CS 475 Machine Learning: Homework 2
Supervised Classifiers 1
Due: Wednesday February 19, 2014, 12:00pm
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 write a logistic regression classifier. Your implementation will be very similar to the algorithm we covered in class. Your code needs to handle data with binary and continuous features and only binary labels (no multi-class).

1.1 Logistic Regression

The logistic regression model is used to model binary classification data. Logistic regression is a special case of generalized linear regression where the labels Y are modeled as a linear combination of the data X, but in a transformed space specified by g, sometimes called the "link function":

$$E[y \mid \mathbf{x}] = g(\mathbf{w}\mathbf{x} + \epsilon) \tag{1}$$

where ϵ is a noise term, usually taken to be Gaussian.

This "link function" allows you to model inherently non-linear data with a linear model. In the case of logistic regression, the link function is the logistic function:

$$g(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

1.2 Gradient Descent

In this assignment, we will solve for the parameters $\mathbf{w}$ in our logistic regression model using gradient descent to find the maximum likelihood estimate.

Gradient descent (GD) is an optimization technique that is both very simple and powerful. It works by taking the gradient of the objective function and taking steps in directions where the gradient is negative, which decreases the objective function[1].

1.3 Maximum Conditional Likelihood

Since we seek to maximize the objective, we will use gradient *ascent*. We begin by writing the conditional likelihood:

$$P(Y \mid \mathbf{w}, X) = \prod_{i=1}^{n} p(y_i \mid \mathbf{w}, \mathbf{x}_i) \tag{3}$$

[1] Gradient descent decreases the objective function if the gradient (first order approximation) is locally accurate, see Taylor expansion.

Since $y_i \in \{0, 1\}$, we can write the conditional probability inside the product as:

$$P(Y \mid \mathbf{w}, X) = \prod_{i=1}^{n} p(y_i = 1 \mid \mathbf{w}, \mathbf{x}_i)^{y_i} \times (p(y_i = 0 \mid \mathbf{w}, \mathbf{x}_i))^{1-y_i} \tag{4}$$

Note that one of these terms in the product will have an exponent of 0, and will evaluate to 1.

For ease of math and computation, we will take the log:

$$\ell(Y, X, \mathbf{w}) = \log P(Y \mid \mathbf{w}, X) = \sum_{i=1}^{n} y_i \log(p(y_i = 1 \mid \mathbf{w}, \mathbf{x}_i)) + (1 - y_i) \log(p(y_i = 0 \mid \mathbf{w}, \mathbf{x}_i)) \tag{5}$$

Plug in our logistic function for the probability that y is 1:

$$\ell(Y, X, \mathbf{w}) = \sum_{i=1}^{n} y_i \ \log(g(\mathbf{w}\mathbf{x}_i)) + (1 - y_i) \ \log(1 - g(\mathbf{w}\mathbf{x}_i))) \tag{6}$$

Recall that the link function, g, is the logistic function. It has the nice property $1 - g(z) = g(-z)$.

$$\ell(Y, X, \mathbf{w}) = \sum_{i=1}^{n} y_i \ \log(g(\mathbf{w}\mathbf{x}_i)) + (1 - y_i) \ \log(g(-\mathbf{w}\mathbf{x}_i)) \tag{7}$$

We can now use the chain rule to take the gradient with respect to $\mathbf{w}$:

$$\nabla\ell(Y, X, \mathbf{w}) = \sum_{i=1}^{n} y_i \ \frac{1}{g(\mathbf{w}\mathbf{x}_i)} \nabla g(\mathbf{w}\mathbf{x}_i) + (1 - y_i) \ \frac{1}{g(-\mathbf{w}\mathbf{x}_i)} \nabla g(-\mathbf{w}\mathbf{x}_i) \tag{8}$$

Since $\frac{\partial}{\partial z} g(z) = g(z)(1 - g(z))$:

$$\begin{aligned} \nabla\ell(Y, X, \mathbf{w}) \ &= \ \sum_{i=1}^{n} y_i \ \frac{1}{g(\mathbf{w}\mathbf{x}_i)} g(\mathbf{w}\mathbf{x}_i)(1 - g(\mathbf{w}\mathbf{x}_i)) \nabla \mathbf{w}\mathbf{x}_i & (9) \\ &\quad + (1 - y_i) \ \frac{1}{g(-\mathbf{w}\mathbf{x}_i)} g(-\mathbf{w}\mathbf{x}_i)(1 - g(-\mathbf{w}\mathbf{x}_i)) \nabla(-\mathbf{w}\mathbf{x}_i) & (10) \end{aligned}$$

Simplify again using $1 - g(z) = g(-z)$ and cancel terms

$$\nabla\ell(Y, X, \mathbf{w}) = \sum_{i=1}^{n} y_i g(-\mathbf{w}\mathbf{x}_i) \nabla \mathbf{w}\mathbf{x}_i + (1 - y_i) g(\mathbf{w}\mathbf{x}_i) \nabla(-\mathbf{w}\mathbf{x}_i) \tag{11}$$

