Homework 6: Multiclass, Trees, and Gradient Boosting

Instructions: Your answers to the questions below, including plots and mathematical work, should be submitted as a single PDF file. It's preferred that you write your answers using software that typesets mathematics (e.g. LaTeX, LyX, or MathJax via iPython), though if you need to you may scan handwritten work. You may find the minted package convenient for including source code in your LaTeX document. If you are using LyX, then the listings package tends to work better.

1 Reformulations of Multiclass Hinge Loss

1.1 Multiclass setting review

Consider the multiclass output space $\mathcal{Y} = \{1, \ldots, k\}$. Suppose we have a base hypothesis space $\mathcal{H} = \{h : \mathcal{X} \times \mathcal{Y} \to \mathbf{R}\}$ from which we select a compatibility score function. Then our final multiclass hypothesis space is $\mathcal{F} = \left\{f(x) = \arg\max_{y \in \mathcal{Y}} h(x, y) \mid h \in \mathcal{H}\right\}$. Since functions in $\mathcal{F}$ map into $\mathcal{Y}$, our action space $\mathcal{A}$ and output space $\mathcal{Y}$ are the same. Nevertheless, we will write our class-sensitive loss function as $\Delta : \mathcal{Y} \times \mathcal{A} \to \mathbf{R}$, even though $\mathcal{Y} = \mathcal{A}$. We do this to indicate that the true class goes in the first slot of the function, while the prediction (i.e. the action) goes in the second slot. This is important because we do not assume that $\Delta(y, y') = \Delta(y', y)$. It would not be unusual to have this asymmetry in practice. For example, false alarms may be much less costly than no alarm when indeed something is going wrong.

In the spirit of empirical risk minimization, we would like to find $f \in \mathcal{F}$ minimizing the empirical cost-sensitive loss:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \Delta\left(y_i, f(x_i)\right),$$

possibly with some regularization. But this is clearly intractable, since we already know binary classification is intractable and that's a special case of this formulation. In lecture we proposed an alternative, tractable objective function: the multiclass SVM based on the convex multiclass hinge loss.

1.2 Two versions of multiclass hinge loss (or generalized hinge loss)

In lecture, we defined the **margin** of the compatibility score function h on the ith example (x_i, y_i) for class y as

$$m_{i,y}(h) = h(x_i, y_i) - h(x_i, y).$$

We also gave a formulation of a multiclass SVM objective function, where the loss on an individual example (x_i, y_i) was

$$\ell_1(h,(x_i,y_i)) = \max_{y\in\mathcal{Y}-\{y_i\}} \left(\max\left[0, \Delta(y_i,y) - m_{i,y}(h)\right]\right).$$

There's an alternative formulation, called the **generalized hinge loss** in SSBD Section 17.2. There they define

$$\ell_2(h,(x_i,y_i)) = \max_{y\in\mathcal{Y}} \left[\Delta\left(y_i,y\right) + h(x_i,y) - h(x_i,y_i)\right].$$

1. Show that if $\Delta(y,y) = 0$ for all $y \in \mathcal{Y}$, then $\ell_2\left(h,(x_i,y_i)\right) = \ell_1(h,(x_i,y_i))$. [Hint: Note that $\max_{y\in\mathcal{Y}} \phi(y) = \max\left(\phi(y_i), \max_{y\in\mathcal{Y}-\{y_i\}} \phi(y_i)\right)$.]
   SOLUTION:

$$\begin{aligned}
\ell_2\left(h,(x_i,y_i)\right) &= \max_{y\in\mathcal{Y}} \left[\Delta\left(y_i,y\right) + h(x_i,y) - h(x_i,y_i)\right] \\
&= \max_{y\in\mathcal{Y}} \left[\Delta\left(y_i,y\right) - m_{i,y}(h)\right] \\
&= \max\left(\underbrace{\Delta\left(y_i,y\right) - m_{i,y}(h)}_{y=y_i}, \max_{y\in\mathcal{Y}-\{y_i\}} \left[\Delta\left(y_i,y\right) - m_{i,y}(h)\right]\right) \\
&= \max\left(0, \max_{y\in\mathcal{Y}-\{y_i\}} \left[\Delta\left(y_i,y\right) - m_{i,y}(h)\right]\right) \\
&= \max_{y\in\mathcal{Y}-\{y_i\}} \left(\max\left[0, \Delta\left(y_i,y\right) - m_{i,y}(h)\right]\right) \\
&= \ell_1(h,(x_i,y_i))
\end{aligned}$$

