# Is He Bourgeois?

## Income classification using US census data

Theodore Nguyen
Dillon Zhi

# Problem

Using the UCI "Adult" data set, containing data from the US census, we attempted to predict whether a person's annual income was over or under $50000, based on demographic attributes of the person as described below. This is a binary classification problem. We used a support vector machine (SVM) model to perform the classification.

# Data set

The UCI "Adult" data set contains data from the "1994 Census database" [1]. We used the data in the file "adult.data" only, which has data for 32560 people, provided in CSV (comma-separated values) format. For each entry, there are 15 attributes such as age, education, marital status, and native country - see [1] for the full list. In particular, one of the attributes indicates whether the annual income was over or under $50000. (The actual annual incomes are not provided in the data set.) Among the data points, approximately 1/4 have income over $50000, and the other 3/4 have income under $50000.

We performed some light pre-processing of the data to encode string attributes in a form which MATLAB could readily process. In particular, we used one-of-K encoding to encode categorical attributes such as race.

# Machine learning method

This is essentially a psuedo-summary of the soft SVM machine learning method outlined in the book.

Given $n$ data points $x_i \in \mathbb{R}^d$ and class labels $t_i = \{-1, 1\}$ we want to find the maximum-margin hyperplane of form $y_i(x_i) = w^T x_i + b$ ideally separates the points into the two classes, where the sign of $y_i$ is a prediction of the sign of $t_i$, from $x_i$. We define slack variables $\xi_i \geq 0$ for each point s.t. $\xi_i = 0$ for points on or inside correct margin boundary and $\xi_i = |t_i - y(x_i)|$ for others. The distance from point to the hyperplane is

$\frac{t_i y(x_i)}{\|w\|}$, so the maximum margin solution is $\arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|w\|} min(t_i y(x_i)) \right\}$. We convert this to an equivalent problem $t_i y(x_i) = 1$ for points closest to the boundary, and using the slack variables we get the constraint $t_i y(x_i) \geq 1 - \xi_i$. Once the margin has been maximized, there will at least be two points that satisfy the inequality. This problem then just requires maximization of $\|w\|^{-1}$, equivalent to minimization of $\|w\|^2$. Therefore, the final soft SVM problem becomes:

$$\min_{w,b,\xi_i} \frac{1}{2}\|w\|^2 + C\sum \xi_i \;\; s.t. \;\; \begin{cases} \xi_i \geq 0 \\ t_i y(x_i) \geq 1 - \xi_i \end{cases}$$

Where $C$ is a parameter that controls the tradeoff between slack variable penalty and the margin.

# Results

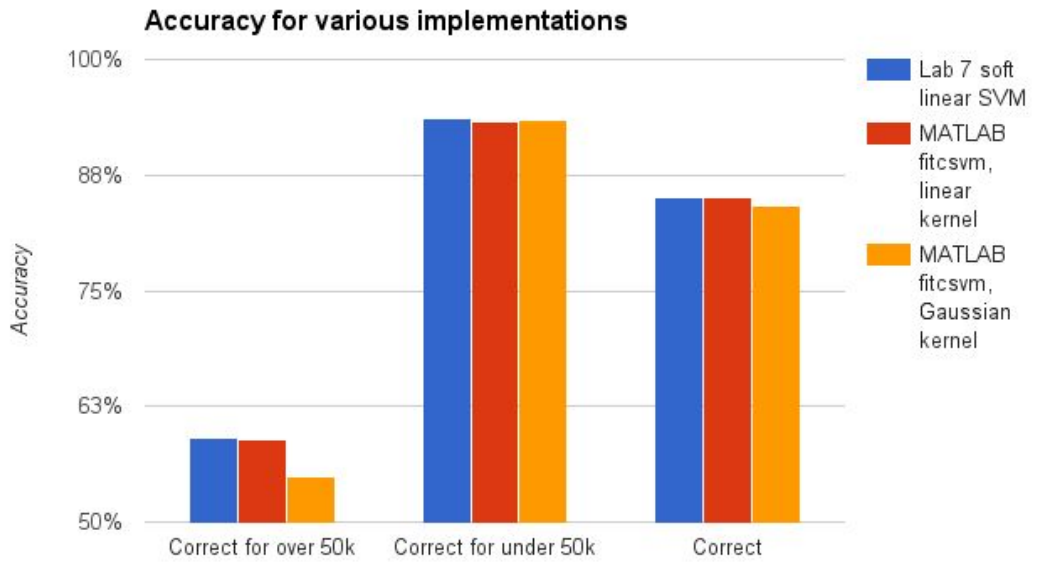
Our main implementation of the SVM classifier, using linear SVM with 20000 training samples, was 85.1% accurate (i.e. when applied to the test set, 85.1% percent of the predictions were correct) . This is comparable to the accuracy of the numerous other methods (e.g. naive Bayes, C4.5, NBTree) cited by the data set authors, whose accuracies were at best 86% [2, 3]. The accuracy for the people with income over $50000 was 59%; the accuracy for those under $50000 was 93.6%. Accuracy for the population with income below $50000 was always far better than for those over $50000: for better or worse, this was likely because the below-$50000 samples outnumbered the above-$50000 samples, and all the samples were weighted equally.

