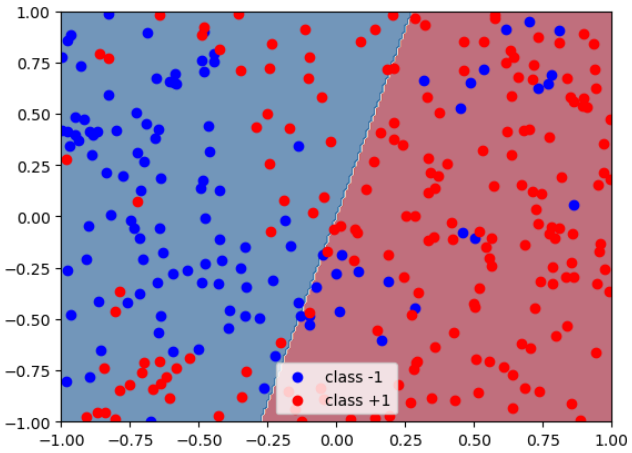
CPSC 340 Assignment 5 (due Wednesday March 29 at 11:55pm)

Name(s) and Student ID(s):

1 Kernel Logistic Regression

If you run `python main.py 1` it will load a synthetic 2D data set, split it into train/validation sets, and then perform kernel logistic regression (without an intercept term, for simplicity). The resulting boundary is linear and identical to regular logistic regression, because the kernel being used is the linear kernel (i.e., the kernel corresponding to no change of basis):



1.1 Implementing Kernels

Inside `kernels.py`, you will see classes named `PolynomialKernel` and `GaussianRBFKernel`, whose `__call__` methods are yet to be written. Implement the polynomial kernel and the RBF kernel for logistic regression. Report your training/validation errors and submit the plots from `utils.plot_classifier` for each case. You should use the kernel hyperparameters $p = 2$ and $\sigma = 0.5$ respectively, and $\lambda = 0.01$ for the regularization strength. For the Gaussian kernel, please do *not* use a $1/\sqrt{2\pi\sigma^2}$ multiplier.

1.2 Hyperparameter search

For the RBF kernel logistic regression, consider the hyperparameter values $\sigma = 10^m$ for $m = -2, -1, \ldots, 2$ and $\lambda = 10^m$ for $m = -4, -3, \ldots, 2$. The function `q1_2()` has a little bit in it already to help set up to run a grid search over the possible combination of these parameter values. You'll need to fill in the `train_errs` and `val_errs` arrays with the results on the given training and validation sets, respectively; then the code already in the function will produce a plot of the error grids. Submit this plot. Also, for each of the training and validation errors, pick the best (or one of the best, if there's a tie) hyperparameters for that error metric, and report the parameter values and the corresponding error, as well as a plot of the decision boundaries (plotting only the training set). While you're at it, submit your code. To recap, for this question you should be submitting: the error grid of your hyperparameter search, two decision boundary plots, the values of two hyperparameter pairs with corresponding errors, and your code.

1.3 Reflection

Briefly discuss the best hyperparameters you found in the previous part, and their associated plots. Was the training error minimized by the values you expected, given the ways that σ and λ affect the fundamental tradeoff?

2 MAP Estimation

In class, we considered MAP estimation in a regression model where we assumed that:

- The likelihood $p(y_i \mid x_i, w)$ for each example i is a normal distribution with a mean of $w^T x_i$ and a variance of 1.
- The prior $p(w_j)$ for each variable j is a normal distribution with a mean of zero and a variance of λ^{-1}.

Under these assumptions we showed that computing the MAP estimate with n training examples leads to the standard L2-regularized least squares objective function:

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

For each of the alternate assumptions below, write down the objective function that results (from minimizing the negative log-posterior, and simplifying as much as possible):

1. We use a Laplace likelihood with a mean of $w^T x_i$ and a scale of 1, and we use a zero-mean Laplace prior for each variable with a scale parameter of λ^{-1},

$$p(y_i \mid x_i, w) = \frac{1}{2}\exp(-|w^T x_i - y_i|), \quad p(w_j) = \frac{\lambda}{2}\exp(-\lambda|w_j|).$$

2. We use a normal likelihood with a mean of $w^T x_i$ but where each example i has its own positive variance σ_i^2, and a normal prior with a variance of λ^{-1} and a mean that is some "guess" w^0 of the optimal parameter vector,

$$p(y_i \mid x_i, w) = \frac{1}{\sqrt{2\sigma_i^2\pi}}\exp\left(-\frac{(w^T x_i - y_i)^2}{2\sigma_i^2}\right), \quad p(w_j) \propto \exp\left(-\frac{\lambda(w_j - w_j^0)^2}{2}\right).$$

   The standard notation for this case is to use Σ as a diagonal matrix with the σ_i^2 values along the diagonal.