2. In the context of the generalized hinge loss, we've said that $\Delta(y_i,y)$ is like the "target margin" between the score for true class y_i and the score for class y. Suppose that for our compatibility function h, all target margins are reached or exceeded on x_i. That is

$$m_{i,y}(h) = h(x_i,y_i) - h(x_i,y) \geq \Delta(y_i,y),$$

   for all $y \in \mathcal{Y} - \{y_i\}$. Assume that $\Delta(y_i,y) > 0\ \forall y \neq y_i$ and $\Delta(y_i,y) = 0$ for $y = y_i$.]

   (a) Show that under the conditions above, $\ell_1(h,(x_i,y_i)) = \ell_2(h,(x_i,y_i)) = 0$.
   SOLUTION:

$$\begin{aligned}
\ell_1\left(h,(x_i,y_i)\right) &= \max_{y\in\mathcal{Y}-\{y_i\}} \left(\max\left[0, \underbrace{\Delta\left(y_i,y\right) - m_{i,y}(h)}_{\leq 0}\right]\right) \\
&= \max_{y\in\mathcal{Y}-\{y_i\}} (0) = 0
\end{aligned}$$

   and since we've assumed $\Delta(y_i,y_i) = 0$, we also have $\ell_1 = \ell_2$, by a previous problem.

   (b) Show that under the conditions above, we make the correct prediction on x_i. That is, $f(x_i) = \arg\max_{y\in\mathcal{Y}} h(x_i,y) = y_i$.SOLUTION: By the margin assumption, we know that for every $y \neq y_i$ we have

$$h(x_i,y_i) \geq \Delta(y_i,y) + h(x_i,y).$$

So

$$\begin{aligned} h(x_i, y_i) &\geq \max_{y \neq y_i} \left[\underbrace{\Delta(y_i, y)}_{>0} + h(x_i, y) \right] \\ &> \max_{y \neq y_i} \left[h(x_i, y) \right]. \end{aligned}$$

Thus $h(x_i, y_i)$ is strictly larger than $h(x_i, y)$ for $y \neq y_i$. Thus

$$f(x_i) = \arg\max_{y \in \mathcal{Y}} h(x_i, y) = y_i.$$

2 SGD for Multiclass Linear SVM

Suppose our output space and our action space are given as follows: $\mathcal{Y} = \mathcal{A} = \{1, \ldots, k\}$. Given a non-negative class-sensitive loss function $\Delta : \mathcal{Y} \times \mathcal{A} \to [0, \infty)$ and a class-sensitive feature mapping $\Psi : \mathcal{X} \times \mathcal{Y} \to \mathbf{R}^d$. Our prediction function $f : \mathcal{X} \to \mathcal{Y}$ is given by

$$f_w(x) = \arg\max_{y \in \mathcal{Y}} \langle w, \Psi(x, y) \rangle$$

For training data $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$, let $J(w)$ be the ℓ_2-regularized empirical risk function for the multiclass hinge loss. We can write this as

$$J(w) = \lambda \|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} \max_{y \in \mathcal{Y}} \left[\Delta\left(y_i, y\right) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle \right],$$

for some $\lambda > 0$.

1. [Optional] Show that $J(w)$ is a convex function of w. You may use any of the rules about convex functions described in our notes on Convex Optimization, in previous assignments, or in the Boyd and Vandenberghe book, though you should cite the general facts you are using. [Hint: If $f_1, \ldots, f_m : \mathbf{R}^n \to \mathbf{R}$ are convex, then their pointwise maximum $f(x) = \max\{f_1(x), \ldots, f_m(x)\}$ is also convex.]
   SOLUTION: The function $w \mapsto \Delta(y_i, y) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle$ is an affine function of w and thus convex (since with respect to w, $\Delta(y_i, y)$ is a constant scalar and $\Psi(x_i, y) - \Psi(x_i, y_i)$ is a constant vector). Thus $w \mapsto \max_{y \in \mathcal{Y}} \left[\Delta\left(y_i, y\right) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle \right]$ is convex, since it's the pointwise maximum of a set of convex functions. Next

   $$w \mapsto \sum_{i=1}^{n} \max_{y \in \mathcal{Y}} \left[\Delta\left(y_i, y\right) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle \right]$$

   is convex, since it's a nonnegative combination of convex functions. The norm piece $w \mapsto \|w\|^2$ is convex, since norms in $\mathbf{R}^d$ are convex. Finally, $J(w)$ is convex, since it's a nonnegative combination of things we've already shown to be convex.

