

# CPSC 340 Assignment 5 (due Monday March 18 at 11:55pm)

Student name: José Abraham Torres Juárez

Student id: 79507828

## Instructions

Rubric: {mechanics:5}

**IMPORTANT!!!** Before proceeding, please carefully read the general homework instructions at <https://www.cs.ubc.ca/~fwood/CS340/homework/>. The above 5 points are for following the submission instructions. You can ignore the words “mechanics”, “reasoning”, etc.

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

## 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).

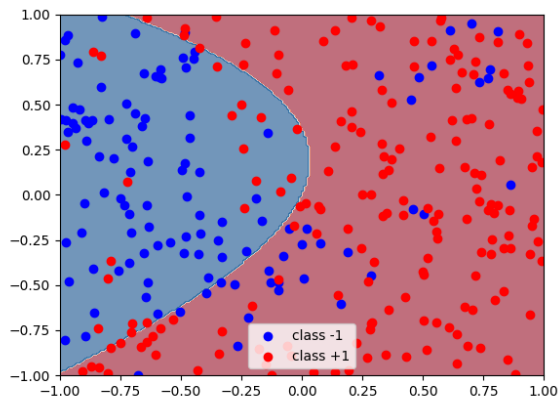
## 1.1 Implementing kernels

Rubric: {code:5}

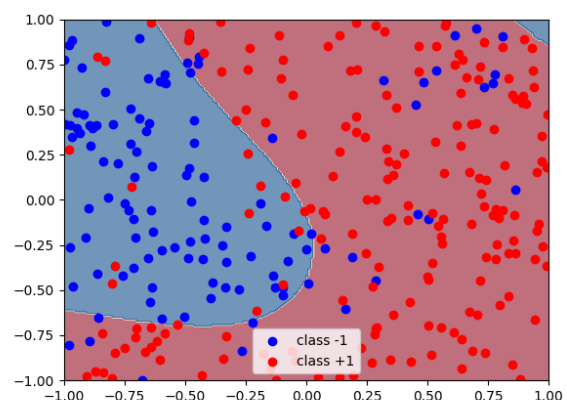
Implement the polynomial kernel and the RBF kernel for logistic regression. Report your training/validation errors and submit the plots generated for each case. You should use the hyperparameters  $p = 2$  and  $\sigma = 0.5$  respectively, and  $\lambda = 0.01$  for the regularization strength.

```
def kernel_RBF(X1, X2, sigma=1):
    n1,d = X1.shape
    n2,d = X2.shape
    Z = np.zeros((n1,n2))
    for i in range(n1):
        for j in range(n2):
            Z[i,j] = np.exp(-((np.sum((X1[i] - X2[j])**2))/(2*sigma)))
    return Z

def kernel_poly(X1, X2, p=2):
    return (1+X1@X2.T)**p
```



Polynomial Kernel



RBF Kernel

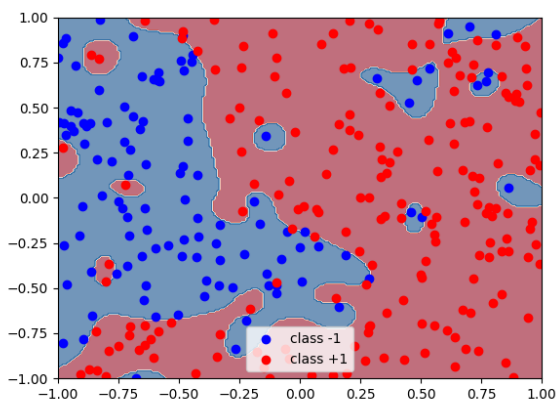
	Training error	Validation Error
Polynomial kernel	0.183	0.170
RBF kernel	0.150	0.130

