CS 475 Machine Learning: Homework 3
Supervised Classifiers 2

Due: Friday October 16, 2015, 11:59pm

100 Points Total Version 1.0

Make sure to read from start to finish before beginning the assignment.

1 Programming (50 points)

In this assignment you will implement a simple but effective stochastic gradient descent algorithm to solve the Support Vector Machine for binary classification problems. The approach is called *Primal Estimated sub-GrAdient SOlver for SVM* (Pegasos).[1] The number of iterations required by Pegasos to obtain a solution of accuracy ϵ is $O(1/\epsilon)$, while previous stochastic gradient descent methods require $O(1/\epsilon^2)$.

1.1 SVM

A Support Vector Machine constructs a hyperplane in high dimensional space, which separates training points of different classes while keeping a large margin with regards to the training points closest to the hyperplane. SVMs are typically formulated as a constrained quadratic programming problem. We can also reformulate it as an equivalent unconstrained problem of empirical loss plus a regularization term. Formally, given a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^M$ and $y_i \in \{+1, -1\}$, we want to find the minimizer of the problem:

$$\min_{\mathbf{w}} \lambda \frac{1}{2}||\mathbf{w}||^2 + \frac{1}{N}\sum_{i=1}^{N} l(\mathbf{w}; (\mathbf{x}_i, y_i)) \tag{1}$$

where λ is the regularization parameter,

$$l(\mathbf{w}; (\mathbf{x}, y)) = \max\{0, 1 - y\langle \mathbf{w}, \mathbf{x}\rangle\} \tag{2}$$

and $\langle \cdot, \cdot \rangle$ is the inner product operator. As you might have noticed, we omit the parameter for the bias term throughout this homework (i.e., the hyperplane is always across the origin).

1.2 Pegasos

Pegasos performs stochastic gradient descent on the primal objective Eq. 1. Similar to the stochastic version of logistic regression in homework 2, the learning rate decreases with each iteration to guarantee convergence. But unlike AdaGrad, where the learning

[1] Shalev-Shwartz, S., Singer, Y., Srebro, N., Cotter, A. (2011). Pegasos: Primal estimated sub-gradient solver for svm. Mathematical programming, 127(1), 3-30.

rate is adaptively chosen based on historical information, here we adopt a simple strategy and make the learning rate a function of the time steps.

On each time step Pegasos operates as follow. Initially, we set $\mathbf{w}_0 = 0$. On time step t of the algorithm (starting from $t = 1$) we first choose a random training example $(\mathbf{x}_{i_t}, y_{i_t})$ by picking an index $i_t \in \{1, \dots, m\}$ uniformly at random (see Section 1.6). We then replace the objective in Eq. 1 with an approximation based on the training example $(\mathbf{x}_{i_t}, y_{i_t})$, yielding:

$$f(\mathbf{w}; i_t) = \lambda \frac{1}{2} ||\mathbf{w}||^2 + l(\mathbf{w}; (\mathbf{x}_{i_t}, y_{i_t})) \tag{3}$$

We consider the sub-gradient of the above approximate objective, given by:

$$\frac{\partial f(\mathbf{w}; i_t)}{\partial \mathbf{w}_t} = \lambda \mathbf{w}_t - \mathbb{1}[y_{i_t} \langle \mathbf{w}_t, \mathbf{x}_{i_t} \rangle < 1] y_{i_t} \mathbf{x}_{i_t} \tag{4}$$

where $\mathbb{1}[\cdot]$ is the indicator function which takes a value of 1 if its argument is true, and 0 otherwise. We then update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \frac{\partial f(\mathbf{w}; i_t)}{\partial \mathbf{w}_t}$ using a step size of $\eta_t = 1/(\lambda t)$. Note that this update can be written as:

$$\mathbf{w}_{t+1} \leftarrow (1 - \frac{1}{t}) \mathbf{w}_t + \frac{1}{\lambda t} \mathbb{1}[y_{i_t} \langle \mathbf{w}_t, \mathbf{x}_{i_t} \rangle < 1] y_{i_t} \mathbf{x}_{i_t} \tag{5}$$

After a predetermined number T of iterations, we output the last iterate $\mathbf{w}_T$.

You will note that the above update rule changes $\mathbf{w}$ even for cases where the value in $\mathbf{x}_i$ is 0. This is a non-sparse update: every feature is updated every time. While there are sparse versions of Pegasos, for this homework you should implement the non-sparse solution. This means that every time you update, every parameter must be updated. This will make training slower on larger feature sets (such as NLP) but it should still be reasonable. We suggest you focus testing on the smaller and thus faster datasets.