## Varying the implementation

We tried three different variations on the machine learning method:

- An implementation of soft linear SVM based on Lab 7 from this class, making use of MATLAB's quadprog function for the computation
- MATLAB's provided SVM functionality (fitcsvm) using the default linear kernel
- MATLAB's fitcsvm with a Gaussian kernel

The results are given in the figure below. The training set consisted of 20000 samples, and the remaining samples were used to test the accuracy of the classification. Our own implementation of linear SVM and MATLAB's linear SVM performed essentially equally well, which was what we expected.




| SVM method | Lab 7 soft linear SVM | MATLAB fitcsvm, linear kernel | MATLAB fitcsvm, Gaussian kernel |
|---|---|---|---|
| Accuracy for over $50k | 59% | 59% | 55% |
| Accuracy for under $50k | 93.6% | 93.2% | 93.5% |
| Accuracy | 85.1% | 85.1% | 84.2% |

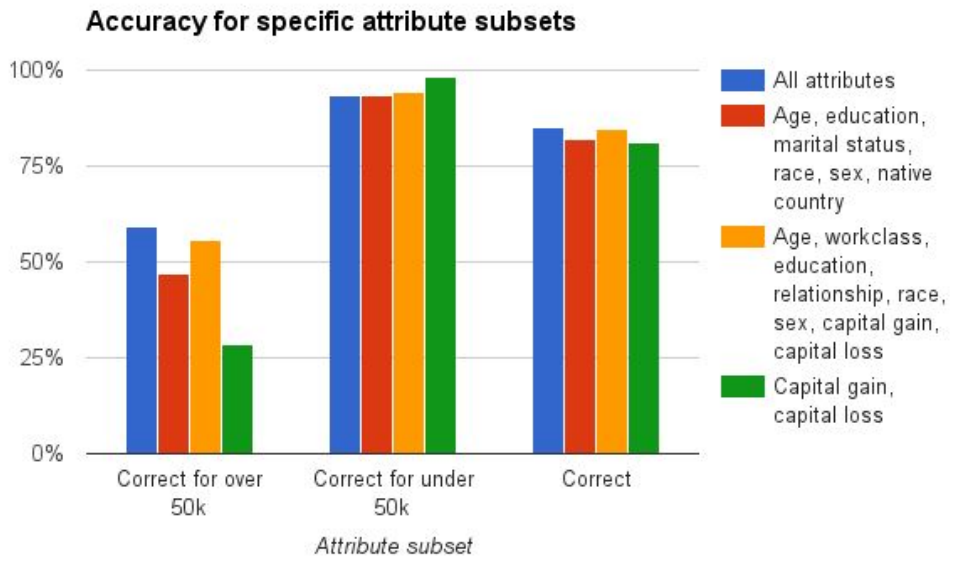
## Prediction using specific attribute subsets

We examined using only selected subsets of the data attributes rather than the full set of data attributes. Listed below are the subsets we tried and the reasons we chose them.

