6.867 Machine Learning

Problem Set 2

Due date: Monday October 6

Please address all questions and comments about this problem set to `6.867-staff@ai.mit.edu`. You will need to use MATLAB for some of the problems but most of the code is provided. If you are not familiar with MATLAB, please consult

`http://www.ai.mit.edu/courses/6.867/matlab.html`

and the links therein.

Problem 1: active regression

Reference: Lecture three; Chapter 5-5.4.1

Let's start by defining our model and assumptions. We use additive regression models with fixed basis functions $\{\phi_i(\mathbf{x})\}_{i=1,\ldots,m}$ so that

$$f(\mathbf{x}; w) = \phi(\mathbf{x})' w = \sum_{i=1}^{m} \phi_i(\mathbf{x}) w_i \tag{1}$$

where the first basis function, $\phi_1(\mathbf{x})$, is typically a constant. The basis functions may be of various different types. They may, for example, simply return specific components of the input vector as in $\phi_i(\mathbf{x}) = x_i$ reducing the above model to a linear model in the input space. Alternatively, the basis functions could measure similarities to "prototypes" as in radial basis functions:

$$\phi_i(\mathbf{x}) = \exp\left\{ -\frac{1}{2s^2} \|\mathbf{x} - \mu_i\|^2 \right\} \tag{2}$$

where μ_i specifies a point in the input space (e.g., representing a cluster or an individual training point). The parameter s specifies the spread around μ_i, i.e., how quickly the basis function goes to zero as we deviate from the prototype.

The observed outputs y are assumed to be related to the inputs x through some "*true*" weights $w^* = (w_1^*, \ldots, w_m^*)'$, and independently corrupted with zero mean Gaussian noise with variance σ^2. We do not know a priori what these weights w^* are but only that they exist. This is nevertheless a strong assumption. Our class of statistical models, linear

models with a specific set of basis functions, is assumed to contain the true model. We will look at examples of what happens when the assumption is not correct.

Now, for any n input points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we model the corresponding outputs $\{y_1, \ldots, y_n\}$ as

$$\begin{bmatrix} y_1 \\ \cdots \\ y_n \end{bmatrix} = \begin{bmatrix} \phi_1(\mathbf{x}_1) & \cdots & \phi_m(\mathbf{x}_1) \\ \cdots & \cdots & \cdots \\ \phi_1(\mathbf{x}_n) & \cdots & \phi_m(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} w_1 \\ \cdots \\ w_m \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \cdots \\ \epsilon_n \end{bmatrix} \tag{3}$$

$$\mathbf{y} = \mathbf{X}w + \mathbf{e} \tag{4}$$

where $\mathbf{e} \sim N(\mathbf{0}, \sigma^2 I)$. The overall noise variance σ^2 is unknown but largely insignificant for our purposes here. The value of σ^2 will certainly affect any measure of uncertainty we will use about the model parameters or predictions but has little effect on our decisions concerning which points we should query next. We define $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x}))'$ as the *feature vector* (column vector) corresponding to an input point $\mathbf{x}$, and therefore write the matrix $\mathbf{X}$ as $(\phi(\mathbf{x}_1) \cdots \phi(\mathbf{x}_n))'$.

We already know from lectures that given $\mathbf{X}$, i.e., given the input points, the noise in the observed outputs cause our parameter estimates $\hat{w}$ to be Gaussian random variables: $\hat{w} \sim N(w^*, \sigma^2(\mathbf{X}'\mathbf{X})^{-1})$. In other words, if we repeatedly requested a new set of outputs corresponding to the input points, and estimated $\hat{w}$ based on each such sample, then the resulting estimates $\hat{w}$ would appear as points drawn from a Gaussian distribution with mean w^* and covariance $\sigma^2(\mathbf{X}'\mathbf{X})^{-1})$.

For simplicity (and with no real loss of generality), we will assume hereafter that $\sigma^2 = 1$. It will be also helpful to define the inverse covariance matrix $A_n = (\mathbf{X}'\mathbf{X})$, where the subscript indicates the number of input points.

Our predictions $\hat{y}(\mathbf{x}) = f(\mathbf{x}, \hat{w}) = \phi(\mathbf{x})'\hat{w}$ are *unbiased* in the sense that

$$E\{\, \hat{y}(\mathbf{x}) \,\} = E\{\, \phi(\mathbf{x})'\hat{w} \,\} = \phi(\mathbf{x})'E\{\, \hat{w} \,\} = \phi(\mathbf{x})'w^* = y^*(\mathbf{x}) \tag{5}$$

where the expectation is over the noise in the outputs (uncertainty in the parameter estimates $\hat{w}$).

