i=1}^n$ from $\{w_i^b\}_{i=1}^n$:
      i. Increase weights for all points misclassified by $\hat{y}^{(b)}(x)$.
      ii. Decrease weights for all points correctly classified by $\hat{y}^{(b)}(x)$.

The predictions of the $B$ classifiers, $\hat{y}^{(1)}(x), \ldots, \hat{y}^{(B)}(x)$, are combined using a weighted majority vote:

$$\hat{y}_{\text{boost}}(x) = \text{sign} \left( \sum_{b=1}^{B} \alpha^{(b)} \hat{y}^{(b)}(x) \right).$$
ex) Boosting illustration

\[ \hat{y}_{\text{boost}}(x) = \text{sign}\left( \sum_{b=1}^{3} \alpha^{(b)} \hat{y}^{(b)}(x) \right) \]
The technical details...

Q1: How do we reweight the data?

Q2: How are the coefficients $\alpha^{(1)}, \ldots, \alpha^{(B)}$ computed?
**Exponential loss**

Loss functions for binary classifier $\hat{y}(x) = \text{sign}(c(x))$. 

Exponential loss function $L(y, c(x)) = \exp(-y \cdot c(x))$ plotted vs. margin $y \cdot c(x)$. The misclassification loss $\mathbb{I}\{y \neq \hat{y}(x)\} = \mathbb{I}\{y \cdot c(x) < 0\}$ is plotted as comparison.
AdaBoost pseudo-code

1. Assign weights $w_i^1 = 1/n$ to all data points.
2. For $b = 1$ to $B$
   (a) Train a weak classifier $\hat{y}^{(b)}(x)$ on the weighted training data $\{(x_i, y_i, w_i^b)\}_{i=1}^n$.
   (b) Update the weights $\{w_i^{b+1}\}_{i=1}^n$ from $\{w_i^b\}_{i=1}^n$:
      i. Increase weights for all points misclassified by $\hat{y}^{(b)}(x)$.
      ii. Decrease weights for all points correctly classified by $\hat{y}^{(b)}(x)$.
      iii. Compute weighted classification error $E^{b}_{\text{train}} = \sum_{i=1}^n w_i^b \{y_i \neq \hat{y}^{(b)}(x_i)\}$
      iv. Compute classifier “confidence” $\alpha^b = 0.5 \log((1 - E^{b}_{\text{train}})/E^{b}_{\text{train}})$.
      v. Compute new weights $w_i^{b+1} = w_i^b \exp(-\alpha^b y_i \hat{y}^{(b)}(x_i))$, $i = 1, \ldots, n$
   iv. Normalize. Set $w_i^{b+1} \leftarrow w_i^{b+1} / \sum_{j=1}^n w_i^{b+1}$, for $i = 1, \ldots, n$.
3. Output $\hat{y}^{(B)}(x) = \text{sign} \left( \sum_{b=1}^B \alpha^b \hat{y}^{(b)}(x) \right)$.

# Boosting vs. bagging

<table>
<thead>
<tr>
<th>Bagging</th>
<th>Boosting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learns base models in parallel</td>
<td>Learns base models sequentially</td>
</tr>
<tr>
<td>Uses bootstrapped datasets</td>
<td>Uses reweighted datasets</td>
</tr>
<tr>
<td>Does not overfit as $B$ becomes large</td>
<td>Can overfit as $B$ becomes large</td>
</tr>
<tr>
<td>Reduces variance but not bias (requires deep trees as base models)</td>
<td>Also reduces bias! (works well with shallow trees)</td>
</tr>
</tbody>
</table>

**N.B.** Boosting does *not* require each base model to have low bias. Thus, a shallow classification tree (say, 4-8 terminal nodes) or even a tree with a single split (2 terminal nodes, a “stump”) is often sufficient.
ex) The Viola-Jones face detector
ex) The Viola-Jones face detector

The Viola-Jones face detector:

- Revolutionized computer face detection in 2001.
- First real-time system — soon built into standard point-and-shoot cameras.
- Based on AdaBoost!

ex) The Viola-Jones face detector

Uses extremely simple (and fast!) base models:

1. Place a “mask” of type $A$, $B$, $C$ or $D$ on top of the image
2. Value $= \sum$ (pixels in black area) $- \sum$ (pixels in white area)
3. Classify as face or noface.

Base models combined using AdaBoost, resulting in a fast and accurate face detector.
Robust loss functions

Why exponential loss?

- *Main reason* – computational simplicity (cf. squared loss in linear regression)

However, models using exponential loss is sensitive to "outliers", e.g. mislabeled or noisy data.
Robust loss functions

Instead of exponential loss we can use a robust loss function with a more gentle slope for negative margins.
Gradient boosting

No free lunch! The optimization problem

$$(\alpha^{(b)}, \hat{y}^{(b)}) = \arg\min_{(\alpha, \hat{y})} \sum_{i=1}^{n} L(y_i, c^{(b-1)}(x_i) + \alpha \hat{y}(x_i))$$

lacks a closed form solution for a general loss function $L(y, c(x))$.

Gradient boosting methods address this by using techniques from numerical optimization.
Gradient descent

Consider the generic optimization problem $\min_\theta J(\theta)$. A numerical method for solving this problem is gradient descent.

Initialize $\theta^{(0)}$ anditerate:

$$\theta^{(b)} = \theta^{(b-1)} - \gamma^{(b)} \nabla_\theta J(\theta^{(b-1)}) .$$

where $\alpha^b$ is the step size at the $b$th iteration.

Can be compared with boosting – sequentially construct an ensemble of models as:

$$c^{(b)}(x) = c^{(b-1)}(x) + \alpha^{(b)} \hat{y}^{(b)}(x)$$

for some base model $\hat{y}^{(b)}(x)$ and “step size” $\alpha^{(b)}$. 
Gradient boosting mimics the gradient descent algorithm by fitting the $b$th base model to the negative gradient of the loss function,

$$g_i^{(b)} = - \left[ \frac{\partial L(y_i, c)}{\partial c} \right]_{c = c^{(b-1)}(x_i)} , \quad i = 1, \ldots, n.$$ 

That is, $\hat{y}^{(b)}(x)$ is learned using $\{(x_i, g_i^{(b)})\}_{i=1}^n$ as training data.

Gradient boosting (with some additional bells and whistles) is used by the very popular libraries XGBoost and LightGBM,

xgboost.readthedocs.io/
lightgbm.readthedocs.io/
Classification methods (covered so far)

So far in the course we have discussed the following methods for classification...

**Parametric classifiers**
1. Logistic regression — *linear*
2. Linear discriminant analysis — *linear*
3. Quadratic discriminant analysis — *nonlinear*

**Nonparametric classifiers**
4. $k$-nearest neighbors
5. Classification trees

**Ensemble-based methods**
6. Bagging (bagged trees / random forests)
7. Boosting (AdaBoost)
A few concepts to summarize lecture 7

**Boosting:** Sequential ensemble method, where each consecutive model tries to correct the mistakes of the previous one.

**AdaBoost:** The first successful boosting algorithm. Designed for binary classification.

**Exponential loss:** The classification loss function used by AdaBoost.

**Gradient boosting:** A boosting procedure that can be used with any differentiable loss function.