You can now get the partial derivatives (components of the gradient) out of this gradient function by:

$$\frac{\partial}{\partial \mathbf{w}_j} \ell(Y, X, \mathbf{w}) = \sum_{i=1}^{n} y_i g(-\mathbf{w}\mathbf{x}_i) \ \mathbf{x}_{ij} + (1 - y_i) g(\mathbf{w}\mathbf{x}_i)(-\mathbf{x}_{ij}) \tag{12}$$

1.4 Learning Rate

In gradient descent, the update rule is $\mathbf{w}' = \mathbf{w} + \eta \nabla \ell(y_i, \mathbf{w}, \mathbf{x}_i)$. Choosing η intelligently is a non-trivial task, and there are many ways to choose it in the literature. In this assignment, we will use a constant η. By default it should be 0.01, but your code should allow the user to change it by setting a command line argument called **gd_eta**.

1.5 Feature selection using information gain

Often times there are irrelevant features in a given representation. In some cases, including these features in learning yields worse results than if they were excluded. Feature selection is a general technique that selects only a subset of the features to use for learning.

There are many ways to do feature selection. In this assignment, you will use information gain (IG) as a feature selection criteria.

You will use a command line parameter `num_features_to_select` that takes an integer value. When the option is provided you will first run feature selection to select the given number of features and then train the appropriate classifier using only those features. For example, if you get the argument `num_features_to_select` with a value of 10, then you will first select only 10 features and then only use those 10 features in learning. Note that this may result in having instances with no features since all of them were removed. In this case, you will not be able to use the example for training. At test time, you will end up with a value at the threshold (e.g. $y = 0.5$.) See the prediction rule below for details on how to translate a prediction into a binary output value.

To select features, first compute IG for all the features in the training data. After you have IG values for all features, select the `num_features_to_select` features to use for training that have the highest IG. Note that there is no reason to compute IG in test mode. Features that are unknown to the classifier at test time should be ignored by the classifier (receive 0 weight.)

To handle continuous features, select the mean of the feature in the training data as the threshold for IG calculation. Note that thresholding and IG are only used for feature selection. After you select `num_features_to_select` features from the data, use the original value of the features to perform your learning.

If `num_features_to_select` option is not provided, you will use all the features.

Recall that entropy is defined as:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i),$$

conditional entropy:

$$H(Y|X) = -\sum_{i=1}^{m} \sum_{j=1}^{n} p(y_i, x_j) \log \frac{p(y_i, x_j)}{p(x_j)}$$

and information gain:

$$IG(Y|X) = H(Y) - H(Y|X)$$

Remember that $H(Y)$ is a constant when comparing $IG(Y|X)$ for different values of X so it should be dropped for efficiency.

1.6 Offset Feature

None of the math above mentions an offset feature (bias feature), a $\mathbf{w}_0$, 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 $E[y] = 0$. While this may or may not be true, for this assignment you should assume that your data is centered. Do not include another feature that is always 1 (x_0) or weight ($\mathbf{w}_0$) for it.

1.7 Convergence

In real gradient descent, you must decide when you have 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. Usually people check that the L_p norm of your 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 **gd_iterations** which is *exactly* how many iterations you should run (not an upper bound). An iteration is a single pass over every training example. The default of **gd_iterations** should be 20.

1.8 Implementation Notes

1. Even though logistic regression predicts a probability that the label is 1, the output of your program should be binary. Round your solution based on whether the probability is greater than or equal to 0.5:

   $\hat{y}_{new} = 1$ if $p(y = 1|\mathbf{w}, \mathbf{x}) = g(\mathbf{wx}) = \frac{1}{1+e^{-\mathbf{wx}}} \geq 0.5$
   $\hat{y}_{new} = 0$ otherwise

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

1.9 How Your Code Will Be Called

To train a model we will call:

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

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

```
-gd_eta k         // sets the update rule for the gradient step, default = 0.01
-gd_iterations t  // sets the number of GD iterations, default = 20
-num_features_to_select // sets the number of features to select, uses
all the features by default. You will not calculate IG in the default case.
```

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

To make predictions using a model we will call:

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

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

Recall from previous assignments how to add a command line parameter. You can add options by adding the following code block to the `createCommandLineOptions` method of `Classify`.

```
registerOption("gd_eta", "int", true, "The step size parameter for GD.");
registerOption("gd_iterations", "int", true, "The number of GD iterations.");
registerOption("num_features_to_select", "int", true, "The number of features to select.");
```

