CPSC 540 Assignment 1 (due January 11th at midnight)

IMPORTANT!!!!! Before proceeding, please carefully read the homework instructions: `www.cs.ubc.ca/~schmidtm/Courses/540-W18/assignments.pdf`

We will deduct 50% on assignments that do not follow the instructions.

Most of the questions below are related to topics covered in CPSC 340, or other courses listed on the prerequisite form. There are several "notes" available on the webpage which can help with some relevant background.

If you find this assignment to be difficult overall, that is an early warning sign that you may not be prepared to take CPSC 540 at this time. Future assignments will be longer and more difficult than this one.

We use blue to highlight the deliverables that you must answer/do/submit with the assignment.

Basic Information

1. Name: Dingqing Yang
2. Student ID: 38800141
3. Graduate students in CPSC/EECE/STAT must submit the prerequisite form as part of a1sol.zip: `https://www.cs.ubc.ca/~schmidtm/Courses/540_prereqs.pdf`

1 Very-Short Answer Questions

Give a short and concise 1-2 sentence answer to the below questions.

1. Why was I unimpressed when a student who thought they did not need to take CPSC 340 said they were able to obtain 97% training accuracy (on their high-dimensional supervised learning problem) as the main result of their MSc thesis?
   Our goal is to minimize test error instead of training error.
2. What is the difference between a test set error and the test error?
   Test error is expected error of our model. Test set error is error on test set. We use test set error to approximate test error.
3. Suppose that a famous person in the machine learning community is advertising their "extremely-deep convolutional fuzzy-genetic Hilbert-long-short recurrent neural network" classifier, which has 500 hyper-parameters. This person claims that if you take 10 different famous (and very-difficult) datasets, and tune the 500 hyper-parameters based on each dataset's validation set, that you can beat the current best-known validation set error on all 10 datasets. Explain whether or not this amazing claim is likely to be meaningful.
   Probably not meaningful. Tuning 500 hyperparameter results in huge optimization bias.
4. In a parametric model, what is the effect of the number of training examples n that our model uses on the training error and on the approximation error (the difference between the training error and test

error)?
As we use more training examples, E_{train} reduces and plateaus, E_{approx}will reduce

5. Give a way to set the random tree depth in a random forest model that makes the model parametric, and a choice that makes the model non-parametric.
When fixing maximum tree depth, random forest is parametric model. Otherwise it is a non-parametric model.

6. In the regression setting, the popular software XGBoost uses the squared error at the leaves of its regression tree, which is different than the "number of training errors" ($\sum_{i=1}^{n}(\hat{y}^i \neq y^i)$) we used for decision trees in 340. Why does it use the squared error instead of the number of training errors?
"number of training error" only works in classification setting where label is discrete. In regression setting, labels are continuous.

7. Describe a situation where it could be better to use gradient descent than Newton's method (known as IRLS in statistics) to fit the parameters of a logistic regression model.
In a situation that one iteration of Newton method too costly (i.e. d huge)

8. How does λ in an L2-regularizer affect the sparsity pattern of the solution (number of w_j set to exactly 0), the training error, and the approximation error?
No Sparsity, increase training error, reduce approximation error

9. Minimizing the squared error used by in k-means clustering is NP-hard. Given this, does it make sense that the standard k-means algorithm is easily able to find a local optimum?
No

10. Suppose that a matrix X is non-singular. What is the relationship between the condition number of the matrix, $\kappa(X)$, and the matrix L2-norm of the matrix, $\|X\|_2$.
$\kappa(X) = \|X^{-1}\|\|X\|$

11. How many regression weights do we have in a multi-class logistic regression problem with k classes?
kd

12. Give a supervised learning scenario where you would use the sigmoid likelihood and one where you would use a Poisson likelihood.
Sigmoid likelihood: binary classification.
Poisson likelihood: predict number of times that something happens.

