CPSC 340 Assignment 5 (due Friday November 16 at 11:55pm)

Instructions

Rubric: {mechanics:5}

The above points are allocated for following the general homework instructions.

1 Kernel Logistic Regresion

If you run `python main.py -q 1` it will load a synthetic 2D data set, split it into train/validation sets, and then perform regular logistic regression and kernel logistic regression (both without an intercept term, for simplicity). You'll observe that the error values and plots generated look the same since the kernel being used is the linear kernel (i.e., the kernel corresponding to no change of basis).

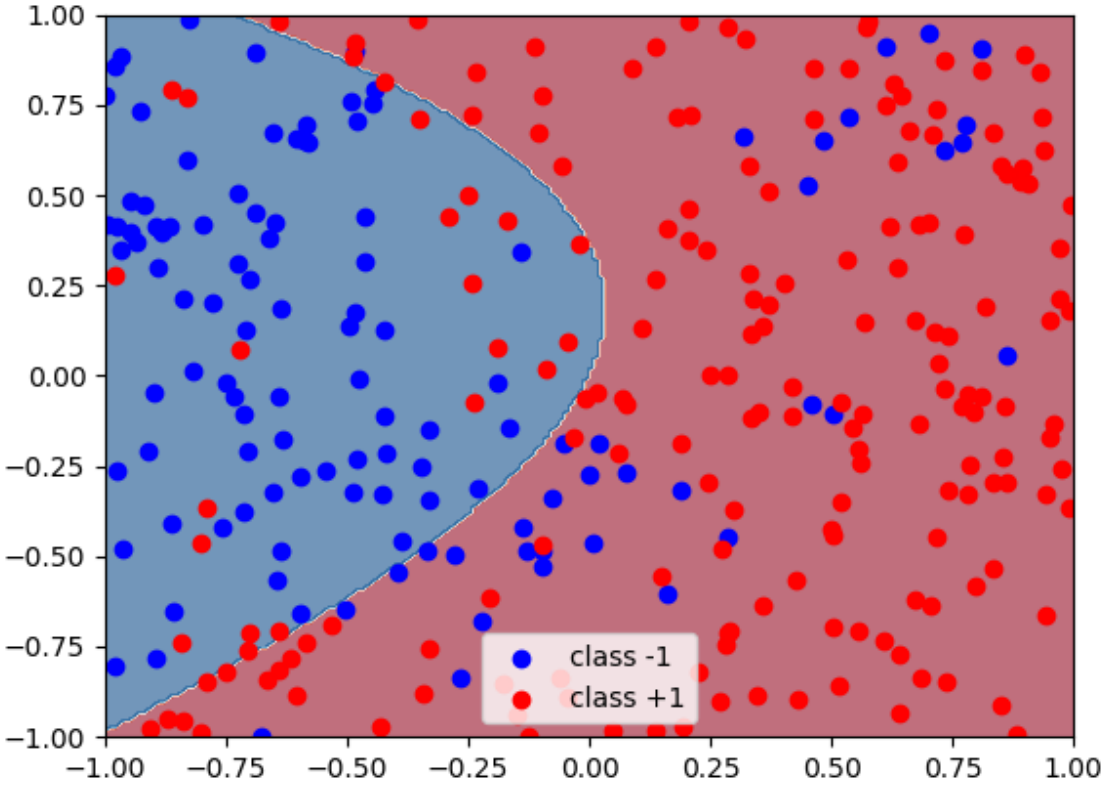