2. Since $J(w)$ is convex, it has a subgradient at every point. Give an expression for a subgradient of $J(w)$. You may use any standard results about subgradients, including the result from an

earlier homework about subgradients of the pointwise maxima of functions. (Hint: It may be helpful to refer to $\hat{y}_i = \arg\max_{y \in \mathcal{Y}} [\Delta(y_i, y) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle]$.)
SOLUTION: A subgradient of $J(w)$ at w is given by the following expression for $g \in \mathbf{R}^d$:

$$g = 2\lambda w + \frac{1}{n} \sum_{i=1}^{n} \left(\Psi(x_i, \hat{y}_i) - \Psi(x_i, y_i) \right).$$

3. Give an expression for the stochastic subgradient based on the point (x_i, y_i).
SOLUTION:
$$g = 2\lambda w + \left(\Psi(x_i, \hat{y}) - \Psi(x_i, y_i) \right).$$

4. Give an expression for a minibatch subgradient, based on the points $(x_i, y_i), \ldots, (x_{i+m-1}, y_{i+m-1})$.
SOLUTION:
$$g = 2\lambda w + \frac{1}{m} \sum_{j=i}^{i+m-1} \left(\Psi(x_j, \hat{y}) - \Psi(x_j, y_j) \right)$$

3 [Optional] Hinge Loss is a Special Case of Generalized Hinge Loss

Let $\mathcal{Y} = \{-1, 1\}$. Let $\Delta(y, \hat{y}) = 1(y \neq \hat{y})$. If $g(x)$ is the score function in our binary classification setting, then define our compatibility function as

$$\begin{aligned} h(x, 1) &= g(x)/2 \\ h(x, -1) &= -g(x)/2. \end{aligned}$$

Show that for this choice of h, the multiclass hinge loss reduces to hinge loss:

$$\ell(h, (x, y)) = \max_{y' \in \mathcal{Y}} [\Delta(y, y')) + h(x, y') - h(x, y)] = \max\{0, 1 - yg(x)\}$$

SOLUTION: We have

$$\begin{aligned} \ell(h, (x, y)) &= \max_{y' \in \mathcal{Y}} [\Delta(y, y')) + h(x, y') - h(x, y)] \\ &= \max\{\Delta(y, y), [\Delta(y, -y)) + h(x, -y) - h(x, y)]\} \\ &= \max\left\{0, \left[1 + \frac{1}{2} \begin{cases} -g(x) - g(x) & \text{for } y = 1 \\ g(x) - [-g(x)] & \text{for } y = -1 \end{cases}\right]\right\} \\ &= \max\left\{0, \left[1 + \begin{cases} -g(x) & \text{for } y = 1 \\ g(x) & \text{for } y = -1 \end{cases}\right]\right\} \\ &= \max\{0, 1 - yg(x)\} \end{aligned}$$

4 Multiclass Classification - Implementation

In this problem we will work on a simple three-class classification example, similar to the one given in lecture. The data is generated and plotted for you in the skeleton code.

4.1 One-vs-All (also known as One-vs-Rest)

In this problem we will implement one-vs-all multiclass classification. Our approach will assume we have a binary base classifier that returns a score, and we will predict the class that has the highest score.

1. Complete the class OneVsAllClassifier in the skeleton code. Following the OneVsAllClassifier code is a cell that extracts the results of the fit and plots the decision region. Include these results in your submission.

4.2 Multiclass SVM

In this question, we will implement stochastic subgradient descent for the linear multiclass SVM, as described in lecture and in this problem set. We will use the class-sensitive feature mapping approach with the "multivector construction", as described in our multiclass classification lecture and in SSBD Section 17.2.1.

1. Complete the skeleton code for multiclass SVM. Following the multiclass SVM implementation, we have included another block of test code. Make sure to include the results from these tests in your assignment, along with your code.

