CPSC 340 Assignment 4 (due Monday November 14 at 11:55pm)

Name(s) and Student ID(s):
David Pérez Carrasco - 25984360
Lucas Amar - 37082823

1 Gaussian RBFs and Regularization

Unfortunately, in practice we often do not know what basis to use. However, if we have enough data then we can make up for this by using a basis that is flexible enough to model any reasonable function. These may perform poorly if we do not have much data, but can perform almost as well as the optimal basis as the size of the dataset grows. In this question you will explore using Gaussian radial basis functions (RBFs), which have this property. These RBFs depend on a parameter σ, which (like p in the polynomial basis) can be chosen using a validation set. In this question, you will also see how cross-validation allows you to tune parameters of the model on a larger dataset than a strict training/validation split would allow.

1.1 Regularization

If you run the demo *example_RBF.jl*, it will load a dataset and randomly split the training examples into a "train" and a "validation" set (it does this randomly since the data is sorted). It will then search for the best value of σ for the RBF basis. Once it has the "best" value of σ, it re-trains on the entire dataset and reports the training error on the full training set as well as the error on the test set.

A strange behaviour appears: if you run the script more than once it might choose different values of σ. Sometimes it chooses a large value of σ (like 32) that follows the general trend but misses the oscillations. Other times it sets $\sigma = 1$ or $\sigma = 2$, which fits the oscillations better but overfits so achieves a similar test error.[1] Write a function *leastSquaresRBF*, that fits the model with L2-regularization. Hand in your code, and report the test error you obtain if you train on the full dataset with $\sigma = 1$ and $\lambda = 10^{-12}$ (a very small value).

Hint: to construct an identity matrix in Julia, use the linear algebra package (*using LinearAlgebra*) and then use I to make an identity matrix of the appropriate size (Julia figures out the dimensions for you).

Answer: Test error with $\sigma = 1$ and $\lambda = 10^{-12}$: 71.17

---

[1] This behaviour seems to be dependent on your exact setup. Because the $Z^T Z$ matrix with the RBF matrix is really-badly behaved numerically, different floating-point and matrix-operation implementations will handle this in different ways: in some settings it will actually regularizer for you!

```
function leastSquaresRBF(X,y,sigma,lambda)
    (n,d) = size(X)

    Z = rbf(X,X,sigma)

    w = (Z'*Z + lambda*I)\(Z'*y)

    predict(Xhat) = rbf(Xhat,X,sigma)*w

    return LinearModel(predict,w)
end
```

Figure 1: 1.1 - LeastSquaresRBF function

1.2 Cross-Validation

Even with regularization, the randomization of the training/validation sets has an effect on the value of σ that we choose (on some runs it still chooses a large σ value). This variability would be reduced if we had a larger "train" and "validation" set, and one way to simulate this is with *cross-validation*. Modify the training/validation procedure to use 10-fold cross-validation to select σ (with λ fixed at 10^{-12}). Hand in your code and report how this affects the selection of σ compared to the original code.

```
1   using Printf
2   using Random
3   using LinearAlgebra
4
5   # Load X and y variable
6   using JLD
7   data = load("basisData.jld")
8   (X,y,Xtest,ytest) = (data["X"],data["y"],data["Xtest"],data["ytest"])
9
10  # Data is sorted, so *randomly* split into train and validation:
11  n = size(X,1)
12  perm = randperm(n)
13
14
15  # Find best value of RBF variance parameter,
16  #   training on the train set and validating on the test set
17  include("leastSquares.jl")
18  minErr = Inf
19  bestSigma = []
20  for sigma in 2.0.^(-15:15)
21
22      # Apply 10-fold CV
23      FoldErr = 0
24      for fold in 0:9
25
26          local validStart = Int64(fold*n/10+1) # Start of fold indices
27          local validEnd = Int64((fold+1)*n/10) # End of fold incides
28          local validNdx = perm[validStart:validEnd] # Indices of fold examples
29          local trainNdx = perm[setdiff(1:n,validStart:validEnd)] # Indices of training examples
```

Figure 2: 1.2 - 10-fold cross-validation procedure code Part 1

Answer: Now our code is able to select the proper σ value practically always, while the original code sometimes returned values which were not optimal, due to the randomness. This is because applying 10-fold Cross Validation procedure, we neglect the overfitting possibility of choosing a random validation set, since we switch between 10 different validation and training sets and we use the average, and it turns into a more accurate validation error, allowing us to get the actual best value of the hyperparameter more commonly.