1.1 Implementing kernels

Rubric: {code:5}

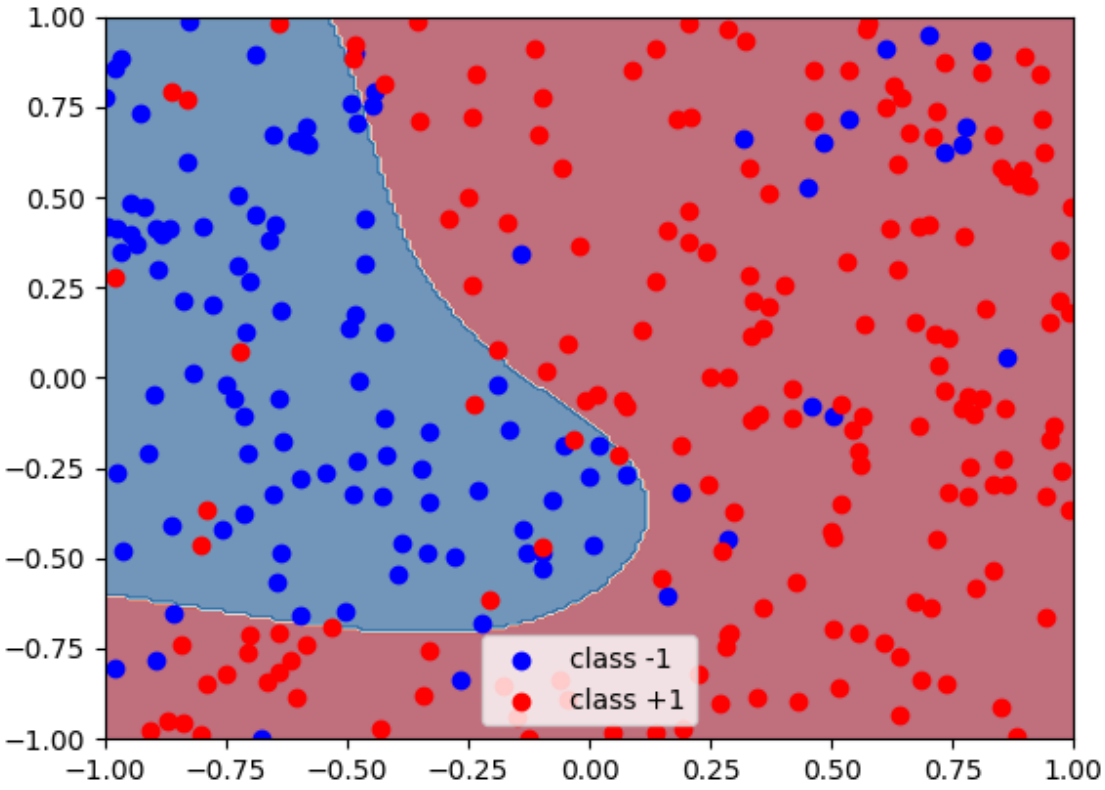
Implement the polynomial kernel and the RBF kernel for logistic regression. Report your training/validation errors and submit the plots generated for each case. You should use the hyperparameters $p = 2$ and $\sigma = 0.5$ respectively, and $\lambda = 0.01$ for the regularization strength.

The errors related to the polynomial kernel are: Training error 0.183,Validation error 0.170. The errors related to the RBF kernel are: Training error 0.127,Validation error 0.090

The graph for polynomial kernel



The graph for RBF kernel is:



