18-661 Introduction to Machine Learning

Ensemble Methods

Spring 2020

ECE – Carnegie Mellon University

Announcements: Remote classes

In accordance with university guidelines, all Pittsburgh and SV students should expect to complete this course entirely remotely. We will **not** be holding any course-related activities on either campus.

- All lectures and recitations will be taught remotely for the rest of the semester. See Canvas/Piazza for the Zoom link.
  - Kigali students are expected to attend lecture and recitations in person as usual at this time.
  - Please stay on mute unless you would like to ask a question. You can also use the chat box to type questions; we'll try to pause and monitor them at regular intervals.
  - Lecture and recitation recordings will be made available on Canvas.
  - Please contact the instructors or ECE/SV IT teams if you are having trouble connecting to Zoom.
- Office hours for me, Gauri, and all TAs will be conducted remotely (see Canvas/Piazza for the Zoom link), at the same times as before. Moise will continue to conduct in-person office hours. Any student can attend any office hour.

Announcements: Remote coursework

- Homework 4's deadline has been extended to 3/23 (a week from today) at 11:59pm ET.
- Deadlines for the subsequent homework assignments will be adjusted accordingly later this week.
- The final exam will be conducted entirely online. We are working out the exact format and will update you closer to the exam date.
- We do not expect to make significant changes to the existing course grading policy.

Please keep checking Piazza and Canvas for announcements and updates as we refine the logistics.

Social distancing

Please take measures to keep yourself and everyone else safe!

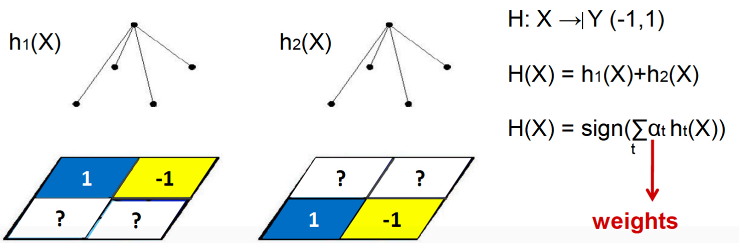
- Our healthcare system does not have the ability to absorb a new large-scale disease. With uncontrolled transmission, more people will die.
- "Lowering the exponent" is our best hope to not overwhelm the system.
- Hence the social distancing. These measures seem radical, but are necessary.
- Early interventions are key. This requires a change in behavior from all of us.

Stay safe, hang out in small sets of people, and, despite everything, try to not worry too much.

Ensemble methods

Ensemble methods

- Instead of learning a single (weak) classifier, learn many weak classifiers, preferably those that are good at different parts of the input spaces
- Predicted Class: (Weighted) Average or Majority of output of the weak classifiers
- Strength in Diversity!



$h_1(X)$
$h_2(X)$
1
-1
?
?
?
?
1
-1
$H: X \rightarrow Y\ (-1,1)$
$H(X) = h_1(X)+h_2(X)$
$H(X) = \text{sign}(\sum_t \alpha_t h_t(X))$
weights

Ensemble methods

In this lecture, we will cover the following ensemble methods:

- Bagging or Bootstrap Aggregation
- Random Forests
- AdaBoost

Method	**Training data per classifier**	**Features per classifier**	**Combination method**
Bagging	Random subset	All	Sum
Random Forests	Random subset	Random subset	Sum
Adaboost	All (weighted)	All	Weighted sum

Bagging or Bootstrap Aggregating

To avoid overfitting a decision tree to a given dataset we can average an ensemble of trees learnt on random subsets of the training data.

Bagging Trees (Training Phase)

- For $b = 1, 2, \cdots, B$
  - Choose n training samples $(\mathbf{x}_i, y_i)$ from $\mathcal{D}$ uniformly at random
  - Learn a decision tree h_b on these n samples
- Store the B decision trees $h_1, h_2, \ldots h_B$
- Optimal B (typically in 1000s) chosen using cross-validation

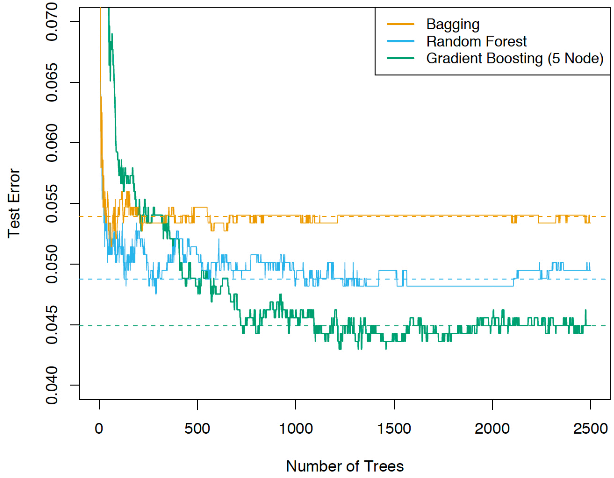
Bagging Trees (Test Phase)

- For a test unlabeled example $\mathbf{x}$
- Find the decision from each of the B trees
- Assign the majority (or most popular) label as the label for $\mathbf{x}$

