CMSC 25400 — Homework 3

University of Chicago, Winter 2019 — due Tuesday, February 29 at 11:59pm

1. In class we have derived an explicit formula for least squares regression when the hypothesis class is the class of linear functions $h(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots \theta_d x_d$. However, not all data can be well modeled by just a linear equation. An alternative, generally richer hypothesis class can be defined by defining a set of n basis functions $\{\phi_1(\mathbf{x}), \ldots, \phi_n(\mathbf{x})\}$, and considering regressors of the form

$$h(\mathbf{x}) = \sum_{i=1}^{n} \theta_i \, \phi_i(\mathbf{x}).$$

Now each θ_i parameter is the coefficient of the corresponding ϕ_i in the linear combination of basis functions. The $\{\phi_i\}$ can really be any collection of functions, but as an illustration in the $d = 1$ case you might consider something like a family of Gaussian bumps

$$\phi_i = e^{(x-i)^2/(2\sigma^2)} \qquad\qquad i = 1, 2, \ldots, n.$$

In principle one could also consider a infinite sequence of basis functions, but for simplicity here we only consider a finite set. As before, let the loss function be the squared error loss

$$J(\boldsymbol{\theta}) = \frac{1}{2} \sum_{j=1}^{m} (h(\mathbf{x}_j) - y_j)^2.$$

Show that similarly to the linear case, the optimal solution can be found in the form

$$\boldsymbol{\theta} = (A^\top A)^{-1} A^\top \vec{\mathbf{y}},$$

where $\vec{\mathbf{y}} = (y_1, \ldots, y_m)^\top$, and derive the form of the matrix A. This problem illustrates that the least squares technique (including its SGD version) has much broader applicability than just linear regression.

2. In class we have derived that given data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$, the log-likelihood for logistic regression is

$$\ell(\theta) = \sum_{i=1}^{m} \big[u_i \log(h(\mathbf{x}_i)) + (1 - u_i) \log(1 - h(\mathbf{x}_i)) \big], \tag{1}$$

where $h(\mathbf{x})$ is the logistic function

$$h(\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta} \cdot \mathbf{x}}} = g(\boldsymbol{\theta} \cdot \mathbf{x}) \qquad\qquad g(z) = \frac{1}{1 + e^{-z}},$$

and the u_i's are just the 0/1 analogs of the y_i's, i.e., $u_i = (1 + y_i)/2$. There is no closed form solution for the MLE of logistic regression.

   (a) For simplicity consider (1) for a single data point $(\mathbf{x}, u)$. Derive the form of the gradient $\nabla \ell(\theta)$. The formula $g'(z) = g(z)\,(1 - g(z))$ that we found in class might come in handy.

   (b) Conclude that the SGD step based on a single datapoint $(\mathbf{x}_i, u_i)$ in the dataset is

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \big[(h(\mathbf{x}_i) - u_i)\, \mathbf{x}_i \big].$$

3. An online algorithm is said to be conservative if it changes its hypothesis only when it makes a mistake. Let $\mathcal{C}$ be a concept class and A be a (not necessarily conservative) online algorithm which has a finite mistake bound M on $\mathcal{C}$. Prove that there is a conservative algorithm A' for $\mathcal{C}$ which also has mistake bound M.

4. Recall that the k–class perceptron maintains k separate weight vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k$, and predicts

$$\widehat{y} = \arg \max_{i \in \{1,2,\ldots,k\}} (\mathbf{w}_i \cdot \mathbf{x}).$$

If this prediction is incorrect, and the correct label should have been y, it updates the weights by setting

$$\mathbf{w}_y \leftarrow \mathbf{w}_y + \mathbf{x}/2$$
$$\mathbf{w}_{\widehat{y}} \leftarrow \mathbf{w}_{\widehat{y}} - \mathbf{x}/2.$$

Let $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots\}$ be the training data. Assume that $\| \mathbf{x}_t \| = 1$ for all t, and that this dataset is separable with a margin δ, which in this case means that there exist unit vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ such that for each example $(\mathbf{x}_t, y_t)$

$$\mathbf{v}_{y_t} \cdot \mathbf{x}_t - \mathbf{v}_{\overline{y}} \cdot \mathbf{x}_t \geq 2\delta \qquad \overline{y} \in \{1, 2, \ldots, k\} \setminus \{y_t\}.$$

(a) Show that in the $k = 2$ case this notion of margin is equivalent to the margin that we saw in class.

(b) In the $k = 2$ case we saw that the number of mistakes that the perceptron can make is upper bounded by $1/\delta^2$. Derive a similar bound for the $k = 3$ case. Hint: Two quantities that you may wish to consider are $a = \mathbf{v}_1 \cdot \mathbf{w}_1 + \mathbf{v}_2 \cdot \mathbf{w}_2 + \mathbf{v}_3 \cdot \mathbf{w}_3$ and $b = \| \mathbf{w}_1 \|^2 + \| \mathbf{w}_2 \|^2 + \| \mathbf{w}_3 \|^2$. Part of your derivation might involve showing that $a \leq 3\sqrt{b}$.

5. The file `train35.digits` contains 2000 images of 3's and 5's from the famous MNIST database of handwritten digits in text format. The size of each image is 28×28 pixels. Each row of the file is a representation one image, with the 28×28 pixels flattened into a vector of size 784. A value of 1 for a pixel represents black, and value of 0 represents white. The corresponding row of `train35.labels` is the class label: $+1$ for the digit 3, or -1 for the digit 5. The file `test35.digits` contains 200 testing images in the same format as `train35.digits`.

Implement the perceptron algorithm and use it to label each test image in `test35.digits`. Submit the predicted labels in a file named `test35.predictions`. In the lectures, the perceptron was presented as an online algorithm. To use the perceptron as a batch algorithm, train it by simply feeding it the training set M times. The value of M can be expected to be less than 10, and should be set by cross validation. Naturally, in this context, the "mistakes" made during training are not really errors. Nonetheless, it is intructive to see how the frequency of mistakes decreases as the hypothesis improves. Include in your write-up a plot of the cumulative number of "mistakes" as a function of the number of examples seen.

Since the data is fairly large, for debugging purposes it might be helpful to run your code on just subsets of the 2000 training test images. Also, it may be helpful to normalize each example to unit norm.