CSC2515 – Assignment #2

Due: Nov.7, 2006, 2pm at the **START** of class
Worth: 18%
Late assignments not accepted.

1 Pseudo-Bayesian Linear Regression (2%)

In this question you will dabble in Bayesian statistics and extend the probabilistic model formulation to include not only the data but also the weights.

Consider the following conditional model of a scalar output y given some inputs $\mathbf{x}$:

$$\begin{aligned} p(y|\mathbf{x}) &= \int_{\mathbf{w}} p(y|\mathbf{x},\mathbf{w})p(\mathbf{w})d\mathbf{w} \\ p(y|\mathbf{x},\mathbf{w}) &= \mathcal{N}(y;\mathbf{w}^\top\mathbf{x},b) \\ p(\mathbf{w}) &= \mathcal{N}(\mathbf{w};\mathbf{0},a\mathbf{I}) \end{aligned}$$

Here the weight prior $p(\mathbf{w})$ is a Gaussian with mean zero and covariance $a\mathbf{I}$ ($\mathbf{I}$ is the identity matrix) and the data model is a conditional linear-Gaussian with mean $\mathbf{w}^\top\mathbf{x}$ and variance b.

Notice that the model generates a value of y given $\mathbf{x}$ by stochastically selecting a set of weights and using these to linearly combine the components of $\mathbf{x}$. Given this generative model, the correct way to make predictions on future data given a finite training set is to *average* over all possible settings of the weights, letting each one make a prediction, and weight the predictions in the average by the posterior of the weights given the training data.

- Find the posterior distribution over weights $p(\mathbf{w}|\{\mathbf{x}_n, y_n\}, a, b)$ given a finite training set $\{\mathbf{x}_n, y_n\}$ and the variances a, b. Hint: your answer should be another Gaussian distribution, now over $\mathbf{w}$. To do this, you can either form the joint Gaussian $p(y, \mathbf{w}|\mathbf{x})$, marginalize out the weights to form $p(y|\mathbf{x})$ and then use Bayes rule to find $p(\mathbf{w}|\mathbf{x}, y)$ or you can try completing the square, using vector and matrix operations and the apply the conditioning formulas from the probability and statistics review notes.
- Show that the ridge regression weights, which minimize the cost $\lambda\mathbf{w}^\top\mathbf{w} + \sum_n (y_n - \mathbf{w}^\top\mathbf{x}_n)^2$ are equal to the mean (and mode) of the posterior above.
- Give the ridge regression penalty λ in terms of the variances a, b above.
- Integrate out the posterior over the weights, and work out the conditional predictive distribution over a new output y_{new} given a new test input $\mathbf{x}^{new}$ and the training set:

$$p(y_{new}|\mathbf{x}^{new}, \{\mathbf{x}_n, y_n\}, a, b) = \int_{\mathbf{w}} p(y^{new}|\mathbf{x}^{new}, \mathbf{w})p(\mathbf{w}|\{\mathbf{x}_n, y_n\}, a, b)d\mathbf{w}$$

It is permissible to consult external sources (textbooks, research reports on the web, notes from Bayesian stats courses, etc) when working out the answer to this question but it is not permissible to copy an answer directly from another source without understanding it and working it out for yourself.

2 Reminder: Training a Neural Network

In the experiments section you will train a one-hidden-layer neural network to perform simple scalar output regression. As a reminder, the neural network is a generalized linear model of the form

$$\hat{y}(\mathbf{x}) = \sum_{j=1}^{K} w_j h_j(\mathbf{x}) \qquad\qquad h_j(\mathbf{x}) = \sigma(\sum_i b_{ij} x_i)$$

where $\sigma(\cdot)$ is the sigmoid or tanh function and we always include a special "bias" input unit $x_0 = 1$ and a "bias" hidden unit $h_0 = 1$. Assuming that we are trying to minimize squared error (maximize likelihood under a Gaussian noise model), our objective is:

$$\phi^N = \sum_{n=1}^{N} \left(\hat{y}(\mathbf{x}^n) - y^n\right)^2 + \lambda \left[\sum_J w_j^2 + \sum_{ij} b_{ij}^2\right]$$

Of course, since there are local minima we must be very carefuly about how we initialize the weights; one very simple trick is to restart several times from different (small) random weights.

3 Locally Weighted Regression

A very useful and common regression model is to just do "local smoothing" on the training set by placing a basis function on every training input point, and setting the weights so that they interpolate the training outputs:

$$\hat{y}(\mathbf{x}) = \sum_{n=1}^{N} y^n h_n(\mathbf{x})$$
$$h_n(\mathbf{x}) = \frac{\exp\left[-\alpha(\mathbf{x} - \mathbf{x}^n)^\top(\mathbf{x} - \mathbf{x}^n)\right]}{\sum_{m=1}^{N} \exp\left[-\alpha(\mathbf{x} - \mathbf{x}^m)^\top(\mathbf{x} - \mathbf{x}^m)\right]}$$

This model is highly "non-parametric": it has only a single parameter $\alpha > 0$ sometimes called the "bandwidth". We'd like to set the bandwidth by optimizing an objective function numerically. We can't use just simple training error, since this would drive α to infinity. Instead, let's minimize the leave-one-out estimate of squared error:

$$\phi^L = \sum_{n=1}^{N} \left(\tilde{y}(\mathbf{x}^n) - y^n\right)^2$$

where $\tilde{y}(\mathbf{x}^n)$ is computed by leaving out the point $\mathbf{x}^n$ when making predictions, ie

$$\tilde{y}(\mathbf{x}^n) = \sum_{k \neq n} y^k \tilde{h}_k(\mathbf{x}^n)$$
$$\tilde{h}_k(\mathbf{x}^n) = \frac{\exp\left[-\alpha(\mathbf{x}^n - \mathbf{x}^k)^\top(\mathbf{x}^n - \mathbf{x}^k)\right]}{\sum_{m \neq n} \exp\left[-\alpha(\mathbf{x}^n - \mathbf{x}^m)^\top(\mathbf{x}^n - \mathbf{x}^m)\right]}$$

3.1 Derivation [1%]

- Find the derivative $d\phi^L / d\log\alpha$ with respect to $\log\alpha$, given the training set and the current value of α.

