RECITATION 10: LEARNING THEORY AND ENSEMBLE METHODS

10-301/10-601 Introduction to Machine Learning (Summer 2024)
http://www.cs.cmu.edu/~hchai2/courses/10601

1 Learning Theory

1.1 PAC Learning

Some Important Definitions
Basic notation:

1.
   - Probability distribution (unknown): $X \sim p^*$
   - **True function** (unknown): $c^* : X \to Y$
   - **Hypothesis space** $\mathcal{H}$ and **hypothesis** $h \in \mathcal{H} : X \to Y$
   - Training dataset $\mathcal{D} = \{x^{(1)}, \ldots, x^{(N)}\}$
2. **True Error (expected risk)**

$$R(h) = P_{x \sim p^*(x)}(c^*(x) \neq h(x))$$

3. **Train Error (empirical risk)**

$$\begin{aligned}\hat{R}(h) &= P_{x \sim \mathcal{D}}(c^*(x) \neq h(x)) \\ &= \frac{1}{N}\sum_{i=1}^{N} 1(c^*(x^{(i)}) \neq h(x^{(i)})) \\ &= \frac{1}{N}\sum_{i=1}^{N} 1(y^{(i)} \neq h(x^{(i)}))\end{aligned}$$

The **PAC criterion** is that we produce a high accuracy hypothesis with high probability. More formally,

$$P(\forall h \in \mathcal{H}, \_\_\_\_\_ \leq \_\_\_\_\_) \geq \_\_\_\_\_$$

$$P(\forall h \in \mathcal{H}, |R(h) - \hat{R}(h)| \leq \epsilon) \geq 1 - \delta$$

Sample Complexity is the minimum number of training examples N such that the PAC criterion is satisfied for a given ϵ and δ

Sample Complexity for 4 Cases: See Figure 1. Note that

- **Realizable** means $c^* \in \mathcal{H}$
- **Agnostic** means c^* may or may not be in $\mathcal{H}$

	Realizable	Agnostic
Finite $\lvert\mathcal{H}\rvert$	**Thm. 1** $N \geq \frac{1}{\epsilon}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{1}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ all $h \in \mathcal{H}$ with $\hat{R}(h) = 0$ have $R(h) \leq \epsilon$.	**Thm. 2** $N \geq \frac{1}{2\epsilon^2}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{2}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ for all $h \in \mathcal{H}$ we have that $\lvert R(h) - \hat{R}(h)\rvert \leq \epsilon$.
Infinite $\lvert\mathcal{H}\rvert$	**Thm. 3** $N{=}O(\frac{1}{\epsilon}\left[\text{VC}(\mathcal{H})\log(\frac{1}{\epsilon}) + \log(\frac{1}{\delta})\right])$ labeled examples are sufficient so that with probability $(1-\delta)$ all $h \in \mathcal{H}$ with $\hat{R}(h) = 0$ have $R(h) \leq \epsilon$.	**Thm. 4** $N = O(\frac{1}{\epsilon^2}\left[\text{VC}(\mathcal{H}) + \log(\frac{1}{\delta})\right])$ labeled examples are sufficient so that with probability $(1-\delta)$ for all $h \in \mathcal{H}$ we have that $\lvert R(h) - \hat{R}(h)\rvert \leq \epsilon$.

12

Figure 1: Sample Complexity for 4 Cases

The **VC dimension** of a hypothesis space $\mathcal{H}$, denoted VC($\mathcal{H}$) or $d_{VC}(\mathcal{H})$, is the maximum number of points such that there exists at least one arrangement of these points and a hypothesis $h \in \mathcal{H}$ that is consistent with any labelling of this arrangement of points.

To show that VC($\mathcal{H}$) $= n$:

- Show there exists a set of points of size n that $\mathcal{H}$ can shatter
- Show $\mathcal{H}$ cannot shatter any set of points of size $n+1$

Questions

1. For the following examples, write whether or not there exists a dataset with the given properties that can be shattered by a linear classifier.
   - 2 points in 1D
   - 3 points in 1D
   - 3 points in 2D
   - 4 points in 2D

   How many points can a linear boundary (with bias) classify exactly for d-Dimensions?

