Assignment 5
CSE 446: Machine Learning

University of Washington

Due: November 28, 2017

You will submit the following files or documents for this homework, compressed into a gzipped tarball named `A5.tgz`.

- Your **report** as a **pdf file** named `A5.pdf` containing answers to written questions. You must include the plots and explanation for programming questions (if required) in this document only. We *strongly* recommend typesetting your scientific writing using LaTeX. Some free tools that might help: ShareLaTeX (`www.sharelatex.com`), TexStudio (Windows), MacTex (Mac), TexMaker (cross-platform), and Detexify^2 (online). If you want to type, but don't know (and don't want to learn) LaTeX, consider using a markdown editor with real-time preview and equation editing (e.g., `stackedit.io`, `marxi.co`). Writing solutions by hand is fine, as long as they are neat; you will need to scan them into a single pdf.
  Part of the training we aim to give you in this advanced class includes practice with technical writing. Organize your report as neatly as possible, and articulate your thoughts as clearly as possible. We prefer quality over quantity. Do not flood the report with tangential information such as low-level documentation of your code that belongs in code comments or the README. Similarly, when discussing the experimental results, do not copy and paste the entire system output directly to the report. Instead, create tables and figures to organize the experimental results.
- **Source code** for the programming assignment. Please note that we will not accept Jupyter notebooks. You will submit your code together with a neatly written README file to instruct how to run your code with different settings (if applicable). We assume that you always follow good practice of coding (commenting, structuring), and these factors are not central to your grade.

You are welcome to use any Python libraries for data munging, visualization, and numerical linear algebra. Examples includes Numpy, Pandas, and Matplotlib. **For the programming part of this assignment, you *may* use machine learning libraries such as Scikit-Learn or TensorFlow, as long as you report all of the tools that you used. See details in problem 4.**

1 Multiclass Classification [30 points]

In this problem, you will learn about generalizing classification methods that you learned about in lectures to problems where the output space includes three or more classes. Let $\mathcal{Y}$ denote the set of output labels; formerly we talked about "positive" examples and negative examples (i.e., $\mathcal{Y} = \{+1, -1\}$). Now we treat $\mathcal{Y}$ as an abstract, discrete, finite set.

First, consider linear models where the binary classifier took the form:

$$f^{(\text{binary})}(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) \tag{1}$$

One way to generalize such a model so that f maps to label set $\mathcal{Y}$ is to assign a different set of parameters to each label $y \in \mathcal{Y}$, and let:

$$f^{(\text{multiclass})}(\mathbf{x}) = \operatorname*{argmax}_{y \in \mathcal{Y}} \mathbf{w}_y \cdot \mathbf{x} + b_y \tag{2}$$

Returning to the binary case, you can see that Equation 2 is a generalization of Equation 1:

- $\mathcal{Y} = \{+1, -1\}$, so we now have two weight vectors, $\mathbf{w}_{+1}$ and $\mathbf{w}_{-1}$, and two biases, b_{+1} and b_{-1}.
- Let the binary classifier's weights and bias be defined by:

$$\begin{aligned} \mathbf{w} &= \mathbf{w}_{+1} - \mathbf{w}_{-1} \\ b &= b_{+1} - b_{-1} \end{aligned}$$

- Now, $\text{sign}(\mathbf{w} \cdot \mathbf{x} + b) > 0$ if and only if $\mathbf{w}_{+1} \cdot \mathbf{x} + b_{+1} > \mathbf{w}_{-1} \cdot \mathbf{x} + b_{-1}$.

This means that, all along, our linear binary classifier was really estimating the *differences* between positive-class weights (and bias) and negative-class weights (and bias)![1]

It may be helpful to stack the weight vectors into a matrix (and the biases into a vector) and think about the activation function as mapping an input $\mathbf{x}$ to a $|\mathcal{Y}|$-length vector:

$$\mathbf{a}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{3}$$

$$\mathbf{W} = \left.\begin{bmatrix} \mathbf{w}_{y^1}^\top \\ \vdots \\ \mathbf{w}_{y^{|\mathcal{Y}|}}^\top \end{bmatrix}\right\} \text{one row per } y \in \mathcal{Y} \tag{4}$$

where $\mathbf{W} \in \mathbb{R}^{|\mathcal{Y}| \times d}$ and $\mathbf{b} \in \mathbb{R}^{|\mathcal{Y}|}$. Instead of a sign, the classifier f uses "argmax" to select a label that indexes the most-activated element in $\mathbf{a}(\mathbf{x})$.

Of course, all of the above can be done for nonlinear models as well. In fact, Equation 3 should remind you of the first layer in our two-layer neural network.