(1) Age, education, marital status, race, sex, native country: Basic demographic information; we wondered how well this alone could predict income.
(2) Age, workclass, education, relationship, race, sex, capital gain, capital loss: A hand-picked subset of attributes that we hypothesized might be a good predictor for income.
(3) Capital gain, capital loss: The only attributes that were directly about finances; we wondered how well just these attributes could predict income.

Again, we used the soft linear SVM implementation based on Lab 7, with 20000 training samples. The results are shown below. The accuracy of (2) was comparable to the accuracy of using the full data set, while (1) and (3) performed substantially worse.

(We did not attempt to systematically learn which small subset of attributes could best predict the income, as this was out of scope for the project. This is a possible direction for future work.)




| Attribute subset | All attributes | Age, education, marital status, race, sex, native country | Age, workclass, education, relationship, race, sex, capital gain, capital loss | Capital gain, capital loss |
|---|---|---|---|---|
| Accuracy for over $50k | 59% | 47% | 56% | 28% |
| Accuracy for under $50k | 93.6% | 93.2% | 94.1% | 98.2% |
| Accuracy | 85.1% | 81.8% | 84.7% | 81.1% |

## Varying the training set size

We considered whether the 20000 training samples we used was enough to produce the most accurate predictions possible with this machine learning method on this data set. To test this, we tried various numbers of training samples; some results are shown below. Increasing the number of training samples past approximately 5000 did not appear to impact the overall accuracy in any significant way. We concluded that 20000 training samples was certainly sufficient for our purposes.


| Number of training data points | 500 | 1000 | 2500 | 5000 | 7500 | 20000 |
|---|---|---|---|---|---|---|
| Accuracy | 84.1% | 84.6% | 84.8% | 85.1% | 85.2% | 85.1% |

# Conclusion

We found that SVM was an effective method for predicting income in the “Adult” data set. The overall classification accuracy was approximately 85%, comparable to the results obtained with numerous other methods [2, 3]. We were surprised at how well the method performed essentially out-of-the-box, without requiring major preprocessing, parameter tuning, sophisticated nonlinear models, etc. at all. It is not clear if such added effort can further improve the prediction accuracy. This is one possible direction for future work. The limitation of using SVM for this problem was the somewhat “black-box” nature of the learned model: the model was effective at prediction, but it was difficult to interpret for insight on the structure of the data.

# Works Cited:


[1] Adult Data Set. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Adult

[2] https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.names

[3] Kohavi, Ron (1996). "Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid". *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*.

[4] Dua, Rishi (2014). Linear Separability Test in MATLAB. https://github.com/rishirdua/linear-separability-matlab

[5] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*.

# Appendix: MATLAB CODE

## Table of Contents



# A1: data_converison.m

This script essentially converts the data we are given into a format we can use in MATLAB.

