CS 446/ECE 449: Machine Learning

Shenlong Wang

University of Illinois at Urbana-Champaign, 2024

L08: Boosting, Ensembles and Regularization/Cross-validation

Goals of this lecture

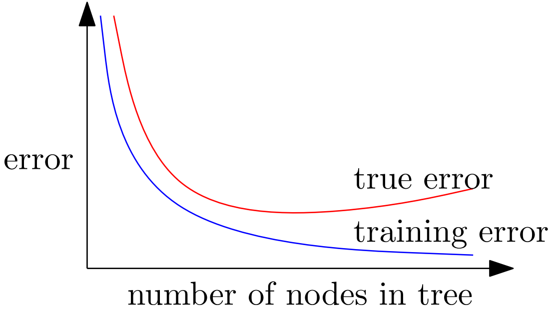
- Getting to know Boosting
- Getting to know Cross-Validation

Reading material:

- Shai Shalev-Shwartz & Shai Ben-David, Understanding Machine Learning: From Theory to Algorithms, Chapter 4

Recap:
Decision trees.

Aside: Overfitting.



- Training error **goes to zero** as number of tree nodes increases.
- True error (measured by test error) decreases initially, but eventually **increases** (i.e. overfitting).

Classifiers we've seen so far:

- Nearest Neighbor
- Logistic regression
- Linear SVM
- Kernel SVM
- Decision tree

Suppose we train one of each.

Do we choose best and throw rest away? Can we somehow combine?

What can we do with model ensembles?

Standard machine learning practice:

- We have some data, we train 10 different predictors.
- Rather than taking the *best*, can we combine them and do better?

Combining classifiers.

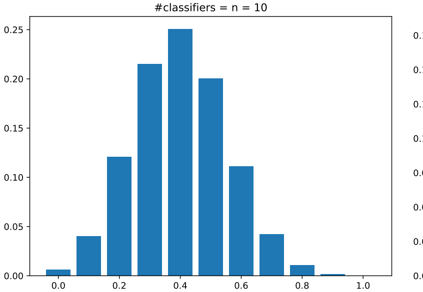
- Suppose we have n classifiers.
- Suppose each is wrong *independently* with probability 0.4.
- Model error of classifiers as random variables
- We can model the distribution of errors with $\text{Binom}(n, 0.4)$.

  Red: *all* wrong.

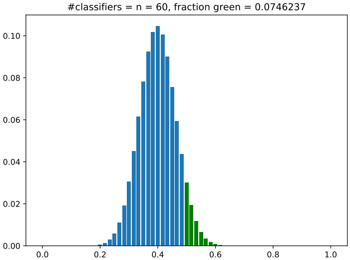
  Green: *at least half* wrong.

  Green region is error of majority vote

  Error rate (n = 60): $0.075 \ll 0.4$!



What does this mean for majority vote classifiers?



Green region is error of majority vote!

Error of majority vote classifier goes down **exponentially** in n.

Error rate of majority classifier (with individual error probability p):

$$\Pr[\text{Binom}(n, p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp\left(-n(1/2 - p)^2\right).$$

Let's use it in practice!

- Version 1: assuming independent errors.
- Version 2: allowing non-independent errors with adaptive classifiers.

Practical majority vote: bagging and random forests.

Combining decision trees.

Let's majority vote a few decision trees.

Problem: Decision tree method we suggested is deterministic.

Bagging

Bagging = **B̲ootstrap a̲g̲g̲regatin̲g̲** (Leo Breiman, 1994).

Input: training data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$ from $\mathcal{X} \times \{-1, +1\}$.

For $t = 1, 2, \ldots, T$:

1. Randomly pick n examples *with replacement* from training data $\longrightarrow$ $\{(\mathbf{x}_t^{(i)}, y_t^{(i)})\}_{i=1}^{n}$ (a *bootstrap* sample).
2. Run learning algorithm on $\{(\mathbf{x}_t^{(i)}, y_t^{(i)})\}_{i=1}^{n}$ $\longrightarrow$ classifier f_t.

Return a majority vote classifier over $f_1, f_2, ..., f_T$.

