Day 8: Ensemble methods, boosting
Topics so far

• Linear regression

• Classification
  o Logistic regression
  o Maximum margin classifiers, kernel trick
  o Generative models
  o Neural networks, backpropagation, NN training – optimization and regularization, special architectures – CNNs, RNNs, encoder-decoder

• Remaining Topics
  o Ensemble methods, boosting
  o Unsupervised learning – clustering, dimensionality reduction
  o Review and topics not covered!
Ensemble learning

- **Ensemble learning**
  - Create a population of base learning \( f_1, f_2, \ldots, f_M: \mathcal{X} \rightarrow \mathcal{Y} \)
  - Combine the predictors to form a composite predictor

- **Example in classification with** \( \mathcal{Y} = \{-1,1\} \rightarrow \)** assign “votes” \( \alpha_m \) to each classifier \( f_m \) and take weighted-majority vote

\[
F(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m f_m(x) \right)
\]

- Individual classifiers can be very simple, e.g., \( x_1 \geq 10, x_5 \leq 5 \)

- **Why?**
  - more powerful models \( \rightarrow \) reduce bias
    - e.g., majority vote of linear classifiers can give decision boundaries that are intersections of halfspaces
  - reduce variance
    - averaging classifiers \( f_1, f_2, \ldots, f_M \) trained independently on different iid datasets \( S_1, S_2, \ldots, S_M \) can reduce variance of composite classifier
Reducing bias using ensembles
Decision trees

• Each non-leaf node tests a binary condition on some feature $x_k$
  o if condition satisfies then go left, else go right
  o leaf nodes have label (typically the label of majority class of training examples at that node)

• Classifying a point by decision tree can be seen as a sequence of classifiers refined as we follow the path to a leaf.
Combining “simple” models

- Smooth-ish tradeoff between bias-complexity
  - start with simple models with large bias and low variance
  - learn more complex classes by composing simple models

- For example consider classifiers $f_1, f_2, \ldots, f_M$ based on only one feature (decision stumps), i.e., each
  \[
  f_m(x; \theta_m) = 1(x_{k_m} \geq \tau_m) \text{ where } \theta_m = (k_m, \tau_m)
  \]

- $\mathcal{H} = \{x \mapsto \text{majority}(\alpha_1 f_1(x; \theta_1), \alpha_2 f_2(x; \theta_2), \ldots, \alpha_M f_M(x; \theta_M))\}$ contains very complex boundaries

- demo (by Nati Srebro)

- So clearly combining simple classifiers can reduce bias. How do we combine classifiers?

Figure credit: Nati Srebro
Combining “simple” models

• Given a family of models $f_1, f_2, \ldots : \mathcal{X} \rightarrow \mathcal{Y}$, we want to combine?

• Weighted averaging of models:
  o parameterize combined classifier using $\alpha_m$ as
    $$F_\alpha(x) = \sum_{m=1}^{M} \alpha_m f_m(x)$$
  o minimize loss over combined model
    $$\min_{\alpha} \sum_{i=1}^{N} \ell (F_\alpha(x^{(i)}), y^{(i)})$$

• Alternative algorithm: greedy approach
  o $F_0(x) = 0$
  o for each round $t = 1, 2, \ldots, T$
    ▪ find the best model to minimize the incremental change from $F_{t-1}$
      $$\min_{\alpha_t, f^{(t)}} \sum_{i=1}^{N} \ell (F_{t-1}(x^{(i)}) + \alpha_t f^{(t)}(x^{(i)}), y^{(i)})$$
  o Output classifier $F_T(x) = \sum_{t=1}^{T} \alpha_t f^{(t)}(x)$
Adaboost

Training data $S = \{(x^{(i)}, y^{(i)}): i = 1, 2, ..., N\}$

• Maintain weights $W_i^{(t)}$ for each example $(x^{(i)}, y^{(i)})$, initially all $W_i^{(1)} = \frac{1}{N}$
Adaboost

Training data $S = \{(x^{(i)}, y^{(i)}): i = 1, 2, \ldots, N\}$

- Maintain weights $W_i^{(t)}$ for each example $(x^{(i)}, y^{(i)})$, initially all $W_i^{(1)} = \frac{1}{N}$

- For $t = 1, 2, \ldots, T$
  - Normalize weights $D_i^{(t)} = \frac{W_i^{(t)}}{\sum_i W_i^{(t)}}$