[1] There is an alternative formulation for multiclass models that fixes one special class's weights and bias at zero, effectively estimating the *differences* between that special class's parameters and every other class's parameters. That version uses $d + 1$ fewer free parameters, but is more confusing.

1. The log loss for multiclass classification is:

$$-(\mathbf{w}_{y_n} \cdot \mathbf{x}_n + b_{y_n}) + \log \sum_{y \in \mathcal{Y}} \exp\left(\mathbf{w}_y \cdot \mathbf{x}_n + b_y\right) \tag{5}$$

   Show that the log loss for binary classification that we discussed in class is a special case. [5 points]
2. Derive the stochastic gradient descent update for multiclass logistic regression. [5 points]
3. The perceptron loss for multiclass classification is:

$$-(\mathbf{w}_{y_n} \cdot \mathbf{x}_n + b_{y_n}) + \max_{y \in \mathcal{Y}} \left(\mathbf{w}_y \cdot \mathbf{x}_n + b_y\right) \tag{6}$$

   Show that the perceptron loss for binary classification that we discussed in class is a special case. [5 points]
4. Derive the update for multiclass perceptron, with the linear model used here. [5 points]
5. The hinge loss for multiclass classification is:

$$-(\mathbf{w}_{y_n} \cdot \mathbf{x}_n + b_{y_n}) + \max_{y \in \mathcal{Y}} \left(\mathbf{w}_y \cdot \mathbf{x}_n + b_y + [\![y \neq y_n]\!]\right) \tag{7}$$

   Show that the hinge loss for binary classification that appeared in A4 is a special case. [5 points]
6. Derive the stochastic subgradient descent update for multiclass hinge loss minimization, with the linear model used here. [5 points]

Note: if you have a vague sense that all of these loss functions are fairly similar to each other, you are correct. Here's a general loss function:

$$-(\mathbf{w}_{y_n} \cdot \mathbf{x}_n + b_{y_n}) + \frac{1}{\beta} \log \sum_{y \in \mathcal{Y}} \exp\left(\beta \cdot \left(\mathbf{w}_y \cdot \mathbf{x}_n + b_y + \gamma \cdot [\![y \neq y_n]\!]\right)\right) \tag{8}$$

The log loss sets $\beta = 1$ and $\gamma = 0$; the perceptron and loss take a limit as $\beta \to \infty$ with perceptron setting $\gamma = 0$ and hinge setting $\gamma = 1$. ($\beta = \gamma = 1$ corresponds to yet another loss function.) You might be able to solve the above problems on this generalized loss once, then specialize to the three cases.

2 Bonus [up to 5 points]

Multiclass logistic regression corresponds to maximum likelihood estimation for a probabilistic model of Y given X, where Y ranges over values in $\mathcal{Y}$. For an arbitrary $y \in \mathcal{Y}$ and $\mathbf{x} \in \mathbb{R}^d$, what form does this model give to $p(Y = y \mid \mathbf{x})$?

3 Kernels and Linear Separability [10 points]

(Original source: Norvig and Russell, Exercise 18.16.) The power of kernel methods stems from the observation that data which are not linearly separable in their input feature space may in fact be separable in a higher-dimensional space, which is a function of the input features (this function was denoted ϕ in lecture slides). For example, the data shown in Figure 1 are not linearly separable in the feature space $(\mathbf{x}[1], \mathbf{x}[2])$, but they *are* separable in the transformed space where we transform each point $\mathbf{x} = (\mathbf{x}[1], \mathbf{x}[2])$ according to $\phi(\mathbf{x}) = (\mathbf{x}[1]^2, \mathbf{x}[2]^2, \sqrt{2} \cdot \mathbf{x}[1] \cdot \mathbf{x}[2])$. Further, we can observe that for two data points $\mathbf{x}$ and $\mathbf{y}$, we have that

$$K(\mathbf{x}, \mathbf{v}) \triangleq \phi(\mathbf{x}) \cdot \phi(\mathbf{v}) = \mathbf{x}[1]^2 \cdot \mathbf{v}[1]^2 + \mathbf{x}[2]^2 \cdot \mathbf{v}[2]^2 + 2\mathbf{x}[1] \cdot \mathbf{v}[1] \cdot \mathbf{x}[2] \cdot \mathbf{v}[2] = (\mathbf{x} \cdot \mathbf{v})^2, \quad (9)$$

where K denotes a *kernel function*.[2]

In this problem, we will generalize the situation shown in Figure 1.