(1-1) (5pts) Show that the variance of the predictions at any specific point $\mathbf{x}$ is given by $\mathrm{Var}(\, \hat{y}(\mathbf{x}) \,) = \phi(\mathbf{x})'(\mathbf{X}'\mathbf{X})^{-1}\phi(\mathbf{x})$, where we continue to assume that $\sigma^2 = 1$.

We consider here a simple variant of the active learning problem where we know the distribution, $Q(\mathbf{x})$, over the inputs $\mathbf{x}$ that we will be tested on. For example, in practice, we might know the dataset of test examples for which we need to generate predictions. The goal in this case is to minimize the average variance of our predictions

$$J(\mathbf{X}) = E_{\mathbf{x}\sim Q}\{\, \mathrm{Var}(\, \hat{y}(\mathbf{x}) \,) \,\} \tag{6}$$

$$= E_{\mathbf{x}\sim Q}\left\{\, \phi(\mathbf{x})'(\mathbf{X}'\mathbf{X})^{-1}\phi(\mathbf{x}) \,\right\} \tag{7}$$

$$= E_{\mathbf{x}\sim Q}\left\{ \text{trace}\left[\phi(\mathbf{x})'(\mathbf{X}'\mathbf{X})^{-1}\phi(\mathbf{x})\right]\right\} \quad (8)$$

$$= E_{\mathbf{x}\sim Q}\left\{ \text{trace}\left[(\mathbf{X}'\mathbf{X})^{-1}\phi(\mathbf{x})\phi(\mathbf{x})'\right]\right\} \quad (9)$$

$$= \text{trace}\left[(\mathbf{X}'\mathbf{X})^{-1}E_{\mathbf{x}\sim Q}\left\{\,\phi(\mathbf{x})\phi(\mathbf{x})'\,\right\}\right] \quad (10)$$

where we have used the fact that $\text{trace}(AB) = \text{trace}(BA)$ whenever both matrix products make sense. The trace is defined as the sum over the diagonal elements: $\text{trace}(A) = \sum_i A_{ii}$. The trace of a real number is just the number since the number can be viewed as a 1x1 matrix. This is how we introduced the trace in the above derivation.

Note that the expectation with respect to Q is over the test points whereas the variance is evaluated with respect to the outputs corresponding to the existing training inputs. How should we select the input points to minimize this criterion? Perhaps as random samples from $Q(\mathbf{x})$? We need a few steps to get to the answer.

For simplicity we'll assume that the input is one dimensional, $x \in [-1, 1]$, and $\phi(x) = (1, x)'$. We will also try to select all the training inputs $x_1, \ldots, x_n$ at once rather than picking them sequentially.

(1-2) (5pts) Let A_n be the inverse covariance matrix based on n training examples.

a) Show that

$$A_n = \sum_{i=1}^{n} \phi(x_i)\phi(x_i)' \quad (11)$$

Make sure you understand that A_n is symmetric and positive (semi-)definite for all choices of $x_1, \ldots, x_n$.

b) Write A_n explicitly in terms of n, $\sum_i x_i$, and $\sum_i x_i^2$.

(1-3) (5pts) Let $Q(x)$ be any symmetric distribution around zero with variance v^2. For example, $N(0, v^2)$ would qualify. Evaluate $E_{x\sim Q}\left\{\,\phi(x)\phi(x)'\,\right\}$ and the resulting objective $J(\mathbf{X})$.

(1-4) (10pts) Show that we would still choose all the input points to be $x = -1$ or $x = 1$ with equal proportions (assuming n is even). The result holds even as $v^2 \to 0$, i.e., even when we only care about predictions at $x = 0$.

(*Hint:* use the fact that the input points would have to be in a symmetric arrangement around zero. What can you say about $\sum_i x_i$ in this case?).

(1-5) (optional) Can you find any distribution $Q(x)$ to avoid selecting $x = -1$ and $x = 1$ equally often?

Problem 2: active parameter estimation

In this problem we consider a simple conditional model of the form $P(y|x)$ where $x \in \{0,1\}$ and $y \in \{0,1\}$. We parameterize this model by two parameters $\theta_{1|0} = P(1|0)$ and $\theta_{1|1} = P(1|1)$ so that the conditional probability matrix is of the form

$$P(y|x,\theta): \quad \begin{pmatrix} P(0|0) & P(0|1) \\ P(1|0) & P(1|1) \end{pmatrix} = \begin{pmatrix} 1-\theta_{1|0} & 1-\theta_{1|1} \\ \theta_{1|0} & \theta_{1|1} \end{pmatrix} \tag{12}$$

Our model does not specify $P(x)$.

Let's start by finding the maximum likelihood setting of the parameter $\theta = (\theta_{1|0}, \theta_{1|1})'$, setting that maximizes the conditional log-likelihood criterion. On the basis of these estimates we can then successively select inputs x that improve the expected accuracy of the parameters.