Example credit: Greg Shaknarovich
Adaboost

Training data $S = \{(x^{(i)}, y^{(i)}): i = 1, 2, \ldots, N\}$

- Maintain weights $W_i^{(t)}$ for each example $(x^{(i)}, y^{(i)})$, initially all $W_i^{(1)} = \frac{1}{N}$

- For $t = 1, 2, \ldots, T$
  - Normalize weights $D_i^{(t)} = \frac{W_i^{(t)}}{\sum_i W_i^{(t)}}$
  - Pick a classifier $f_t$ has better than 0.5 weighted loss
    $\epsilon_t = \sum_{i=1}^{N} D_i^{(t)} \ell^{01}(f_t(x^{(i)}), y^{(i)})$

Example credit: Greg Shaknarovich
Adaboost

Training data \( S = \{(x^{(i)}, y^{(i)}): i = 1, 2, ..., N\} \)

- Maintain weights \( W_i^{(t)} \) for each example \((x^{(i)}, y^{(i)})\), initially all \( W_i^{(1)} = \frac{1}{N} \)
- For \( t = 1, 2, ..., T \)
  - Normalize weights \( D_i^{(t)} = \frac{w_i^{(t)}}{\sum_i w_i^{(t)}} \)
  - Pick a classifier \( f_t \) has better than 0.5 weighted loss
    \[ \epsilon_t = \sum_{i=1}^{N} D_i^{(t)} \ell_{01}(f_t(x^{(i)}), y^{(i)}) \]
  - Set \( \alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t} \)

Example credit: Greg Shaknarovich
Adaboost

Training data $S = \{(x^{(i)}, y^{(i)}): i = 1, 2, ..., N\}$

- Maintain weights $W_i^{(t)}$ for each example $(x^{(i)}, y^{(i)})$, initially all $W_i^{(1)} = \frac{1}{N}$
- For $t = 1, 2, ..., T$
  - Normalize weights $D_i^{(t)} = \frac{W_i^{(t)}}{\sum_i W_i^{(t)}}$
  - Pick a classifier $f_t$ has better than 0.5 weighted loss
    $\epsilon_t = \sum_{i=1}^{N} D_i^{(t)} \ell_{01}(f_t(x^{(i)}), y^{(i)})$
  - Set $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  - Update weights
    $W_i^{(t+1)} = W_i^{(t)} \exp \left(-\alpha_t y^{(i)} f_t(x^{(i)})\right)$

Example credit: Greg Shaknarovich
Adaboost

Training data $S = \{(x^{(i)}, y^{(i)}): i = 1, 2, ..., N\}$

- Maintain weights $W_i^{(t)}$ for each example $(x^{(i)}, y^{(i)})$, initially all $W_i^{(1)} = \frac{1}{N}$
- For $t = 1, 2, ..., T$
  - Normalize weights $D_i^{(t)} = \frac{W_i^{(t)}}{\sum_i W_i^{(t)}}$
  - Pick a classifier $f_t$ has better than 0.5 weighted loss
    $\epsilon_t = \sum_{i=1}^{N} D_i^{(t)} \ell^{01}(f_t(x^{(i)}), y^{(i)})$
  - Set $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  - Update weights $W_i^{(t+1)} = W_i^{(t)} \exp\left(-\alpha_t y^{(i)} f_t(x^{(i)})\right)$

Example credit: Greg Shaknarovich
Adaboost

Training data $S = \{(x^{(i)}, y^{(i)}): i = 1,2, ..., N\}$

- Maintain weights $W_i^{(t)}$ for each example $(x^{(i)}, y^{(i)})$, initially all $W_i^{(1)} = \frac{1}{N}$
- For $t = 1,2, ..., T$
  - Normalize weights $D_i^{(t)} = \frac{W_i^{(t)}}{\sum_i W_i^{(t)}}$
  - Pick a classifier $f_t$ has better than 0.5 weighted loss
    $\epsilon_t = \sum_{i=1}^N D_i^{(t)} \ell^{01}(f_t(x^{(i)}), y^{(i)})$
  - Set $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  - Update weights
    $W_i^{(t+1)} = W_i^{(t)} \exp \left( -\alpha_t y^{(i)} f_t(x^{(i)}) \right)$

