10-701 Introduction to Machine Learning

Homework 2, version 1.0

Due Oct 16, 11:59 am

Rules:

1. Homework submission is done via CMU Autolab system. Please package your writeup and code into a zip or tar file, *e.g.*, let `submit.zip` contain `writeup.pdf` and `ps2_code/*.m`. Submit the package to https://autolab.cs.cmu.edu/courses/10701-f15.
2. Like conference websites, repeated submission is allowed. So please feel free to refine your answers. We will only grade the latest version.
3. Autolab may allow submission after the deadline, note however it is because of the late day policy. Please see course website for policy on late submission.
4. We recommend that you typeset your homework using appropriate software such as LaTeX. If you are writing please make sure your homework is cleanly written up and legible. The TAs will not invest undue effort to decrypt bad handwriting.
5. You are allowed to collaborate on the homework, but you should write up your own solution and code. Please indicate your collaborators in your submission.

---

1 Bayes Optimal Classification (20 Points) (Yan)

In classification, the loss function we usually want to minimize is the 0/1 loss:

$$\ell\left(f(x), y\right) = \mathbf{1}\left\{f(x) \neq y\right\} \tag{1}$$

where $f(x), y \in \{0, 1\}$ (*i.e.*, binary classification). In this problem we will consider the effect of using an asymmetric loss function:

$$\ell_{\alpha,\beta}\left(f(x), y\right) = \alpha\mathbf{1}\left\{f(x) = 1, y = 0\right\} + \beta\mathbf{1}\left\{f(x) = 0, y = 1\right\}. \tag{2}$$

Under this loss function, the two types of errors receive different weights, determined by $\alpha, \beta > 0$.

1. Determine the Bayes optimal classifier, *i.e.*, the classifier that achieves minimum risk assuming $P(x, y)$ is known, for the loss $\ell_{\alpha,\beta}$ where $\alpha, \beta > 0$.

2. Suppose that the class $y = 0$ is extremely uncommon (*i.e.*, $P(y = 0)$ is small). This means that the classifier $f(x) = 1$ for all x will have good risk. We may try to put the two classes on even footing by considering the risk:

   $$R = P\left(f(x) = 1 \mid y = 0\right) + P\left(f(x) = 0 \mid y = 1\right). \tag{3}$$

   Show how this risk is equivalent to choosing a certain α, β and minimizing the risk where the loss function is $\ell_{\alpha,\beta}$.

3. Consider the following classification problem. I first choose the label $Y \sim \mathrm{Ber}\left(\frac{1}{2}\right)$, which is 1 with probability $\frac{1}{2}$. If $Y = 1$, then $X \sim \mathrm{Ber}\left(p\right)$; otherwise, $X \sim \mathrm{Ber}\left(q\right)$. Assume that $p > q$. What is the Bayes optimal classifier, and what is its risk?

4. Now consider the regular 0/1 loss ℓ, and assume that $P(y = 0) = P(y = 1) = \frac{1}{2}$. Also, assume that the class-conditional densities are Gaussian with mean μ_0 and co-variance Σ_0 under class 0, and mean μ_1 and co-variance Σ_1 under class 1. Further, assume that $\mu_0 = \mu_1$.

   For the following case, draw contours of the level sets of the class conditional densities and label them with $p(x \mid y = 0$ and $p(x \mid y = 1)$. Also, draw the decision boundaries obtained using the Bayes optimal classifier in each case and indicate the regions where the classifier will predict class 0 and where it will predict class 1.

   $$\Sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}. \tag{4}$$

2 Regularized Linear Regression Using Lasso (20 Points) (Yan)

Lasso is a form of regularized linear regression, where the L1 norm of the parameter vector is penalized. It is used in an attempt to get a sparse parameter vector where features of little "importance" are assigned to zero weight. But why does lasso encourage sparse parameters? For this question, you are going to examine this.

Let $\mathbf{X}$ denote an $n \times d$ matrix where rows are training points, $\mathbf{y}$ denotes an $n \times 1$ vector of corresponding output value, $\mathbf{w}$ denotes a $d \times 1$ parameter vector and $\mathbf{w}^\star$ denotes the optimal parameter vector. To make the analysis easier we will consider the special case where the training data is whitened (*i.e.*, $X^\top X = I$). For lasso regression, the optimal parameter vector is given by

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}\left\|\mathbf{y} - X\mathbf{w}\right\|^2 + \lambda\left\|\mathbf{w}\right\|_1, \tag{5}$$

where $\lambda > 0$.

1. Show that whitening the training data nicely decouples the features, making $\mathbf{w}_i^\star$ determined by the ith feature and the output regardless of other features. To show this, write $J_\lambda(\mathbf{w})$ in the form

$$J_\lambda(\mathbf{w}) = g(\mathbf{y}) + \sum_{i=1}^{d} f\left(X_{\cdot i}, \mathbf{y}, w_i, \lambda\right), \tag{6}$$

   where $X_{\cdot i}$ is the ith column of X.

