COMPSCI 671D Fall 2019
Homework 2

1 Convexity (25 points)

In this question, we will explore the convexity of some common empirical risk functions in machine learning. Suppose we have a dataset $\{X, \mathbf{y}\} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ that we will perform classification on, where $\mathbf{x}_i \in \mathbb{R}^p$ is a feature vector, and $y_i \in \{-1, +1\}$ is a binary label.

Please identify which of the following empirical risk functions are convex *with respect to weight parameter vector* $\mathbf{w}$ and show by proof whether or not they are convex. In the models below, $\mathbf{w} = [w_1, ..., w_p]^T \in \mathbb{R}^p, b \in \mathbb{R}$ are the weight and bias. We would like to show convexity with respect to $\mathbf{w}$ and b.

(Hint: you might not want to take the second derivative for some empirical risk functions, especially when some are not everywhere differentiable. Instead, you can use properties of convex functions along with the fact that affine functions are convex.)

(a) Logistic Regression (5 points)

$$L(\mathbf{w}, b) = \sum_{i=1}^{n} \log(1 + \exp(-y_i(\mathbf{w}^T\mathbf{x}_i + b)))$$

(b) SVM (5 points)

$$L(\mathbf{w}, b; C) = \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)), \quad C \geq 0$$

where C is the penalty parameter for errors.

Now suppose we have another dataset $\{X, \mathbf{z}\} = \{\mathbf{x}_i, z_i\}_{i=1}^n$ that we will perform regression on, where $\mathbf{x}_i \in \mathbb{R}^p$ is a feature vector, and $z_i \in \mathbb{R}$ is a real-valued target. Let us continue the problem, determining the convexity of empirical risk functions.

(c) LASSO (5 points)

$$L(\mathbf{w}, b; \lambda) = ||\mathbf{z} - (X\mathbf{w} + b\mathbf{1}_n)||_2^2 + \lambda||\mathbf{w}||_1, \quad \lambda \geq 0$$

where λ is the tuning parameter for L_1 penalty.

(d) Linear Regression with SCAD penalty (10 points)

$$L(\mathbf{w}, b; \lambda) = ||\mathbf{z} - (X\mathbf{w} + b\mathbf{1}_n)||_2^2 + \sum_{j=1}^{p} P(w_j; \lambda, \gamma), \quad \lambda > 0, \gamma > 2$$

where $P(x; \lambda, \gamma)$ is the penalty function

$$P(x; \lambda, \gamma) = \begin{cases} \lambda|x| & \text{if } |x| \leq \lambda \\ \frac{2\gamma\lambda|x| - x^2 - \lambda^2}{2(\gamma - 1)} & \text{if } \lambda < |x| < \gamma\lambda \\ \frac{\lambda^2(\gamma+1)}{2} & \text{if } |x| \geq \gamma\lambda \end{cases}$$

2 Support Vector Machine (35 points)

In this part, let's go through the basic concepts of support vector machines and use visualization to help with our understanding of how SVM works.

(a) Least-squares Support Vector Machine (10 points)

Given a dataset $\{x_i, y_i\}_{i=1}^n, x_i \in \mathbb{R}^p, y_i \in \{-1, +1\}$, the soft SVM learns the classifier by the following optimization problem:

$$\min L(w, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i, \quad s.t. \begin{cases} y_i[w^T \phi(x_i) + b] \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases}$$

where $\phi(x)$ is the feature map from the original space to the higher- or infinite-dimensional space.

The least-squares SVM introduces the sum of squared errors within the objective function of the standard SVM, which reformulates the optimization problem as

$$\min L(w, b, e) = \frac{1}{2} w^T w + \frac{1}{2} C \sum_{i=1}^{n} e_i^2, \quad s.t. \quad y_i[w^T \phi(x_i) + b] = 1 - e_i.$$

Using similar techniques (KKT conditions) as in the standard soft-margin SVM, derive the explicit solution of LS-SVM with the kernel matrix $\Omega \in \mathbb{R}^{n \times n}$ defined by $\Omega_{ij} = \phi(x_i)^T \phi(x_j) = K(x_i, x_j)$ and remember to write out the formula for the classifier so that you can make predictions from it.

You are permitted to look the answer up on the internet but you should solve it on your own first to make sure you understand KKT conditions and how to apply them.

(b) Implement SVM and LS-SVM (15 points)

With the result derived in part (a), **write your own code** to implement LS-SVM to construct a classifier with **rbf** kernel. Make sure hyperparameters γ and C in your code can be user-specified, where γ is the tuning parameter in rbf kernel, $K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2)$ and C is the tuning parameter for the penalty on error.