13. Suppose we need to multiply a huge number of matrices to compute a product like $A_1A_2A_3\cdots A_k$. The matrices have wildly-different sizes so the order of multiplication will affect the runtime (e.g., $A_1(A_2A_3)$ may be faster to compute than $(A_1A_2)A_3$). Describe (at a high level) an $O(k^3)$-time algorithm that finds the lowest-cost order to multiply the matrices.
Use dynamic programming to solve this. Divide the sequence into sub-sequences, evaluate sub-sequences cost and the cost for multiplying resulting. Use memoization to avoid evaluating sub-problem again

14. You have a supervised learning dataset $\{X, y\}$. You fit a 1-hidden-layer neural network using stochastic gradient descent to minimize the squared error, that makes predictions of the form $\hat{y}^i = v^\top W x^i$ where W and v are the parameters. You find that this gives the same training error as using the linear model ($\hat{y}^i = w^\top x^i$) that minimizes the squared error. You thought the accuracy might be higher for the neural network. Explain why or why not this result is reasonable.
Did not introduce non-linearity (activation), so this is still a linear model. Definitely reasonable to have same accuracy.

15. Is it possible that the neural network and training procedure from the previous question results in a higher training error than the linear least squares model? Is it possible that it results in a lower training error?

SGD only guarantees convergence within a ball, so it is reasonable for the trained model to have different accuracy.

16. What are two reasons that convolutional neural networks overfit less than classic neural networks?
Weight is sparse and contains repeated partterns (i.e. filter shared.)

2 Calculation Questions

2.1 Minimizing Strictly-Convex Quadratic Functions

Solve for the minimizer w of the below strictly-convex quadratic functions:

1. $f(w) = \frac{1}{2}\|w-u\|_\Sigma$ (projection of u onto the real space under the quadratic norm defined by Σ).
2. $f(w) = \frac{1}{2\sigma^2}\|Xw-y\|^2 + w^\top \Lambda w$ (ridge regression with known variance and weighted L2-regularization).
3. $f(w) = \frac{1}{2}\sum_{i=1}^n v_i(w^\top x^i - y^i)^2 + \frac{1}{2}(w-u)^\top \Lambda(w-u)$ (weighted least squares shrunk towards u).

Above we use our usual supervised learning notation. In addition, we assume that u is $d \times 1$ and v is $n \times 1$, while Σ and Λ are symmetric positive-definite $d \times d$ matrices. You can use V as a diagonal matrix with v along the diagonal (with the v_i non-negative). Hint: positive-definite matrices are invertible.

1. Any arbitrary norm $\|x\|_p$ is minimized at $x = 0$, therefore, the minimizer $w^* = u$
2. $f(w) = \frac{1}{2\sigma^2}(w^\top X^\top Xw - 2y^\top Xw + y^\top y) + w^\top Aw$. Therefore, $\nabla f(w) = (\frac{1}{\sigma^2}X^\top X + 2\Lambda)w - \frac{1}{\sigma^2}X^\top y = 0$.

   Since $\frac{1}{\sigma^2}X^\top X + 2\Lambda \succ 0$, the minimizer $w^* = (\frac{1}{\sigma^2}X^\top X + 2\Lambda)^{-1}X^\top y$.
3. $f(w) = \frac{1}{2}(Xw-y)^\top V(Xw-y) + \frac{1}{2}(w-u)^\top \Lambda(w-u)$

   $\nabla f(w) = X^\top VXw - X^\top Vy + \Lambda(w-u) = 0$.

   $(X^\top VX + \Lambda)w = X^\top Vy + \Lambda u$.

   Since $X^\top VX + \Lambda \succ 0$, the minimizer $w^* = (X^\top VX + \Lambda)^{-1}X^\top Vy$

2.2 Norm Inequalities

Show that the following inequalities hold for vectors $w \in \mathbb{R}^d$, $u \in \mathbb{R}^d$, and $X \in \mathbb{R}^{n\times d}$:

1. $\|w\|_\infty \leq \|w\|_2 \leq \|w\|_1$ (relationship between decreasing p-norms)
2. $\frac{1}{2}\|w+u\|_2^2 \leq \|w\|_2^2 + \|u\|_2^2$ ("not the triangle inequality" inequality)
3. $\|X\|_2 \leq \|X\|_F$ (matrix norm induced by L2-norm is smaller than Frobenius norm)

You should use the definitions of the norms, but should not use the known equivalences between these norms (since these are the things you are trying to prove). Hint: for many of these it's easier if you work with squared values (and you may need to "complete the square"). Beyond non-negativity of norms, it may also help to use the Cauchy-Schwartz inequality, to use that $\|x\|_1 = x^\top \text{sign}(x)$, and to use that $\sum_{i=1}^n \sum_{j=1}^d x_{ij}^2 = \sum_{c=1}^{\min\{n,d\}} \sigma_c(X)^2$ (where $\sigma_c(X)$ is singular value c of X).