Example credit: Greg Shaknarovich
Adaboost

Training data $S = \{(x^{(i)}, y^{(i)}): i = 1, 2, ..., N\}$

- Maintain weights $W_i^{(t)}$ for each example $(x^{(i)}, y^{(i)})$, initially all $W_i^{(1)} = \frac{1}{N}$
- For $t = 1, 2, ..., T$
  - Normalize weights $D_i^{(t)} = \frac{W_i^{(t)}}{\sum_i W_i^{(t)}}$
  - Pick a classifier $f_t$ has better than 0.5 weighted loss
    $\epsilon_t = \sum_{i=1}^{N} D_i^{(t)} \ell_{01}(f_t(x^{(i)}), y^{(i)})$
  - Set $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  - Update weights
    $W_i^{(t+1)} = W_i^{(t)} \exp \left( -\alpha_t y^{(i)} f_t(x^{(i)}) \right)$
Adaboost

Training data \( S = \{(x^{(i)}, y^{(i)}): i = 1,2, ..., N\} \)

- Maintain weights \( W_i^{(t)} \) for each example \( (x^{(i)}, y^{(i)}) \), initially all \( W_i^{(1)} = \frac{1}{N} \)
- For \( t = 1,2, ..., T \)
  - Normalize weights \( D_i^{(t)} = \frac{W_i^{(t)}}{\sum_i W_i^{(t)}} \)
  - Pick a classifier \( f_t \) has better than 0.5 weighted loss
    \[
    \epsilon_t = \sum_{i=1}^{N} D_i^{(t)} \ell^{01}(f_t(x^{(i)}), y^{(i)})
    \]
  - Set \( \alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t} \)
  - Update weights
    \[
    W_i^{(t+1)} = W_i^{(t)} \exp \left( -\alpha_t y^{(i)} f_t(x^{(i)}) \right)
    \]

Example credit: Greg Shaknarovich
Adaboost

Training data \( S = \{(x^{(i)}, y^{(i)}): i = 1, 2, \ldots, N\} \)

- Maintain weights \( W^{(t)}_i \) for each example \((x^{(i)}, y^{(i)})\), initially all \( W^{(1)}_i = \frac{1}{N} \)
- For \( t = 1, 2, \ldots, T \)
  - Normalize weights \( D^{(t)}_i = \frac{W^{(t)}_i}{\sum_i W^{(t)}_i} \)
  - Pick a classifier \( f_t \) has better than 0.5 weighted loss
    \[ \epsilon_t = \sum_{i=1}^{N} D^{(t)}_i \ell^01(f_t(x^{(i)}), y^{(i)}) \]
  - Set \( \alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t} \)
  - Update weights
    \[ W^{(t+1)}_i = W^{(t)}_i \exp \left( -\alpha_t y^{(i)} f_t(x^{(i)}) \right) \]
- Output strong classifier \( F_T(x) = \text{sign}(\sum_t \alpha_t f_t(x)) \)

Example credit: Greg Shaknarovich
Adaboost

- Demo again (code by Nati Srebro)
- What are we doing in Adaboost?
  - Some algorithm to do ensembles
  - Learning *sparse* linear predictors with large (infinite?) dimensional features
    - Sparsity controls complexity
    - Number of iterations controls sparsity
      $\Rightarrow$ early stopping as regularization
  - Coordinate descent on exponential loss (briefly next)
- Variants of AdaBoost
  - FloatBoost: After each round, see if removal of a previously added classifier is helpful.
  - Totally corrective AdaBoost: update the $\alpha$’s for all weak classifiers selected so far by minimizing loss
Exponential loss

- Exponential loss $\ell(f(x), y) = \exp(-f(x)y)$ another surrogate loss

- Ensemble classifier $F_\alpha(x) = \text{sign}(\sum_t \alpha_t f_t(x))$

- We will not derive, but can show that adaboost updates correspond to coordinate descent on ERM with exp loss

$$\min_\alpha \sum_{i=1}^N \exp \left( - \sum_t \alpha_t f_t(x^{(i)}) y^{(i)} \right)$$
Example: Viola-Jones Face Detector

• Classify each square in an image as “face” or “no-face”

$X = \text{patches of 24x24 pixels, say}$
Viola-Jones “Weak Predictors”/Features

\[ \mathcal{B} = \left\{ 1(g_{r,t}(x) < \theta) \mid \theta \in \mathbb{R}, \text{rect } r \text{ in image}, t \in \{A, B, C, D, \bar{A}, \bar{B}, \bar{C}, \bar{D}\} \right\} \]

where \( g_{r,t}(x) = \text{sum of “blue” pixels – sum of “red” pixels} \)
Viola-Jones Face Detector

