CSE 353: Homework 4 — Due Friday Nov. 4

1. *Machine learning fundamentals.* **(3 pts, 1 pt each)**

   There is a growing need for machine models to have *interpretability*; that is, when it tells you the label of your new data sample, you want to also understand *why* it told you this. One of the most common answers to the "why" is to find which features (or which combination of features) best contributed to the answer given, or which samples best contributed to the model. In each case, describe how you could use parts of the learned model to return not only a new data label, but which features (or samples) were most important. Be specific and detailed; if I can't implement your suggestion based on what you wrote, you won't get full credit.

   (a) Decision trees

   (b) Linear regression

   (c) Support vector machines.

(source: xkcd)

2. *Adaboost***(3pts)** A popular and computationally cheap boosting method is adaboost, described in Algorithm 1. In particular, it is a greedy coordinate-wise method that minimizes the empirical exponential loss, e.g. given a predictor $h(x) = y$, we find h which minimizes

$$f(h) = \frac{1}{m}\sum_{i=1}^{m} \exp(-y_i h(x_i)).$$

In this problem, we will implement Adaboost and analyze its greedy structure.

Algorithm 1: Discrete Adaboost (source: https://en.wikipedia.org/wiki/AdaBoost)

Data: Samples: $x_1, ..., x_m$, training labels $y_i \in \{-1, 1\}$, weak learners $\mathcal{H}$.
Result: Classifier $H(x) = \sum_{t=1}^{T} \alpha_t h^{(t)}(x)$
Initial weights $w_i^{(0)} = \frac{1}{m}$ for $i = 1, ..., m$;
for $t = 1, ..., T$ **do**

Choose $h^{(t)}(x)$ which minimizes the weighted sum error for misclassified points

$$h^{(t)}(x) = \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{\substack{i=1 \\ h(x_i) \neq y_i}}^{m} w_i^{(t-1)}$$

Update

$$\alpha^{(t)} = \frac{1}{2}\log\left(\frac{1-\epsilon^{(t)}}{\epsilon^{(t)}}\right), \qquad \epsilon^{(t)} := \sum_{h^{(t)}(x_i) \neq y_i} w_i^{(t-1)} \qquad (\star)$$

Update weights

$$\hat{w}_i^{(t)} = w_i^{(t-1)} \exp(-y_i \alpha^{(t)} h^{(t)}(x_i)), \qquad w_i^{(t+1)} = \frac{\hat{w}_i^{(t+1)}}{\sum_j \hat{w}_j^{(t+1)}}$$

end

(a) Open the mnist_adaboost_release directory and in the iPython notebook, download the data. We are again going to do 4/9 handwriting disambiguation. Only minimal preprocessing was used in this dataset. You may now use sklearn's tree implementation.

- Write a function that computes the train and misclassification rates , as well as the train exponential loss value, using just this decision stump (tree of depth = 1). Over the given dataset, these values should be:

$$10.98\% \text{ train misclassification}, 15.27\% \text{ test misclasification}, 0.6259675480492947 \text{ exponential loss}$$

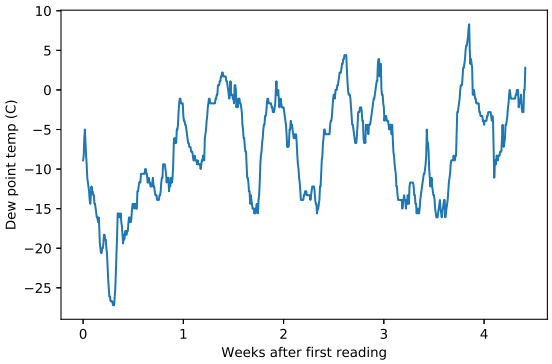
- **Deep trees.** Using the tree library from sklearn, train trees with depth 1 to 100, and plot the train and test misclassification rate, as well as the train exponential loss, as a function of depth. Report also the smallest train and test misclassification rate, and smallest train exponential loss, over the sweep.
- **Boosted decision stumps. (1 pt)** Now build a decision stump, e.g. a tree with depth = 1. Initialize weights as $w_i = 1/m$ for all $i = 1, ..., m$, and fit the decision tree over the *weighted* misclassification error, using the code snippet

`clf = clf.fit(Xtrain, ytrain, sample_weight = w)`.