HOW TO USE: Make sure you have adult.data.dat and adult.test.dat in the current working directory. We actually never use adult.test.dat at all, but since work was done planning to do so, we never got to changing it. Note that the data from UCI is given as adult.data and adult.test - you have to simply rename the files to adult.data.dat and adult.test.dat respectively for this to work. Simply run the script once those conditions are satisfied

```
adult_data = readtable('adult.data.dat');
adult_test = readtable('adult.test.dat');

for j = 2:-1:1
 if j == 1
  a = table2cell(adult_data);
  N = size(a,1);
 elseif j == 2
  a = table2cell(adult_test);
  N = size(a,1);
 end

 %age
 age = [];
 for i = 1:N
  age(end+1) = a{i,1};
 end
 age = age';

 %work class
 workclass = zeros(N, 8);
 for i = 1:N
  if strcmp(a{i,2}, 'Private')
   workclass(i,1) = 1;
  elseif strcmp(a{i,2},'Self-emp-not-inc')
   workclass(i,2) = 1;
  elseif strcmp(a{i,2},'Self-emp-inc')
   workclass(i,3) = 1;
  elseif strcmp(a{i,2}, 'Federal-gov')
   workclass(i,4) = 1;
```

```
 elseif strcmp(a{i,2}, 'Local-gov')
  workclass(i,5) = 1;
 elseif strcmp(a{i,2}, 'State-gov')
  workclass(i,6) = 1;
 elseif strcmp(a{i,2}, 'Without-pay')
  workclass(i,7) = 1;
 elseif strcmp(a{i,2}, 'Never-worked')
  workclass(i,8) = 1;
 end
end

%fnclwgt
fnlwgt = [];
for i = 1:N
 fnlwgt(end + 1) = a{i,3};
end
fnlwgt = fnlwgt';

%education-cont
education = [];
for i = 1:N
 education(end + 1) = a{i,5};
end
education = education';

%martial-status
martialstatus = zeros(N, 7);
for i = 1:N
 if strcmp(a{i,6}, 'Married-civ-spouse')
  martialstatus(i,1) = 1;
 elseif strcmp(a{i,6}, 'Divorced')
  martialstatus(i,2) = 1;
 elseif strcmp(a{i,6}, 'Never-married')
  martialstatus(i,3) = 1;
 elseif strcmp(a{i,6}, 'Separated')
  martialstatus(i,4) = 1;
 elseif strcmp(a{i,6}, 'Widowed')
  martialstatus(i,5) = 1;
 elseif strcmp(a{i,6}, 'Married-spouse-absent')
  martialstatus(i,6) = 1;
 elseif strcmp(a{i,6}, 'Married-AF-spouse')
  martialstatus(i,7) = 1;
 end
end

%occupation
occupation = zeros(N, 14);
for i = 1:N
 if strcmp(a{i,7}, 'Tech-support')
  occupation(i,1) = 1;
 elseif strcmp(a{i,7}, 'Craft-repair')
  occupation(i,2) = 1;
 elseif strcmp(a{i,7}, 'Other-service')
  occupation(i,3) = 1;
```

```
 elseif strcmp(a{i,7}, 'Other-service')
  occupation(i,4) = 1;
 elseif strcmp(a{i,7}, 'Sales')
  occupation(i,5) = 1;
 elseif strcmp(a{i,7}, 'Exec-managerial')
  occupation(i,6) = 1;
 elseif strcmp(a{i,7}, 'Prof-specialty')
  occupation(i,7) = 1;
 elseif strcmp(a{i,7}, 'Machine-op-inspct')
  occupation(i,8) = 1;
 elseif strcmp(a{i,7}, 'Adm-clerical')
  occupation(i,9) = 1;
 elseif strcmp(a{i,7}, 'Farming-fishing')
  occupation(i,10) = 1;
 elseif strcmp(a{i,7}, 'Transport-moving')
  occupation(i,11) = 1;
 elseif strcmp(a{i,7}, 'Priv-house-serv')
  occupation(i,12) = 1;
 elseif strcmp(a{i,7}, 'Protective-serv')
  occupation(i,13) = 1;
 elseif strcmp(a{i,7}, 'Armed-Forces')
  occupation(i,14) = 1;
 end
end
%relationship
relationship = zeros(N, 6);
for i = 1:N
 if strcmp(a{i,8}, 'Wife')
  relationship(i,1) = 1;
 elseif strcmp(a{i,8}, 'Own-child')
  relationship(i,2) = 1;
 elseif strcmp(a{i,8}, 'Husband')
  relationship(i,3) = 1;
 elseif strcmp(a{i,8}, 'Not-in-family')
  relationship(i,4) = 1;
 elseif strcmp(a{i,8}, 'Other-relative')
  relationship(i,5) = 1;
 elseif strcmp(a{i,8}, 'Unmarried')
  relationship(i,6) = 1;
 end
end

%race
race = zeros(N, 5);
for i = 1:N
 if strcmp(a{i,9}, 'White')
  race(i,1) = 1;
 elseif strcmp(a{i,9}, 'Asian-Pac-Islander')
  race(i,2) = 1;
 elseif strcmp(a{i,9}, 'Amer-Indian-Eskimo')
  race(i,3) = 1;
 elseif strcmp(a{i,9}, 'Other')
  race(i,4) = 1;
 elseif strcmp(a{i,9}, 'Black')
```

```
  race(i,5) = 1;
 end
end