   - Yes
   - No
   - Yes
   - No

   $d+1$

2. Consider a rectangle classifier (i.e. the classifier is uniquely defined 3 points $x_1, x_2, x_3 \in \mathbb{R}^2$ that specify 3 out of the four corners), where all points within the rectangle must equal 1 and all points outside must equal -1

   (a) Which of the configurations of 4 points in figure 2 can a rectangle shatter?

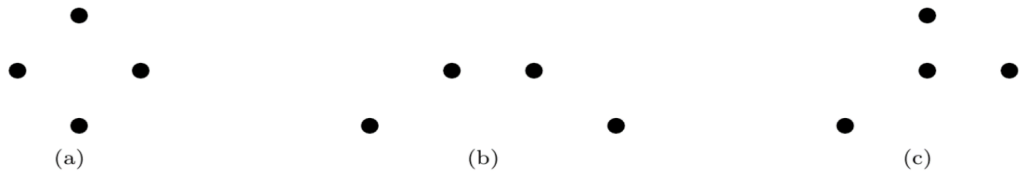

Figure 2

(a), (b), since the rectangle can be scaled and rotated it can always perfectly classify the points. (c) is not perfectly classifiable in the case that all the exterior points are positive and the interior point is negative.

   (b) What about the configurations of 5 points in figure 3?

Figure 3

None of the above. For (d), consider (from left to right) the labeling 1, 1 -1, -1, 1. For (e), same issue as (c).

3. Let $x_1, x_2, ..., x_n$ be n random variables that represent binary literals ($x \in \{0,1\}^n$). Let the hypothesis class $\mathcal{H}_n$ denote the conjunctions of no more than n literals in which each variable occurs at most once. Assume that $c^* \in \mathcal{H}_n$.

   Example: For $n = 4$, $(x_1 \wedge x_2 \wedge x_4), (x_1 \wedge \neg x_3) \in \mathcal{H}_4$

   Find the minimum number of examples required to learn $h \in \mathcal{H}_{10}$ which guarantees at least 99% accuracy with at least 98% confidence.

   $|H_n| = 3^n$

   $|H_{10}| = 3^{10}, \epsilon = 0.01, \delta = 0.02$

   $N(H_{10}, \epsilon, \delta) \geq \lceil \frac{1}{\epsilon}[\ln |H_{10}| + \ln \frac{1}{\delta}] \rceil = \lceil 1489.81 \rceil = 1490$

2 Ensemble Methods

The idea of ensemble methods is to build a model for prediction by combining the strengths of a group of simpler models. We'll cover two examples of ensemble methods: random forests and AdaBoost.

2.1 Random Forests

1. What are some downsides of decision trees, and how can we explain this in the context of the bias-variance tradeoff?

   learned greedily, can overfit if depth isn't controlled, low bias but high variance

   Random Forests = Sample Bagging + Split-Feature Randomization

2. What is **sample bagging**?

   Bagging stands for bootstrap aggregating. A bootstrapped dataset has the same number of rows as the original dataset, but its rows are drawn from the original dataset with replacement. Models are trained on each individual dataset, but since the training datasets are not as similar to each other, the learned models tend to be different and more variable as well. Aggregating refers to the model predictions being combined to form the final prediction.

3. What is **split-feature randomization**?

   In the splits for each decision tree, instead of choosing the split feature from the set of all features, we limit the feature set to a randomly chosen subset of all the features. This further reduces correlation between the learned decision trees as the "best" feature may not always be available to split on.

4. How do these techniques affect the bias and variance of an individual tree? Both increase bias and decrease variance by limiting the information available to train on, compared to a decision tree trained on the full original dataset.

5. How do these techniques affect the bias and variance of an ensemble of trees?

   The increase in bias carries over from the individual trees. The ensemble has lower variance because we are aggregating predictions over a set of trees that are not fully correlated. Both techniques are designed to make the trees less correlated with one another, as reducing covariance between trees reduces variance of the average over trees.