4 Experiments

This part of the assignment is set with some basic experiments (worth 9%), which you should do all of, two advanced experiments (worth an additional 3% each) and a bonus (worth up to 3% extra). **Note**: Do not hand in any of your code. Really, we're not kidding, we won't look at it and it will annoy us that you ignored this instruction.

4.1 Datasets

On the course website you will find three datasets, `a2antigen`, `a2motorcycle` and `a2weather`, which are regression problems each having a single scalar output. The antigen data has input dimension 8, 80 training cases, 17 test cases and represents the prostate specific antigen level in a medical prediction task as a function of various risk factors. The motorcycle data set has input dimension 1, 103 training points, 30 test points and represents the acceleration force (in g's) on a motorcycle helmet as a function of time (in ms) during a collision. The weather data set has data input dimension 2, 188 training points, and 50 test points and represents the annual precipitation (in mm) in the state of Washington, USA, as a function of grid location on a map.

4.2 Basic Experiments [9%]

Write programs to perform both locally weighted regression and to train multilayer neural networks (with sigmoid/tanh units) on the three prediction tasks. For locally weighted regression use gradient minimization on ϕ^L to set α. For the neural networks, use gradient minimization on the penalized objective ϕ^N to set the weights and biases. For each of the three datasets do the following:

1. Fit locally weighted regression. Set α by optimizing the leave-one-out error. On the datasets in this assignment, if you carefully follow the gradient of leave-one-out error with respect to α (or better $\log \alpha$), starting from a relatively small value (on the order of the spacing between the training points) you should not need to do any random restarts. If you have some spare cycles on your hands you can check that your search is working by exhaustively evaluating α on a fine grid.
2. Fit a multilayer neural network with one hidden layer having 64 hidden units (with sigmoid/tanh nonlinearities at the hidden layer but no nonlinearity at the output). Use both $\lambda = 0$ (no weight decay) and $\lambda = .001$ (weight decay).

What to hand in:

1. Error/Objective Values: For all three datasets, report the average training error (for the neural nets) or average leave-one-out error (for locally weighted regression) and the average test error for all three methods. Make sure you do not include the weight decay term when computing average test/train errors (but of course you should include it when you compute the gradient to adjust the weights). For the neural nets, report test results for the parameters that achieved the lowest value of the penalized objective function at training time (for most optimization methods this will be the parameters at the final iteration). For locally weighted regression report test results for the α which minimzed the leave-one-out error. Indicate the value of the objective function that you attained at training time and the α your method chose for locally weighted regression.
2. Motorcycle Data Plots: For each regression model, produce a plot showing the training points as squares, the test points as circles and the interpolant $\hat{y}(\mathbf{x})$ as a curve.
3. Weather Data Plots: For each regression model, plot the training points in the input plane using squares, the test points using circles and the interpolant using contour lines overlaid on the data.
4. Antigen Data Plots: Hand in a single plot, clearly labelled, showing four lines, representing the average training and testing errors of the neural network (using $\lambda = 0$ and $\lambda = .001$) as training proceeds (i.e. plot training iterations horizontally and error vertically). Mark on your plot (as a vertical line) the iteration at which the

minimum of training objective occurs, and indicate what the test error is at that point. Also mark on your plot (as another vertical line) the iteration at which the minimum of *test* error occurs and indicate what that minimum test error is. Finally, show (as horizontal lines) the value of training, leave-one-out and testing error achieved by locally weighted regression at the α chosen by minimizing leave-one-out error. This plot may look better if either (or both) iterations/error are shown on a logarithmic scale.

4.3 Advanced Experiments [6%]

1. In the locally weighted regression model, use a different bandwidth α_d for each input dimension. Pick one dataset, and retrain your model and replot the training and testing errors, as well as reporting the optimal values of α_d for all dimensions d.

2. In the neural network model (with $\lambda = .001$), try all numbers of hidden units units in the set $k = [1, 2, 4, 8, 16, 32, ..., M]$ (where M is $\geq$ the number of training samples). For each number of units k, compute the best training error you can achieve after multiple restarts and the test error of the corresponding network. Do this for one dataset of your choice. Hand in a plot of each error measure as a function of the (log base 2 of) number of hidden units k. What k gives the best test error?

4.4 Bonus Experiment [up to 3%]

1. Pick one of the datasets. For that dataset, set both the number of hidden units k *and* the value of the weight decay parameter λ simultaneously by using leave-one-out cross-validation when training your neural network. Chose sensible ranges for k and λ to search over, and display your results graphically, as well as reporting the best pair of (k,λ) for test and validation. Successfully completing this bonus question requires being smart about how you train and re-train your networks, so tell us a bit about how you did that.

IMPORTANT: please do your computing on CDF or your own machine and not on CSLAB machines since they have limited software licenses and other resources.