1. This is equivalent to show (i) $\|w\|_1 \geq \|w\|_2$ and (ii) $\|w\|_2 \geq \|w\|_\infty$

   To show (i), we can show $\|w\|_1^2 \geq \|w\|_2^2$.

$\|w\|_1^2 - \|w\|_2^2 = (\sum_{i=1}^d |w_i|)^2 - \sum_{i=1}^d w_i^2 = \sum_{i \neq j} 2|w_i||w_j| \geq 0$

Therefore, $\|w\|_1 \geq \|w\|_2$

To show (ii), we can show $\|w\|_2^2 \geq \|w\|_\infty^2$.

$\|w\|_2^2 - \|w\|_\infty^2 = \sum_{i=1}^d w_i^2 - \max_i\{w_i^2\} \geq 0$

Therefore, $\|w\|_2 \geq \|w\|_\infty$

2. Move LHS of the inequality to the RHS and expand the squared norm, we have

   $\frac{1}{2}w^\top w - w^\top u + \frac{1}{2}u^\top u \geq 0$

   Complete the square and we get $\frac{1}{2}\|w-u\|^2 \geq 0$

3. $\|X\|_2^2 = \max_c\{\sigma_c(X)^2\} \leq \sum_{c=1}^{\min\{n,d\}} \sigma_c(X)^2 = \|X\|_F^2$

2.3 MAP Estimation

In 340, we showed that under the assumptions of a Gaussian likelihood and Gaussian prior,

$$y^i \sim \mathcal{N}(w^\top x^i, 1), \quad w_j \sim \mathcal{N}\left(0, \frac{1}{\lambda}\right),$$

that the MAP estimate is equivalent to solving the L2-regularized least squares problem

$$f(w) = \frac{1}{2}\sum_{i=1}^n (w^\top x^i - y^i)^2 + \frac{\lambda}{2}\sum_{j=1}^d w_j^2,$$

in the "loss plus regularizer" framework. For each of the alternate assumptions below, write it in the "loss plus regularizer" framework (simplifying as much as possible):

1. Laplace likelihood (with a scale of 1) for each training example and Gaussian prior with separate variance σ_j^2 for each variable

   $$y^i \sim \mathcal{L}(w^\top x^i, 1), \quad w_j \sim \mathcal{N}\left(0, \sigma_j^2\right).$$

2. Robust student-t likelihood and Gaussian prior centered at u.

   $$p(y^i|x^i, w) = \frac{1}{\sqrt{\nu}B\left(\frac{1}{2}, \frac{\nu}{2}\right)}\left(1 + \frac{(w^\top x^i - y^i)^2}{\nu}\right)^{-\frac{\nu+1}{2}}, \quad w_j \sim \mathcal{N}\left(u_j, \frac{1}{\lambda}\right),$$

   where u is $d \times 1$, B is the "Beta" function, and the parameter ν is called the "degrees of freedom".[1]

3. We use a Poisson-distributed likelihood (for the case where y_i represents counts), and we use a uniform prior for some constant κ,

   $$p(y^i|w^\top x^i) = \frac{\exp(y^i w^\top x^i)\exp(-\exp(w^\top x^i))}{y^i!}, \quad p(w_j) \propto \kappa.$$

   (This prior is "improper" since $w \in \mathbb{R}^d$ but it doesn't integrate to 1 over this domain, but nevertheless the posterior will be a proper distribution.)

[1] This likelihood is more robust than the Laplace likelihood, but leads to a non-convex objective.

For this question, you do not need to convert to matrix notation.

1. $f(w) = \frac{1}{2}\sum_{i=1}^{n} |w^\top x^i - y^i| + \frac{1}{2}\sum_{j=1}^{d}(w_j/\sigma_j)^2$
2. $f(w) = -\frac{\nu+1}{2}\sum_{i=1}^{n} \log\left(1 + \frac{(w^\top x^i - y^i)^2}{\nu}\right) + \frac{\lambda}{2}\sum_{j=1}^{d}(w_j - \mu_j)^2$
3. $f(w) = \sum_{i=1}^{n} -y^i w^\top x^i + \exp(w^\top x^i)$

2.4 Gradients and Hessian in Matrix Notation

Express the gradient $\nabla f(w)$ and Hessian $\nabla^2 f(w)$ of the following functions in matrix notation, simplifying as much as possible:

1. Regularized and tilted Least Squares

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2 + w^\top u.$$

   wher u is $d \times 1$.