   Consider random variables X_1, X_2, each with variance σ^2. The variance of their average is given

by

$$\begin{aligned}
\text{Var}(\frac{1}{2}X_1 + \frac{1}{2}X_2) &= \text{Var}(\frac{1}{2}X_1) + \text{Var}(\frac{1}{2}X_2) + 2\text{Cov}(\frac{1}{2}X_1, \frac{1}{2}X_2) \\
&= \frac{1}{4}\text{Var}(X_1) + \frac{1}{4}\text{Var}(X_2) + \frac{1}{2}\text{Cov}(X_1, X_2) \\
&= \frac{1}{4}\sigma^2 + \frac{1}{4}\sigma^2 + \frac{1}{2}\sigma^2\frac{\text{Cov}(X_1, X_2)}{\sigma \cdot \sigma} \\
&= \frac{1}{4}\sigma^2 + \frac{1}{4}\sigma^2 + \frac{1}{2}\rho\sigma^2 \\
&= \sigma^2\frac{1}{2}(1+\rho)
\end{aligned}$$

where ρ is the correlation between X_1, X_2.

If $\rho = 1$ (fully correlated trees), then we achieve no variance reduction: the average has variance σ^2. In general, reducing ρ makes the trees less correlated and reduces variance.

Note that our techniques are generally not extreme enough to generate anticorrelated trees, where tree predictions would oppose each other. The more anticorrelated our predictions are, the closer our model gets to always predicting 0, which does reduce variance, but at the cost of performance on our task.

6. For each data point $\mathbf{x}^{(i)}$, define $t^{(-i)}$ to be the set of decision trees that $\mathbf{x}^{(i)}$ was not used to train. Use each tree in $t^{(-i)}$ to make a prediction for $\mathbf{x}^{(i)}$, and use these predictions to make an aggregated prediction $\overline{t^{(-i)}}(\mathbf{x}^{(i)})$ (i.e. for classification take the majority vote). Then, we can define the *out-of-bag* error as follows:

$$E_{OOB} = \frac{1}{N}\sum_{i=1}^{N} 1\left(\overline{t^{(-i)}}(\mathbf{x}^{(i)}) \neq y^{(i)}\right)$$

Why can we use E_{OOB} for hyperparameter optimization even though it was calculated using training points we used to learn the decision trees with?

While every point was used to train a certain set of decision trees, the calculation of E_{OOB} takes advantage of the fact that the nature of bootstrapped datasets means that there will generally be a reasonably large proportion of them that do not contain any particular point.

Therefore, for the decision trees that were trained on these datasets, the training point is equivalent to a test point as the tree has never seen it before. Since each tree is trained independently, there will never be a scenario in which a tree is both trained on a data point and also evaluated on it.

7. **Random Forest Example:** Suppose we train a random forest with two decision trees on the following dataset, using the provided bootstrap samples. Assume that for ties, we predict $Y = 1$.

All	X_0	X_1	X_2	X_3	Y
1	1	0	0	0	1
2	0	0	1	0	1
3	0	0	0	1	1
4	0	0	0	0	0
5	0	1	0	1	1

Sample 1	X_0	X_1	X_2	X_3	Y
1	1	0	0	0	1
4	0	0	0	0	0
5	0	1	0	1	1

Sample 2	X_0	X_1	X_2	X_3	Y
3	0	0	0	1	1
4	0	0	0	0	0
5	0	1	0	1	1

(a) Suppose we train our first tree on Sample 1 and the split feature randomization chooses $\{X_1, X_2\}$ for the feature candidates at the root. What feature will we split on at the root? X_1

(b) Suppose we then recurse on the left child (with feature value 0) of the root and split feature randomization chooses $\{X_0, X_2\}$ for the feature indices. What feature will we split on? X_0

(c) Suppose we train our second tree on Sample 2 and the split feature randomization chooses $\{X_2, X_3\}$ for the feature candidates at the root. What feature will we split on at the root? X_3

(d) What is the training error of the ensemble? $1/5$, as only point 2 is incorrect.

