Lecture 7 – Boosting

UPPSALA
UNIVERSITET

Fredrik Lindsten
Division of Systems and Control
Department of Information Technology
Uppsala University.

Email: fredrik.lindsten@it.uu.se

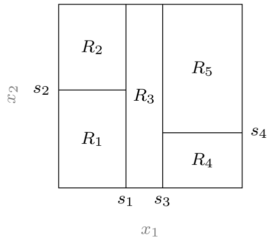
UPPSALA UNIVERSITET

Summary of Lecture 6 (I/III)

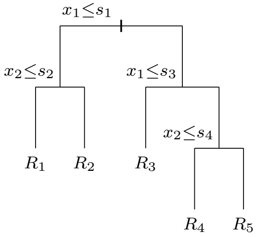
CART: Classification And Regression Trees

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

Partitioning of input space



Tree representation



- **Classification:** Majority vote within the region.
- **Regression:** Mean of training data within the region.

Summary of Lecture 6 (II/III)

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 $\widetilde{\mathcal{T}}^b$ from $\mathcal{T}$.
   (b) Train a model $\widetilde{y}^b(\mathbf{x})$ using the bootstrapped data $\widetilde{\mathcal{T}}^b$.
2. Aggregate the B models by outputting,
   - **Regression:** $\widehat{y}^{\text{bag}}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} \widetilde{y}^b(\mathbf{x})$.
   - **Classification:** $\widehat{y}^{\text{bag}}(\mathbf{x}) = \text{MajorityVote}\{\widetilde{y}^b(\mathbf{x})\}_{b=1}^{B}$.

Summary of Lecture 6 (III/III)

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

Boosting

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

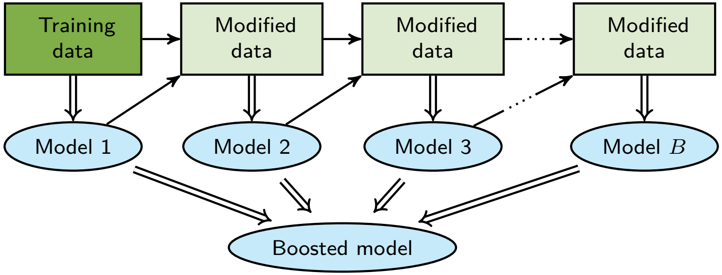
Can we then learn an ***ensemble*** of "weak models", each describing some part of this relationship, and combine these into one "strong model"?

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!

UPPSALA UNIVERSITET

Boosting



The models are built **sequentially** such that each models tries to **correct the mistakes** made by the previous one!

fredrik.lindsten@it.uu.se

UPPSALA UNIVERSITET

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.

Note. Using labels -1 and 1 is mathematically convenient as it allows us to express a majority vote between B classifiers $\widehat{y}^1(\mathbf{x}), \dots, \widehat{y}^B(\mathbf{x})$ as

$$\text{sign}\left(\sum_{b=1}^{B} \widehat{y}^b(\mathbf{x})\right) = \begin{cases} +1 & \text{if more plus-votes than minus-votes,} \\ -1 & \text{if more minus-votes than plus-votes.} \end{cases}$$

Boosting procedure (for classification)

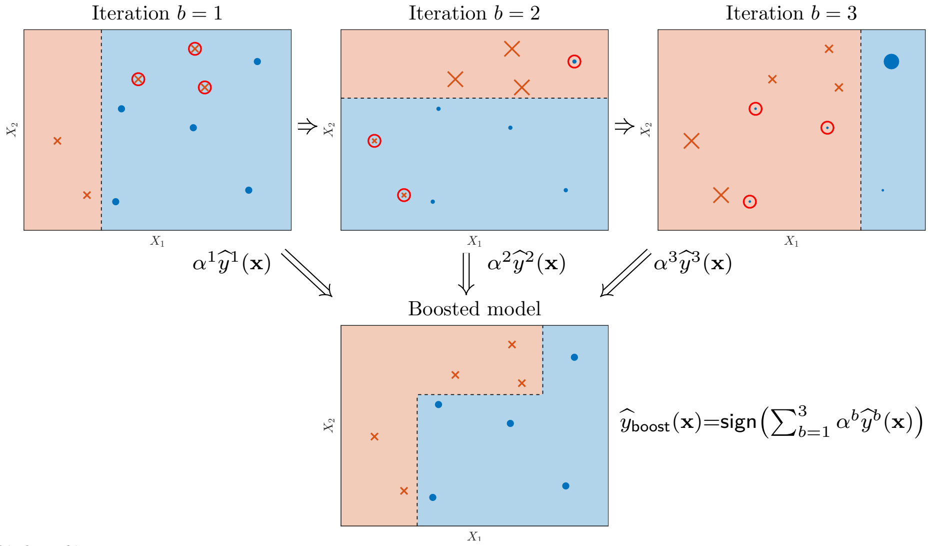
Boosting procedure:

1. Assign weights $w_i^1 = 1/n$ to all data points.
2. For $b = 1$ to B
   (a) Train a weak classifier $\widehat{y}^b(\mathbf{x})$ on the **weighted training data** $\{(\mathbf{x}_i, y_i, w_i^b)\}_{i=1}^n$.
   (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 $\widehat{y}^b(\mathbf{x})$.
      ii. Decrease weights for all points correctly classified by $\widehat{y}^b(\mathbf{x})$.