5 [Optional] Audio Classification

In this problem, we will work on the urban sound dataset URBANSOUND8K from the Center for Urban Science and Progress (CUSP) at NYU. (You should download the data from that link.) We will first extract features from raw audio data using the LibROSA package, and then we will train multiclass classifiers to classify the sounds into 10 sound classes. URBANSOUND8K dataset contains 8732 labeled sound excerpts. For this problem, you may use the file UrbanSound8K.csv to randomly sample 2000 examples for training and 2000 examples for validation.

1. In LibROSA, there are many functions for visualizing audio waves and spectra, such as display.waveplot() and display.specshow(). Load a random audio file from each class as a floating point time series with librosa.load(), and plot their waves and linear-frequency power spectrogram. If you are interested, you can also play the audio in the notebook with functions display() and Audio() in IPython.display.

2. Mel-frequency cepstral coefficients (MFCC) are a commonly used feature for sound processing. We will use MFCC and its first and second differences (like discrete derivatives) as our features for classfication. First, use function feature.mfcc() from LibROSA to extract MFCC features from each audio sample. (The first MFCC coefficient is typically discarded in sound analysis, but you do not need to. You can test whether this helps in the optional problem below.) Next, use function feature.delta() to calculate the first and second differences of MFCC. Finally, combine these features and normalize each feature to zero mean and unit variance.

3. Train a linear multiclass SVM on your 2000 example training set. Evaluate your results on the validation set in terms of 0/1 error and generate a confusion table. Compare the results to a one-vs-all classifier using a binary linear SVM as the base classifier. For each model, may use your code from the previous problem, or you may use another implementation (e.g. from sklearn).

4. [More Optional] Compare results to any other multiclass classification methods of your choice.

5. [More Optional] Try different feature sets and see how they affect performance.

6 [Optional] Decision Tree Implementation

In this problem we'll implement decision trees for both classification and regression. The strategy will be to implement a generic class, called `Decision_Tree`, which we'll supply with the loss function we want to use to make node splitting decisions, as well as the estimator we'll use to come up with the prediction associated with each leaf node. For classification, this prediction could be a vector of probabilities, but for simplicity we'll just consider hard classifications here. We'll work with the classification and regression data sets from previous assignments.

1. [Optional] Complete the class `Decision_Tree`, given in the skeleton code. The intended implementation is as follows: Each object of type `Decision_Tree` represents a single node of the tree. The depth of that node is represented by the variable self.depth, with the root node having depth 0. The main job of the fit function is to decide, given the data provided, how to split the node or whether it should remain a leaf node. If the node will split, then the splitting feature and splitting value are recorded, and the left and right subtrees are fit on the relevant portions of the data. Thus tree-building is a recursive procedure. We should have as many`Decision_Tree` objects as there are nodes in the tree. We will not implement pruning here. Some additional details are given in the skeleton code.

2. [Optional] Complete either the `compute_entropy` or `compute_gini` functions. Run the code provided that builds trees for the two-dimensional classification data. Include the results. For debugging, you may want to compare results with sklearn's decision tree. For visualization, you'll need to install `graphviz`.

3. [Optional] Complete the function `mean_absolute_deviation_around_median` (MAE). Use the code provided to fit the `Regression_Tree` to the krr dataset using both the MAE loss and median predictions. Include the plots for the 6 fits.

7 Gradient Boosting Machines

Recall the general gradient boosting algorithm[1], for a given loss function ℓ and a hypothesis space $\mathcal{F}$ of regression functions (i.e. functions mapping from the input space to $\mathbf{R}$):

1. Initialize $f_0(x) = 0$.

2. For $m = 1$ to M:

   (a) Compute:

$$\mathbf{g}_m = \left(\frac{\partial}{\partial f(x_j)} \sum_{i=1}^{n} \ell\left(y_i, f(x_i)\right) \Big|_{f(x_i)=f_{m-1}(x_i),\, i=1,\ldots,n} \right)_{j=1}^{n}$$

[1] Besides the lecture slides, you can find an accessible discussion of this approach in http://www.saedsayad.com/docs/gbm2.pdf, in one of the original references http://statweb.stanford.edu/~jhf/ftp/trebst.pdf, and in this review paper http://web.stanford.edu/~hastie/Papers/buehlmann.pdf.