2. Assume that $w_i^\star > 0$, what is the value of $w_i^\star$ in this case?

3. Assume that $w_i^\star < 0$, what is the value of $w_i^\star$ in this case?

4. From 2 and 3, what is the condition for $w_i^\star$ to be 0? How can you interpret that condition?

5. Now consider ridge regression where the regularization term is replaced by $\frac{1}{2}\lambda \|\mathbf{w}\|_2^2$. What is the condition for $w_i^\star = 0$? How does it differ from the condition you obtained in 4?

3 Multinomial Logistic Regression (20 Points) (Yan)

Multinomial logistic regression is a classification method that generalizes logistic regression to multiclass problems. It has the form

$$p\left(y = c \mid \mathbf{x}, \mathbf{W}\right) = \frac{\exp\left(w_{c0} + \mathbf{w}_c^\top \mathbf{x}\right)}{\sum_{k=1}^{C} \exp\left(w_{k0} + \mathbf{w}_k^\top \mathbf{x}\right)}, \tag{7}$$

where C is the number of classes, and $\mathbf{W}$ is a $C \times (d+1)$ weight matrix, and d is the dimension of input vector $\mathbf{x}$. In other words, $\mathbf{W}$ is a matrix whose rows are the weight vectors for each class.

1. Show that in the special case where $C = 2$, the multinomial logistic regression reduces to logistic regression.

2. In the training process of the the multinomial logistic regression model, we are given a set of training data $\{\mathbf{x}_i, y_i\}_{i=1}^n$, and we want to learn a set of weight vectors that maximize the conditional likelihood of the output labels $\{y_i\}_{i=1}^n$, given the input data $\{\mathbf{x}_i\}_{i=1}^n$ and $\mathbf{W}$. That is, we want to solve the following optimization problem (assuming the data points are *i.i.d*).

$$\mathbf{W}^\star = \underset{\mathbf{W}}{\operatorname{argmax}} \prod_{i=1}^{n} P(y_i \mid \mathbf{x}_i, \mathbf{W}) \tag{8}$$

   In order to solve this optimization problem, most numerical solvers require that we provide a function that computes the objective function value given some weight and the gradient of that objective function (*i.e.*, its first derivative). Some solvers usually also require the Hessian of the objective function (*i.e.*, its second derivative). So, in order to implement the algorithm we need to derive these functions.

   (a) Derive the conditional log-likelihood function for the multinomial logistic regression model. You may denote this function as $\ell(W)$.

   (b) Derive the gradient of $\ell(W)$ with respect to the weight vector of class c ($\mathbf{w}_c$). That is, derive $\nabla_{\mathbf{w}_c} \ell(W)$. You may denote this function as gradient $g_c(W)$. Note: The gradient of a function $f(\mathbf{x})$ with respect to a vector $\mathbf{x}$ is also a vector, whose i-th entry is defined as $\frac{\partial f(\mathbf{x})}{\partial x_i}$, where x_i is the i-th element of $\mathbf{x}$.

   (c) Derive the block submatrix of the Hessian with respect to weight vector of class c ($\mathbf{w}_c$) and class c' ($\mathbf{w}_{c'}$). You may denote this function as $H_{c,c'}(\mathbf{W})$. Note: The Hessian of a function $f(\mathbf{x})$ with respect to vector $\mathbf{x}$ is a matrix, whose $\{i,j\}$th entry is defined as $\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}$. In this case, we are asking a block submatrix of Hessian of the conditional log-likelihood function, taken with respect to only two classes c and c'. The $\{i,j\}$th entry of the submatrix is defined as $\frac{\partial^2 \ell(\mathbf{W})}{\partial w_{ci} \partial w_{c'j}}$.

4 Perceptron Mistake Bounds (20 Points) (Xun)

Suppose $\{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in \mathbb{R}^n, y_i \in \{+1, -1\}, i = 1, \ldots, m\}$ can be linearly separated by a margin $\gamma > 0$, *i.e.*

$$\exists \mathbf{w} \in \mathbb{R}^n \text{ s.t. } \|\mathbf{w}\|_2 = 1, \ \langle y_i \mathbf{x}_i, \mathbf{w} \rangle \geq \gamma, \ \forall i = 1, \ldots, m, \tag{9}$$

where $\langle a, b \rangle = a^\top b$ is the dot product between two vectors. Further assume $\|\mathbf{x}_i\|_2 \leq M, \ \forall i$. Recall that Perceptron algorithm starts from $\mathbf{w}^{(0)} = \mathbf{0}$ and updates $\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + y^{(t)}\mathbf{x}^{(t)}$, where $(\mathbf{x}^{(t)}, y^{(t)})$ is the t-th misclassified example. We will prove that Perceptron learns an exact classifier in finitely many steps.