3. We use a Poisson likelihood with a mean of $\exp(w^T x_i)$,[1] and we use a uniform prior for some constant κ,

$$p(y_i \mid x_i, w) = \frac{\exp(y_i w^T x_i)\exp(-\exp(w^T x_i))}{y_i!}, \quad p(w_j) \propto \kappa$$

   For this sub-question you don't need to put likelihood in matrix notation.

4. We use a Laplace likelihood with a mean of $w^T x_i$ where each example i has its own positive scale paramater v_i^{-1}, and a student t prior (which is very robust to irrelevant features) with ν degrees of freedom,

$$p(y_i \mid x_i, w) = \frac{1}{2}\exp\left(-v_i|w^T x_i - y_i|\right), \quad p(w_j) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\left(1 + \frac{w_j^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

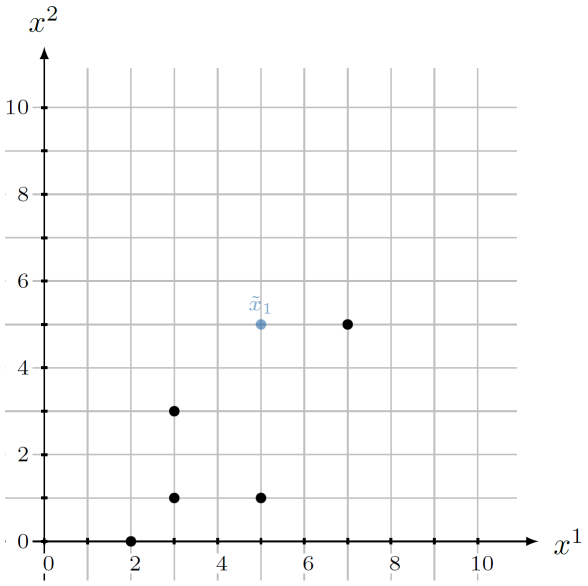
   where you use can V as a diagonal matrix with the v_i along the diagonal and Γ is the "gamma" function (which is always non-negative). You do not need to put the log-prior in matrix notation.

[1] This is one way to use regression to model *counts*, like "number of Facebook likes".

3 Principal Component Analysis (PCA)

3.1 PCA by Hand

Consider the following dataset, containing $n = 5$ training examples and $t = 1$ test example with $d = 2$ features each:



	x^1	x^2
	2	0
	3	1
X	7	5
	3	3
	5	1
$\widetilde{X}$	5	5

1. Principal component analysis requires a centered feature matrix. Find the vector of means $\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$ and compute the centered feature matrix X_c.

2. Writing $G_c = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} = X_c^\top X_c$, the first principal component $w_1 = \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix}$ is given by the **normalized** solution to the linear system $(G_c - \mathbf{I}_d \lambda_1) \cdot w_1 = 0$, where λ_1 is the largest solution to the quadratic equation $\chi(\lambda) = (\lambda - g_{11})(\lambda - g_{22}) - g_{12}g_{21} = 0$.[2] The second principal component $w_2 = \begin{bmatrix} w_{21} \\ w_{22} \end{bmatrix}$ is given by the **normalized** solution to the linear equation $w_1^\top w_2 = 0$. Compute the two principal components w_1 and w_2 (show your work).

3. The two principal components w_1 and w_2 form an orthonormal basis $W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$ of $\mathbb{R}^2$. Represent the test point $\tilde{x}_1$ in this basis by performing an orthogonal projection $\tilde{z}_1 = \begin{bmatrix} \tilde{z}_{11} \\ \tilde{z}_{12} \end{bmatrix} = W(\tilde{x}_1 - \mu)$ onto w_1 and w_2 (show your work).

[2] χ is called the *characteristic polynomial* of G_c and its roots are eigenvalues λ whose corresponding eigenvectors w solve the equation $G_c w = \lambda w$.

