CS 498 Probability and Statistics for Computer Scientists Final Homework

This homework is more ambitious than previous homeworks, and I'll take this into account in weighting grades. This homework is due at the start of the final exam (which you can find on the web). You may do this homework in pairs.

From the UC Irvine machine learning repository (http://archive.ics.uci.edu/ml/), download the "Adult" dataset (http://archive.ics.uci.edu/ml/datasets/Adult). This contains 48842 data items. Each consists of a set of numeric and categorical features describing a person, together with whether their annual income is larger than or smaller than 50K$. You will use these features to predict whether the annual income of a test person is greater than, or less than, 50K$.

1) First use only the numeric data. Hold out 1,000 examples as a validation set, and a further 5,000 examples as a test set. These should be chosen at random. Now use the remaining examples to build a linear support vector machine, which you should train with stochastic gradient descent. You should implement this yourself. I used 1000 epochs, and 1000 randomly chosen examples per epoch; you may get better results if you try different numbers.
    1) Use the validation set to choose a regularization constant (the lambda in the notes). To do this, train with different values of lambda, then test on the validation set, and use the value of lambda that gives the best result. It should be quite small. The values you test can be quite widely separated, for example 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, etc.
    2) Once you have a reasonable lambda, plot a curve illustrating the effects of training using this lambda. To do this, compute the accuracy (percentage of examples correctly classified) on the validation set at the end of each epoch. Then plot this value against the epoch. It should wiggle a bit, but generally go up; eventually, it should settle down.
    3) Once you are happy with your training results, compute the accuracy of your classifier on the 5,000 example test set.

2) Now use both numeric and categorical features. You can turn categorical features into numeric features by the following scheme. For example, the second data field can take the values Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked . You can represent this with an 8 dimensional vector. If the value of the field is Private, the first entry of the vector is 1 and the others are zero; if it is Self-emp-not-inc, the second is 1 and the others are zero; and so on. You should notice that some examples have fields where the value is unknown (it's given by a ?). Omit these examples from test and training for this part of the process. Again, hold out 1,000 examples as a validation set, and a further 5,000 examples as a test set. These should be chosen at random. Again, train using stochastic gradient descent which you implement yourself.
    1) Use the validation set to choose a regularization constant (the lambda in the notes). To do this, train with different values of lambda, then test on the

validation set, and use the value of lambda that gives the best result. It should be quite small. The values you test can be quite widely separated, for example 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, etc.

2) Once you have a reasonable lambda, plot a curve illustrating the effects of training. To do this, compute the accuracy (percentage of examples correctly classified) on the validation set at the end of each epoch. Then plot this value against the epoch. It should wiggle a bit, but generally go up; eventually, it should settle down.
3) Once you are happy with your training results, compute the accuracy of your classifier on the 5,000 example test set.

3) Now write your test and training sets for the second case to a file, download SVMlib from http://www.csie.ntu.edu.tw/~cjlin/libsvm/ and use that to train a linear SVM on your training set. Compute the accuracy of this classifier on a 5000 example test set. How do they compare? Do the categorical features make things better? What is the difference between your classifier and the SVMlib classifier?

4) (Optional: A tiny number of extra marks available for this part, but you can have a great deal of fun like this) Can you make a more accurate classifier using the various options in SVMLib?