1.3 Offset Feature

None of the math above mentions an offset feature (bias feature) $\mathbf{w}_0$, that corresponds to a $x_{i,0}$ that is always 1. It turns out that we don't need this if our data is centered. By centered we mean that $E[y] = 0$. For simplicity, assume that the data used in this assignment is centered (even though this may not be true). Do not include another feature that is always 1 (x_0) or weight ($\mathbf{w}_0$) for it.

1.4 Convergence

In practice, SGD optimization requires the program to determine when it has converged. Ideally, a maximized function has a gradient value of 0, but due to issues related to your step size, random noise, and machine precision, your gradient will likely never be exactly zero. Common practice is to check that the L_p norm of the gradient is less than some δ, for some p. For the sake of simplicity and consistent results, we will not do this in this assignment. Instead, your program should take a parameter **sgd_iterations** which is *exactly* how many iterations you should run (not an upper bound). An iteration is a single pass over every training example. **Note that the "t" in Section 1.2 indexes a time step, which is different from the "iterations" defined here.** The default of **sgd_iterations** should be 20.

You can add a command line parameter by adding the following code block to the `createCommandLineOptions` method of `Classify`.

```
registerOption("sgd_iterations", "int", true, "The number of SGD iterations.");
```

Be sure to add the option name exactly as it appears above. A common mistake is to change underscores to dashes.

You can read the value from the command line by adding the following to the main method of `Classify`:

```
int sgd_iterations = 20;
if (CommandLineUtilities.hasArg("sgd_iterations"))
    sgd_iterations = CommandLineUtilities.getOptionValueAsInt("sgd_iterations");
```

Note that this is the same argument added in the last assignment. You can reuse that argument for this algorithm.

1.5 Regularization Parameter

Pegasos uses a Regularization Parameter λ to adjust the relative strength of regularization. Your default value for λ should be 10^{-4}. You *must* add a command line argument to allow this value to be adjusted via the command line.

Add this command line option by adding the following code to the `createCommandLineOptions` method of `Classify`.

```
registerOption("pegasos_lambda", "double", true, "The regularization parameter for Pegasos.");
```

Be sure to add the option name exactly as it appears above. A common mistake is to change underscores to dashes.

You can then read the value from the command line by adding the following to the main method of `Classify`:

```
double pegasos_lambda = 1e-4;
if (CommandLineUtilities.hasArg("pegasos_lambda"))
    pegasos_lambda = CommandLineUtilities.getOptionValueAsFloat("pegasos_lambda");
```

1.6 Sampling Instances

Pegasos relies on sampling examples for each time step. For simplicity, we will NOT randomly sample instances. Instead, on round i you should update using the ith example. An iteration involves a single pass in order through all the provided instances. You will make several passes through the data based on `sgd_iterations`.

1.7 Implementation Notes

1. Many descriptions of SGD call for shuffling your data before learning. This is a good idea to break any dependence in the order of your data. In order to achieve consistent results for everyone, we are requiring that you **do not shuffle your data**. When you are training, you will go through your data in the order it appeared in the data file.

2. In SVM, $y \in \{+1, -1\}$, but the class labels in our data files are $0/1$ valued. To resolve this inconsistency, convert class label 0 to -1 before training.

3. During prediction, a new instance $\mathbf{x}$ is predicted by the following rule:

   $\hat{y}_{new} = 1$ if $\langle \mathbf{w}, \mathbf{x} \rangle \geq 0$
   $\hat{y}_{new} = 0$ otherwise

4. Initialize the parameters $\mathbf{w}$ to 0.

1.8 Deliverables

You need to implement the Pegasos algorithm. Your Pegasos predictor will be selected by passing the string `pegasos` as the argument for the algorithm parameter.

1.9 How Your Code Will Be Called

To train a model we will call:

```
java cs475.Classify -mode train -algorithm pegasos \
        -model_file speech.pegasos.model \
        -data speech.train
```

There are some additional parameters which your program must support during training:

```
-pegasos_lambda lambda   // sets the the constant scalar, default = 1e-4
-sgd_iterations t  // sets the number of SGD iterations, default = 20
```

All of these parameters are *optional*. If they are not present, they should be set to their default values.

To make predictions using a model we will call:

```
java cs475.Classify -mode test -algorithm pegasos \
        -model_file speech.pegasos.model \
        -data speech.test \
        -predictions_file speech.test.predictions
```

Remember that your output should be 0/1 valued, not real valued.

2 Analytical (50 points)

The following problems consider a standard binary classification setting: we are given n observations with m features, $x_1, ..., x_n \in \mathbb{R}^m$.

1) Overfitting (8 points) SVMs using nonlinear kernels usually have two tuning parameters (regularization parameter C and kernel parameter γ), which are usually determined by cross validation.