2. L2-regularized weighted least squares with non-Euclidean quadratic regularizaiton,

$$f(w) = \frac{1}{2}\sum_{i=1}^{n} v_i(w^\top x^i - y^i)^2 + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} w_i w_j \lambda_{ij}$$

   where you can use V as a matrix with the v_i along the diagonal and Λ as a positive-definite $d \times d$ (symmetric) matrix with λ_{ij} in position (i, j).

3. Squared hinge loss,

$$f(w) = \frac{1}{2}\sum_{i=1}^{n} \left(\max\{0, 1 - y^i w^\top x^i\}\right)^2.$$

Hint: You can use the results from the linear and quadratic gradients and Hessians notes to simplify the derivations. You can use 0 to represent the zero vector or a matrix of zeroes and I to denote the identity matrix. It will help to convert the second question to matrix notation first. For the last question you'll need to define new vectors to express the gradient and Hessian in matrix notation and you can use $\circ$ as element-wise multiplication of vectors. As a sanity check, make sure that your results have the right dimension.

1. $f(w) = \frac{1}{2}(w^\top X^\top X w - 2y^\top X w + y^\top y) + \frac{\lambda}{2} w^\top w + w^\top u.$

   $\nabla f(w) = (X^\top X + \lambda I)w - X^\top y + u.$

   $\nabla^2 f(w) = X^\top X + \lambda I$

2. $f(w) = \frac{1}{2}(Xw - y)^\top V(Xw - y) + \frac{1}{2} w^\top \Lambda w$

   $\nabla f(w) = (X^\top V X + \Lambda)w - X^\top V y$

   $\nabla^2 f(w) = X^\top V X + \Lambda$

3. $\partial f / \partial w_j = \sum_{i=1}^{n} x_j^i y^i \delta(1 - y^i w^\top x^i)$,

   where p is the defined as $p(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise.} \end{cases}$

   $\therefore \nabla f(w) = X^\top r$, with $r = y \circ P(\mathbb{1} - y \circ Xw)$

where $\circ$ denote element-wise vector product, $\mathbb{1}$ represent d dimensional vector with all element = 1, and P is defined as above but applied element-wise on the vector.

$\partial^2 f/\partial w_j \partial w_k = \sum_{i=1}^{n} -x_j^i y^i y^i x_k^i \delta(1 - y^i w^\top x^i)$,

where δ is the indicator function defined as $\delta(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise.} \end{cases}$

$\nabla^2 f(w) = X^\top D X$, where D is a diagonal matrix with $d_{ii} = -(y^i)^2 \delta(1 - y^i w^\top x^i)$

3 Coding Questions

If you have not previously used Julia, there is a list of useful Julia commands (and syntax) among the list of notes on the course webpage.

3.1 Regularization and Hyper-Parameter Tuning

Download *a1.zip* from the course webpage, and start Julia (latest version) in a directory containing the extracted files. If you run the script *example_nonLinear*, it will:

1. Load a one-dimensional regression dataset.
2. Fit a least-squares linear regression model.
3. Report the test set error.
4. Draw a figure showing the training/testing data and what the model looks like.

This script uses the *JLD* package to load the data and the *PyPlot* package to make the plot. If you have not previously used these packages, they can be installed using:[2]

```
using Pkg
Pkg.add("JLD")
Pkg.add("PyPlot")
```

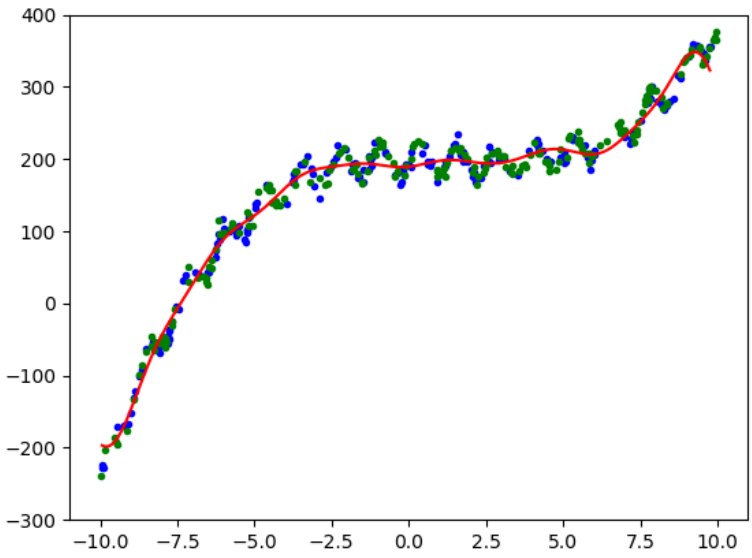
Unfortunately, this is not a great model of the data, and the figure shows that a linear model is probably not suitable.