Random Forests

- Limitation of Bagging: If one or more features are very informative, they will be selected by almost every tree in the bag, reducing the diversity (and potentially increasing the bias).
- Key Idea behind Random Forests: Reduces correlation between trees in the bag without increasing variance too much
    - Same as bagging in terms of sampling training data
    - Before each split, select $m \leq d$ features at random as candidates for splitting $m \sim \sqrt{d}$
    - Take majority vote of B such trees

Comparing Bagging with Random Forests



Limitations of Bagging and Random Forests

- Bagging: Significant correlation between trees that are learnt on different training datasets
- Random Forests try to resolve this by doing "feature bagging" but some correlation still remains
- All B trees are given the same weight when taking the average

Boosting methods: Force classifiers to learn on different parts of the feature space, and take their *weighted* average

Boosting

High-level idea: combine a lot of classifiers

- Sequentially construct / identify these classifiers, $h_t(\cdot)$, one at a time
- Use *weak* classifiers to arrive at a complex decision boundary (*strong* classifier), where β_t is the contribution of each weak classifier

$$h[\boldsymbol{x}] = \text{sign}\left[\sum_{t=1}^{T} \beta_t h_t(\boldsymbol{x})\right]$$

Our plan:

- Describe AdaBoost algorithm
- Derive the algorithm

AdaBoost algorithm

- Given: N samples $\{\boldsymbol{x}_n, y_n\}$, where $y_n \in \{+1, -1\}$, and some way of constructing weak (or base) classifiers
- Initialize weights (distribution) $w_1(n) = \frac{1}{N}$ for every training sample
- For $t = 1$ to T
  1. Train a weak classifier $h_t(\boldsymbol{x})$ using current weights $w_t(n)$, i.e., find the classifier that minimizes

     $$\epsilon_t = \sum_n w_t(n) \mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)] \quad \text{(the weighted classification error)}$$

  2. Compute contribution for this classifier: $\beta_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  3. Update weights on training points

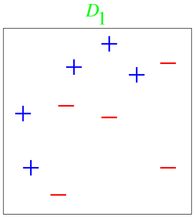
     $$w_{t+1}(n) \propto w_t(n) e^{-\beta_t y_n h_t(\boldsymbol{x}_n)}$$

     and normalize them such that $\sum_n w_{t+1}(n) = 1$.
- Output the final classifier

$$h[\boldsymbol{x}] = \text{sign}\left[\sum_{t=1}^{T} \beta_t h_t(\boldsymbol{x})\right]$$

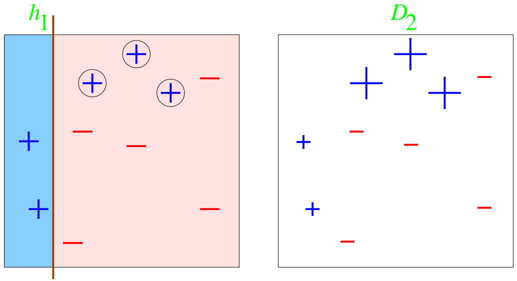
Example

10 data points and 2 features



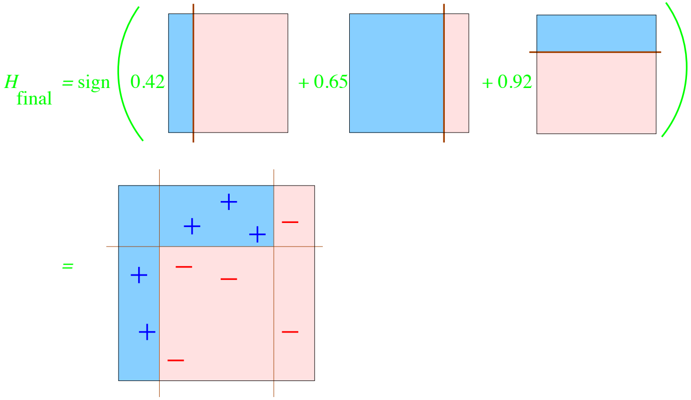
- The data points are clearly not linearly separable
- In the beginning, all data points have equal weights (the size of the data markers "$+$" or "-")
- Base classifier $h(\cdot)$: horizontal or vertical lines ('decision stumps')
  - Depth-1 decision trees, i.e., classify data based on a single attribute.

Round 1: $t = 1$



- 3 misclassified (with circles): $\epsilon_1 = 0.3 \rightarrow \beta_1 = 0.42$.
- Recompute the weights; the 3 misclassified data points receive larger weights

Final classifier: Combining 3 classifiers


$H_{\text{final}} = \text{sign}\left(0.42 \quad + 0.65 \quad + 0.92 \quad\right)$
=
+
−

- All data points are now classified correctly!

Why does AdaBoost work?

Suppose a classifier $f_{t-1}(\boldsymbol{x})$, and want to add a weak learner $h_t(\boldsymbol{x})$

$$f(\boldsymbol{x}) = f_{t-1}(\boldsymbol{x}) + \beta_t h_t(\boldsymbol{x})$$

Note: $h_t(\cdot)$ outputs -1 or 1, as does $\text{sign}\left[f_{t-1}(\cdot)\right]$

How can we 'optimally' choose $h_t(\boldsymbol{x})$ and combination coefficient β_t?