1. Consider data as in Figure 1, but now separated by a circular decision boundary of arbitrary center and radius. Such a decision boundary can be expressed as $(\mathbf{x}[1]-a)^2+(\mathbf{x}[2]-b)^2-r^2 = 0$. Show that a linear decision boundary can always be found if the data are mapped into the four-dimensional feature space $(\mathbf{x}[1], \mathbf{x}[2], \mathbf{x}[1]^2, \mathbf{x}[2]^2)$. [4 points]
2. For data separated by an ellipse of the form $c(\mathbf{x}[1] - a)^2 + d(\mathbf{x}[2] - b)^2 - 1 = 0$, show that a linear decision boundary can be found if the data are projected into the five-dimensional feature space $(\mathbf{x}[1], \mathbf{x}[2], \mathbf{x}[1]^2, \mathbf{x}[2]^2, \mathbf{x}[1] \cdot \mathbf{x}[2])$. [6 points]

4 Implementation: Multiclass Classification [60 points, plus up to 10 bonus points]

For this final assignment, you are to train a classifier for the “Fashion MNIST” task. Training and testing data, as well as documentation, are available at https://www.kaggle.com/zalando-research/fashionmnist.

Your solution must involve either a non-linear kernel, a neural network with at least two layers, or both. You may use existing libraries and software packages to implement your solution. You may report on **only one** classifer.

Your one-page report must include:

- a clear and concise description of the learning method you used to obtain your classifier;
- a clear and concise description of any hyperparameter tuning you did;
- a list of libraries or tools that you used in implementation (include any machine learning code that you did not personally implement);

[2] A notation detail: in case the notation $\mathbf{x} \cdot \mathbf{v}$ is new to you, it’s fine to think of it as the same as $\mathbf{x}^\top \mathbf{v}$. Unless you really like math, stop reading here. For math people, we use the “dot” notation here because it turns out that inner products can be defined on spaces more general than $\mathbb{R}^n$; in particular, they can be defined over infinite-dimensional spaces such as Hilbert space. In such a space, the notation $\mathbf{x}^\top \mathbf{v}$ doesn’t have an obvious semantics — what does it mean to transpose an infinite-dimensional object? However, the inner product is still well-defined. Kernel methods thus allow you to manipulate *infinite-dimensional objects* with *finite memory*, by working only with similarities between object pairs and not representing the objects explicitly. Did that just totally blow your mind? If this sort of things interests you, search online for the phrase “reproducing kernel Hilbert space.” On the other hand, if this discussion sounds like total nonsense, forget you ever read this.

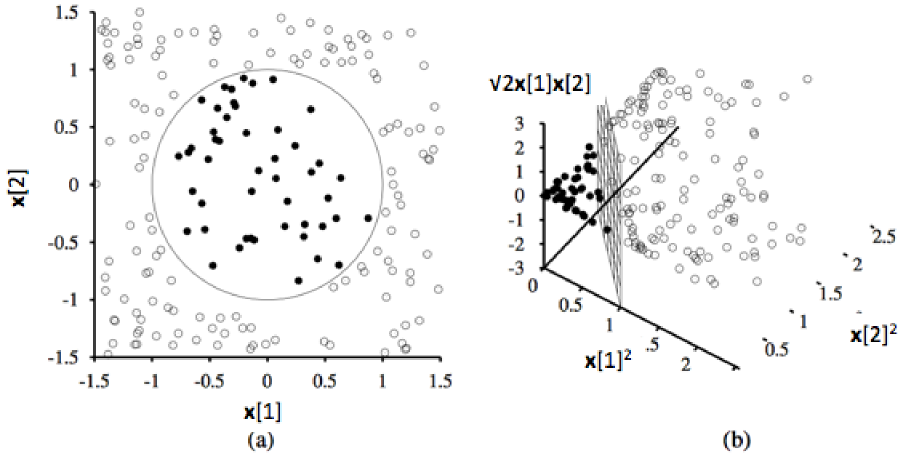

Figure 1: In (a), a two-dimensional training set with positive examples as black circles and negative as white circles. The true decision boundary, $\mathbf{x}[1]^2 + \mathbf{x}[2]^2 \leq 1$, is also shown. In (b), the data have been mapped into the three-dimensional input space $(\mathbf{x}[1]^2, \mathbf{x}[2]^2, \sqrt{2} \cdot \mathbf{x}[1] \cdot \mathbf{x}[2])$. The circular decision boundary in (a) becomes a linear decision boundary in three dimensions.

- an explanation of what *you* actually implemented;[3]
- accuracy on the training, development, and testing sets.

Include your source code as well. *Bonus points will be awarded for the most creative solutions.*

[3] We suspect in some cases that the description will say, “I didn’t implement anything; I simply called function `foo` with the following arguments . . . ” This is acceptable. It is also acceptable to use code that your team produced for the project, as long as you acknowledge it like everything else.