1. Write a function called *leastSquaresRBFL2* that implements *least squares using Gaussian radial basis functions (RBFs) and L2-regularization.*
   You should start from the *leastSquares* function and use the same conventions: n refers to the number of training examples, d refers to the number of features, X refers to the data matrix, y refers to the targets, Z refers to the data matrix after the change of basis, and so on. Note that you'll have to add two additional input arguments (λ for the regularization parameter and σ for the Gaussian RBF variance) compared to the *leastSquares* function. To make your code easier to understand/debug, you may want to define a new function *rbfBasis* which computes the Gaussian RBFs for a given training set, testing set, and σ value. Hand in your function and the plot generated with $\lambda = 1$ and $\sigma = 1$.
   New testError (loss) = 369.22
   RBF code included in leastSquares.jl:

[2] Last term, several people (eventually including myself) had a runtime problem on some system. This seems to be fixed using the answer of K. Gkinis at this url: `https://stackoverflow.com/questions/46399480/julia-runtime-error-when-using-pyplot`

```
function rbfBasis(X1, X2, sigma)
    # pass Xtest, X will generate Ztest
    # pass X, X will generate Z
    # form Z
    Z = exp.(-distancesSquared(X1,X2)./(2sigma^2))

    return Z
end

function leastSquaresRBFL2(X,y,lambda, sigma)

    n = size(X,1)
    Z = rbfBasis(X, X, sigma)
    # Find regression weights minimizing squared error
    w = (Z'*Z + lambda*Matrix{Float64}(I, n, n))\(Z'*y)

    # Make linear prediction function
    predict(Xtilde) = rbfBasis(Xtilde, X, sigma)*w

    # Return model
    return LinearModel(predict,w)
end
```

plots are included below:



2. 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. What is the cost in big-O notation of training the model on n training examples with d features under (a) the linear basis, and (b) Gaussian

RBFs (for a fixed σ)? What is the cost of classifying t new examples under these two bases? Assume that multiplication by an n by d matrix costs $O(nd)$ and that solving a d by d linear system costs $O(d^3)$.
Training cost: (a) $O(nd^2 + d^3)$, (b) $O(n^3 + dn^2)$
Test Cost: (a) $O(td)$, (b) $O(ntd)$

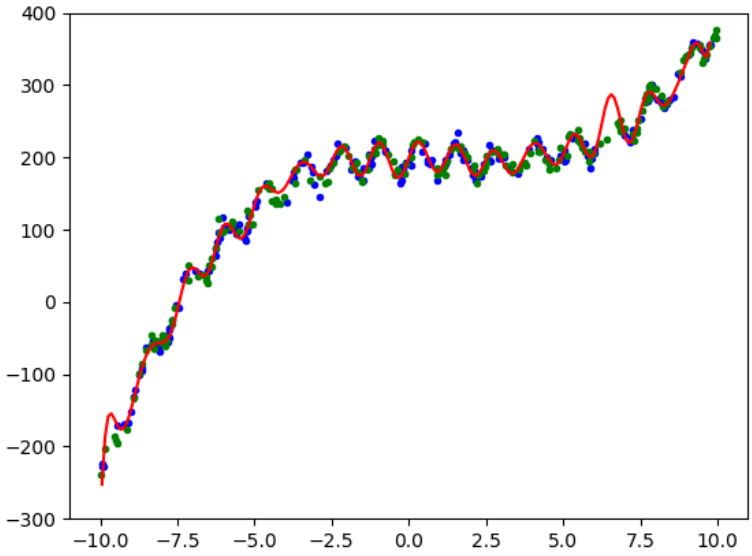
3. Modify the script to split the training data into a "train" and "validation" set (you can use half the examples for training and half for validation), and use these to select λ and σ. Hand in your modified script and the plot you obtain with the best values of λ and σ.
I did a coarse-grained grid search for σ and λ from 0.001, 0.01, 0.1, 1, 10, 100. Then based on the best validation result of coarse-grained search, I applied a fine-grained grid search and obtained the best validation results and hyperparameters as follow:

```
julia> include("example_nonLinear.jl")
Validation Error = 93.44
Best lambda = 0.000
Best sigma = 0.930
(-300, 400)
```