- **(1 pt)** Implement the Adaboost method, as shown in algorithm 1. Plot the training exponential loss, and train and test misclassification rate. Report also the smallest train and test misclassification rate, and smallest train exponential loss, over the sweep. Comment on how the boosted decision stumps performed compared to the deep decision tree.
- **(1 pt)** Plot also $\epsilon^{(t)}$ and $\alpha^{(t)}$ as a function of t. For what values of $\epsilon^{(t)}$ is $\alpha^{(t)}$ really large and positive? really large and negative? close to 0? Interpret this mechanism; what is it saying about how boosting uses classifiers, in terms of their weighted performance?

3. *Polyfit via linear regression.* **(4 pts)**

- Download weatherDewTmp.mat. Plot the data. It should look like the following



- We want to form a polynomial regression of this data. That is, given w = weeks and d = dew readings, we want to find $\theta_1, ..., \theta_p$ as the solution to

$$\underset{\theta \in \mathbb{R}^p}{\text{minimize}} \quad \frac{1}{2} \sum_{i=1}^{m} (\theta_1 + \theta_2 w_i + \theta_3 w_i^2 + \cdots + \theta_p w_i^{p-1} - d_i)^2. \tag{1}$$

Form X and y such that (1) is equivalent to the least squares problem

$$\underset{\theta \in \mathbb{R}^p}{\text{minimize}} \quad \frac{1}{2} \|X\theta - y\|_2^2. \tag{2}$$

That is, for w the vector containing the week number, and y containing the dew data, form

$$X = \begin{bmatrix} 1 & w_1 & w_1^2 & w_1^3 & \cdots & w_1^{p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 1 & w_m & w_m^2 & w_m^3 & \cdots & w_m^{p-1} \end{bmatrix}.$$

(a) *Linear regression.***(1pt)**

i. Write down the normal equations for problem (2).

ii. Fill in the code to solve the normal equations for θ, and use it to build a predictor. To verify your code is running correctly, the number after `check number` should be 1.759 (implemented correctly) or 1.341 (also accepted).

iii. Implement a polynomial fit of orders $p = 1, 2, 3, 10, 100$, for the weather data provided. Include a figure that plots the original signal, overlaid with each polynomial fit. Comment on the "goodness of fit" for each value of p.

(b) *Ridge regression.* **(1pt)** Oftentimes, it is helpful to add a *regularization term* to (2), to improve stability. In other words, we solve

$$\underset{\theta \in \mathbb{R}^p}{\text{minimize}} \quad \tfrac{1}{2}\|X\theta - y\|_2^2 + \tfrac{\rho}{2}\|\theta\|_2^2. \tag{3}$$

for some $\rho > 0$.

i. Again, write down the normal equations for (3). Your equation should be of form $A\theta = b$ for some matrix A and vector b that you specify.

ii. Write the code for solving the ridge regression problem and run it. To verify your code is running correctly, the number after `check number` should be 1.206.

iii. Using $\rho = 1.0$, plot the weather data with overlaying polynomial fits with ridge regression. Provide these plots for $p = 1, 2, 3, 10, 100$. Comment on the "goodness of fit" and the stability of the fit, and also compare with the plots generated without using the extra penalty term.

(c) *Conditioning.* **(1pt)**

i. An *unconstrained quadratic problem* is any problem that can be written as

$$\underset{\theta}{\text{minimize}} \quad \frac{1}{2}\theta^T Q\theta + c^T\theta + r \tag{4}$$

for some symmetric positive semidefinite matrix Q, and some vector c and some scalar r. Show that the ridge regression problem (3) is an unconstrained quadratic problem by writing down Q, c, and r in terms of X and y such that (4) is equivalent to (3).

ii. In your code, write a function that takes in X and y, constructs Q as specified in the previous problem, and returns the condition number of Q. Report the condition number $\kappa(Q)$ for varying values of p and ρ, by filling in the following table. Here, $m = 742$ is the total number of data samples. Report at least 2 significant digits. Comment on how much ridge regression is needed to affect conditioning.

p	$\rho = 0$	$\rho = m$	$\rho = 10m$	$\rho = 100m$
1				
2				
5				
10				

iii. Under the *same experimental parameters* as the previous question, run ridge regression for each choice of p and ρ, and fill in the table with the mean squared error of the fit:

$$\textbf{mean squared error} = \frac{1}{m}\sum_{i=1}^{m}(x_i^T\theta - y[i])^2$$

where x_i is the ith row of X. Comment on the tradeoff between using larger ρ to improve conditioning vs its affect on the final performance.

p	$\rho = 0$	$\rho = m$	$\rho = 10m$	$\rho = 100m$
1				
2				
5				
10				

(d) *Forcasting.* **(1pt)** Picking your favorite set of hyperparameters (p, ρ), forecast the next week's dew point temperature. Plot the forecasted data over the current observations. Do you believe your forecast? Why?