Point 1: tree 1 predicts 1, tree 2 predicts 0, so prediction is 1

Point 2: tree 1 predicts 0, tree 2 predicts 0, so prediction is 0

Point 3: tree 1 predicts 0, tree 2 predicts 1, so prediction is 1

Point 4: tree 1 predicts 0, tree 2 predicts 0, so prediction is 0

Point 5: tree 1 predicts 1, tree 2 predicts 1, so prediction is 1

(e) What is the out of bag error of the ensemble? $4/5$, as only point 5 is correct

Point 1: only tree 2 is involved, prediction is 0

Point 2: both trees are involved, prediction is 0

Point 3: only tree 1 is involved, prediction is 0

Points 4 and 5: majority vote over 0 trees predicts 1

2.2 AdaBoost

2.2.1 AdaBoost Definitions

- T: The number of iterations used to train AdaBoost.
- N: The number of training samples.
- $S = \{(x^{(1)}, y^{(1)}), \cdots, (x^{(N)}, y^{(N)})\}$: The training samples with binary labels ($y^{(i)} \in \{-1, +1\}$).
- $\omega_t^{(i)}$: The weight assigned to training example i at time t. Note that $\sum_i \omega_t^{(i)} = 1$.
- h_t: The weak learner constructed at time t (a function $X \to \{-1, +1\}$).
- ϵ_t: The weighted (by ω_t) error of h_t.
- $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$: The normalization factor for the distribution update at time t.
- $\alpha_t = \frac{1}{2}\ln((1-\epsilon_t)/\epsilon_t)$: The weight assigned to the learner h_t in the composite hypothesis.
- $H_t(x) = \left(\sum_{t'=1}^{t} \alpha_{t'} h_{t'}(x)\right) / \left(\sum_{t'=1}^{t} \alpha_{t'}\right)$: The majority vote of the weak learners, rescaled based on the total weights.
- $g_t(x) = \text{sign}(H_t(x))$: The voting classifier decision function.

2.2.2 AdaBoost Weighting

AdaBoost relies on building an ensemble of weak learners, assigning them weights based on their errors during training.

1. Assume we are in the binary classification setting. What happens to the weight $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ of classifier h_t if its error $\epsilon_t > 0.5$? Why is this useful? It becomes negative (check the log term). This "inverts" the output of this weak learner for every input, turning it into a classifier with $\epsilon_t < 0.5$.

   Note that if we can find weak learners h_t with $\epsilon_t < 0.5$ for all t, training error will decrease exponentially fast in the total number of iterations T.

2. AdaBoost also assigns weights $\omega_t^{(i)}$ for each data point. Explain in broad terms how the weights assigned to examples get updated in each iteration.

   Generally, points that get incorrectly classified get up-weighted and points that get correctly clas-

sified get down-weighted. The amount by which they're weighted depends on the importance of the weak learner - better (lower error) learners lead to stronger up-weights and down-weights. The weights are also normalized to have sum 1.

Update rule: $\omega_t^{(i)} \propto \omega_{t-1}^{(i)} \exp\left(-\alpha_t y^{(i)} h_t(x^{(i)})\right)$

2.2.3 The Margin

In the following question, we will examine the generalization error of AdaBoost using a concept known as the *classification margin*.

For a binary classification task, assume that we use a probabilistic classifier that provides a probability distribution over the possible labels (i.e. $p(y|x)$ for $y \in \{+1, -1\}$). The classifier output is the label with highest probability. We define the *classification margin* for an input as the signed difference between the probability assigned to the correct label and the incorrect label $p_{correct} - p_{incorrect}$, which takes on values in the range $[-1, 1]$.

1. Let $\text{margin}_t(x, y)$ represent the margin for our AdaBoost classifier at iteration t on the sample (x, y). Write a single inequality in terms of $\text{margin}_t(x, y)$ that is true if and only if the classifier makes a mistake on the input (x, y) (i.e., provide a bound on the margin in the case the classifier is incorrect). Assume the classifier makes a mistake on ties. $\text{margin}_t(x, y) \leq 0$.

2. For a given input and label $(x^{(i)}, y^{(i)})$, write $\text{margin}_t(x^{(i)}, y^{(i)})$ in terms of $x^{(i)}, y^{(i)}$, and f_t.

   $y^{(i)} H_t(x^{(i)})$

   INTUITION: H_t tells us the difference between the probabilities on the positive and negative labels, which is the margin in the case that the real label is positive. In the case that the real label is negative, H_t is the opposite of the margin. In both cases, this gives us the margin as $H_t(x^{(i)})$ multiplied by the true label $y^{(i)}$.