Fine grained grid search code is included in example_nonLinear.jl :

```julia
include("leastSquares.jl")
global best_val = Inf
#println("Xtrain size", size(Xtrain))
for i = 0.00 : 0.001: 0.02
    for j = 0.85 : 0.01 : 1
        lambda = i
        sigma = j
        valErrorSum = 0
        for k = 1:3
            # shuffle training set using random permutation
            shuffled = randperm(n)
            boundary = div(n, 2) # 100 in this case
            train = shuffled[1:boundary]
            val = shuffled[boundary+1:end]

            Xtrain = reshape(X[train], boundary, 1)
            ytrain = y[train]
            Xval = reshape(X[val], boundary, 1)
            yval = y[val]

            model = leastSquaresRBFL2(Xtrain,ytrain,lambda,sigma)
            yhat = model.predict(Xval)
            valErrorSum += sum((yhat - yval).^2)/size(Xval,1)
        end
        valError = valErrorSum/3
        if valError < best_val
            global best_lambda = lambda
            global best_sigma = sigma
            global best_val = valError
        end
    end
end


# Report the error on the validation set
using Printf
@printf("Validation Error = %.2f\n",best_val)
@printf("Best lambda = %.3f\n",best_lambda)
@printf("Best sigma = %.3f\n",best_sigma)
```

Plot is included below:



4. There are reasons why this dataset is particularly well-suited to Gaussian RBFs are that (i) the period of the oscillations stays constant and (ii) we have evenly sampled the training data across its domain. If either of these assumptions are violated, the performance with our Gaussian RBFs might be much worse. Consider a scenario where either (i) or (ii) is violated, and describe a way that you could address this problem.
   Use multiple σ values in one training pass.

Note: the *distancesSquared* function in *misc.jl* is a vectorized way to quickly compute the squared Euclidean distance between all pairs of rows in two matrices.

3.2 Multi-Class Logistic Regression

The script *example_multiClass.jl* loads a multi-class classification dataset and fits a "one-vs-all" logistic regression classifier, 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 even predicts some of the classes.

Using a one-vs-all classifier hurts performance because the classifiers are fit independently, so there is no attempt to calibrate the columns of the matrix W. An alternative to this independent model is to use the softmax probability,

$$p(y^i|W, x^i) = \frac{\exp(w_{y^i}^\top x^i)}{\sum_{c=1}^k \exp(w_c^\top x^i)}.$$

Here c is a possible label and w_c is column c of W. Similarly, y^i is the training label, w_{y^i} is column y^i of W. The loss function corresponding to the negative logarithm of the softmax probability is given by

$$f(W) = \sum_{i=1}^n \left[-w_{y^i}^\top x^i + \log\left(\sum_{c'=1}^k \exp(w_{c'}^\top x^i) \right) \right].$$

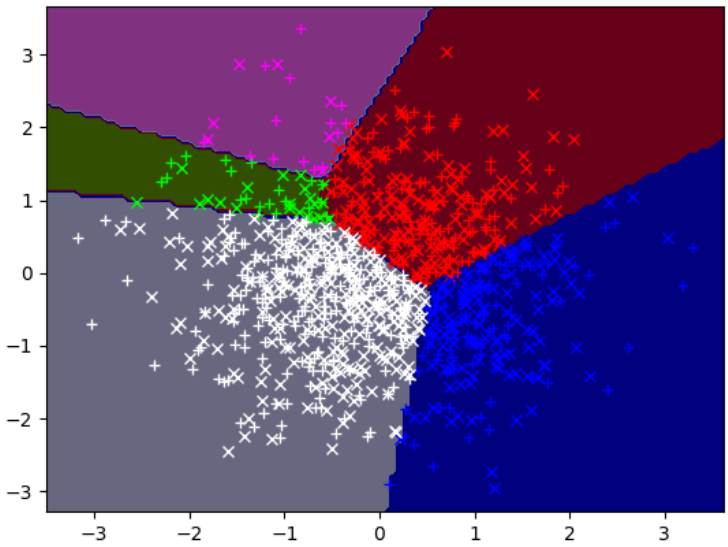
Make a new function, *softmaxClassifier*, which fits W using the softmax loss from the previous section instead of fitting k independent classifiers. Hand in the code and report the validation error.

