CPSC 440/540 Machine Learning (Jan-Apr 2023)
Assignment 1 – due Friday January 20th at **noon**

IMPORTANT!!!!! Please carefully read the submission instructions that will be posted on Piazza. We will deduct up to 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 overly difficult, that is an early warning sign that you may not be prepared to take CPSC 440 at this time. Future assignments may be longer and more difficult than this one.

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

Cite any sources outside of course materials that you refer to. Even if you've written code for some of these questions before (e.g. in 340), **do it again**.

1 Very-Short Answer Questions [24 points]

Give a short and concise 1-2 sentence answer to the below questions. All acronyms, symbols, and methods are as used in CPSC 340, and should be fairly standard. Each part is worth [1.5 points].

(a) Suppose we want to solve the binary classification problem of determining whether 26 by 26 images were images of digits or images of letters. What is an easy way to make our classifier roughly invariant to small translations of the training images?

Answer: TODO

(b) Ensemble methods are models that combine the predictions of multiple individual models. Give a reason why an ensemble method could do better than the best individual model in the ensemble.

Answer: TODO

(c) Suppose we are given two vectors of integers, x=$(x_1, x_2, \ldots, x_n)$ and y=$(y_1, y_2, \ldots, y_n)$, and we want to find the longest sequence of numbers that appears consecutively in both x and y. Describe an $\mathcal{O}(n^2)$ algorithm to do this using dynamic programming, assuming comparing integers takes constant time.

Answer: TODO

(d) A common method for outlier detection is to collect a large number of outliers, and build a binary classifier that distinguishes these outliers from standard "inlier" datapoints. What is an advantage and a drawback of this approach to outlier detection, compared to unsupervised outlier detection methods?

Answer: TODO

(e) Why might we add a column of 1 values to X when fitting a linear model with predictions of the form $w^T x$?

Answer: TODO

(f) If a function is convex, what does that imply about stationary points of the function? Does convexity imply that a stationary point exists?

Answer: TODO

(g) Consider fitting a neural network with one hidden layer. Would we call this a parametric or a non-parametric model if the hidden layer had n units (where n is the number of training examples)? What about one with d^2 units (where d is the number of input features)?

Answer: TODO

(h) Suppose you are hired by company to look at a dataset, and they want to "find which features are relevant" for their prediction task. What are two warnings you might give them regarding the features found by feature selection methods?

Answer: TODO

(i) Given a vector of n positive integers $(x_1, x_2, \ldots, x_n)$, give an $O(n)$ time algorithm for computing, for each position i, the sum of all numbers except x_i. Next, describe how you could solve this problem in $O(n)$ even if you were not allowed to use subtraction or negation, just addition. *Hint: you can start at the beginning or the end of the list.*

Answer: TODO

(j) If we fit a linear regression model with L1-regularization of the parameters, what is the effect of the regularization parameter λ on the sparsity level of the solution? What is the effect of λ on the two parts of the "fundamental trade-off" (the training error, and the amount of overfitting, aka the generalization gap)?

Answer: TODO

(k) Suppose we fit a one-feature linear regression model with a polynomial basis, and this leads to non-zero regression weights. What would the prediction of this model be on a test point with value x going to ∞? How would this change if you used Gaussian RBFs as features?

Answer: TODO

(l) Suppose that a famous machine learning TikTok influencer is advertising their "extremely-deep convolutional fuzzy-genetic Hilbert-long-short adversarial-recurrent neural network" classifier, which has 5,000 hyper-parameters. They claim that if you take 10 different famous datasets, and tune the 5,000 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.

Answer: TODO

(m) When searching for a good w for a linear classifier, why might we use the logistic loss instead of just minimizing the number of classification errors?

Answer: TODO

(n) What is the computational cost, in $\mathcal{O}()$ notation, for fitting a linear regression model with n examples and 1 feature using least squares (the normal equations)? Explain under what conditions it would make sense to use gradient descent, instead of the normal equations for fitting a linear regression model with 1 feature.

Answer: TODO

(o) With stochastic gradient descent, the loss might go up or down each time the parameters are updated. However, we don't actually know which of these cases occurred. Explain why it doesn't make sense to check whether the loss went up/down after each update.

Answer: TODO

(p) Consider using a fully-connected neural network for 3-class classification on a problem with $d = 10$. If the network has two hidden layers, of size $k_1 = 100$ and $k_2 = 50$, how many parameters (including biases) does the network have?