%sex
sex = zeros(N, 2);
for i = 1:N
 if strcmp(a{i,10},'Female')
  sex(i,1) = 1;
 elseif strcmp(a{i,10},'Male')
  sex(i,2) = 1;
 end
end

%captial-gain
capitalgain = [];
for i = 1:N
 capitalgain(end + 1) = a{i, 11};
end
capitalgain = capitalgain';

%capital-loss
capitalloss = [];
for i = 1:N
 capitalloss(end + 1) = a{i, 12};
end
capitalloss = capitalloss';

%hours-per-week
hoursperweek = [];
for i = 1:N
 hoursperweek(end + 1) = a{i,13};
end
hoursperweek = hoursperweek';

%native country
nativecountry = zeros(N, 41);
for i = 1:N
 if strcmp(a{i,14}, 'United-States')
  nativecountry(i,1) = 1;
 elseif strcmp(a{i,14}, 'Cambodia')
  nativecountry(i,2) = 1;
 elseif strcmp(a{i,14}, 'England')
  nativecountry(i,3) = 1;
 elseif strcmp(a{i,14}, 'Puerto-Rico')
  nativecountry(i,4) = 1;
 elseif strcmp(a{i,14}, 'Canada')
  nativecountry(i,5) = 1;
 elseif strcmp(a{i,14}, 'Germany')
  nativecountry(i,6) = 1;
 elseif strcmp(a{i,14}, 'Outlying-US(Guam-USVI-etc)')
  nativecountry(i,7) = 1;
 elseif strcmp(a{i,14}, 'India')
  nativecountry(i,8) = 1;
```

```
elseif strcmp(a{i,14}, 'Japan')
 nativecountry(i,9) = 1;
elseif strcmp(a{i,14}, 'Greece')
 nativecountry(i,10) = 1;
elseif strcmp(a{i,14}, 'South')
 nativecountry(i,11) = 1;
elseif strcmp(a{i,14}, 'China')
 nativecountry(i,12) = 1;
elseif strcmp(a{i,14}, 'Cuba')
 nativecountry(i,13) = 1;
elseif strcmp(a{i,14}, 'Iran')
 nativecountry(i,14) = 1;
elseif strcmp(a{i,14}, 'Honduras')
 nativecountry(i,15) = 1;
elseif strcmp(a{i,14}, 'Philippines')
 nativecountry(i,16) = 1;
elseif strcmp(a{i,14}, 'Italy')
 nativecountry(i,17) = 1;
elseif strcmp(a{i,14}, 'Poland')
 nativecountry(i,18) = 1;
elseif strcmp(a{i,14}, 'Jamaica')
 nativecountry(i,19) = 1;
elseif strcmp(a{i,14}, 'Vietnam')
 nativecountry(i,20) = 1;
elseif strcmp(a{i,14}, 'Mexico')
 nativecountry(i,21) = 1;
elseif strcmp(a{i,14}, 'Portugal')
 nativecountry(i,22) = 1;
elseif strcmp(a{i,14}, 'Ireland')
 nativecountry(i,23) = 1;
elseif strcmp(a{i,14}, 'France')
 nativecountry(i,24) = 1;
elseif strcmp(a{i,14}, 'Dominican-Republic')
 nativecountry(i,25) = 1;
elseif strcmp(a{i,14}, 'Laos')
 nativecountry(i,26) = 1;
elseif strcmp(a{i,14}, 'Ecuador')
 nativecountry(i,27) = 1;
elseif strcmp(a{i,14}, 'Taiwan')
 nativecountry(i,28) = 1;
elseif strcmp(a{i,14}, 'Haiti')
 nativecountry(i,29) = 1;
elseif strcmp(a{i,14}, 'Columbia')
 nativecountry(i,30) = 1;
elseif strcmp(a{i,14}, 'Hungary')
 nativecountry(i,31) = 1;
elseif strcmp(a{i,14}, 'Guatemala')
 nativecountry(i,32) = 1;
elseif strcmp(a{i,14}, 'Nicaragua')
 nativecountry(i,33) = 1;
elseif strcmp(a{i,14}, 'Scotland')
 nativecountry(i,34) = 1;
elseif strcmp(a{i,14}, 'Thailand')
 nativecountry(i,35) = 1;
```

```
 elseif strcmp(a{i,14}, 'Yugoslavia')
  nativecountry(i,36) = 1;
 elseif strcmp(a{i,14}, 'El-Salvador')
  nativecountry(i,37) = 1;
 elseif strcmp(a{i,14}, 'Trinadad&Tobago')
  nativecountry(i,38) = 1;
 elseif strcmp(a{i,14}, 'Peru')
  nativecountry(i,39) = 1;
 elseif strcmp(a{i,14}, 'Hong')
  nativecountry(i,40) = 1;
 elseif strcmp(a{i,14}, 'Holand-Netherlands')
  nativecountry(i,41) = 1;
 end
end

if j == 1
 over50k = zeros(N,1);
 for i=1:N
  if strcmp(a{i,15}, '<=50K')
   over50k(i,1) = -1;
  elseif strcmp(a{i,15}, '>50K')
   over50k(i,1) = 1;
  end
 end
elseif j == 2
 testY = zeros(N,1);
 for i=1:N
  if strcmp(a{i,15}, '<=50K')
   testY(i,1) = -1;
  elseif strcmp(a{i,15}, '>50K')
   testY(i,1) = 1;
  end
 end
end

if j == 1
 data = [age workclass fnlwgt education martialstatus occupation
relationship race sex capitalgain capitalloss hoursperweek
nativecountry];
elseif j == 2
 test = [age workclass fnlwgt education martialstatus occupation
relationship race sex capitalgain capitalloss hoursperweek
nativecountry];
 end
end
```