Softmax function is defined in logReg.jl. The validation error is: 0.026

```
function softmaxObj(w,X,y)
    (n,d) = size(X)
    k = maximum(y)
    W = reshape(w, k, d)

    # objective function
    f = 0
    for i = 1:n
        f += log.(sum(exp.(W*X[i, :])))-dot(W[y[i], :], X[i, :])
    end

    # gradient
    G = zeros(k,d)
    prob = zeros(k)
    for c = 1:k
        for j = 1:d
            for i = 1:n
                num = exp(dot(W[c, :], X[i, :]))
                denom = sum(exp.(W*X[i, :]))
                prob[c] = num/denom
                if c == y[i]
                    prob[c] -= 1
                end
                G[c, j] += X[i, j] * prob[c]
            end
        end
    end
    g = reshape(G, k*d)
    return (f,g)
end
```

```
# Multi-class softmax (assumes y_i in {1,2,...,k})
function softmax(X,y)
    (n,d) = size(X)
    k = maximum(y)

    # Each column of 'w' will be a logistic regression classifier
    W = zeros(k,d)
    w = reshape(W, k*d)

    funObj(w) = softmaxObj(w,X,y)

    w = findMin(funObj,w,derivativeCheck=true,verbose=false)
    W = reshape(w, k, d)

    # Make linear prediction function
    predict(Xhat) = mapslices(argmax,Xhat*W',dims=2)

    return LinearModel(predict,W)
end
```

Plot is included below:



Hint: you can use the *derivativeCheck* option when calling *findMin* to help you debug the gradient of the softmax loss. Also, note that the *findMin* function treats the parameters as a vector (you may want to use *reshape* when writing the softmax objective).

3.3 Robust and Brittle Regression

The script *example_outliers.jl* loads a one-dimensional regression dataset that has a non-trivial number of "outlier" data points. These points do not fit the general trend of the rest of the data, and pull the least squares model away from the main cluster of points. One way to improve the performance in this setting is simply to remove or downweight the outliers. However, in high-dimensions it may be difficult to determine whether points are indeed outliers (or the errors might simply be heavy-tailed). In such cases, it is preferable to replace the squared error with an error that is more robust to outliers.

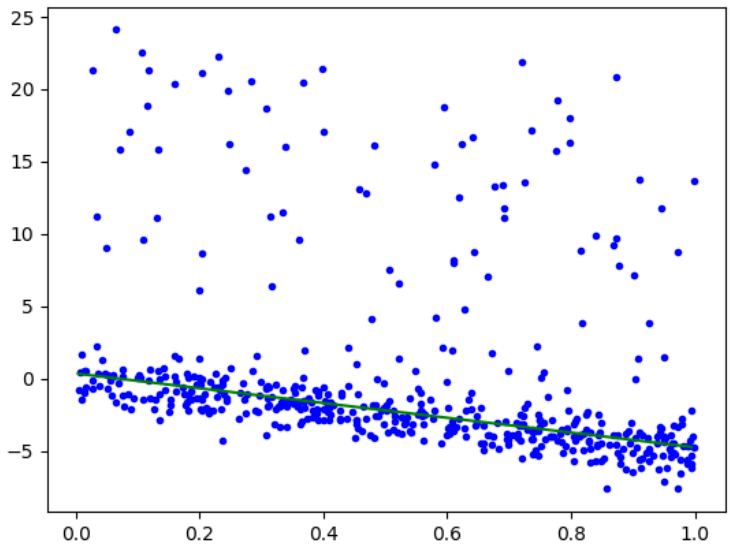
1. Write a new function, *leastAbsolutes(X,y)*, that adds a bias variable and fits a linear regression model by minimizing the absolute error instead of the square error,