Use **your LS-SVM classifier** and separately, use a **library for a standard SVM** (for example, *sklearn.svm* in python) to fit a variety of SVM classifiers on the toy dataset by varying the kernel, as well as hyperparameters C and γ in appropriate ranges. How do your classification errors change in the training set and the test set? Draw box plots showing the classification training and test errors for each algorithm and comment on your findings in light of your knowledge of overfitting and underfitting.

(c) Support Vectors (10 points)

Why is support vector machine called a support vector machine? What do support vectors mean? In fact, some people say that this terminology is misleading. It would be useful to do visualization to help with our understanding of the concepts. In fact, these kinds of visualizations can be very helpful in general, particularly for low-dimensional datasets. Figure 1 shows classification results of a linear kernel SVM on the **training set** of the toy dataset. The 2 gray dashed lines are contours for $-1, 1$ levels of the decision function and the dash-dotted line is for the decision boundary at the 0-level. We've also put in some nice colors at various level sets, just to make it look nicer.

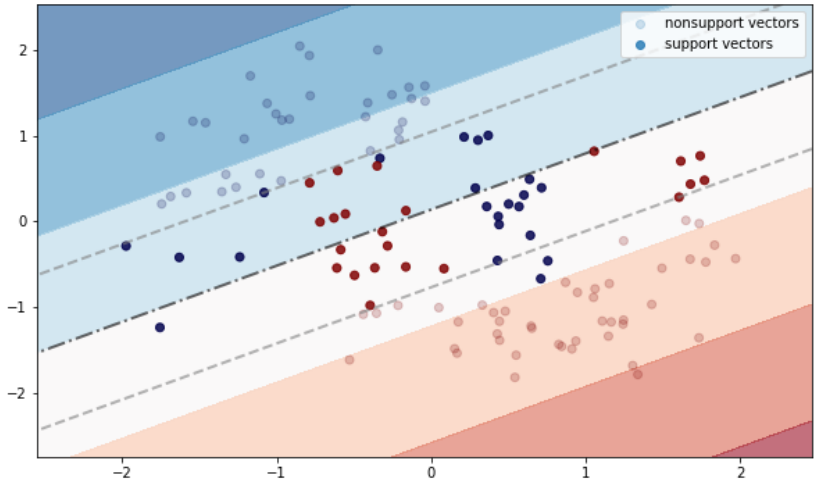

Figure 1: Support vectors in linear SVM

(i) Use the rbf kernel in SVM (built-in libraries are fine), set your γ and C parameters to be 1 and plot similar classification results on the **training set**. (Use *contourf* and *contour* in *matplotlib.pyplot* if you are coding with python). Your visualization should include:

- Scatter plot with points colored by their true labels
- Contour of different levels of decision function (Please set the colormap of the contour to match or complement the colors of your scatter plot. You don't have to specify specific levels for this contour.)
- Highlighted $[-1, 0, 1]$ levels on the contour
- Identified support vectors (You can make non-support vectors more transparent similar to Figure 1.)

(ii) Write an expression for a region that contains your support vectors with **positive labels**. Specifically, simplify $\{x : f(x) \leq \theta\}$ for a value of θ that contains all of your support vectors.

Calculate this in terms of $\{x_i, y_i, \alpha_i \text{ (dual variable)}\}_{i \in \mathcal{I}}$, rbf kernel function $K(x, y)$ and bias b, where $\mathcal{I}$ is the index set of support vectors.

(iii) Vary your γ in appropriate ranges. Generally speaking, how does the non-linearity of your boundary change? Then vary your C in appropriate ranges. How does the Euclidean distance between the 0 and 1 levels of your classifiers change?

Can you relate these observations to your findings on overfitting and underfitting in (b)?

3 AdaBoost with Stumps Fits an Additive Model (30 points + 5 bonus)

In class, we derived AdaBoost with exponential loss for classification. In this problem, we would like to derive AdaBoost with L2 loss for regression. Our goal is to model Californian house prices as a summation of bias and component functions on individual features, where we are given data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^p$, $y_i \in \mathbb{R}$.

$$\hat{y}_i = f(\mathbf{x}_i) = b + \sum_{q=1}^{p} f_q(\mathbf{x}_i). \tag{1}$$

We will choose the form of each component function $f_q(\mathbf{x}_i)$ to be a piece-wise constant function. The sum of component functions can then be written as a weighted summation of **decision stumps**: $h_j(x) = \mathbf{1}_{[x_{\cdot,q} > I_j]}$, where $\mathbf{1}_{[x_{\cdot,q} > I_j]}$ means that the value of the function is 1 if the qth component of x is greater than threshold I_j. We can add up weighted sums of these functions to produce the final combined classifier,

$$\hat{y}_i = f(x_i) = b + \sum_{j=1}^{m} \lambda_j h_j(x_i) = \mathbf{h}(\mathbf{x}_i)^T \lambda, \tag{2}$$

where $\mathbf{h}(\mathbf{x}_i) = [1, h_1(\mathbf{x}_i), \ldots, h_m(\mathbf{x}_i)]$ and $\lambda = [b, \lambda_1, \ldots, \lambda_m]$, $\lambda \in \mathbb{R}^m$ is a vector of weights.