1.2 Hyperparameter search

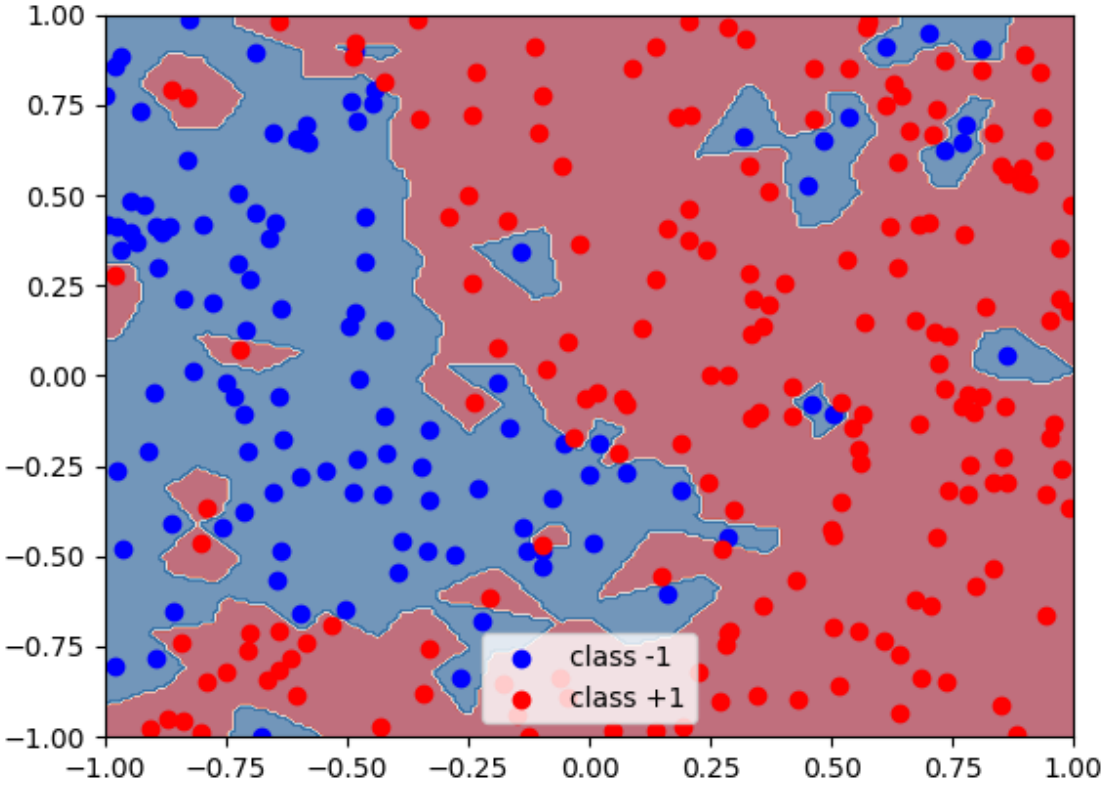
Rubric: {code:3}

For the RBF kernel logistic regression, consider the hyperparameters values $\sigma = 10^m$ for $m = -2, -1, \ldots, 2$ and $\lambda = 10^m$ for $m = -4, -3, \ldots, 0$. In `main.py`, sweep over the possible combinations of these hyperparameter values. Report the hyperparameter values that yield the best training error and the hyperparameter values that yield the best validation error. Include the plot for each.

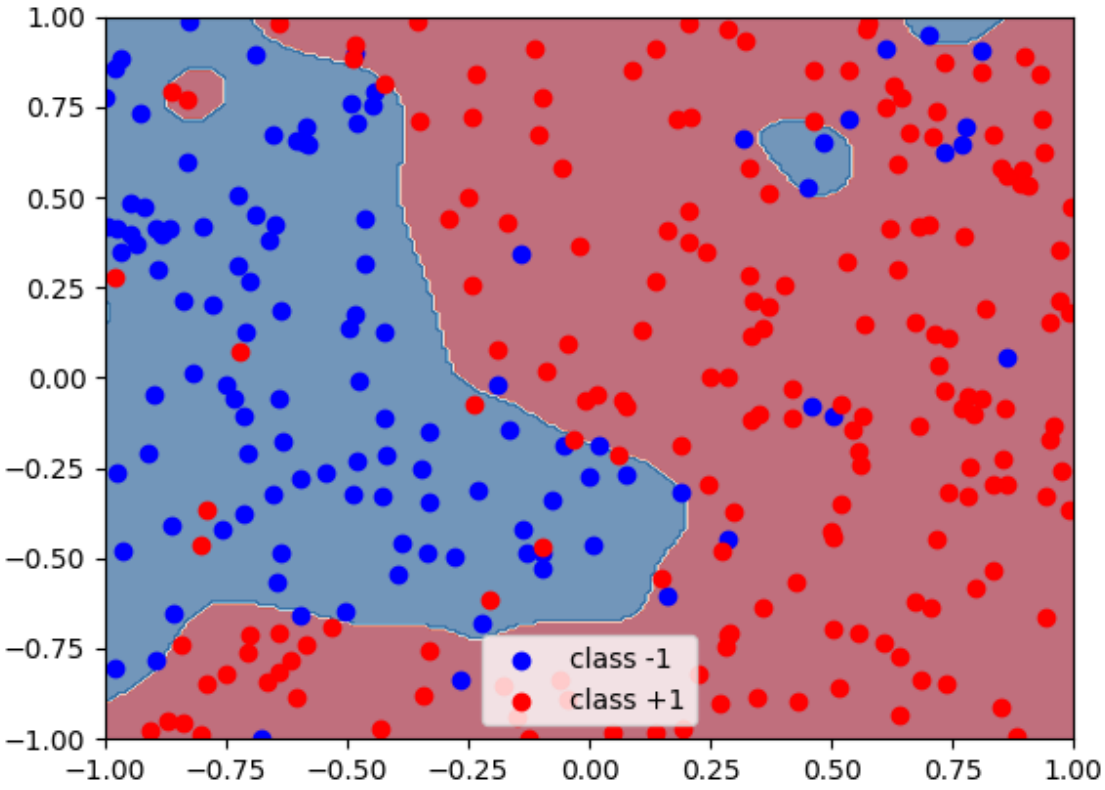
Note: on the job you might choose to use a tool like scikit-learn's `GridSearchCV` to implement the grid search, but here we are asking you to implement it yourself by looping over the hyperparameter values.