2

```
26          local validStart = Int64(fold*n/10+1) # Start of fold indices
27          local validEnd = Int64((fold+1)*n/10) # End of fold incides
28          local validNdx = perm[validStart:validEnd] # Indices of fold examples
29          local trainNdx = perm[setdiff(1:n,validStart:validEnd)] # Indices of training examples
30          local Xtrain = X[trainNdx,:]
31          local ytrain = y[trainNdx]
32          local Xvalid = X[validNdx,:]
33          local yvalid = y[validNdx]
34
35
36          model_sigma = leastSquaresRBF(Xtrain,ytrain,sigma, 10^-12)
37
38          # Compute the error on the validation set
39          yhat_sigma = model_sigma.predict(Xvalid)
40          FoldvalidError = sum((yhat_sigma - yvalid).^2)/(n/2)
41
42          FoldErr+=FoldvalidError
43
44      end
45
46      validError = FoldErr/10
47
48      @printf("With sigma = %.3f, Average validError with 10-fold CV = %.2f\n",sigma,validError)
49
50      # Keep track of the lowest validation error
51      if validError < minErr
52          global minErr = validError
53          global bestSigma = sigma
54      end
```

Figure 3: 1.2 - 10-fold cross-validation procedure code Part 2

```
50          # Keep track of the lowest validation error
51          if validError < minErr
52              global minErr = validError
53              global bestSigma = sigma
54          end
55
56      end
57
58      # Now fit the model based on the full dataset
59      model = leastSquaresRBF(X,y,bestSigma, 10^-12)
60
61      # Report the error on the test set
62      t = size(Xtest,1)
63      yhat = model.predict(Xtest)
64      testError = sum((yhat - ytest).^2)/t
65      @printf("With best sigma of %.3f, testError = %.2f\n",bestSigma,testError)
66
67      # Plot model
68      using Plots
69      scatter(X,y,legend=false,linestyle=:dot)
70      scatter!(Xtest,ytest,legend=false,linestyle=:dot)
71      Xhat = minimum(X):.1:maximum(X)
72      Xhat = reshape(Xhat,length(Xhat),1) # Make into an n by 1 matrix
73      yhat = model.predict(Xhat)
74      plot!(Xhat,yhat,legend=false)
```

Figure 4: 1.3 - 10-fold cross-validation procedure code Part 3

3

1.3 Cost of Non-Parametric Bases

When dealing with larger datasets, an important issue is the dependence of the computational cost on the number of training examples n and the number of features d.

1. What is the cost in big-O notation of training a linear regression model with Gaussian RBFs on n training examples with d features (for fixed σ and λ)?

   Answer: $O(n^2 d + n^3)$

2. What is the cost of classifying t new examples with this model?

   Answer: O(tnd)

3. When is it cheaper to train using Gaussian RBFs than using the original linear basis?

   Answer: When d is larger than n it is cheaper to use.

4. When is it cheaper to predict using Gaussian RBFs than using the original linear basis?

   Answer: Never, since the prediction cost of the original basis is O(td)

2   Logistic Regression with Sparse Regularization

If you run the function *example_logistic.jl*, it will:

1. Load a binary classification dataset containing a training and a validation set.

2. "Standardize" the columns of X and add a bias variable.