# A2: linear_sep.m

This script is the property of Rishi Dua; it has been modified here and is used to check if the convex hulls of the two classes of points overlap - that is, if they are linearly separable. The algorithm does not converge on our data with any modified parameter of maxiter, so we assume that the data is not linearly separable.

https://github.com/rishirdua/linear-separability-matlab

HOW TO USE: Have this file in the same directory as data_conversion.m, adult.test.dat, and adult.data.dat, outlined identically in section A1's HOW TO USE. Simply run the script once those conditions are satisfied

```
%clc; clear; close all;
data_conversion;
Ytrain = over50k;
Xtrain = data;
%set max iteration
maxiter = 10000;

%load data
%[Ytrain, Xtrain] = libsvmread('sample.txt'); %libsvm format
mtrain = size(Xtrain,1);
n = size(Xtrain,2);


% learn perceptron
Xtrain_perceptron = [ones(mtrain,1) Xtrain];
alpha = 0.1;
%initialize
theta_perceptron = zeros(n+1,1);
trainerror_mag = 1000000;
iteration = 0;

%loop
while (trainerror_mag>0)
 fprintf('iteration is %i, trainerror is %i\n', iteration,
 trainerror_mag);
 iteration = iteration+1;
 if (iteration == maxiter)
  break;
 end;
 for i = 1 : mtrain
  Ypredict_temp = sign(theta_perceptron'*Xtrain_perceptron(i,:)');
  theta_perceptron = theta_perceptron + alpha*(Ytrain(i)-
Ypredict_temp)*Xtrain_perceptron(i,:)';
 end
 Ytrainpredict_perceptron =
 sign(theta_perceptron'*Xtrain_perceptron')';
 trainerror_mag = (Ytrainpredict_perceptron -
 Ytrain)'*(Ytrainpredict_perceptron - Ytrain);
end

if (trainerror_mag==0)
 fprintf('Data is Linearly seperable\n');
 fprintf('Parameters are:\n');
 disp(theta_perceptron)
else
 fprintf('Data is not linearly seperable');
end;
```