The sigma, lambda values for the least training error is [0.01, 0.0001] and the related training error is 0.0.
The sigma, lambda values for the least validation error is [0.1, 1.0] and the related validation error is 0.12.

The graph for the least training error:



The graph for the least validation error



1.3 Reflection

Rubric: {reasoning:1}

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?

The sigma, lambda values for the least training error is [0.01, 0.0001] and the related training error is 0.0. The sigma, lambda values for the least validation error is [0.1, 1.0] and the related validation error is 0.12. The values are as expected the least training error is achieved with the least lambda and the least sigma. The least validation error is achieved by the highest lambda and a largest sigma value

2 MAP Estimation

Rubric: {reasoning:8}

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

- The likelihood $p(y_i \mid x_i, w)$ is a normal distribution with a mean of $w^T x_i$ and a variance of 1.

- The prior for each variable j, $p(w_j)$, is a normal distribution with a mean of zero and a variance of λ^{-1}.

Under these assumptions, we showed that this leads to the standard L2-regularized least squares objective function,

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

which is the negative log likelihood (NLL) under these assumptions (ignoring an irrelevant constant). For each of the alternate assumptions below, show how the loss function would change (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 Gaussian prior with a variance of σ^2.

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

2. We use a Gaussian likelihood where each datapoint has its own variance σ_i^2, and where we use a zero-mean Laplace prior with a vairance of λ^{-1}.

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

   You can use Σ as a diagonal matrix that has the values σ_i^2 along the diagonal.

3. We use a (very robust) student t likelihood with a mean of $w^T x_i$ and ν degrees of freedom, and a zero-mean Gaussian prior with a variance of λ^{-1},

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

   where Γ is the "gamma" function (which is always non-negative).

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

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

   (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.)

★ const means constant and I ignore them because they do not change the solution

$$(2|1) - \sum_{i=1}^{n} \log\left(\tfrac{1}{2}\exp\left(-|w^T x_i - y_i|\right)\right) \sum_{j=1}^{d} \log\left(\underbrace{\frac{1}{\sqrt{2\pi}\sigma}}_{\text{const}} \exp\left(-\frac{w_j^2}{2\sigma^2}\right)\right)$$

$$= -\sum_{i=1}^{n} \underbrace{\log(1/2)}_{\text{const}} + \left(-|w^T x_i - y_i|\right) + \sum_{j=1}^{d} \frac{w_j^2}{2\sigma^2}$$