(b) Fit regression model to $-\mathbf{g}_m$:

$$h_m = \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} \left((-\mathbf{g}_m)_i - h(x_i) \right)^2 .$$

(c) Choose fixed step size $\nu_m = \nu \in (0, 1]$, or take

$$\nu_m = \arg\min_{\nu > 0} \sum_{i=1}^{n} \ell \left(y_i, f_{m-1}(x_i) + \nu h_m(x_i) \right) .$$

(d) Take the step:

$$f_m(x) = f_{m-1}(x) + \nu_m h_m(x)$$

3. Return f_M.

In this problem we'll derive two special cases of the general gradient boosting framework: ℓ_2-Boosting and BinomialBoost.

1. Consider the regression framework, where $\mathcal{Y} = \mathbf{R}$. Suppose our loss function is given by

$$\ell(\hat{y}, y) = \frac{1}{2} (\hat{y} - y)^2 ,$$

and at the beginning of the m'th round of gradient boosting, we have the function $f_{m-1}(x)$. Show that the h_m chosen as the next basis function is given by

$$h_m = \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} \left[(y_i - f_{m-1}(x_i)) - h(x_i) \right]^2 .$$

In other words, at each stage we find the base prediction function $h_m \in \mathcal{F}$ that is the best fit to the residuals from the previous stage. [Hint: Once you understand what's going on, this is a pretty easy problem.] **Solution:**
The i^{th} component of $\mathbf{g}_m$ is computed as the partial derivative of $\ell(f(x_i), y_i)$ with respect to $f(x_i)$ at $f(x_i) = f_{m-1}(x_i)$.

$$\begin{aligned} \partial_{f(x_i)} \ell(f(x_i), y_i) &= \partial_{f(x_i)} \left[\frac{1}{2} (f(x_i) - y_i)^2 \right] \\ &= f(x_i) - y_i \end{aligned}$$

Evaluating this at $f(x_i) = f_{m-1}(x_i)$ we get $f_{m-1}(x_i) - y_i$. Thus the expression for $\mathbf{g}_m$ is

$$\mathbf{g}_m = (f_{m-1}(x_i) - y_i)_{i=1}^{n} .$$

Thus,

$$h_m = \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} \left((-\mathbf{g}_m)_i - h(x_i) \right)^2 = \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} \left[(y_i - f_{m-1}(x_i)) - h(x_i) \right]^2 .$$

2. Now let's consider the classification framework, where $\mathcal{Y} = \{-1, 1\}$. In lecture, we noted that AdaBoost corresponds to forward stagewise additive modeling with the exponential loss, and that the exponential loss is not very robust to outliers (i.e. outliers can have a large effect on the final prediction function). Instead, let's consider the logistic loss

$$\ell(m) = \ln\left(1 + e^{-m}\right),$$

where $m = yf(x)$ is the margin. Similar to what we did in the ℓ_2-Boosting question, write an expression for h_m as an argmin over $\mathcal{F}$. **Solution:**
Proceeding as in the previous question, we get

$$\mathbf{g}_m = \left(\frac{-y_i}{1 + e^{y_i f_{m-1}(x_i)}}\right)_{i=1}^{n}$$

and,

$$h_m = \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} \left((-\mathbf{g}_m)_i - h(x_i)\right)^2 = \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} \left[\left(\frac{y_i}{1 + e^{y_i f_{m-1}(x_i)}}\right) - h(x_i)\right]^2.$$

8 Gradient Boosting Implementation

This method goes by many names, including gradient boosting machines (GBM), generalized boosting models (GBM), AnyBoost, and gradient boosted regression trees (GBRT), among others. Although one of the nice aspects of gradient boosting is that it can be applied to any problem with a subdifferentiable loss function, here we'll keep things simple and consider the standard regression setting with square loss.

1. Complete the `gradient_boosting` class. As the base regression algorithm, you may use sklearn's regression tree. You should use the square loss for the tree splitting rule and the mean function for the leaf prediction rule. Run the code provided to build gradient boosting models on the classification and regression data sets, and include the plots generated. Note that we are using square loss to fit the classification data, as well as the regression data.

2. [Optional] Repeat the previous runs on the classification data set, but use a different classification loss, such as logistic loss or hinge loss. Include the new code and plots of your results. Note that you should still use the same regression tree settings for the base regression algorithm.