3. Apply the same transformation to $Xvalidate$.

4. Fit a least squares model, using the sign of $w^T x_i$ to make predictions.

5. Report the number of features selected by the model (number of non-zero regression weights).

6. Report the error on the training and validation sets.

Least squares does ok as a binary classifier on this dataset, but it uses all the features (even though only the prime-numbered features are relevant) and the validation error is above the minimum achievable for this model (which is 1 percent, if you have enough data and know which features are relevant). In this question, you will modify this demo to use the logistic loss and to use different forms of regularization to improve on these aspects.

2.1   Logistic Regression

Instead of least squares, modify the script to use logistic regression. You can use the *logReg.jl* file, which implements the training and prediction function for a logistic regresion classifier (using a version of the *findMin* function that does derivative checking for you and that uses more-clever choices of step-sizes). When you switch to using logistic regression, report how the following quantities change: the training error, validation error, and number of features.

Answer:
Training error: 0.038 to 0.0
Validation error: 0.106 to 0.082
Number of features: 101 to 101

2.2   L2-Regularization

Make a new function, *logRegL2*, that takes an input parameter λ and fits a logistic regression model with L2-regularization. Specifically, while *logReg* computes w by minimizing

$$f(w) = \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i)),$$

your new function *logRegL2* should compute w by minimizing

$$f(w) = \sum_{i=1}^{n} \left[\log(1 + \exp(-y_i w^T x_i)) \right] + \frac{\lambda}{2} \|w\|^2.$$

Hand in the objective function that your updated code minimizes, and using $\lambda = 1.0$ report how the following quantities change: the training error, the validation error, the number of features used, and the number of gradient descent iterations.

Answer:
Training error: 0.0 to 0.002
Validation error: 0.082 to 0.074
Number of features: 101 to 101
30 iterations of gradient descent

```
function logisticObjL2(w,X,y,lambda)
    yXw = y.*(X*w)
    f = sum(log.(1 .+ exp.(-yXw)))+(lambda/2)*sqrt(sum(w'w))
    g = -X'*(y./(1 .+ exp.(yXw)))
    return (f,g)
end
```

Figure 5: Function to minimize for L2-Regularization

2.3   L1-Regularization

Make a new function, *logRegL1*, that takes an input parameter λ and fits a logistic regression model with L1-regularization,

$$f(w) = \sum_{i=1}^{n} \left[\log(1 + \exp(-y_i w^T x_i)) \right] + \lambda \|w\|_1.$$

Hand in your *logRegL1* code. Using this new code and $\lambda = 1$, report the following quantities: the training error, the validation error, and the number of features the model uses.

You should use the function *findMinL1*, which implements a proximal-gradient method to minimize the sum of a differentiable function g and $\lambda \|w\|_1$,

$$f(w) = g(w) + \lambda \|w\|_1.$$

This function has a similar interface to *findMin*, except that you (a) only provide the code to compute the function/gradient of the differentiable part g and (b) need to provide the value λ.

```
function logisticObjL1(w,X,y,lambda)
    yXw = y.*(X*w)
    f = sum(log.(1 .+ exp.(-yXw)))+(lambda)*(sum(broadcast(abs, w)))
    g = -X'*(y./(1 .+ exp.(yXw)))
    return (f,g)
end
```

Figure 6: Function to minimize for L3-Regularization

Answer:
Training error: 0.002
Validation error: 0.052
Number of features: 74

6

2.4   L0-Regularization

The function *logRegL0* contains part of the code needed to implement the *forward selection* algorithm, which approximates the solution with L0-regularization,

$$f(w) = \sum_{i=1}^{n} \left[\log(1 + \exp(-y_i w^T x_i)) \right] + \lambda \|w\|_0.$$

The 'for' loop in this function is missing the part where we fit the model using the subset *Sj*, then compute the score and updates the *minScore/minS*. Modify the 'for' loop in this code so that it fits the model using only the features *Sj*, computes the score above using these features, and updates the *minScore/minS* variables (if you want to turn off the diagonistics generated by *findMin*, you can use *verbose = false*).[2] Hand in your updated code. Using this new code, set $\lambda = 1$ and report: the training error, the validation error, and the number of features used.