$$= \sum_{i=1}^{n} |w^T x_i - y_i| + \sum_{j=1}^{d} \frac{w_j^2}{2\sigma^2}$$

$$= \|(w)_2 \|Xw - y\| + \frac{\|w\|^2}{2\sigma^2}$$

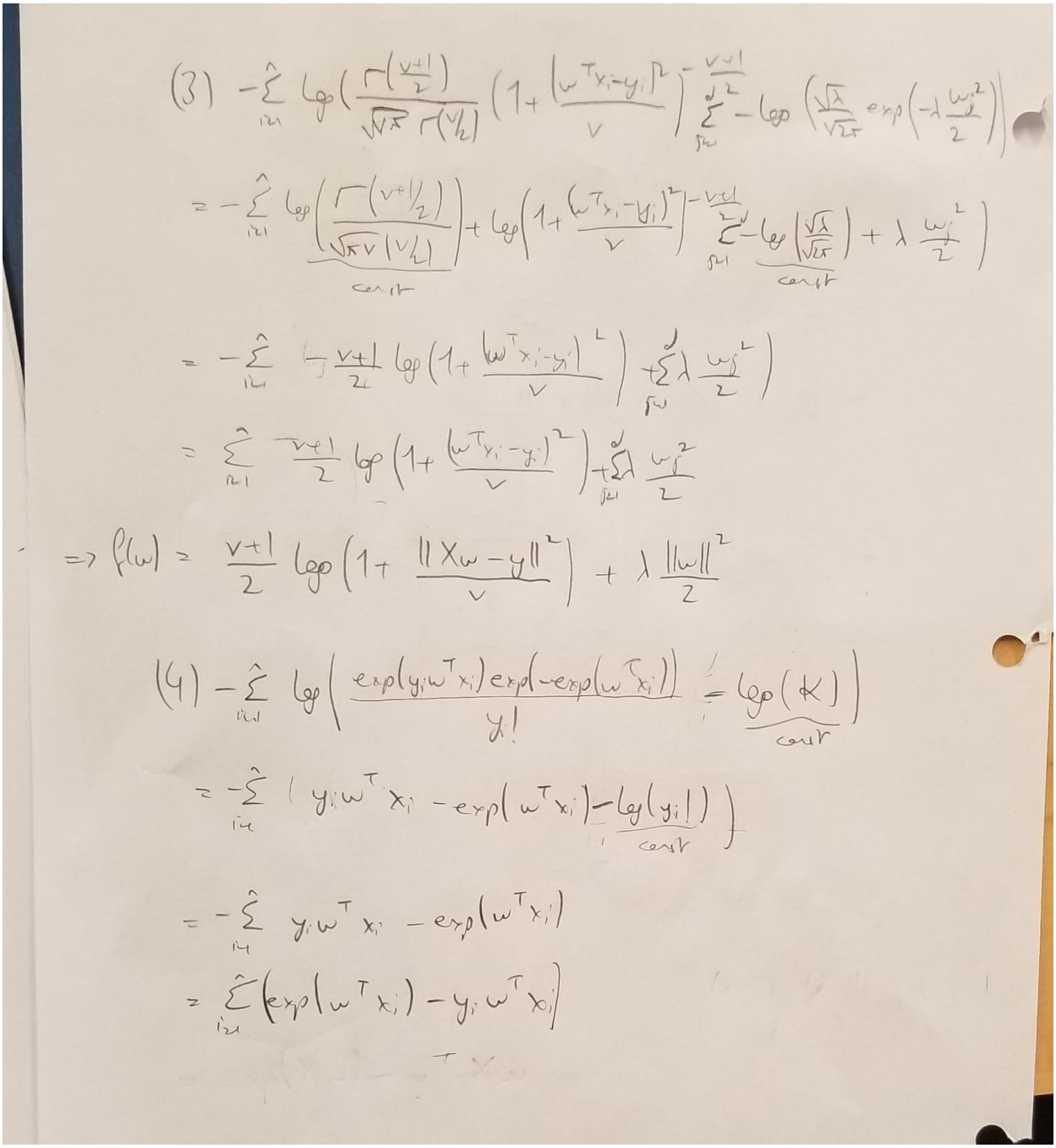
$$(2) - \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\sigma_i^2 \pi}} \exp\left(-\frac{(w^T x_i - y_i)^2}{2\sigma_i^2}\right)\right) \sum_{j=1}^{d} \log\left(\lambda/2 \exp\left(-\lambda |w_j|\right)\right)$$

$$= -\sum_{i=1}^{n} \underbrace{\log\left(\frac{1}{2\sigma_i^2 \pi}\right)}_{\text{const}} + \frac{-(w^T x_i - y_i)^2}{2\sigma_i^2} \sum_{j=1}^{d} \underbrace{\log(\lambda/2)}_{\text{const}} + \lambda |w_j|$$

$$= \sum_{i=1}^{n} \frac{(w^T x_i - y_i)^2}{2\sigma_i^2} + \sum_{j=1}^{d} \lambda |w_j|$$

(Define $\tilde{\Sigma} = \sqrt{\Sigma}^2$ meaning $\tilde{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \ddots \\ & & & \sigma_n \end{bmatrix}$

$$= \tfrac{1}{2} \|\tilde{\Sigma}^{-1}(Xw - y)\|^2 - \lambda \|w\|$$



3 Principal Component Analysis

Rubric: {reasoning:3}

Consider the following dataset, containing 5 examples with 2 features each:

x_1	x_2
-4	3
0	1
-2	2
4	-1
2	0

Recall that with PCA we usually assume that the PCs are normalized ($\|w\| = 1$), we need to center the data before we apply PCA, and that the direction of the first PC is the one that minimizes the orthogonal distance to all data points.

1. What is the first principal component?
2. What is the reconstruction loss (L2 norm squared) of the point (-3, 2.5)? (Show your work.)
3. What is the reconstruction loss (L2 norm squared) of the point (-3, 2)? (Show your work.)