# A3: softsvm.m

This function is identical to the one submitted in Lab 7. Essentially the implementation of soft SVM

HOW TO USE: Below is a description of the function input and output

```
% Learns an approximately separating hyperplane for the provided data.
% In the following, N = number of data samples, D = dimensionality of
 data.

% Inputs:
% X - Matrix with observations in each row. This is an N x D matrix.
% t - Vector of length equal to the number of columns of X, with
 either a 1 or -1
%     indicating the class label. This is a 1 x N array/vector/matrix.
% gamma - Slack penalty parameter. Higher implies greater violation
 penalty. We used
%       gamma = 0.005 by default here.

% Outputs:
% w - Normal vector for the output hyperplane (plane equation is <w,x>
 + b = 0).
% b - Constant offset for the output hyperplane.


function [w, b] = softsvm(X, t, gamma)
 N = size(X, 1);
 D = size(X, 2);
 %make H
 H = eye(N+D+1);
 H(1,1) = 0;
 H(N+D+1, N+D+1) = 0;
 %define f
 ft = [];
 for i = 1:N+D+1
  ft(end + 1) = gamma;
 end
 ft(N+D+1) = 0;
 ft(N+D) = 0;
 f = ft';
 %make A.
 In = eye(N) * -1;
 T = diag(t);
 TX = -1*T*X;
 negt = -1 * t;
 A = [ In TX negt ];
 %define b
 B = [];
 for i = 1:N
  B(i) = -1;
 end
 %define lb
 lbt = [0 -inf -inf];
```

```
 lb = lbt';

 xi_w_b = quadprog(H, f, A, B, [], [], lb);

 w = [];
 for i = 1:D
  w(end + 1) = xi_w_b(N+i);
 end

 b = xi_w_b(N+D+1);

end
```