Note that the code differs a bit from what we discussed in class, since we assume that the first feature is the bias variable and assume that the bias variable is always included. Also, note that for this particular case using the L0-norm with $\lambda = 1$ is equivalent to what is known as the Akaike information criterion (BIC) for variable selection.

```
for j in setdiff(1:d,S)
    # Fit the model with 'j' added to the feature set 'S'
    # then compute the score and update 'minScore' and 'minS'
    Sj = [S;j]
    Xs = X[:,Sj]
    # PUT YOUR CODE HERE

    w = zeros(length(Sj),1)
    w = findMin(funObj,w,verbose=false)
    (f,~) = funObj(w)
    score = f + lambda*length(S)

    if score < minScore
        minScore = score # Lowest score we've found
        minS = Sj # Best set of features we've found
    end

end
```

Figure 7: Function to minimize for L0-Regularization

Answer:
Training error: 0.0
Validation error: 0.018
Number of features: 24

---

[2]Note that Julia doesn't like when you re-define functions, but if you change the variable *Xs* it will actually change the behaviour of the *funObj* that is already defined.

2.5 L1-Regularization vs. L0-Regularization

For this problem, the relevant features are the bias variable and the featurs with prime numbers. Given this, explain how each of the 3 regularizers (L2-regularization, L1-regularization, and L0-regularization) performed in terms of false positives for feature selection (a false positive would be when a feature is selected but it is not relevant). And then explain how each method did in terms of false negatives..

Answer:
L0-Regularization: 1 false positive & 3 false negatives
L1-Regularization: 48 false positives & 0 false negatives
L2-Regularization: 75 false positives & 0 false negatives

3   Multi-Class Logistic

The function *example_multiClass* loads a multi-class classification dataset with $y_i \in \{1, 2, 3, 4, 5\}$ and fits a 'one-vs-all' classification model using binary logistic regression, then reports the validation error and shows a plot of the data/classifier. The performance on the validation set is ok, but could be much better. For example, this classifier never predicts that examples will be in class 1 (corresponding to the blue circles).

3.1   Softmax Classification

Linear classifiers make their decisions by finding the class label c maximizing the quantity $w_c^T x_i$, so we want to train the model to make $w_{y_i}^T x_i$ larger than $w_{c'}^T x_i$ for all the classes c' that are not the true label y_i. Here, c is a possible label and $w_{c'}$ is **row** c' of W. Similarly, y_i is the training label, w_{y_i} is **row** y_i of W, and in this setting we are assuming a discrete label $y_i \in \{1, 2, \ldots, k\}$. Before we move on to implementing the softmax classifier to fix the issues raised in the introduction, let's do a simple example:

Consider the dataset below, which has 30 training examples, 2 features, and 3 class labels:
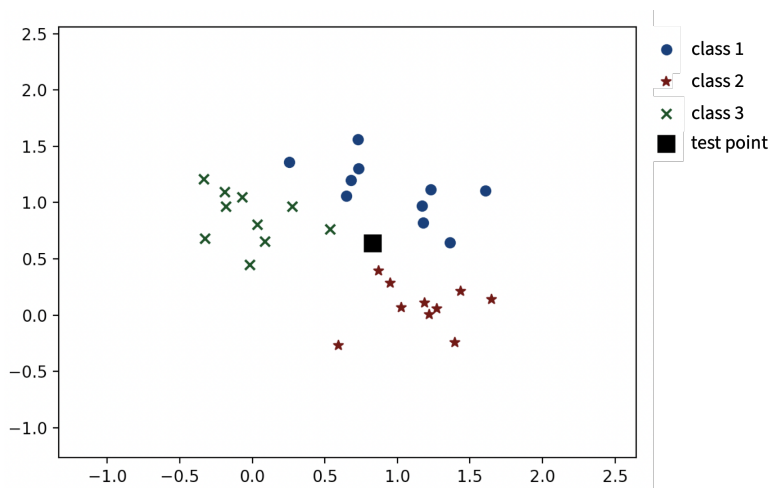
Suppose that we want to classify the black square at the location