   To find $\text{margin}_t(x^{(i)}, y^{(i)})$, we must find expressions for $p_{correct}$ and $p_{incorrect}$. We will do this by first finding expressions for p_+ and p_-, the probabilities assigned to the positive and negative classes during our decision.

   First, note that each weak learner has an associated weight α_t. These weights are not normalized: the weight sum $\alpha = \sum_{t'=1}^{t} \alpha_{t'}$ is not necessarily 1.

   When we classify a point $x^{(i)}$, we use the function $H_t(x^{(i)})$. This function computes the weighted summation over the outputs of all individual weak learners and normalizes this by the total sum of

weights. Because each weak learner outputs ± 1, we have $H_t(x^{(i)}) \in [-1, 1]$.

Now, let's rewrite $H_t(x^{(i)}) = \sum_{t'=1}^{t} \frac{\alpha'_t}{\alpha} h_{t'}(x^{(i)})$ (pushing in that denominator summation α we defined earlier). We can think of $\frac{\alpha'_t}{\alpha}$ as the probability assigned to the weak learner $h_{t'}$ from time t'.

Let's look at a single element of this summation $\frac{\alpha'_t}{\alpha} h_{t'}(x^{(i)})$. If $h_{t'}$ predicts positive on $x^{(i)}$, then $h_{t'}(x^{(i)}) = 1$ and this element is $\frac{\alpha'_t}{\alpha}$. Similarly, if $h_{t'}$ predicts negative, this is $-\frac{\alpha'_t}{\alpha}$.

We can then rewrite our summation f_t as follows:

$$H_t(x^{(i)}) = \sum_{t':h_{t'}(x^{(i)})=1} \frac{\alpha_{t'}}{\alpha} - \sum_{t':h_{t'}(x^{(i)})=-1} \frac{\alpha_{t'}}{\alpha}$$

where the first summation contains all the weights of weak learners that predicted positive and the second summation contains all the weights of weak learners that predicted negative. Note that $\sum_{t'} \frac{\alpha_{t'}}{\alpha} = \frac{1}{\alpha} \sum_{t'} \alpha_{t'} = \frac{\alpha}{\alpha} = 1$, s the total weight here does sum to 1. Thus, we can now define $p_+ = \sum_{t':h_{t'}(x^{(i)})=1} \frac{\alpha_{t'}}{\alpha}$ and $p_- = \sum_{t':h_{t'}(x^{(i)})=-1} \frac{\alpha_{t'}}{\alpha}$, and $H_t(x^{(i)}) = p_+ - p_-$.

Note that if the true label $y^{(i)} = +1$, our margin is $p_{correct} - p_{incorrect} = p_+ - p_- = H_t(x^{(i)})$. If the true label is $y^{(i)} = -1$, our margin is $p_{correct} - p_{incorrect} = p_- - p_+ = -H_t(x^{(i)})$. Thus, our margin is always given by $y^{(i)} H_t(x^{(i)})$.

2.2.4 Weak Learners

We always talk using AdaBoost with "weak" learners; why can't we ensemble together "stronger" learners? Let's take a look at bounds on the test error of AdaBoost, fixing the number of samples N and number of training iterations T, but allowing variation in the hypothesis class of weak learners $\mathcal{H}$.

Let d be the VC-dimension of the hypothesis class. Consider the following bounds on the error of the ensemble g_T with respect to d:

$$\text{Bound 1 (PAC Learning)}: \quad \text{True Error} \leq \text{Train Error} + O\left(\sqrt{T \log T}\sqrt{d}\sqrt{\frac{\log N}{N}}\right)$$

$$\text{Bound 2 (Margin Analysis)}: \quad \text{True Error} \leq \hat{P}_S\left[\text{margin}_T \leq \theta\right] + O\left(\frac{1}{\theta}\sqrt{d}\sqrt{\frac{\log^2 N}{N}}\right)$$

1. What happens to our bounds on true error if we increase the VC dimension of the weak learner hypothesis space?

   The bounds loosen/increase.

2. What concept does this connection between classifier complexity and error relate to? Overfitting