Hint: it may help (a lot) to plot the data before you start this question.

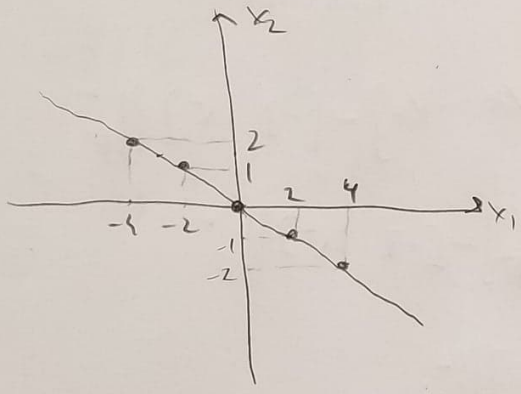
(3) (1) Center data, so the new data points are:

x_1	x_2
-4	2
0	0
-2	1
4	-2
2	-1

↓

since mean is 0

Plot these points:



We can fit a line perfectly with equation

$y = -\frac{1}{2}x$

a vector which is in the span of this line is: $[2, -1]$

normalize this by dividing by $\sqrt{2^2+1} = \sqrt{5}$

$\Rightarrow$ Principal component $= \left[\frac{2}{\sqrt{5}}, -\frac{1}{\sqrt{5}}\right]$

(2) $\tilde{X} = (-3, 1.5)$

$X = Z \times W$ (1×d, n×k, k×d)

$\tilde{Z} = \tilde{X} W^T (W W^T)^{-1}$

$= \begin{bmatrix} -3 & 1.5 \end{bmatrix} \begin{bmatrix} 2/\sqrt{5} \\ -1/\sqrt{5} \end{bmatrix} \left(\begin{bmatrix} 2/\sqrt{5} & -1/\sqrt{5} \end{bmatrix} \begin{bmatrix} 2/\sqrt{5} \\ -1/\sqrt{5} \end{bmatrix} \right)^{-1}$

$= \begin{bmatrix} -3 & 1.5 \end{bmatrix} \begin{bmatrix} 2/\sqrt{5} \\ -1/\sqrt{5} \end{bmatrix} (1)^{-1}$

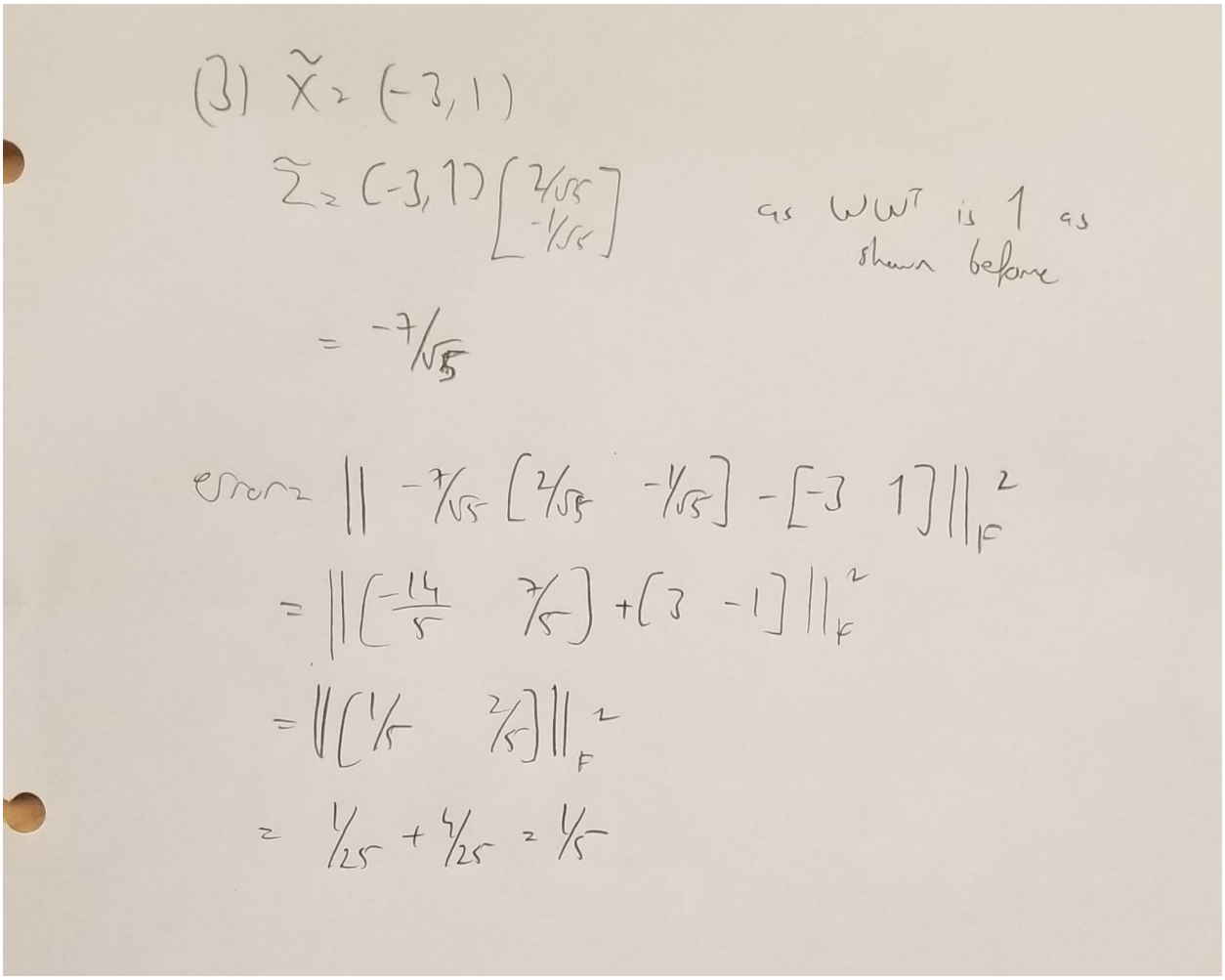
$= \begin{bmatrix} -7.5 \\ \hline \sqrt{5} \end{bmatrix}$ (2×1, 1×2)