(2-1) (10pts) Suppose we sample the conditional model with preselected x values $\mathbf{x}^{(n)} = (x_1 \ldots x_n)'$ and obtain the corresponding sample outputs $\mathbf{y}^{(n)} = (y_1 \ldots y_n)'$. Based on the data $(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$, derive the maximum-likelihood parameter estimates $\hat{\theta}$

$$\hat{\theta} = \arg\max_{\theta} \log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, \theta) \tag{13}$$

Express your answer in terms of the statistics $N_{\mathrm{x,y}}(0,1)$, $N_{\mathrm{x,y}}(1,1)$, $N_{\mathrm{x}}(0)$, and $N_{\mathrm{x}}(1)$ where $N_{\mathrm{x,y}}(x,y)$ is the number of times $(\mathrm{x},\mathrm{y}) = (x,y)$ and $N_{\mathrm{x}}(x)$ is the number of times $\mathrm{x} = x$.

Is the resulting *estimator* $\hat{\theta}(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$ (viewed as a function of the data) *unbiased*? Are $\hat{\theta}_{1|0}(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$ and $\hat{\theta}_{1|1}(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$ *uncorrelated* given $\mathbf{x}^{(n)}$? Briefly explain your answer.

(2-2) (10pts) Derive expressions for the variances

$$\mathrm{Var}(\hat{\theta}_{1|0}(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})|\mathbf{x}^{(n)}) \tag{14}$$

$$\mathrm{Var}(\hat{\theta}_{1|1}(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})|\mathbf{x}^{(n)}) \tag{15}$$

in terms of the statistics $N_x(0)$ and $N_x(1)$ and the parameters $\theta = (\theta_{1|0}, \theta_{1|1})'$.

(*Hint.* the variance of a biased 0/1 coin is $p(1-p)$ where p is the probability of 1).

(2-3) (10pts) We should be able to choose the inputs $\mathbf{x}^{(n)}$ so as to improve the resulting parameter estimates. Specifically, we'd like to choose the inputs so as to minimize the overall variance of the estimators

$$J(\mathbf{x}^{(n)}; \theta) = \mathrm{Var}(\hat{\theta}_{1|0}(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})|\mathbf{x}^{(n)}) + \mathrm{Var}(\hat{\theta}_{1|1}(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})|\mathbf{x}^{(n)}) \tag{16}$$

which is a function of both θ and the inputs. We cannot directly use this criterion to select inputs since we do not know θ. We can, however, *plug in* our current best

estimate of θ or $\hat{\theta}$ and use $J(\mathbf{x}^{(n)}; \hat{\theta})$ as the criterion to select the next input x_{n+1} to query. Derive the selection rule for picking $x_{n+1} = 0$ or $x_{n+1} = 1$ based upon which choice reduces the overall variance the most:

$$x_{n+1} = \arg \max_{x_{n+1}=0,1} [J(\mathbf{x}^{(n)}; \hat{\theta}) - J(\mathbf{x}^{(n+1)}; \hat{\theta})] \tag{17}$$

This should reduce to a simple comparison involving the statistics $N_x(\cdot)$ and the estimates $\hat{\theta} = (\hat{\theta}_{1|0}, \hat{\theta}_{1|1})'$. Note that $\hat{\theta}$ is viewed here as fixed since it represents our prior knowledge at the time of selecting x_{n+1}. The actual value naturally depends on all the outputs we have observed so far in response to the inputs $\mathbf{x}^{(n)}$.

(2-4) (20pts) We have provided you with a MATLAB function `y = query(x)` which simulates the conditional model for some unknown parameters θ^* (which you are trying to estimate). Write a MATLAB script to implement your active learning procedure. Initilize your procedure with $\mathbf{x}^{(10)} = (0\,0\,0\,0\,0\,1\,1\,1\,1\,1)'$ and $\mathbf{y}^{(10)} = (0\,0\,1\,0\,1\,1\,1\,1\,1\,0)'$. Run your active learning method for 500 steps[1]. Plot the scaled estimated variance $n \cdot J(\mathbf{x}^{(n)}; \hat{\theta})$, where the scaling comes from the fact that we'd expect the variance to go down roughly as $1/n$. Also plot $n \cdot J(\mathbf{x}^{(n)}; \theta^*)$, where $\theta^* = (0.4, 0.9)'$ are the true parameters for this problem. We typically don't have these but they are useful for comparison.

Hint. To implement your procedure, it is sufficient to accrue just the statistics $N_{\mathrm{x,y}}(0,1)$, $N_{\mathrm{x,y}}(1,1)$, $N_{\mathrm{x}}(0)$, and $N_{\mathrm{x}}(1)$.