Random Forests.

Random Forests (Leo Breiman, 2001).

Input: training data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n$ from $\mathbb{R}^d \times \{-1, +1\}$.

For $t = 1, 2, \ldots, T$:

1. Randomly pick n examples *with replacement* from training data
   $\longrightarrow$ $\{(\mathbf{x}_t^{(i)}, y_t^{(i)})\}_{i=1}^n$ (a *bootstrap* sample).
2. Run variant of decision tree learning algorithm on $\{(\mathbf{x}_t^{(i)}, y_t^{(i)})\}_{i=1}^n$, where each split is chosen by only considering a random subset of $\sqrt{d}$ features (rather than all d features)
   $\longrightarrow$ decision tree classifier f_t.

Return a majority vote classifier over $f_1, f_2, \ldots, f_T$.

Non-independent errors with majority vote: boosting.

Non-independent errors.

So far, we combined classifiers assuming **independent errors**.
(This never happens, can be expensive to approximate independent errors.)

Reminder:
old setting is we have n classifiers handed to us and then majority vote over them.

How can we handle dependent errors?

We'll use an assumption on how we get classifiers.

- We can adaptively choose classifiers.
- and **reweight the dataset**.

Boosting.

Suppose you have a weak classifier i.e. error rate lower than 50%

We call this a **weak learner** with **weak learning rate** $\epsilon > 0$.

- We have a black box "**weak learning oracle (WLO)**"
  Given a *reweighted* data set, it gives us back a classifier with error $\leq 1/2 - \epsilon$.

Algorithm scheme.

1. Start with uniform distribution over dataset.
2. Ask *weak learning oracle* for a new classifier.
3. Reweight dataset: examples where current ensemble is bad will have more weight.
4. Go back to step 2.

AdaBoost (Adaptive Boosting).

input Training data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$ from $\mathcal{X} \times \{-1, +1\}$.

1: **initialize** $\gamma_1^{(i)} := 1/n$ for each $i = 1, 2, \ldots, n$ (a probability distribution).
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Get weak classifier f_t from γ_t-weighted samples.
4: Update weights:

$$z_t := \sum_{i=1}^{n} \gamma_t^{(i)} \cdot y^{(i)} f_t(\mathbf{x}^{(i)}) \in [-1, +1] \quad \text{(weighted error rate)}$$

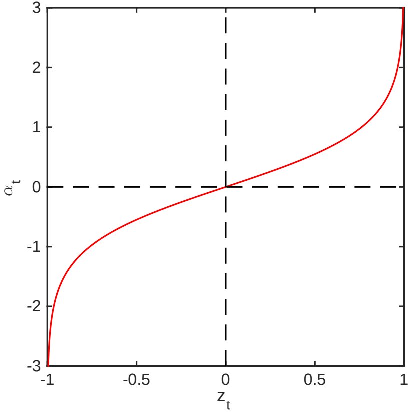
$$\alpha_t := \frac{1}{2} \ln \frac{1 + z_t}{1 - z_t} \in \mathbb{R} \quad \text{(weight of } f_t\text{)}$$

$$\gamma_{t+1}^{(i)} := \gamma_t^{(i)} \exp\left(-\alpha_t \cdot y^{(i)} f_t(\mathbf{x}^{(i)})\right) / Z_t \quad \text{for each } i \quad \text{(sample weight)},$$

where $Z_t > 0$ is normalizer that makes D_{t+1} a probability distribution.
5: **end for**
6: **return** Final classifier $\text{sign}\left(\sum_{t=1}^{T} \alpha_t \cdot f_t(x)\right)$.

Interpretation.

Classifier weights $\alpha_t = \frac{1}{2} \ln \frac{1+z_t}{1-z_t}$



Example weights $\gamma_{t+1}^{(i)}$

$$\gamma_{t+1}^{(i)} \propto \gamma_t^{(i)} \cdot \exp(-\alpha_t \cdot y^{(i)} f_t(\mathbf{x}^{(i)}))\,.$$

Aside: Straightforward to handle importance weights in empirical risk minimization (ERM).