Answer: TODO

2 Machine Learning Model Memory and Time Complexities [18 points]

Answer the following questions using big-O notation, and a brief explanation. Your answers may involve n, d, and perhaps additional quantities defined in the question. As an example, the (linear) least squares model has $\mathcal{O}(d)$ parameters, requires $\mathcal{O}(d)$ time for prediction, and requires $\mathcal{O}(nd^2 + d^3)$ time to train.[1]

Each part is worth [2 points].

(a) What is the storage space required for the k-means clustering algorithm?

Answer: TODO

(b) What is the cost of clustering t examples using an already-trained k-means model?

Answer: TODO

(c) What is the training time for least squares (linear regression by solving the normal equations) with L2 regularization?

Answer: TODO

(d) What is the prediction cost for a trained linear model on t test examples?

Answer: TODO

(e) What is the storage space required to make predictions with a linear regression model using Gaussian RBFs as features?

Answer: TODO

(f) What is the prediction time for linear regression with Gaussian RBFs as features on t test examples? You can use σ^2 as the variance of the Gaussian RBFs.

Answer: TODO

(g) What is the cost of evaluating the support vector regression objective function shown in Question 3 part (i)?

Answer: TODO

(h) What is the cost of trying to minimize the exponential loss shown in Question 3 part (h) by running t iterations of gradient descent?

Answer: TODO

(i) What is the cost of trying to minimize the exponential loss by running t iterations of stochastic gradient descent?

Answer: TODO

Many people got very low grades on this question in past years. If you're not sure how to answer questions like this, get help!

[1] In this course, we assume matrix operations have the "textbook" cost where the operations are implemented in a straightforward way with "for" loops. For example, we'll assume that multiplying two $n \times n$ matrices or computing a matrix inverse simply costs $\mathcal{O}(n^3)$, rather than the $\mathcal{O}(n^\omega)$ where ω is closer to 2.37 as discussed in CS algorithm courses; this is closer to the behaviour observed for actual practical matrix sizes.

3 Matrix Notation, Quadratics, Convexity, and Gradients [18 points]

This and the next question review some mathematical tools used in CPSC 340. You might find some of the notes on the course webpage useful as refreshers. Each part is worth [2 points].

(a) Consider the function

$$f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j + \sum_{i=1}^{n} b_i x_i + c,$$

where x is a vector of length n with elements x_i, b is a vector of length n with elements b_i, and A is an $n \times n$ matrix with elements a_{ij} (not necessarily symmetric). Write this function in matrix notation so that it uses A and b, and does not have summations or references to indices i.

Answer: TODO

(b) Write the gradient of f from the previous question, in matrix notation.

Answer: TODO

(c) Show that f is convex if A is a symmetric, positive semi-definite matrix.

Answer: TODO

(d) In that case, give a linear system whose solution minimizes f in terms of x.

Answer: TODO

(e) Consider weighted linear regression with an L2-regularizer with regularization strength $1/\sigma^2$,

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

where $\sigma > 0$ and we have a vector v of length n containing the elements v^i. The other symbols are our standard regression notation. Write this function in matrix notation.

Note: you can use $\mathbf{X}$ as the matrix containing x^i as the rows, y as the vector containing the elements y^i, and V as a diagonal matrix containing the vector v along the diagonal.

Answer: TODO

(f) Assuming that $v^i \geq 0$ for all i, show that f from the previous part is convex. *(You can use part (c) or not, as you prefer.)*

Answer: TODO

(g) Assuming that we have $v^i \geq 0$ for all i, give a linear system whose solution gives a minimum value of f in terms of w. *(Again, use the previous result or not, your choice.)*

Answer: TODO

(h) If we fit a linear classifier with the exponential loss (used in older boosting algorithms), it would take the form

$$f(w) = \sum_{i=1}^{n} \exp(-y^i w^T x^i).$$

Derive the gradient of this loss function.

Answer: TODO

(i) The support vector regression objective function is

$$f(w) = \sum_{i=1}^{n} \max\{0, |w^T x^i - y^i| - \epsilon\} + \frac{\lambda}{2} \|w\|^2 .$$

where ϵ is a non-negative hyper-parameter. It is similar to the L1 robust regression loss, but where there is no penalty for being within ϵ of the prediction (which can reduce overfitting). Show that this non-differentiable function is convex.

Answer: TODO

4 MAP Estimation [9 points]

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):

(a) [3 points] Gaussian likelihood with a separate variance σ_i^2 for each training example, and Laplace prior with a separate variance $1/\lambda_j$ for each variable,

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

Answer: TODO

(b) [3 points] Robust student-t likelihood and Gaussian prior centered at u.