(2-5) (10pts) Suppose that we instead randomly sampled x s.t $P(x = 1) = 0.5$. Repeat the above experiment and compare the performance of these two learning methods (active vs. stochastic). You may have to run each method a couple of times to convince yourself that the differences (if any) are typical. Which one is better? Briefly explain the differences.

Problem 3: generative and discriminative models

We can solve classification tasks from two different perspectives. In a *generative* approach, we construct a joint distribution over both the examples $\mathbf{x}$ and labels y. In other words, we choose a parametric family $P(\mathbf{x}, y|\theta)$, where the parameters θ specify each distribution in the family. The parameters are estimated by maximizing the *joint* log-likelihood of the data:

$$l(D; \theta) = \sum_{i=1}^{n} \log P(\mathbf{x}_i, y_i|\theta) \tag{18}$$

[1]You could also implement a stopping criterion based on the remaining variance $J(\mathbf{x}^{(n)}; \hat{\theta})$

where the sum is over the training data $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$. We are maximizing here the joint likelihood of labels and examples although we only need the conditional probabilities $P(y|\mathbf{x}, \theta)$ for the classification task.

In a *discriminative* approach, on the other hand, we only care about the conditional distribution $P(y|\mathbf{x}, w)$ such as the logistic regression model

$$P(y = 1|\mathbf{x}, w) = g(\, w_0 + \sum_{i=1}^{d} x_i w_i \,) \tag{19}$$

The parameters w are estimated by maximizing the *conditional* log-likelihood

$$l_c(D; w) = \sum_{i=1}^{n} \log P(y_i|\mathbf{x}_i, w) \tag{20}$$

Note that while the set of possible conditionals $P(y|\mathbf{x}, w)$ may in some cases be the same set as the conditionals derived from a generative model, $P(y|\mathbf{x}, \theta)$, they are typically not the same as the parameters are estimated differently from the data.

In this problem we estimate both class conditional Gaussian models and logistic regression models. The data for this problem is in `data.mat` and you can load it into MATLAB by `load data.mat`. This will give you two structures, `trainD` and `testD`, and access to examples via `trainD.X` and labels `trainD.y`.

The task is to use the training data in order to learn to predict the labels of yet unobserved points (test data).

(3-1) (10pts) Fit generative Gaussian models to the training data `trainD`. We have provided you code for doing this `mix = mixtrain(trainD.X,trainD.y)`. As a result `mix{1}.mu` has the mean (row vector) and `mix{1}.cov` is the covariance of the class conditional Gaussian corresponding to $y = 0$ examples. Similarly, `mix{2}` contains the Gaussian over $y = 1$ examples.

a) Plot the classification error on the training and test sets. To evaluate the classification error you can use `loglik = mixloglik(X,y,mix)` that returns a log-probability $\log P(\mathbf{x}_i, y_i|\theta)$ for each row.

b) Plot the data and the resulting decision boundary using `mixboundary(mix,trainD)`. This is just so you can view the result.

c) We also need the *conditional* log-likelihoods of the training and test sets under this model for comparison with the logistic regression model. Return the code and the values. (*Hint.* use the code provided for evaluating $\log P(\mathbf{x}_i, y_i|\theta)$).

(3-2) (10pts) Fit a linear logistic regression model on the same data. You can do this by `w = logisticreg(trainD.X,trainD.y)`. Note that the first component of `w` will be the bias term.

a) Evaluate the conditional log-likelihood on the training and test sets as well as the classification errors.

b) Plot the data and the resulting decision boundary using `boundary(w,trainD)`. Again, this is just for you to visualize the result.

c) Compare these results to the above Gaussian case. Can you explain the differences?

(3-3) (10pts) Fit a 2nd order logistic regression model on the training set and repeat the above steps. You can expand any input matrix `X` (where rows correspond to each example) to a matrix of second order feature vectors by `phi = degexpand(X,2)`.

Compare the resulting conditional log-likelihoods to your earlier results involving the class-conditional Gaussian and the linear logistic model. Do the differences make sense? (you may wish to solve the next two problems before completing your answer here).

(3-4) (10pts) Show that the conditional distributions $P(y|x)$ possible with a generative Gaussian mixture model (class condititional Gaussians) are exactly the 2nd order logistic conditional distributions models.

(3-5) (10pts) Show that the conditional log-likelihood on the training set attained by a 2nd order logistic regression model will never be lower than the conditional log-likelihood of the maximum likelihood Gaussian mixture model.

(3-6) (10pts) Train logistic regression models with 1-4 degree polynomial feature vectors. Again use `phi = degexpand(X,2)` with different degrees to generate the appropriate feature vectors. Plot or list the training/test log-conditional likelihoods as a function of the polynomial degree. Explain how they behave in terms of structural and approximation errors. Note that we measure "errors" here in terms conditional log-likelihoods that are maximized rather than minimized.