$$f(w) = \|Xw - y\|_1.$$

   You should turn this into a *linear program* as shown in class, and you can solve this linear program using the *linprog* function the *MathProgBase* package. Hand in the new function and the updated plot. Function is included in leastSquares.jl

```
function leastAbsolute(X,y)

    # Add bias column
    n, d = size(X)
    Z = [ones(n,1) X]

    # use linear programming to find solution
    c = [zeros(d+1);ones(n)]
    eye = Matrix{Float64}(I, n, n)
    A = [Z eye; Z -eye]
    b = [ones(n)*Inf; y]
    b = reshape(b, 2n)
    d = [y; ones(n)*-Inf]
    d = reshape(d, 2n)
    solution = linprog(c,A,d,b,-Inf,Inf,GLPKSolverLP())
    x = solution.sol
    w = [x[1]; x[2]]

    # Make linear prediction function
    predict(Xtilde) = [ones(size(Xtilde,1),1) Xtilde]*w

    # Return model
    return LinearModel(predict,w)
end
```

the updated plot is:



2. The previous question assumes that the "outliers" are points that we don't want to model. But what if we want good performance in the worst case across *all* examples (including the outliers)? In this setting we may want to consider a "brittle" regression method that chases outliers in order to improve

the worst-case error. For example, consider minimizing the absolute error in the worst-case,

$$f(w) = \|Xw - y\|_\infty.$$

This objective function is non-smooth because of the absolute value function as well as the max function. Show how to formulate this non-smooth optimization problem as a linear program.

$$f(w) = \|Xw - y\|_\infty = \max_i |w^\top x^i - y^i|$$

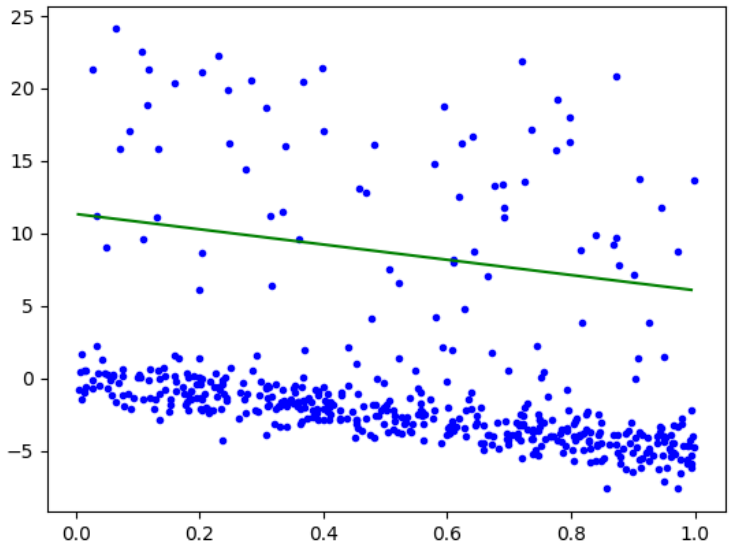
$$\arg\min_w f(w) = \arg\min_{w,t} t,$$

$$\text{where } t \geq w^\top x^i - y^i, t \geq y^i - w^\top x^i, \forall i$$

3. Write and hand in a function, *leastMax*, that fits this model using *linprog* (after adding a bias variable). Hand in the new function and the updated plot.
Function is included in leastSquares.jl

```
function leastMax(X,y)

    # Add bias column
    n, d = size(X)
    Z = [ones(n,1) X]

    # use linear programming to find solution
    c = [zeros(d+1); 1]
    A = [Z ones(n); Z -ones(n)]
    b = [ones(n)*Inf; y]
    b = reshape(b, 2n)
    d = [y; ones(n)*-Inf]
    d = reshape(d, 2n)
    solution = linprog(c,A,d,b,-Inf,Inf,GLPKSolverLP())
    x = solution.sol
    w = [x[1]; x[2]]

    # Make linear prediction function
    predict(Xtilde) = [ones(size(Xtilde,1),1) Xtilde]*w

    # Return model
    return LinearModel(predict,w)
end
```

the updated plot is:



To use the *linprog* function, you can use:

```
using MathProgBase, GLPKMathProgInterface
solution = linprog(c,A,d,b,lb,ub,GLPKSolverLP())
x = solution.sol
```

This requires installing the appropriate packages, and finds a vector x minimizing the function $c^\top x$ subject to $d \leq Ax \leq b$ and $\text{lb} \leq x \leq \text{ub}$. You can set values of c to 0 for variables that don't affect the cost function, and you can set values in $b/d/\text{lb}/\text{ub}$ (or the other variables) to appropriate infinite values if there are no lower/upper bounds. The vectors $c/d/b/lb/ub$ should all be lists.