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

$$\widehat{y}^B_{\text{boost}}(\mathbf{x}) = \operatorname{sign}\left(\sum_{b=1}^{B} \alpha^b \widehat{y}^b(\mathbf{x})\right).$$

ex) Boosting illustration



Iteration $b = 1$
Iteration $b = 2$
Iteration $b = 3$
X_1
X_2
$\alpha^1 \widehat{y}^1(\mathbf{x})$
$\alpha^2 \widehat{y}^2(\mathbf{x})$
$\alpha^3 \widehat{y}^3(\mathbf{x})$
Boosted model

$$\widehat{y}_{\text{boost}}(\mathbf{x}) = \text{sign}\left(\sum_{b=1}^{3} \alpha^b \widehat{y}^b(\mathbf{x})\right)$$

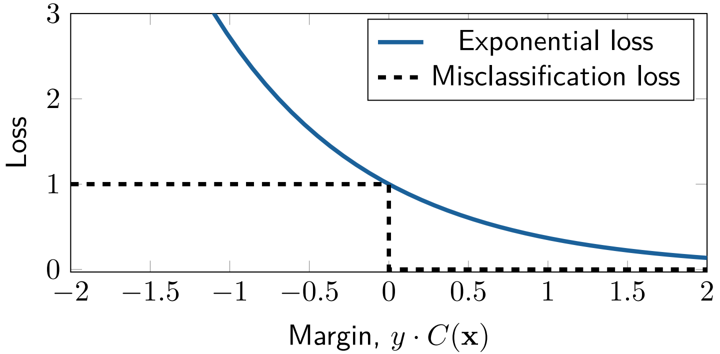
The technical details. . .

Q1: How do we reweight the data?

Q2: How are the coefficients $\alpha^1, \dots, \alpha^B$ computed?

Exponential loss

Loss functions for binary classifier $\widehat{y}(\mathbf{x}) = \text{sign}(C(\mathbf{x}))$.



Exponential loss function $L(y, C(\mathbf{x})) = \exp(-y \cdot C(\mathbf{x}))$ plotted vs. **margin** $y \cdot C(\mathbf{x})$.
The misclassification loss $\mathbb{I}\{y \neq \widehat{y}(\mathbf{x})\} = \mathbb{I}\{y \cdot C(\mathbf{x}) < 0\}$ is plotted as comparison.

AdaBoost pseudo-code

AdaBoost:

```
1. Assign weights $w_i^1 = 1/n$ to all data points.
2. For $b = 1$ to $B$
    (a) Train a weak classifier $\widehat{y}^b(\mathbf{x})$ on the weighted training data $\{(\mathbf{x}_i, y_i, w_i^b)\}_{i=1}^n$.
    (b) Update the weights $\{w_i^{b+1}\}_{i=1}^n$ from $\{w_i^b\}_{i=1}^n$:
        i. Compute $E_{\text{train}}^b = \sum_{i=1}^n w_i^b \mathbb{I}\{y_i \neq \widehat{y}^b(\mathbf{x}_i)\}$
        ii. Compute $\alpha^b = 0.5 \log((1 - E_{\text{train}}^b)/E_{\text{train}}^b)$.
        iii. Compute $w_i^{b+1} = w_i^b \exp(-\alpha^b y_i \widehat{y}^b(\mathbf{x}_i))$, $i = 1, \ldots, n$
        iv. Normalize. Set $w_i^{b+1} \leftarrow w_i^{b+1} / \sum_{j=1}^n w_j^{b+1}$, for $i = 1, \ldots, n$.
3. Output $\widehat{y}_{\text{boost}}^B(\mathbf{x}) = \text{sign}\left(\sum_{b=1}^B \alpha^b \widehat{y}^b(\mathbf{x})\right)$.
```

Y. Freund and R. E. Schapire. **Experiments with a New Boosting Algorithm**. *Proceedings of the 13th International Conference on Machine Learning (ICML)* Bari, Italy, 1996.