$$\hat{x} = \begin{bmatrix} +0.84 \\ +0.64 \end{bmatrix}.$$

Suppose that we fit a multi-class linear classifier including bias variable using the softmax loss. We obtain the weight matrix

$$
\begin{array}{ccc}
x^1 & x^2 & \text{bias}
\end{array}
$$
$$
W = \begin{bmatrix} +1.62 & +3.47 & +0.00 \\ -3.83 & +0.67 & +4.90 \\ +2.22 & -4.13 & +3.37 \end{bmatrix},
$$

where the first two columns correspond to the two features and the *last column* corresponds to the bias variable.

1. What is the meaning of the rows w_i in the matrix W?

   Answer: The row w_i is the set of weights corresponding to the predicted value for class i. In other words, are the weights that multiplied with the corresponding features of a point (plus the bias) are going to give us how probable is the point from class i.

9

2. Under this model, what class label would we assign to the test example $\hat{x}$? (show your work)

Answer: $w_1^T x_1 = 1.62\text{*}0.84 + 3.47\text{*}0.64 + 0.0 = 3.5816$
$w_2^T x_2 = \text{-}3.83\text{*}0.84 + 0.67\text{*}0.64 + 4.9 = 2.1116$
$w_3^T x_3 = 2.22\text{*}0.84\text{-}4.13\text{*}0.64 + 3.37 = 2.5916$
The test example $\hat{x}$ corresponds to class label 1, since its quantity $w_c^T x_i$ is the bigger one.

3.2 Softmax Loss

Using a one-vs-all classifier hurts performance because the classifiers are fit independently, so there is no attempt to calibrate the **rows** of the matrix W. An alternative to this independent model is to use the softmax loss function, which for n training examples is given by

$$f(W) = \sum_{i=1}^{n} \left[-w_{y_i}^T x_i + \log \left(\sum_{c'=1}^{k} \exp(w_{c'}^T x_i) \right) \right].$$

Derive the partial derivative $\frac{\partial f}{\partial W_{cj}}$ of this loss function with respect to a particular element W_{cj} (the variable in row c and column j of the matrix W). Try to simplify the derivative as much as possible (but you can express the result in summation notation).

Hint: for the gradient you can use x_{ij} to refer to element j of example i. For the first term you will need to separately think about the cases where $c = y_i$ and the cases where $c \neq y_i$. You may find it helpful to use an 'indicator' function, $I(y_i = c)$, which is 1 when $y_i = c$ and is 0 otherwise. Note that you can use the definition of the softmax probability to simplify the second term of the derivative.

Answer: $\frac{\partial f}{\partial W_{jc}} = \sum_{i=1}^{n} \left[-I(y_i = c) * x_{ij} + \frac{x_{ij} \exp(w_c^T x_i)}{\sum_{c'=1}^{k} \exp(w_{c'}^T x_i)} \right] = \sum_{i=1}^{n} \left[-I(y_i = c) * x_{ij} + \frac{x_{ij} \exp(z_c)}{\sum_{c'=1}^{k} \exp(z_{c'})} \right] =$

$= \sum_{i=1}^{n} \left[-I(y_i = c) * x_{ij} + x_{ij} * p(y = c | z_1, z_2, z_3, ..., z_k) \right] = \sum_{i=1}^{n} \left[x_{ij} * (p(y = c | z_1, z_2, z_3, ..., z_k) - I(y_i = c)) \right]$

where $z_c = w_c^T x_i$, and $p(y = c | z_1, z_2, z_3, ..., z_k)$ is the softmax function probability of y being class c.

3.3 Softmax Classifier

Make a new function, *softmaxClassifier*, which fits W using the softmax loss from the previous section instead of fitting k independent classifiers. Hand in the code and report the validation error.