$\text{error} = \left\| \begin{bmatrix} -7.5 \\ \hline \sqrt{5} \end{bmatrix} \begin{bmatrix} 2/\sqrt{5} & -1/\sqrt{5} \end{bmatrix} - \begin{bmatrix} -3 & 1.5 \end{bmatrix} \right\|_F^2$

$= \left\| \begin{bmatrix} -15/5 & 7.5/5 \end{bmatrix} - \begin{bmatrix} -3 & 1.5 \end{bmatrix} \right\|_F^2$

$= \left\| \begin{bmatrix} 0 & 0 \end{bmatrix} \right\|_F^2$

$= 0$



4 PCA Generalizations

4.1 Robust PCA

Rubric: {code:10}

If you run `python main -q 4.1` the code will load a dataset X where each row contains the pixels from a single frame of a video of a highway. The demo applies PCA to this dataset and then uses this to reconstruct the original image. It then shows the following 3 images for each frame:

1. The original frame.
2. The reconstruction based on PCA.
3. A binary image showing locations where the reconstruction error is non-trivial.

Recently, latent-factor models have been proposed as a strategy for "background subtraction": trying to separate objects from their background. In this case, the background is the highway and the objects are the cars on the highway. In this demo, we see that PCA does an OK job of identifying the cars on the highway

in that it does tend to identify the locations of cars. However, the results aren't great as it identifies quite a few irrelevant parts of the image as objects.

Robust PCA is a variation on PCA where we replace the L2-norm with the L1-norm,

$$f(Z, W) = \sum_{i=1}^{n} \sum_{j=1}^{d} |\langle w^j, z_i \rangle - x_{ij}|,$$

and it has recently been proposed as a more effective model for background subtraction. Complete the class *pca.RobustPCA*, that uses a smooth approximation to the absolute value to implement robust PCA. Briefly comment on the results.

Note: in its current state, *pca.RobustPCA* is just a copy of *pca.AlternativePCA*, which is why the two rows of images are identical.

Hint: most of the work has been done for you in the class *pca.AlternativePCA*. This work implements an alternating minimization approach to minimizing the (L2) PCA objective (without enforcing orthogonality). This gradient-based approach to PCA can be modified to use a smooth approximation of the L1-norm. Note that the log-sum-exp approximation to the absolute value may be hard to get working due to numerical issues, and a numerically-nicer approach is to use the "multi-quadric" approximation:

$$|\alpha| \approx \sqrt{\alpha^2 + \epsilon},$$

where ϵ controls the accuracy of the approximation (a typical value of ϵ is 0.0001).