Adaboost greedily *minimizes the exponential loss function*!

$$\begin{aligned}(h_t^*(\boldsymbol{x}), \beta_t^*) &= \text{argmin}_{(h_t(\boldsymbol{x}),\beta_t)} \sum_n e^{-y_n f(\boldsymbol{x}_n)} \\ &= \text{argmin}_{(h_t(\boldsymbol{x}),\beta_t)} (e^{\beta_t} - e^{-\beta_t}) \sum_n w_t(n) \mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)] \\ &\qquad + e^{-\beta_t} \sum_n w_t(n)\end{aligned}$$

where we have used $w_t(n)$ as a shorthand for $e^{-y_n f_{t-1}(\boldsymbol{x}_n)}$

Finding the optimal weak learner

Summary

$$
\begin{aligned}
(h_t^*(\boldsymbol{x}), \beta_t^*) &= \text{argmin}_{(h_t(\boldsymbol{x}),\beta_t)} \sum_n w_t(n) e^{-y_n \beta_t h_t(\boldsymbol{x}_n)} \\
&= \text{argmin}_{(h_t(\boldsymbol{x}),\beta_t)} (e^{\beta_t} - e^{-\beta_t}) \sum_n w_t(n) \mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)] \\
&\qquad + e^{-\beta_t} \sum_n w_t(n)
\end{aligned}
$$

What term(s) must we optimize to choose $h_t(\boldsymbol{x}_n)$?

$$
h_t^*(\boldsymbol{x}) = \text{argmin}_{h_t(\boldsymbol{x})} \, \epsilon_t = \sum_n w_t(n) \mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)]
$$

Minimize weighted classification error as noted in step 1 of Adaboost!

How to choose β_t?

Summary

$$\begin{aligned}
(h_t^*(\boldsymbol{x}), \beta_t^*) &= \text{argmin}_{(h_t(\boldsymbol{x}),\beta_t)} \sum_n w_t(n) e^{-y_n \beta_t h_t(\boldsymbol{x}_n)} \\
&= \text{argmin}_{(h_t(\boldsymbol{x}),\beta_t)} (e^{\beta_t} - e^{-\beta_t}) \sum_n w_t(n) \mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)] \\
&\qquad + e^{-\beta_t} \sum_n w_t(n)
\end{aligned}$$

What term(s) must we optimize?

We need to minimize the entire objective function with respect to β_t!

We can take the derivative with respect to β_t, set it to zero, and solve for β_t. After some calculation and using $\sum_n w_t(n) = 1$...

$$\beta_t^* = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$$

which is precisely step 2 of Adaboost! (*Exercise – verify the solution*)

Updating the weights

Once we find the optimal weak learner we can update our classifier:

$$f(\boldsymbol{x}) = f_{t-1}(\boldsymbol{x}) + \beta_t^* h_t^*(\boldsymbol{x})$$

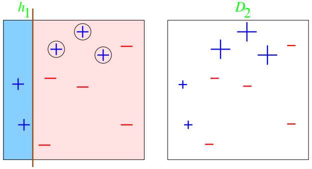
We then need to compute the weights for the above classifier as:

$$\begin{aligned} w_{t+1}(n) &\propto e^{-y_n f(\boldsymbol{x}_n)} = e^{-y_n[f_{t-1}(\boldsymbol{x})+\beta_t^* h_t^*(\boldsymbol{x}_n)]} \\ &= w_t(n) e^{-y_n \beta_t^* h_t^*(\boldsymbol{x}_n)} = \begin{cases} w_t(n)e^{\beta_t^*} & \text{if } y_n \neq h_t^*(\boldsymbol{x}_n) \\ w_t(n)e^{-\beta_t^*} & \text{if } y_n = h_t^*(\boldsymbol{x}_n) \end{cases} \end{aligned}$$

Intuition Misclassified data points will get their weights increased, while correctly classified data points will get their weight decreased

Decision stumps in AdaBoost

How do we choose a decision stump classifier given the weights at the second round of the following distribution?



We can simply enumerate all possible ways of putting vertical and horizontal lines to separate the data points into two classes and find the one with the smallest weighted classification error! Runtime?

- Presort data by each feature in $O(dN \log N)$ time
- Evaluate $N + 1$ thresholds for each feature at each round in $O(dN)$ time
- In total $O(dN \log N + dNT)$ time – this efficiency is an attractive quality of boosting!

Ensemble Methods: Summary

Fight the bias-variance tradeoff by combining (by a weighted sum or majority vote) the outputs of many weak classifiers.

Bagging trains several classifiers on random subsets of the training data. Random forests train several decision trees that are constrained to split on random subsets of data *features*.

- Prevents some correlation between trees due to dominant features.
- All classifiers are weighted equally in making the final decision.

Boosting sequentially adds weak classifiers, increasing the weight of "hard" data points at each step.

- Greedily minimizes a surrogate classification loss
- Commonly uses decision trees as base classifiers