Hint: you will want to use the *derivativeCheck* option in *findMin.jl* to check that your gradient code is correct. Also, note that *findMin.jl* expects that the parameter vector and gradient are *column vectors*. The easiest way to work around these issues is to use the *reshape* command: call *findMin.jl* with a $dk \times 1$ vector w and at the start of your objective function, reshape w to be a $k \times d$ matrix W, then compute the $k \times d$ matrix of partial derivatives, and finally reshape this to be the $dk \times 1$ gradient vector.

```
115   function softMaxObj(w,X,y,k)
116       (n,d) = size(X)
117       w = reshape(w, (k,d))
118
119       f= 0
120       for i in 1:n
121           f += (-w[y[i],:]'X[i,:]) + log(sum(exp.(w*(X[i,:])))))
122       end
123
124       g = zeros(k,d)
125       for c in 1:k
126           for j in 1:d
127               g2 = 0
128               for i in 1:n
129                   g2+= -(y[i] == c)*X[i,j] +X[i,j] * (exp(w[c,:]'X[i,:]))/sum(exp.(w*(X[i,:])))
130               end
131               g[c,j] = g2
132           end
133       end
134
135       g = reshape(g, (k*d,1))
136
137       return (f,g)
138   end
139
```

Figure 8: 3.3 - SoftMax Objective Function code

```
140   function softmaxClassifier(X,y)
141
142       (n,d) = size(X)
143       k = maximum(y)
144
145       # Each row of 'w' will be a logistic regression classifier
146       W = zeros(k*d,1)
147
148       # Function we're going to minimize (and that computes gradient)
149       funObj(W) = softMaxObj(W,X,y,k)
150
151       # Solve least squares problem
152       W = findMin(funObj,W,derivativeCheck=true)
153
154       W = reshape(W, (k,d))
155
156       # Make linear prediction function
157       predict(Xhat) = mapslices(argmax,Xhat*W',dims=2)
158
159       return LinearModel(predict,W)
160   end
```

Figure 9: 3.3 - softmaxClassifier function code

Answer: Validation error: 0.026

3.4   Cost of Multinomial Logistic Regression

Assuming that we have

- n training examples.
- d features.
- k classes.
- t testing examples.
- T iterations of gradient descent for training.

1. In big-$\mathcal{O}$ notation, what is the cost of training the softmax classifier?

2. In big-$\mathcal{O}$ notation, what is the cost of classifying the test examples?

Answer:
Training cost $\rightarrow$ O(ndkT)
Classify cost $\rightarrow$ O(tdk)

4 Very-Short Answer Questions

1. If we fit a linear regression model and then remove all features whose associated weight is small, why is this an ineffective way of performing feature selection?

   Answer: If there 2 irrelevant features they should have a weight of 0 but if their values are such that $value(f_1) = -value(f_2)$, they could have an infinitely large weight, their sum would be 0 and so even though they are irrelevant, their weights would be very large.

2. Given 3 features $\{f_1, f_2, f_3\}$, provide an argument that illustrates why the forward selection algorithm is not guaranteed to find an optimal subset of features.

   Answer: A greedy model will always start by picking f_1 as it will assess optimality at an individual level. Therefore, it will always have f_1 as a feature, while also picking others, even though having just f_2 and f_3 might provide a better solution. Individual optimality will not alwatys ensure global optimality.

3. What is a setting where you would use the L1-loss, and what is a setting where you would use L1-regularization?

   Answer:
   L1-Loss: To be used when training dataset contains some outliers.
   L1-Regularization: To be used for feature selection and prevent overfitting.

4. Among L0-regularization, L1-regularization, and L2-regularization: which yield convex objectives? Which yield unique solutions? Which yield sparse solutions?

   Answer:
   Convex objectives: L1-Regularization & L2-Regularization
   Unique solutions: L2-Regularization
   Sparse solutions: L1-Regularization & L0-Regularization

5. What is the effect of λ in L1-regularization on the sparsity level of the solution? What is the effect of λ on the two parts of the fundamental trade-off?