As expected when we used L1-loss, we are more robust to outliers, namely cars. So, we achieve better background substraction. And in the third frame, we can see the cars more clearly

4.2 Reflection

Rubric: {reasoning:3}

1. Briefly explain why using the L1 loss might be more suitable for this task than L2.
2. How does the number of video frames and the size of each frame relate to n, d, and/or k?
3. What would the effect be of changing the threshold (see code) in terms of false positives (cars we identify that aren't really there) and false negatives (real cars that we fail to identify)?

1. Since L1 is more sensitive to outliers, when we use L1, it achieves better background substraction
2. The number of frames are n so it increases when the number of video frames go up. The size of each frame (or the number of features) is d meaning d = height * width. So as the size of the frame increases, d goes up. k is a hyperparameter, so it does not directly get effected by n or d. But depending on the sizes of the photo, we might choose to increase or decrease it.
3. If we increase the threshold, it increases the number of false negatives because now it is harder to consider something as significant error or car. If we decrease the threshold, it increases the number of false positives because now we consider more things as significant error or car.
4.

5 Very-Short Answer Questions

Rubric: {reasoning:11}

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. 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?
3. In the language of loss functions and regularization, what is the difference between MLE and MAP?
4. What is the difference between a generative model and a discriminative model?
5. With PCA, is it possible for the loss to increase if k is increased? Briefly justify your answer.
6. What does "label switching" mean in the context of PCA?
7. Why doesn't it make sense to do PCA with $k > d$?
8. In terms of the matrices associated with PCA (X, W, Z, $\hat{X}$), where would an "eigenface" be stored?
9. What is an advantage and a disadvantage of using stochatic gradient over SVD when doing PCA?
10. 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) $\alpha^t = 1/\sqrt{t}$.
    (d) $\alpha^t = 1$.
11. We discussed "global" vs. "local" features for e-mail classification. What is an advantge of using global features, and what is advantage of using local features?

1. Other normal equation are faster when n is less than k. The cost with using other normal equations is $O(n^2k + n^3)$ instead of $O(nk^2 + k^3$
2. We can compute euclidean distance with kernels s.t. $|z_i - z_j| = k(x_i, x_i) - 2kx(x_i, x_j) + k(x_j, x_j)$ which will let us train and predict faster than before.
3. In MAP, we have an extra component on top of the likelihood,called prior. When we use negative log likelihood, this prior can be seen to be regularizer in least squares. So MLE is like a loss function without regularization and MAP is like a loss function with regularization
4. Generative models model p(yi,xi) or namely how the features X are generated. We can get p(yi—xi) using rules of probability to make predictions. Discriminative models treat features X as fixed and directly model p(yi—xi)
5. No, it is not. When we have more dimensions to minimize with, we can always achieve the loss we achieved before just by not using the last dimension. So, the loss with higher k is less or equal to the loss with lower k.
6. Label switching is an issue we face in PCA. It is caused by the fact that span(w1,w2)= span(w2,w1). So, when we change the labels we assign to components, we stil have the same PCA. So, we cannot rely on the labels like in k-means
7. Because with d dimensions we are able to represent all points perfectly. We do not need more than d dimensions. We can just set Z=X and W=I and get 0 error with d-dimensions.
8. It would be stored in matrix W.

9. Using SGD, we can fit huge datasets where SVD is too expensive. So with huge datasets, SGD is faster than SVD as it does not depend on n. bUT WHEN WE USE SGD, we do not anymore enforce the uniqueness constraints and we can run into the problems faces by SGD such as non-convergence or imperfect convergence.

10. (a) using the formula $\frac{\sum_{t=1}^{\infty} \frac{1}{\alpha^2}^2}{\sum_{t=1}^{\infty} \frac{1}{\alpha^2}}$. Using wolfram alpha this expression converge to $\pi^2/15$ which is not equal to 0 so SGD does not converge to a stationary point in this case

    (b) The denominator diverges to positive infinity since it is harmonic series and the numerator converges so SGD will converge to a stationary point in this case

    (c) As shown in class, both sequences diverge but denominator diverges faster so SGD converges to a stationary point in this case

    (d) The fraction is equal to one since numerator is equal to the denominator, so SGD does not converge to a stationary point in this case

11. Using global features allow us to predict what is important to a generic user. Using local features allow us to make personalized predictions