(a) Suppose we use cross validation to determine the non-linear kernel parameter and slack variable for an SVM. We find that classifiers using parameters (c_1, γ_1) and (c_2, γ_2) achieve the same cross validation error, but (c_1, γ_1) leads to fewer support vectors than (c_2, γ_2). Explain which set of parameters should we choose for the final model?

(b) The optimization problem of linear SVMs can be in either primal or dual form. As we know, the primal form has m parameters to learn, while the dual form has n parameters to learn. If $m \gg n$, is it true that the dual form reduces over-fitting since it has fewer parameters? Explain.

2) Hinge Loss (12 points) Linear SVMs can be formulated in an unconstrained optimization problem

$$\min_{w,b} \sum_{i=1}^{n} H(y_i(w^T x_i)) + \lambda \|w\|_2^2, \tag{6}$$

where λ is the regularization parameter and $H(a) = \max(1-a, 0)$ is the well known hinge loss function. The hinge loss function can be viewed as a convex surrogate of the 0/1 loss function $I(a \leq 0)$.

(a) Prove that $H(a)$ is a convex function of a.

(b) The function $L(a) = \max(-a, 0)$ can also approximate the 0/1 loss function. What is the disadvantage of using this function instead?

(c) If $H'(a) = \max(0.5 - a, 0)$, show that there exists λ' such that (7) is equivalent to (6). Hint: think about the geometric interpretation of hinge loss.

$$\min_{w,b} \sum_{i=1}^{n} H'(y_i(w^T x_i)) + \lambda' \|w\|_2^2. \tag{7}$$

3) Kernel Trick (10 points) The kernel trick extends SVMs to handle with nonlinear data sets. However, an improper use of a kernel function can cause serious over-fitting. Consider the following kernels.

(a) Polynomial kernel: $K(x, x') = (1 + (xx'^T))^d$, where $d \in \mathbb{N}$. Does increasing d make over-fitting more or less likely?

(b) Gaussian kernel: $K(x, x') = \exp(-||x - x'||^2/2\sigma^2)$, where $\sigma > 0$. Does increasing σ make over-fitting more or less likely?

We say K is a kernel function, if there exists some transformation $\phi : \mathbb{R}^m \to \mathbb{R}^{m'}$ such that $K(x_i, x_{i'}) = \langle \phi(x_i), \phi(x_{i'}) \rangle$.

(c) Let K_1 and K_2 be two kernel functions. Prove that $K(x_i, x_{i'}) = K_1(x_i, x_{i'}) + K_2(x_i, x_{i'})$ is also a kernel function.

4) Prediction using Kernel (8 points) One of the differences between linear SVMs and kernel SVMs concerns computational complexity at prediction time.

(a) What is the computational complexity of prediction of a linear SVM in terms of the examples n and features m?

(b) What is the computational complexity of prediction of a non-linear SVM in terms of the examples n, features m and support vectors s?

5) Stochastic Gradient Algorithm (12 points) The stochastic gradient algorithm is a very powerful optimization tool to solve large-scale machine learning problems. Instead of computing the gradient over the entire data set before making an update, the stochastic gradient algorithm computes the gradient over a single sample, then updates the parameters. By passing over the entire data set of n samples in this fashion we can converge to the optimal parameters.

A single iteration of stochastic gradient considers a single example. While it takes many more iterations, each iteration is much faster, both in terms of memory and computation.

Consider a ridge regression problem with n samples:

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \|\beta\|_2^2. \tag{8}$$

In each iteration, instead of using only one example, we randomly choose k out of n samples and obtain $(x_{1'}, y_{1'}), ..., (x_{k'}, y_{k'})$.

(a) What is the computational complexity of computing the mini-batch stochastic gradient or gradient at each iteration (using k and n samples respectively)?

(b) What are the advantages/disadvantages of increasing k in terms of computational complexity (using k and n samples respectively)? What is traded-off by increasing/decreasing k?

(c) Give one advantage and one disadvantage of using stochastic gradient descent with $k = 1$, ignoring computational considerations. Explain.

3 What to Submit

In each assignment you will submit two things.

1. **Code:** Your code as a zip file named `library.zip`. **You must submit source code (.java files)**. We will run your code using the exact command lines described above, so make sure it works ahead of time. Remember to submit all of the source code, including what we have provided to you.

2. **Writeup:** Your writeup as a **PDF file** (compiled from latex) containing answers to the analytical questions asked in the assignment. Make sure to include your name in the writeup PDF and use the provided latex template for your answers.

Make sure you name each of the files exactly as specified (library.zip and writeup.pdf).

To submit your assignment, visit the "Homework" section of the website (http://www.cs475.org/.)

4 Questions?

Remember to submit questions about the assignment to the appropriate group on the class discussion board: http://bb.cs475.org.