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 gd_iterations = 20;
if (CommandLineUtilities.hasArg("gd_iterations"))
    gd_iterations = CommandLineUtilities.getOptionValueAsInt("gd_iterations");
int gd_eta = .01;
if (CommandLineUtilities.hasArg("gd_eta"))
    gd_eta = CommandLineUtilities.getOptionValueAsFloat("gd_eta");
int num_features;
if (CommandLineUtilities.hasArg("num_features_to_select"))
    num_features = CommandLineUtilities.getOptionValueAsInt("num_features_to_select");
```

2 Analytical (50 points)

1) Decision Tree and Logistic Regression (8 points) Consider a binary classification task (label y) with four features (x):

x_1	x_2	x_3	x_4	y
0	1	1	-1	1
0	1	1	1	0
0	-1	1	1	1
0	-1	1	-1	0

(a) Can this function be learned using a decision tree? If so, provide such a tree (describe each node in the tree). If not, prove it.

(b) Can this function be learned using a logistic regression classifier? If yes, give some example parameter weights. If not, why not.

(c) For the models above where you can learn this function, the learned model may over-fit the data. Propose a solution for each model on how to avoid over-fitting.

2) Stochastic Gradient Descent (8 points) In the programming part of this assignment you implemented Gradient Descent. A stochastic variation of that method (Stochastic Gradient Descent) takes an estimate of the gradient based on a single sampled example, and takes a step based on that gradient. This process is repeated many times until convergence. To summarize:

1. Gradient descent: compute the gradient over all the training examples, take a gradient step, repeat until convergence.
2. Stochastic gradient descent: sample a single training example, compute the gradient over that training example, take a gradient step, repeat until convergence.

In the limit, will both of these algorithms converge to the same optimum or different optimum? Answer this question for both convex and non-convex functions. Prove your answers.

3) Regularizer of Regression (10 points) In linear regression we want to minimize the sum of square loss

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} ||y - X\beta||_2^2 \tag{13}$$

To address overfitting, we might plug in a regularizer $||\beta||_q$ as:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} ||y - X\beta||_2^2 + \lambda||\beta||_q \tag{14}$$

where $||\cdot||_q$ is the q-norm and λ is the regularization parameter

(a) What is the effect on β when $q = 0$, $q = 1$ and $q = 2$? Explain why this is the case.

(b) What is the effect of λ in terms of variance/bias trade-off? How do we usually select a proper λ?

(c) Suppose each example has three features and the corresponding parameters are β_0, β_1 and β_2. If we formulate the regularizer as $||\beta_{\{0,1\}}||_2 + |\beta_2|$, where $\beta_{\{0,1\}}$ is a 2 dimensional vector containing the first two elements of β. Describe the effect of this regularizer.

4) Constructing Generalized linear models. (12 points) Generalized linear models (GLMs), especially logistic regression are heavily used by banks, credit card companies and insurance companies. Actually, when you apply for a credit card, banks may put your information into a logistic regression model to decide whether you are eligible.

(a) The GLMs are closely related to the exponential distribution family, which has the probability density/mass function $f(y;\theta)$ in the form

$$f(y;\theta) = h(y)e^{\eta(\theta)\cdot T(y) - A(\theta)}, \tag{15}$$

where h, η, T, A are some known functions.

Consider a classification or regression problem where we would like to predict the value of some random variable y as a function of x. To derive a GLM for this problem, we will make the following three assumptions about the conditional distribution of y given x and about our model:

1. $y|x;\theta \sim ExponentialFamily(\eta)$. I.e., given x and θ, the distribution of y follows some exponential family distribution, with parameter η.
2. Given x, our goal is to predict the expected value of $T(y)$ given x. In most of our examples, we will have $T(y) = y$, so this means we would like the prediction $h(x)$ output by our learned hypothesis h to satisfy $h(x) = E[y|x]$.

3. The natural parameter η and the inputs x are related linearly: $\eta = \theta^T x$

Derive the expression of logistic regression from the Bernoulli distribution:

$$h_\theta(x) = \frac{1}{1 + \exp(-\theta^T x)} \tag{16}$$

by following the above three assumptions.

(b) The GLMs often contain some transformation, which is non-linear such as the log-odds-ratio transformation in the logistic regression. Why do we still call them "linear"?

5) Convex Optimization (12 points) Jenny at Acme Inc. is working hard on her new machine learning algorithm. She starts by writing an objective function that captures her thoughts about the problem. However, after writing the program that optimizes the objective and getting poor results, she returns to the objective function in frustration. Turning to her colleague Matilda, who took CS 475 at Johns Hopkins, she asks for advice. "Have you checked that your function is convex?" asks Matilda. "How?" asks Jenny.

(a) Jenny's function can be written as $f(g(x))$, where $f(x)$ and $g(x)$ are convex. Prove that $f(g(x))$ is a convex function. (Hint: You may find it helpful to use the definition of convexity. Do not use gradient or Hessian, since f and g may not have them.)

(b) Jenny realizes that she made an error and that her function is instead $f(x) - g(x)$, where $f(x)$ and $g(x)$ are convex functions. Her objective may or may not be convex. Give examples of functions $f(x)$ and $g(x)$ whose difference is convex, and functions $\bar{f}(x)$ and $\bar{g}(x)$ whose difference is non-convex.

"I now know that my function is non-convex," Jenny says, "but why does that matter?"

(c) Why was Jenny getting poor results with a non-convex function?

(d) One approach for convex optimization is to iteratively compute a descent direction and take a step along that direction to have a new value of the parameters. The choice of a proper stepsize is not so trivial. In gradient descent algorithm, the stepsize is chosen such that it is proportional to the magnitude of the gradient at the current point. What might be the problem if we fix the stepsize to a constant regardless of the current gradient? Discuss when stepsize is too small or too large.

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.