   Answer: Given that λ gives the severity of the penalty to give errors in predictions, when λ decreases so does the importance given to the penalty term. Therefore a smaller λ will yield a greater sparsity level as there will be more features and vice versa. Therefore the effect of λ on the fundamental trade off is as follows:
   λ increase $\rightarrow$ higher train error $\rightarrow$ lower test error
   λ decrease $\rightarrow$ lower train error $\rightarrow$ higher test error

6. Suppose you have a feature selection method that tends not generate false positives but has many false negatives (it misses relevant variables). Describe an ensemble method for feature selection that could improve the performance of this method.

   Answer: Bootstrap samples from the training data and perform feature selection by taking the ones that are above the threshold.

7. How does the hyper-parameter σ affect the shape of the Gaussian RBFs bumps? How does it affect the fundamental tradeoff?

   Answer:
   σ decrease $\rightarrow$ decrease in width of bump $\rightarrow$ overfit $\rightarrow$ fall in training error $\rightarrow$ increase in test error
   σ increase $\rightarrow$ increase in width of bump $\rightarrow$ underfit $\rightarrow$ increase in training error $\rightarrow$ fall in test error

8. What is the main problem with using least squares to fit a linear model for binary classification?

Answer: Least squares will predict a continuous value. Therefore in a classification model such as predicting +1 or -1 predictions could be any number. Therefore, even though our goal might be to predict whether a value is positive or negative, even if most of the predictions have the correct sign, the returned error could still be really high.

9. Suppose a binary classification dataset has 3 features. If this dataset is "linearly separable", what does this precisely mean in three-dimensional space?

Answer: It means that there exists a plane that can accurately separate the data into correct groups.

10. Why do we not minimize $\max(0, -y_i w^\top x_i)$ when we fit a binary linear classifier, even though it's a convex approximation to the 0-1 loss?

Answer: Because it has a degenerate solution: f(0)=0 (The lowest possible value of f). However it can be fixed by using hinge loss or logistic loss.

11. For a linearly-separable binary classification problem, how does an SVM classifier differ from a classifier found using the perceptron algorithm?

Answer: The perceptron algorithm will generate a different hyperplane solely based on the initial weights. SVM keeps a classification margin on each side so that it classifies test data points that come near to the boundary properly. This makes SVM more robust to get to the real boundary properly

12. Which of the following methods produce linear classifiers? (a) binary least squares as in Question 3, (b) the perceptron algorithm, (c) SVMs, (d) logistic regression, and (e) KNN with $k = 1$ and $n = 2$.

Answer:
Binary least squares
SVMs
Perceptron algorithm
Logistic regression
KNN with k=1 and n=2

13. Why do we use the polynomial kernel to implement the polynomial basis when d and p (degree of polynomial) are large?

Answer: Because for large d and p values, storing the polynomial becomes intractable. Having $O(d^p)$ columns does not fit in memory.

14. What is the relationship between the softmax loss and the softmax function?

Answer: Softmax loss is the negative logarithm of the probabilities obtained by the function.

Project Proposal (OPTIONAL FOR 340 STUDENTS)

For 532M students, there is a project component to the course that will be worth 20% of your final grade. For 340 students, there is no requirement to do a project. However, 340 students have the option to do a project anyway for the possibility of obtaining a higher grade: your project grade can replace either your 2 lowest assignment scores or your midterm score (whicheve helps you more).[3]

These projects are done in groups of 2-3. The final deliverable will be a 6-page report that is due near the end of the exam period (something like December 22nd, minus a few days so we have time to grade). It is expected that this project will be a literature survey, but research projects are also ok.

There aren't really any restrictions on the group compositions: 340 students can work with 532M students, auditors can work with registered students, and you can combine this project with a project from another one of your classes (assuming you get the other instructor's permission, and even if not all students in the other class are registered in this class). The only combinations I really want to avoid are having students do projects with the TAs (due to the obvious conflict of interest), project groups that have no students enrolled in 340 or 532M, or projects that contain people taking no CPSC classes.