Let's define the error of $F_T = \sum_{t=1}^{T} \alpha_t \cdot f_t(x)$ to be

$$E(F_T) = \sum_i \exp\left(-y^{(i)} F_T(\mathbf{x}^{(i)})\right)$$

Let's also define factors γ_i^t:

$$\gamma_i^1 = 1 \ \forall\ i; \qquad \gamma_i^t = \exp\left(-y^{(i)} F_{t-1}(\mathbf{x}^{(i)})\right)$$

Consequently

$$\begin{aligned}
\min E(F_t) &= \min \sum_i \gamma_i^t \exp\left(-y^{(i)} \alpha_t f_t(\mathbf{x}^{(i)})\right) \\
&= \min \sum_{i:y^{(i)}=f_t(\mathbf{x}^{(i)})} \gamma_i^t e^{-\alpha_t} + \sum_{i:y^{(i)}\neq f_t(\mathbf{x}^{(i)})} \gamma_i^t e^{\alpha_t} \\
&= \min \sum_i \gamma_i^t e^{-\alpha_t} + \sum_{i:y^{(i)}\neq f_t(\mathbf{x}^{(i)})} \gamma_i^t \left(e^{\alpha_t} - e^{-\alpha_t}\right)
\end{aligned}$$

Therefore:

Pick f_t with lowest weighted error $\sum_{i:y^{(i)}\neq f_t(\mathbf{x}^{(i)})} \gamma_i^t$

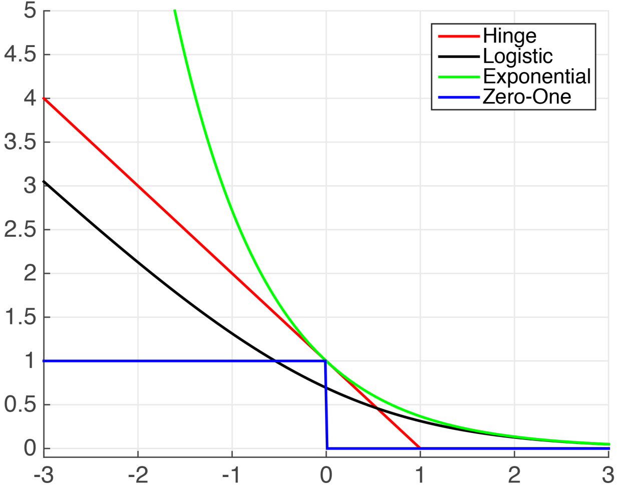
How to pick α_t?

$$\frac{dE(F_t)}{d\alpha_t} = 0$$

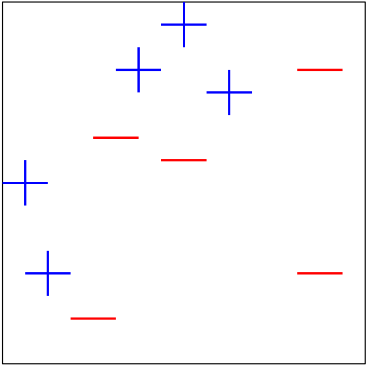
Result:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \qquad \text{where} \quad \epsilon_t = \left(\sum_{i:y^{(i)} \neq f_t(\mathbf{x}^{(i)})} \gamma_i^t \right) / \left(\sum_i \gamma_i^t \right)$$

Adaboost –> minimizing exponential loss.