1. Show that $\left\langle \mathbf{w}^{(t)}, \mathbf{w} \right\rangle \geq t\gamma$.
2. Show that $\|\mathbf{w}^{(t)}\|_2^2 \leq tM^2$.
3. Use the results above, show that number of updates t is upper bounded by M^2/γ^2.
4. True or False: when zero error is achieved, the classifier always has margin γ. Explain briefly.

5 Logistic Regression for Image Classification (20 Points) (Xun)

In this problem, we will implement (almost) from scratch logistic regression for image classification. We will use MNIST (http://yann.lecun.com/exdb/mnist/), which contains in total 70,000 handwritten digits from 0 to 9. Although initially released in 1998, MNIST is still one of the most widely used benchmark data sets for image classification.

5.1 Exploring the data

It is often a good practice to take a careful look at the data before modeling. Run `download_mnist.sh` and `visualize_mnist.m`, and explore the following properties of `images` and `labels`:

1. size of each image
2. range of labels
3. range of pixel values
4. maximum and minimum ℓ_2-norm of the images
5. whether the data is sparse or dense
6. whether the label distribution is skewed or uniform

Please append your code to `visualize_mnist.m`.

5.2 Binary logistic regression

Let's start with a simple binary logistic regression for classifying ONLY between the digits 3 and 8. In `lr.m` we have outlined the training and testing process, as well as some preprocessing routines, such as selecting 3's and 8's and normalizing data to have zero mean and unit variance. Your goal is to perform the following steps:

1. Complete the function `s = sigmoid(a)`. Note that `a` and `s` could be vectors.
2. Complete the function `[f, g] = oracle_lr(w, X, y)`, where w is the weight vector, X is the set of images, and y is the set of labels. This function returns the objective f and the gradient g.

3. Complete the function `err = grad_check(oracle, t)`. This will help you check if the oracle implementation is correct. First recall the definition of derivatives:

$$g(t) = \lim_{h \to 0} \frac{f(t+h) - f(t)}{h}. \tag{10}$$

The idea is to check analytic gradients against numerical gradients. For a small h, say $h \approx 10^{-6}$, the numerical estimate

$$\widehat{g}_j(t) = \frac{f(t + h\mathbf{e}_j) - f(t - h\mathbf{e}_j)}{2h} \approx g_j(t), \tag{11}$$

for all $j \in \{1, \ldots, d\}$, where $\mathbf{e}_j$ is the unit vector at j-th coordinate. If the oracle is implemented correctly, then we should expect small (*e.g.* $\approx 10^{-6}$) average error

$$\text{err} = \frac{1}{d} \sum_{j=1}^{d} \left| \widehat{g}_j(t) - g_j(t) \right|. \tag{12}$$

Try running `oracle_lr_test.m` to see if your oracle can pass the test.

4. Complete the function `w = optimize_lr(w0, X, y)`, where w_0 is the initial value. You will implement a simple gradient descent/ascent algorithm to find the best parameter w.

5. Complete the function `acc = binary_accuracy(w, X, y)`. This function returns the fraction of correct predictions of classifier w on data X.

6. Run `lr.m`, report the number of iterations, final objective function value, final $\|w\|_2^2$, training accuracy, and test accuracy. You should be able to get $\geq 95\%$ test accuracy.

7. Modify `oracle_lr.m` to return ℓ_2-regularized objective and gradient, with tuning parameter λ. Specifically, add/subtract $\frac{\lambda}{2} \|w\|_2^2$ to the objective, depending on whether the objective is log-likelihood or negative log-likelihood. Report the number of iterations, final objective function value, final $\|w\|_2^2$, training accuracy, and test accuracy. Briefly summarize your observation.

   (Note: you can check your implementation again with `oralce_lr_test.m`.)

5.3 Multiclass logistic regression

Now let's learn a classifier for ALL digits using multiclass logistic regression derived above. Again we have the algorithm outline and preprocessing in `mlr.m`. Notice that the labels are now shifted by 1, so that they are 1-indexed. Your goal is to perform the following steps:

1. Complete the function `[f, g] = oracle_mlr(W0, X, y)`. In particular, implement ℓ_2-regularization for each class, again with λ being the tuning parameter, *i.e.* include $\frac{\lambda}{2} \|\mathbf{W}\|_F^2$ in your objective.

   (Note: you can check your implementation with `oralce_mlr_test.m`.)

2. Complete the function `W = optimize_mlr(W0, X, y)`.

3. Complete the function `acc = multiclass_accuracy(W, X, y)`.

4. Run `mlr.m`, report the number of iterations, final objective function value, final $\|\mathbf{W}\|_F^2$, training accuracy, and test accuracy. Also include the visualization of the learned weights. For this task you should be able to get $\geq 92\%$ test accuracy.