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

where u is $d \times 1$, B is the "Beta" function, and the parameter ν is called the "degrees of freedom."[2]

Answer: TODO

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

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

This prior is "improper," since $w \in \mathbb{R}^d$ but it doesn't integrate to 1 over this domain.

Answer: TODO

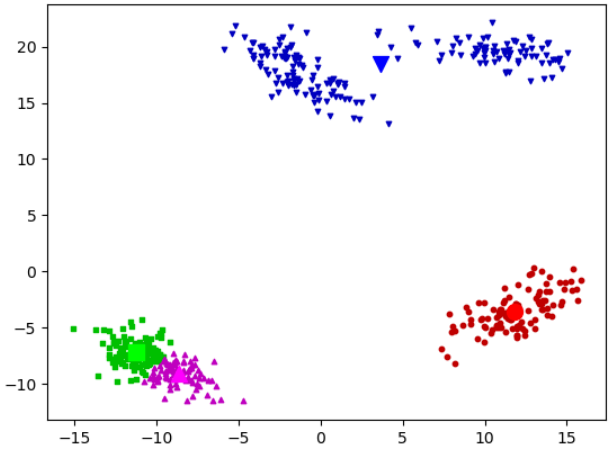
[2] This likelihood is more robust than the Laplace likelihood, but leads to a non-convex objective.

5 K-Means Clustering [11 points]

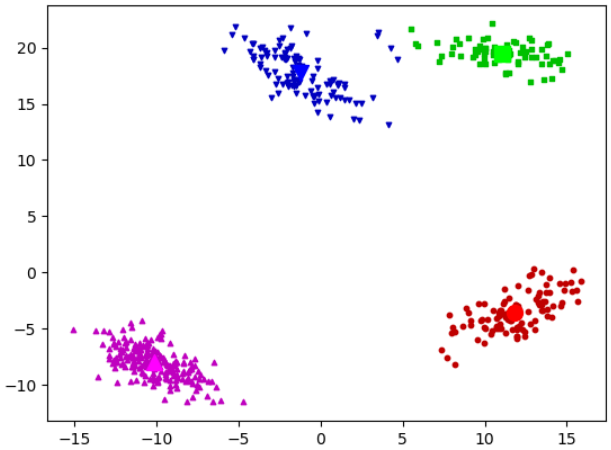
This and following questions use code available from the course webpage; get it from `a1.zip`.

You'll need Python 3 and the packages `numpy`, `scipy`, *and* `matplotlib`. *If you don't have them installed already, install them with* `conda install`, *your system package manager, or* `pip install numpy scipy matplotlib`.

If you run `main.py 5`, it will load a dataset with two features and a very obvious clustering structure. It will then apply the k-means algorithm with a random initialization. The result of applying the algorithm will thus depend on the randomization, but a typical run might look like this:



(Note that the colours are arbitrary due to the label switching problem.) But the "correct" clustering (that was used to make the data) is something more like this:



If you run the demo several times, it will find different clusterings. To select among clusterings for a *fixed* value of k, one strategy is to minimize the sum of squared distances between examples x_i and their means w_{y_i},

$$f(w_1, w_2, \ldots, w_k, y^1, y^2, \ldots, y^n) = \sum_{i=1}^{n} \left\| x^i - w_{y^i} \right\|_2^2 = \sum_{i=1}^{n} \sum_{j=1}^{d} (x_j^i - w_{y_j^i})^2.$$

where $y_i \in \arg\min_{c \in [n]} \|x_i - w_c\|$ is the index of the closest mean to x_i. This is a natural criterion because the steps of k-means alternately optimize this objective function in terms of the w_c and the y_i values.

(a) [2 points] Complete the function `KMeans.loss`, inside `k_means.py`, that takes in a dataset $\mathbf{X}$, a set of cluster assignments y, and a set of cluster means W, and computes this objective function. Hand in your code.

Answer: TODO

(b) [1 points] Modify your code to, instead of/in addition to printing the number of labels that change on each iteration, print the value of `KMeans.loss` after each iteration. What trend do you observe? (No need to hand in code or specific loss values for this, just describe the trend.)

Answer: TODO

(c) [2 points] `main.py 5c` will call the function `q_5c()` in `main.py`, which calls the `best_fit` function to rerun k-means 50 times and take the one with the lowest error. Complete the `best_fit` function, and hand in the resulting plot.

Answer: TODO

(d) [2 points] Explain why `KMeans.loss` function should not be used to choose k – even if you evaluate it on test data.