4. We can now represent the PCA coordinate system in the orginal coordinate system. In the plot above, draw the two coordinate axes defined by w_1 and w_2, using μ as origin. Then visualize the PCA coordinates $\tilde{z}_1$ computed in 3.1.3 by drawing two lines between the test point and its orthogonal projections onto w_1 (given by $\begin{bmatrix} \tilde{z}_{11} \\ 0 \end{bmatrix}$ in the PCA coordinate system) and w_2 (given by $\begin{bmatrix} 0 \\ \tilde{z}_{12} \end{bmatrix}$ in the PCA coordinate system). Finally, use the visualization to read off the coordinates of $\begin{bmatrix} \tilde{z}_{11} \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ \tilde{z}_{12} \end{bmatrix}$ in the **original** coordinate system, which can be interpreted as reconstructions of $\tilde{x}_1$. Report both reconstructions and their L2 reconstruction errors (show your work).

   Note: If you want, you can draw the PCA coordinate system and the orthogonal projections directly in Latex; see the commented lines below this one in a5.tex for an example. Hand-drawn results will also be accepted.

3.2 Data Visualization

If you run `python main.py 3.2`, the program will load a dataset containing 50 examples, each representing an animal. The 85 features are traits of these animals. The script standardizes these features and gives two unsatisfying visualizations of it. First, it shows a plot of the matrix entries, which has too much information and thus gives little insight into the relationships between the animals. Next it shows a scatterplot based on two random features and displays the name of 15 randomly-chosen animals. Because of the binary features even a scatterplot matrix shows us almost nothing about the data.

In `encoders.py`, you will find a class named `PCAEncoder`, which implements the classic PCA method (orthogonal bases via SVD) for a given k, the number of principal components. Using this class, create a scatterplot that uses the latent features z_i from the PCA model with $k = 2$. Make a scatterplot of all examples using the first column of Z as the x-axis and the second column of Z as the y-axis, and use `plt.annotate()` to label the points corresponding to `random_is` in the scatterplot. (It's okay if some of the text overlaps each other; a fancier visualization would try to avoid this, of course, but hopefully you can still see most of the animals.) Do the following:

1. Hand in your modified demo and the scatterplot.
2. Which trait of the animals has the largest influence (absolute value) on the first principal component?
3. Which trait of the animals has the largest influence (absolute value) on the second principal component?

3.3 Data Compression

It is important to know how much of the information in our dataset is captured by the low-dimensional PCA representation. In class we discussed the "analysis" view that PCA maximizes the variance that is explained by the PCs, and the connection between the Frobenius norm and the variance of a centered data matrix X. Use this connection to answer the following:

1. How much of the variance is explained by our two-dimensional representation from the previous question?
2. How many PCs are required to explain 75% of the variance in the data?

Note: you can compute the Frobenius norm of a matrix using the function `np.linalg.norm`, among other ways. Also, note that the "variance explained" formula from class assumes that X is already centred.

4 Very-Short Answer Questions

1. Assuming we want to use the original features (no change of basis) in a linear model, what is an advantage of the "other" normal equations over the original normal equations?
2. Describe the kernel trick in L2-regularized least squares in one sentence.
3. In class we argued that it's possible to make a kernel version of k-means clustering. What would an advantage of kernels be in this context?
4. What is the key advantage of stochastic gradient methods over gradient descent methods?
5. Which of the following step-size sequences lead to convergence of stochastic gradient to a stationary point?
   (a) $\alpha^t = 1/t^2$.
   (b) $\alpha^t = 1/t$.
   (c) $\alpha^t = \gamma/t$ (for $\gamma > 0$).
   (d) $\alpha^t = 1/(t+\gamma)$ (for $\gamma > 0$).
   (e) $\alpha^t = 1/\sqrt{t}$.
   (f) $\alpha^t = 1$.
6. Given discrete data D and parameters w, is the expression $p(D|w)$ a probability mass function, a likelihood function, or both? Briefly justify your answer.
7. Why is it often more convenient to minimize the negative log-likelihood than to maximize the likelihood?
8. How does the impact of the prior in MAP estimation change as we add more data?
9. What is the difference between a generative model and a discriminative model?
10. With PCA, is it possible for the loss to increase if k is increased? Briefly justify your answer.
11. Why doesn't it make sense to do PCA with $k > d$?
12. In terms of the matrices associated with PCA (X, W, Z, $\hat{X}$), where would an "eigenface" be stored?