If you are in 532M, or in 340 and want to do a project, for the final part of this assignment you must a submit a project proposal for your course project. The proposal should be a maximum of 2 pages (and 1 page or half of a page is ok if you can describe your plan concisely). The proposal should be written for the instructors and the TAs, so you don't need to introduce any ML background but you will need to introduce non-ML topics.

You should submit this question as a group on Gradescope, separate from the other assignment questions.

There is quite a bit of flexibility in terms of the type of project you do, as I believe there are many ways that people can make valuable contributions to research. However, note that ultimately the final deliverable for the project will be a report that emphasizes a particular "contribution" (i.e., what doing the project has added to the world). The reason for this, even though it's strange for some possible projects, is that this is the standard way that results are communicated to the research community.

The three mains ingredients of the project proposal are:

1. What problem you are focusing on.

2. What you plan to do.

3. What will be the "contribution".

Also, for the course project note that negative results (i.e., we tried something that we thought would work in a particular setting but it didn't work) are acceptable (and often unavoidable).

I encourage you to follow the following default "template" for the project:

1. **Literature review**: you pick a specific topic in ML, read at least 10 papers on the topic, then write a report summarizing what has been done on the topic and what are the most promising directions of future work. In this case, the contribution would be your summary of the relationships between the existing works, and your insights about where the field is going.

The advantage of the above template is that the project will take a somewhat-predictable amount of time. If you want to explore a different style of project, here are some standard "templates":

2. **Application bake-off**: you pick a specific application (from your research, personal interests, or maybe from Kaggle) or a small number of related applications, and try out a bunch of techniques (e.g., random forests vs. logistic regression vs. generative models). In this case, the contribution would be

---

[3]The course is *not* graded on a curve, so 340 students are not hurt by choosing the to skip the project.

showing that some methods work better than others for this specific application (or your contribution could be that everything works equally well/badly).

3. **New application**: you pick an application where people aren't using ML, and you test out whether ML methods are effective for the task. In this case, the contribution would be knowing whether ML is suitable for the task.

4. **Scaling up**: you pick a specific machine learning technique, and you try to figure out how to make it run faster or on larger datasets. In this case, the contribution would be the new technique and an evaluation of its performance, or could be a comparison of different ways to address the problem.

5. **Improving performance**: you pick a specific machine learning technique, and try to extend it in some way to improve its performance. In this case, the contribution would be the new technique and an evaluation of its performance.

6. **Generalization to new setting**: you pick a specific machine learning technique, and try to extend it to a new setting (for example, making a multi-label version of random forests). In this case, the contribution would be the new technique and an evaluation of its performance, or could be a comparison of different ways to address the problem.

7. **Coding project**: you pick a specific method or set of methods, and build an implementation of them. In this case, the contribution could be the implementation itself or a comparison of different ways to solve the problem.

8. **Theory**: you pick a theoretical topic (like the variance of cross-validation), read what has been done about it, and try to prove a new result (usually by relaxing existing assumptions or adding new assumptions). The contribution could be a new analysis of an existing method, or why some approaches to analyzing the method will not work.

9. **Reproduction**: you take a recent paper published in one of the top machine learning venue (either NeurIPS, ICML, ICLR, AI/Stats, or JMLR), and try to reproduce the results in the paper.

The above are just suggestions, and projects could mix several of these templates together, but if you are having trouble getting going then it's best to stick with one of the above templates (and again I recommend doing the literature review). Also note that the project can focus on topics not covered in the course (like GANs), so there is flexibility in the topic, but the topic should be closely-related to ML.

This question is mandatory but will not be formally marked: it is just a sanity check that you have at least one project idea that has an appropriate topic and scope, that you find a group early, and that you allocate some time to thinking about the project. Also, there is flexibility in the choice of project topics even after the proposal: if you want to explore different topics you can ultimately choose to do a project that is unrelated to the one in your proposal (and changing groups is ok too). If you aren't sure what to do, go bug the TAs in office hours (which is a good idea even if you are sure what you want to).