Homework 3, Due 11:59 p.m., April 28

MPCS 53111 Machine Learning, University of Chicago

Practice problems, do not submit

P-1 The cost function[1] for logistic regression with L_2 regularization is (using Andrew Ng's MLE method and notation (see Section 5)) —

$$-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log h_\theta(\mathbf{x}^{(i)}) + (1-y^{(i)})\log(1-h_\theta(\mathbf{x}^{(i)})\right) \quad + \lambda\sum_{j=1}^{n}\theta_j^2,$$

in which λ is the strength of regularization and $h_\theta(\mathbf{x}^{(i)}) = \frac{1}{1+e^{-\theta^T\mathbf{x}^{(i)}}}$. Show that the corresponding update rule for gradient descent is —

$$\theta_j \leftarrow \theta_j + \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - h_\theta(\mathbf{x}^{(i)})\right)x_j^{(i)} \quad - 2\lambda\theta_j\right].$$

P-2 Solve the following by hand, showing your calculations clearly.

x_1	x_2	Class Label
1	1	0
2	2	0
3	4	0
2	4.5	1
3	6	1

(a) Roughly plot the position of the examples along x_1 and x_2 and choose a simple decision boundary such that all the five training examples are correctly classified.

(b) Compute by hand the parameters of a logistic regression model that correspond to the above decision boundary.

(c) Determine the probability of the 0 class for the first and the last training examples in the table (at (1,1) and (3,6)) according to the your logistic regression model.

[1] Russell-Norvig considers square error, not MLE, and has a different cost function.

Graded problems, submit

1. Run the perceptron learning rule, by hand, on the dataset in Problem P-2 (go through the examples in the order shown), initializing $\mathbf{w}$ to $(0, 0, 1)$, and using a learning rate α of 0.5 (see Russell-Norvig (18.7) on page 724). Stop when the corresponding perceptron classifies all examples correctly. State clearly your final weights.

2. Often a dataset that is not linearly separable in the original input space becomes linearly separable when we include polynomial functions of the original inputs, such as x_1^2 or $x_1 x_2$, as additional inputs. The following problems illustrates this. Solve it by hand, showing your calculations clearly.

x_1	x_2	Class Label
0.5	0.5	+
−0.5	0.5	+
−0.5	−0.5	+
2	2	−
2	−2	−
−2	2	−
−2	−2	−

   (a) Roughly plot the position of the examples along x_1 and x_2 and choose a simple—not necessarily linear—decision boundary such that all the seven training examples are correctly classified.

   (b) Compute by hand the parameters of a logistic regression model that corresponds to the above decision boundary. (Hint: For features use x_1^2, x_2^2.)

   (c) Determine the probability of the + class for the first and the last training examples in the table according to the your logistic regression model.

3. To extend logistic regression to more than two classes, a set of $\mathbf{w}_k$ parameter weight vectors is learnt for each class k such that the probability $P(Y = k \mid \mathbf{x})$ is estimated by

$$\frac{e^{\mathbf{w}_k^T \mathbf{x}}}{\sum_{j \in \text{classes}} e^{\mathbf{w}_j^T \mathbf{x}}}.$$

The predicted class $\hat{y}$ is $\text{argmax}_k P(Y = k \mid \mathbf{x})$. See, e.g, Wikipedia or this Stanford tutorial. Prove that this is a linear classifier, i.e., the class boundaries are linear functions of the input attributes in $\mathbf{x}$.

4. Implement a python function `gradient_descent(X, y, alpha, lambda, T)` that takes an input `numpy` matrix `X` of shape (m,n), a output vector `y` of shape (m,), a scalar learning rate `alpha`, a regularization strength parameter `lambda`, the number of iterations `T`, and returns a vector `theta` of shape (n+1,1). `theta` should be the logistic regression parameter vector θ found by executing the gradient descent algorithm for `T` iterations on the given inputs. The function should also plot the value of the cost (loss + complexity) function vs the iteration number. Use the cost function, with L_2 regularization, from Problem P-1.

   It helps to internally normalize the inputs by ensuring each attribute has mean 0 and standard deviation 1. But return the `theta` values corresponding to the original, unnormalized inputs.

   Make reasonable assumptions about other design choices. You should find it useful to watch the two videos on "Gradient Descent in Practice" on Coursera. Enroll in the course for free to access the videos. In fact, watching all videos related to gradient descent in Weeks 1–3 is recommended. Also while developing, it is best to test your function on a toy dataset with few examples and just a couple of features where you have worked out what should happen at each step and so can debug your code efficiently.

5. Use the above implemenation to fit a logistic regression model on the accompanying breast cancer data, with 30 input variables, and two classes. State the θ values you obtain, and the values of the learning rate α and regularization strength λ you used. Further, submit a plot of the cost function vs the iteration number. The data is the Wisconsin Breast Cancer Dataset. (There is a research article in the accompanying files that describes how this dataset was obtained, which is interesting reading,[2] but not required for this homework.)

6. Fit a logistic regression model on the above breast cancer data using scikit-learn's `LogisticRegression` and state the θ values it returns.

[2] The article by Mangasarian et al. describes a linear programming based customized approach to build a model.

Choose settings to mimic your run in Problem 5 as closely as possible. If the returned θ here is very different, determine why or debug your gradient descent code. Describe reasons for difference in θ values.

7. Fit a logistic regression model on the just the features `mradius` and `mtexture` using scikit-learn's `LogisticRegresion`, but include terms up to degree 3—such as $\texttt{mradius}^3$ and $\texttt{mradius} \times \texttt{mtexture}^2$. Draw a scatter plot in which the malignant cases are in red, the benign cases are in green, and the decision boundary corresponding to your model is in blue. You'll find sample code helpful for drawing such plots in the accompanying files.

Submitting your work

- Submit your homework as a single `.py` and a single `.pdf` file.
- The `.py` file should have functions named, e.g., `run_prob_5()` which when called should run the code required in Problem 5, with possibly a default setting, and produce plot files or print required output. (Copies of plots and output should also be included in your `.pdf` file.)
- Do not submit Jupyter notebooks or scripts not collected into functions.
- The `.pdf` file should contain all answers to theory problems—which may be scans of neatly handwritten solutions—as well as required plots and any results, discussions, or comments related to the programming problems.

Optional reading and problems; do not submit

Opt-1 There is substantial statistics-based work in evaluating the parameter estimators of a linear model, including techniques for determining related confidence intervals and hypothesis testing. Please read Section 3.1 of ISL (pages 61–71), and answer the following.

(a) Briefly describe RSE and R^2 and compare the advantage of each over the other.

(b) Derive the relationship $R^2 = r^2$, when there is a single input variable (simple linear regression).