- Simple implementation of boosting using generic (non-face specific) “weak learners”/features
  - Can be used also for detecting other objects
- Efficient method using dynamic programing and caching to find good weak predictor
- About 1 million possible $g_{r,t}$, but only very few used in returned predictor
- Sparsity:
  - Generalization
  - Prediction speed! (and small memory footprint)
- To run in real-time (on 2001 laptop), use sequential evaluation
  - First evaluate first few $h_t$ to get rough prediction
  - Only evaluate additional $h_t$ on patches where the leading ones are promising
Ensembling to reduce variance
Averaging predictors

• Averaging reduces variance: if $Z_1, Z_2, \ldots, Z_M$ are independent random variables each with mean $\mu$ and variance of $\sigma^2$

\[
\text{var} \left( \frac{1}{M} \sum_{m=1}^{M} Z_m \right) = \frac{\sigma^2}{M}
\]

• What happens to mean?

\[
\mathbb{E} \left( \frac{1}{M} \sum_{m=1}^{M} Z_m \right) = \mu
\]
Averaging predictors

• Averaging reduces variance: if $Z_1, Z_2, ..., Z_M$ are independent random variables each with mean $\mu$ and variance of $\sigma^2$

$$\text{var} \left( \frac{1}{M} \sum_{m=1}^{M} Z_m \right) = \frac{\sigma^2}{M}$$

• What happens to mean?

$$\mathbb{E} \left( \frac{1}{M} \sum_{m=1}^{M} Z_m \right) = \mu$$

• If we had $M$ models $f_1, f_2, ... f_M$ trained independently on different iid datasets $S_1, S_2, ..., S_M$, then averaging the results of the models will
  o reduce variance: it will be less sensitive to specific training data
  o without increasing the bias: on average all classifiers will do as well

• But we have only one dataset! How do we get multiple models
  o Remember the models have to be independent!
Bagging: Bootstrap aggregation

Averaging independent models reduces variance without increasing bias.

- But we don’t have independent datasets!
  - Instead take repeated bootstrap samples from training set $S$
- Bootstrap sampling: Given dataset $S = \{(x^{(i)}, y^{(i)}): i = 1, 2, \ldots, N\}$, create $S'$ by drawing $N$ examples at random with replacement from $S$
- Bagging:
  - Create $M$ bootstrap datasets $S_1, S_2, \ldots, S_M$
  - Train distinct models $f_m: \mathcal{X} \rightarrow \mathcal{Y}$ by training only on $S_m$
  - Output final predictor 
    
    $F(x) = \frac{1}{M} \sum_{m=1}^{M} f_m(x)$ (for regression)
    
    or $F(x) = \text{majority}(f_m(x))$ (for classification)

Figure credit: David Sontag
Bagging

• Most effective while combining high variance, low bias predictors
  o unstable non-linear predictors like decision trees
  o “overfitting quirks” of different trees canceling out

• Not very useful with linear predictors

• Useful property of bagging: “out of bag” (OOB) data
  o in each “bag”, treat the examples that didn't make it to the bag as a kind of validation set
  o while learning predictors, keep track of OOB accuracy
Bagging example

Output of single DT

100 bagged trees

Slide/example credit: David Sontag
Random forests

• Ensemble method specifically built for decision trees

• Two sources of randomness
  o **Sample bagging**: Each tree grown with a bootstrapped training data
  o **Feature bagging**: at each node, best split decided over only a subset of random features → increases diversity among trees

• **Algorithm**
  o Create **bootsrapped datasets** \( S_1, S_2, \ldots, S_M \)
  o For each \( m \), grow a decision tree \( T_m \) by repeating the following at each node until some stopping condition
    ▪ select \( K \) features at random from \( d \) features of \( x \)
    ▪ pick best variable/split threshold among the \( K \) selected features
    ▪ split the node into two child nodes based on above condition
  o Output majority vote of \( \{ T_m \}_{m=1}^{M} \)
Ensembles summary

• Reduce bias:
  o build ensemble of low-variance, high-bias predictors sequentially to reduce bias
  o AdaBoost: binary classification, exponential surrogate loss

• Reduce variance:
  o build ensemble of high-variance, low-bias predictors in parallel and use randomness and averaging to reduce variance
  o random forests, bagging

• Problems
  o Computationally expensive (train and test time)
  o Often loose interpretability