We then define the L2 loss:

$$L(\lambda) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{h}(\mathbf{x}_i)^T \lambda)^2. \tag{3}$$

This will create a beautiful additive model that can be visualized as a set of plots - we can plot all of the component functions and visually see the whole model. It will look something like Figure 2 if we've done it right.

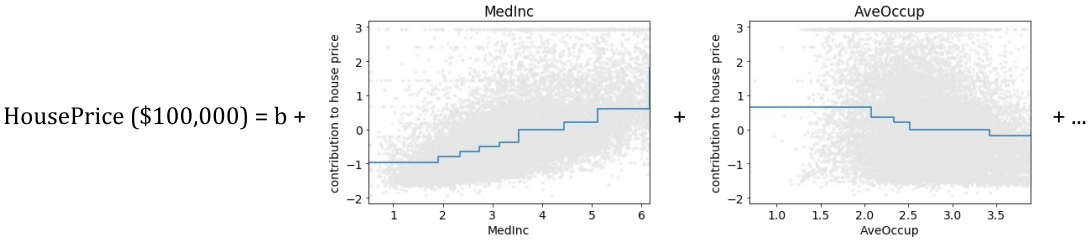

Figure 2: Californian house price explained with an additive model. Grey dots in the background are datapoints, each blue piece-wise constant line is a component function fitted by AdaBoost.

We are now ready to perform coordinate descent and model Californian house prices.

(a) Coordinate descent with L2 loss (10 points)

(i) Following the derivation of AdaBoost in class, derive the optimal direction to move in for AdaBoost with L2 loss.

(ii) After choosing the optimal direction, derive the amount to move by in that direction.

(b) Fit Californian house prices with AdaBoost (15 points)

(i) For each feature, features = ['MedInc','AveOccup',...], plot 'HousePrice' versus each feature q. (These plots should already tell us something intuitive about the model we will produce.)

(ii) Apply AdaBoost. Sample code for downloading the dataset and for creating the decision stumps is provided.

(iii) For each feature, plot the feature value on the horizontal axis and $f_q(x)$ on the vertical axis. Repeat this for each feature q, where features = ['MedInc','AveOccup',...]. Make sure to compute $f_q(x)$ as a weighted sum of all of the stumps on that feature. Describe the trends you see.

(iv) Compute variable importance using a technique shown in class. Discuss the results from variable importance in comparison with the plots in (i) and (iii).

(c) Compare with Linear Regression (5 points)

Using the sample code for boosted decision trees and linear regression, compute the mean square errors. Discuss any differences in mean square error between these algorithms.

(d) Bonus! (5 points)

Can you provide a way to ensure that each $f_q(x_{\cdot,q})$ is monotonically increasing in $x_{\cdot,q}$? How do you make it monotonically decreasing? Hint: the answer is very simple.

4 Random Forest (10 points)

"The generalization error of a random forest depends on the strength of the individual trees and the correlation between them." Leo Breiman (2001).

(a) Strength of individual trees (4 points) Unfortunately we cannot ask Breiman what he meant by "strength of the individual trees" any more, but perhaps we can make a good guess.

Show that decision trees can achieve low bias. In other words, show that there exist a decision tree with training error equal to zero, assuming no noise in the labels of the data. That is, $P(y = 1|\mathbf{x})$ is either 0 or 1. In that case, it is not possible to have conflicting labels, where a conflicting label would have $\mathbf{x}_i = \mathbf{x}_j$ but $y_i \neq y_j$.

(b) Correlation between trees (6 points) Each tree was grown from independent and identically distributed parameters governing subsample and feature selection. The sample selection scheme (on its own) is known as "bagging", and the feature selection scheme is known as choosing a "random subspace".

Suppose we are performing regression, for a datapoint $\mathbf{x}$, we have a set of random variables $\{F_j\}_{j=1}^M$, where each $F_j = f_j(\mathbf{x}) \in \mathbb{R}$ is an output from a tree in the random forest.

(i) If each F_j has variance σ^2, and the pairwise correlation coefficient is ρ, show that the variance of $\frac{1}{M}\sum_{j=1}^M F_j$ is given as:

$$\frac{\sigma^2}{M}(1 + \rho(M - 1)).$$

(ii) What happens to the variance of $\frac{1}{M}\sum_{j=1}^{M} F_j$ as $M \to \infty$?

(iii) How can you decrease the pairwise correlation coefficient ρ? Are there any benefits or disadvantages using the method you suggested? Hint: there are multiple possible answers.