Answer: TODO

(e) [1 points] The data in `clusterData2.pkl` is exactly the same as `clusterData.pkl`, except that it has four outliers that are far away from the data. `main.py 5f` will run your code from above to find the best of 50 runs on this data and save it as `figs/kmeans-outliers.png`. Hand in this plot; are you satisfied with it?

Answer: TODO

(f) [3 points] Implement the *k-medians* algorithm in `kmedians.py`, which assigns examples to the nearest w_c in the L1-norm and then updates the w_c by setting them to the "median" of the points assigned to the cluster (defining the d-dimensional median as the concatenation of the medians along each dimension). For this algorithm it makes sense to use the L1-norm version of the error (where y^i now represents the closest median in the L1-norm),

$$f(w_1, w_2, \ldots, w_k, y^1, y^2, \ldots, y^n) = \sum_{i=1}^{n} \left\| x_i - w_{y^i} \right\|_1 = \sum_{i=1}^{n} \sum_{j=1}^{d} |x_j^i - w_{y^i,j}|,$$

`main.py 5g` runs it for you, once you've finished the `KMedians` class. Hand in your code and plot obtained by taking the clustering with the lowest L1-norm after using 50 random initializations for $k = 4$. Is this better?

Answer: TODO

6 Regularization and Hyper-Parameter Tuning [10 points]

If you run `main.py 6`, it will:

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

Unfortunately, this is not a great model of the data, and the figure shows that a linear model with a y intercept of 0 is probably not suitable.

(a) [2 points] Implement the `LeastSquaresBias` class to include a *y-intercept* variable w_0, and makes predictions using the model

$$y^i = w^T x^i + w_0.$$

Hand in your new class and the updated plot.

Answer: TODO

(b) [3 points] Allowing a non-zero y-intercept improves the prediction substantially, but the model still seems sub-optimal because there are obvious non-linear effects. Complete the model `leastSquaresRBFL2` that implements *least squares using Gaussian radial basis functions (RBFs) with L2-regularization.*

Use `lam` for the L2 regularization parameter, and `sigma` for the lengthscale of the Gaussian features. Note that your L2 regularization should correspond to minimizing the loss function $\|\mathbf{X}w - y\|^2 + \lambda \|w\|^2$; some versions instead correspond to putting a $\frac{1}{n}$ in front of the loss term.

Hand in your function and the plot generated with $\lambda = 1$ and $\sigma = 1$.

Hint: The function `utils.euclidean_dist_squared`, which was also used in our k-means implementation, efficiently computes the squared Euclidean distance between all pairs of rows in two matrices.

Answer: TODO

(c) [3 points] Modify the script, in the function `q_6c`, 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 from some reasonable set of candidate values. You'll probably want to vary these by a few orders of magnitude either smaller or larger from 1.

(Although in real work you'd probably use some pre-coded helpers for this, code it yourself here.)

Hand in your modified function, the selected λ and σ, and the plot you obtain by refitting on the full dataset with the best values of λ and σ.

Answer: TODO

(d) [2 points] Consider a model combining the first two parts of this question,

$$y^i = w^T x^i + w_0 + v^T z^i,$$

where z^i is the Gaussian RBFs and v is a vector of regression weights for those features. Suppose that we first fit w and w_0 assuming that $v = 0$ as in part (a), and then we fit v with w and w_0 fixed (you could use your code for (c) to do this by modifying the y^i values). Why would someone want to do this?

Hint: Think about how this model would behave with $x^i = 10$.

Answer: TODO

7 Multi-Class Logistic Regression [10 points]

Running `main.py 7` loads a multi-class classification dataset and fits a "one-vs-all" logistic regression classifier using gradient descent as implemented in `optimize.py`, then reports the validation error and shows a plot of the data/classifier. The performance on the validation set is okay, but could be much better. For example, this classifier never 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,

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

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

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

Moreover, I'll be nice and remind you that the gradient of this function is given by

$$\frac{\partial f(W)}{\partial W_{jc}} = \sum_{i=1}^n \left[-x_j^i \mathbb{1}(y^i = c) + x_j^i p(y^i = c \mid W, x^i) \right].$$

(Make sure you can see how to derive this!)

Implement the model in `softmax_classifier.py`, which fits W using the softmax loss from the previous section instead of fitting k independent classifiers. Hand in your code, a plot produced by `plot2Dclassifier`, and report the validation error.

Hint: Remember to add a bias variable somewhere, or else code it in explicitly.

Hint: You can pass `check_grad=True` to `find_min` to help you debug the gradient of the softmax loss. Also, note that the `find_min` function treats the parameters as a vector; you may want to use `reshape` when writing the softmax objective.

Answer: TODO