2003 Gödel Prize

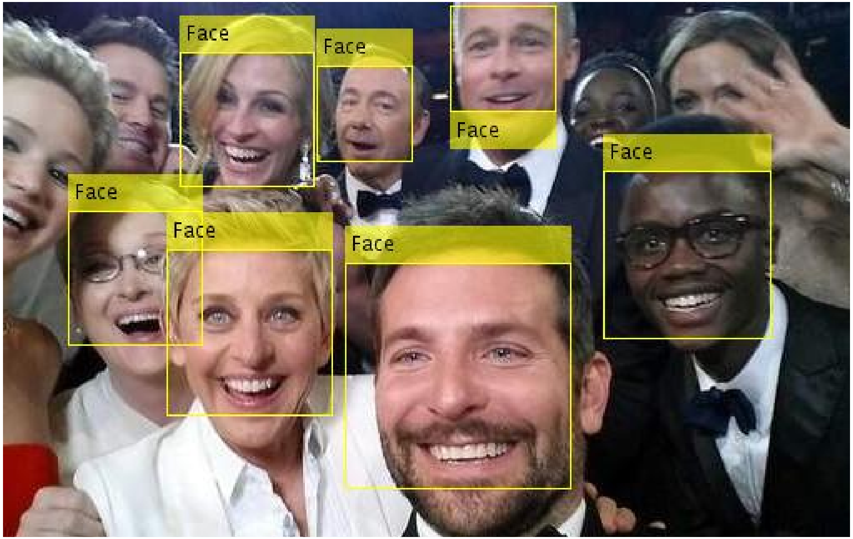
Boosting vs. bagging

Bagging	**Boosting**
Learns base models in parallel	Learns base models sequentially
Uses bootstrapped datasets	Uses reweighted datasets
Reduces variance but not bias (requires deep trees as base models)	Also reduces bias! (works well with shallow trees)

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.

UPPSALA UNIVERSITET

ex) The Viola-Jones face detector



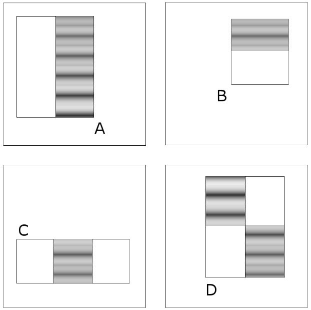
ex) The Viola-Jones face detector

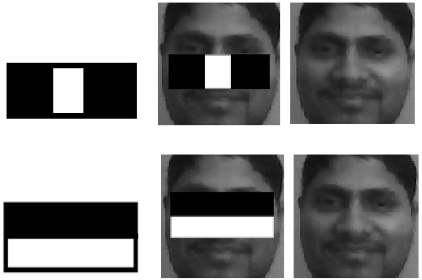
The Viola-Jones face detector:

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

P. Viola and M. Jones **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. Kauai, Hawaii, 2001.

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.

UPPSALA UNIVERSITET

Robust loss functions

Why exponential loss?

- *Main reason* – computational simplicity (cf. squared loss in linear regression)
- Exponential loss is related to the log-odds

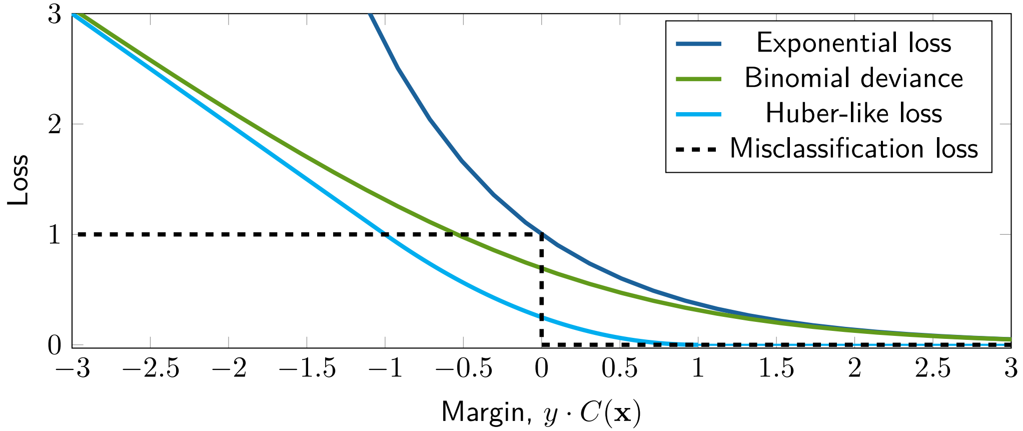
$$\log \left[\frac{\Pr(y = 1 \mid \mathbf{x})}{\Pr(y = -1 \mid \mathbf{x})} \right].$$

However, models using exponential loss is sensitive to ***“outliers”***, e.g. mislabeled or noisy data.

UPPSALA UNIVERSITET

Robust loss functions

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



fredrik.lindsten@it.uu.se

Gradient boosting

No free lunch! The optimization problem

$$(\alpha^b, \widehat{y}^b) = \arg\min_{(\alpha, \widehat{y})} \sum_{i=1}^{n} L(y_i, C^b(\mathbf{x}_i))$$

lacks a closed form solution for a general loss function $L(y, C(\mathbf{x}))$.

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 θ^0 and iterate:

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

where α^b is the **step size** at the bth iteration.

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

$$C^b(\mathbf{x}) = C^{b-1}(\mathbf{x}) + \alpha^b \widehat{y}^b(\mathbf{x})$$

for some base model $\widehat{y}^b(\mathbf{x})$ and “step size” α^b.

Gradient boosting

Gradient boosting mimics the gradient descent algorithm by fitting the bth base model to the ***negative gradient*** of the loss function,

$$g_i^b = -\left[\frac{\partial L(y_i, c)}{\partial c}\right]_{c=C^{b-1}(\mathbf{x}_i)}, \quad i = 1, \dots, n.$$

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

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

`https://xgboost.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)

UPPSALA UNIVERSITET

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.