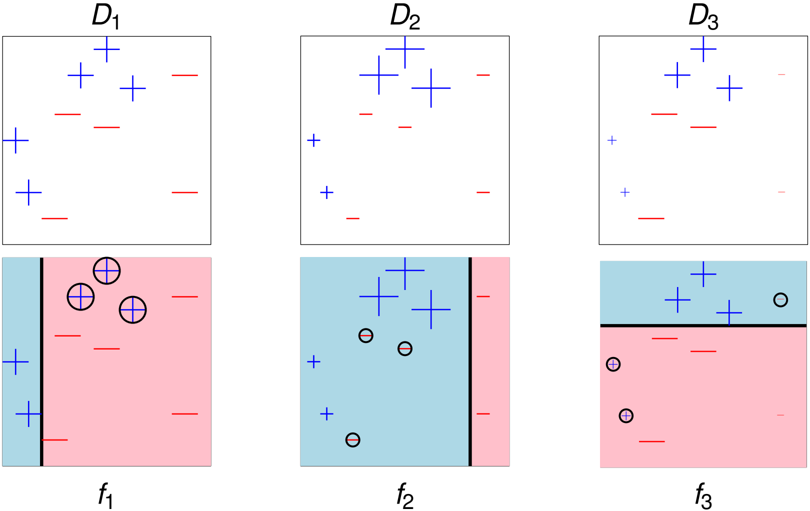
Example: AdaBoost with decision stumps.



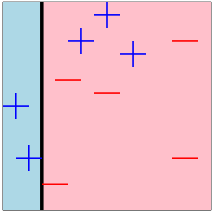
Weak learning algorithm:
ERM with "decision stumps" i.e., axis-aligned threshold functions

$$\mathbf{x} \mapsto \mathrm{sign}(x^{(i)}_{\mathrm{index}_t} - \tau_t).$$

(Example from Figures 1.1 and 1.2 of Schapire & Freund textbook)

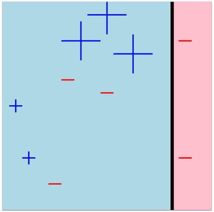


$z_1 = 0.40$, $\alpha_1 = 0.42$ $z_2 = 0.58$, $\alpha_2 = 0.65$ $z_3 = 0.72$, $\alpha_3 = 0.92$



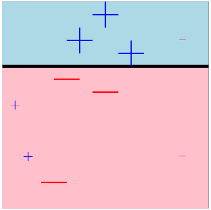
f_1

$z_1 = 0.40, \alpha_1 = 0.42$



f_2

$z_2 = 0.58, \alpha_2 = 0.65$



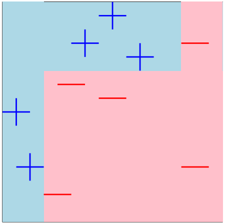
f_3

$z_3 = 0.72, \alpha_3 = 0.92$

Final classifier

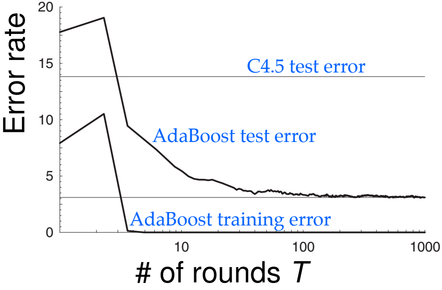
$\hat{f}(x) = \text{sign}(0.42 f_1(x) + 0.65 f_2(x) + 0.92 f_3(x))$

(Zero training error rate!)



A typical run of boosting.

AdaBoost+C4.5 on “letters” dataset.



Training error rate is zero after just five rounds, but test error rate continues to decrease, even up to 1000 rounds!

(# nodes across all decision trees in $\hat{f}$ is $>2 \times 10^6$)

(Figure 1.7 from Schapire & Freund text)

Boosting the margin.

Final classifier from AdaBoost:

$$\hat{f}(x) \;=\; \text{sign}\underbrace{\left(\frac{\sum_{t=1}^{T}\alpha_t f_t(x)}{\sum_{t=1}^{T}|\alpha_t|}\right)}_{g(x)\in[-1,+1]}.$$

Call $y \cdot g(x) \in [-1, +1]$ the **margin** achieved on example (x, y).

Theory [Schapire, Freund, Bartlett, and Lee, 1998]:

- **Larger margins $\Rightarrow$ better resistance to overfitting**, independent of T.
- AdaBoost tends to increase margins on training examples.

(Conceptually similar to SVM margins.)

Code

Quiz:

- What is Boosting?
- How do we construct decision trees?
- What are ensembles?

Important topics of this lecture

- Boosting
- Classification and regression trees
- Ensembles

Up next:

- Structured Models