# A4: runMyLab.m

This script essentially runs the entire lab and gets data for each of our desired subsets and specifications. The only thing that needs to be tweaked is the train_num parameter - this will be the number of points used to train our SVM classifier

```
%HOW TO USE: Have data_conversion, adult.test.dat, adult.data.dat in
 the
%same directory as this. Modify train_num accordingly depending on how
 many
%datapoints you want to train the model with. Simply run the script
clear all
data_conversion

warning('off','all')

train_num = 20000;
acc = [];
underacc = [];
overacc = [];

original_data = data;
for iteration = 0:3

 if iteration == 0
  data = original_data;
 elseif iteration == 1
  data = [age education martialstatus race sex nativecountry];
 elseif iteration == 2
  data = [age workclass education relationship race sex capitalgain
 capitalloss ];
 elseif iteration == 3
  data = [capitalgain capitalloss];
    end

 %this code did things using ONLY the adult_data file,
 proportionalizing it into a test/train

 N = 32560;
 [w, b] = softsvm(data(1:train_num,:), over50k(1:train_num),
 0.005); %used gamma = 0.005 in the lab
```

```
 prediction = data((train_num+1):N,:) * w' + b;
 results = zeros(1, N - train_num);
 for i = 1:N-train_num
  if sign(prediction(i)) == sign(over50k(train_num + i))
   results(i) = 1;
  else
   results(i) = 0;
  end
 end



 total_accuracy = sum(results) / (N - train_num);
 acc(end+1) = total_accuracy;


 num_under_test = 0;
 num_over_test = 0;
 num_under_correct = 0;
 num_over_correct = 0;
 for i = (train_num+1):N
  if over50k(i) == 1
   num_over_test = num_over_test + 1;
   if results(i-train_num) == 1
    num_over_correct = num_over_correct + 1;
   else
    continue;
   end
  elseif over50k(i) == -1
   num_under_test = num_under_test + 1;
   if results(i-train_num) == 1
    num_under_correct = num_under_correct + 1;
   else
    continue;
   end
  end
 end

 overaccuracy = num_over_correct / num_over_test;
 overacc(end + 1) = overaccuracy;

 underaccuracy = num_under_correct / num_under_test;
 underacc(end+1) = underaccuracy;


 fprintf('-------------------------------------\n');

end

fprintf('--------------------------------\n');
for k = 0:3
 if k == 0
  fprintf('Using the original data (no subsets)\n');
  fprintf('Overall accuracy is: %f \n', acc(k+1));
```

```
  fprintf('Accuracy for those Over 50K income: %f\n', overacc(k+1));
  fprintf('Accuracy for those Under 50K income: %f\n', underacc(k+1));
 elseif k == 1
  fprintf('Using only age, education, martialstatus, race, sex, and
 country\n');
  fprintf('Overall accuracy is: %f \n', acc(k+1));
  fprintf('Accuracy for those Over 50K income: %f\n', overacc(k+1));
  fprintf('Accuracy for those Under 50K income: %f\n', underacc(k+1));
 elseif k == 2
  fprintf('Using age, workclass, education, relationship, race, sex,
 and capital gain/loss\n');
  fprintf('Overall accuracy is: %f \n', acc(k+1));
  fprintf('Accuracy for those Over 50K income: %f\n', overacc(k+1));
  fprintf('Accuracy for those Under 50K income: %f\n', underacc(k+1));
 elseif k == 3
  fprintf('Using ONLY capitalgain/loss')
  fprintf('Overall accuracy is: %f \n', acc(k+1));
  fprintf('Accuracy for those Over 50K income: %f\n', overacc(k+1));
  fprintf('Accuracy for those Under 50K income: %f\n', underacc(k+1));
 end
 fprintf('--------------------------------\n');
end
```

# A5: otherSVM.m

This script tests and uses the other SVM MATLAB function, tweakable to using both a Gaussian kernel, like we did for our data, and for a soft SVM, similar to what we did in Lab 7.

HOW TO USE: Read the comments in the code for more information. Refer back to data_conversion's matrices/arrays to see how to pick what subset of data to use for limited_data. also, replace the Kernel-Function with gaussian underneath fitSVM if you want to use a different kernel.

```
% Modify this line to choose the specific subset of the data
 attributes you
% want to use for prediction
limited_data = data;
% Example:
%limited_data = [age education];

data_reordered = data;
over50k_reordered = over50k;
% Uncomment to use a randomly selected training set instead
%perm = randperm(N);
%data_reordered = limited_data(perm, :);
%over50k_reordered = over50k(perm, :);

m = 1000;
X_train = data_reordered(1:m, :);
y = over50k_reordered(1:m);

svm_model = fitcsvm(X_train, y, ...
        'Standardize', true, ...
        'KernelFunction', 'linear', ... % can replace 'linear' with
 'gaussian'
```

```
        'KernelScale', 'auto', ...
        'BoxConstraint', 1);

X_test = data_reordered(m+1:N, :);

[label,Score] = predict(svm_model, X_test);
correct = label==over50k_reordered(m+1:N);
correct_fraction = mean(correct);

correct_for_over50k = correct(over50k_reordered(m+1:N)==1);
correct_for_over50k_fraction = mean(correct_for_over50k);

correct_for_under50k = correct(over50k_reordered(m+1:N)==-1);
correct_for_under50k_fraction = mean(correct_for_under50k);
```

*Published with MATLAB® R2015a*