## 1.2 Hyperparameter search

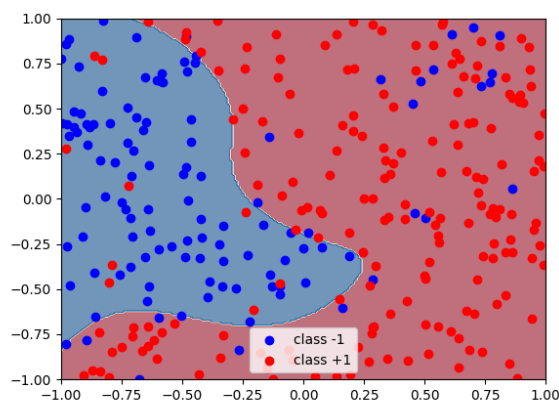
Rubric: {code:3}

For the RBF kernel logistic regression, consider the hyperparameters values  $\sigma = 10^m$  for  $m = -2, -1, \dots, 2$  and  $\lambda = 10^m$  for  $m = -4, -3, \dots, 0$ . 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.



RBF Kernel with best training error



RBF Kernel with best validation error

	Training error	Validation Error	$\sigma$	$\lambda$
Best training	0.017	0.210	0.010	0.0001
Best validation	0.113	0.110	0.100	0.0001

## 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  $\sigma$  and  $\lambda$  affect the fundamental tradeoff?

As we can tell by the images above, the value of  $\lambda$  doesn't change, it appears to be the sweet spot for this data set. On the other hand the value of  $\sigma$  changes, as it grows the validation training error becomes lower but the approximation error also grows.

So we can conclude that as  $\sigma$  grows from 0.01 to 0.1 the approximation error becomes smaller which yields a better validation error.

## 2 MAP Estimation

Rubric: {reasoning:8}

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

- The likelihood  $p(y_i | x_i, w)$  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  $\lambda^{-1}$ .

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  $\sigma^2$ .

$$\begin{aligned} p(y_i | x_i, w) &= \frac{1}{2} \exp(-|w^T x_i - y_i|), \quad p(w_j) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{w_j^2}{2\sigma^2}\right). \\ f(w) &= -\sum_{i=1}^n \log\left[\frac{1}{2} \exp(-|w^T x_i - y_i|)\right] - \log\left[\frac{1}{2\pi\sigma} \exp\left(-\sum_{j=1}^d \frac{w_j^2}{2\sigma^2}\right)\right] \\ f(w) &= -\sum_{i=1}^n \left[\log\left(\frac{1}{2}\right) + \log(\exp(-|w^T x_i - y_i|))\right] - \log\left(\frac{1}{2\pi\sigma}\right) - \log\left[\exp\left(-\sum_{j=1}^d \frac{w_j^2}{2\sigma^2}\right)\right] \\ f(w) &= -\sum_{i=1}^n [\text{constant} + \log(\exp(-|w^T x_i - y_i|))] - \text{constant} + \sum_{j=1}^d \frac{w_j^2}{2\sigma^2} \\ f(w) &= \text{constant} - \sum_{i=1}^n [-|w^T x_i - y_i|] - \text{constant} + \frac{1}{2\sigma^2} \sum_{j=1}^d w_j^2 \\ f(w) &= \text{constant} + \sum_{i=1}^n [|w^T x_i - y_i|] + \frac{1}{2\sigma^2} \|w\|_2^2 \\ f(w) &= \text{constant} + \|Xw - Y\|_1 + \frac{1}{2\sigma^2} \|w\|_2^2 \end{aligned}$$

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

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

You can use  $\Sigma$  as a diagonal matrix that has the values  $\sigma_i^2$  along the diagonal.

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

$$\begin{aligned}
f(w) &= - \sum_{i=1}^n \left[ \log \left( \frac{1}{\sqrt{2\sigma_i^2\pi}} \right) + \log \left( \exp \left( -\frac{(w^T x_i - y_i)^2}{2\sigma_i^2} \right) \right) \right] - \log \left( \frac{\lambda}{2} \right) - \log \left[ \exp \left( -\sum_{j=1}^d \lambda |w_j| \right) \right] \\
f(w) &= - \sum_{i=1}^n \left[ \text{constant} - \frac{(w^T x_i - y_i)^2}{2\sigma_i^2} \right] - \text{constant} + \sum_{j=1}^d \lambda |w_j| \\
f(w) &= \text{constant} - \sum_{i=1}^n \left[ -\frac{(w^T x_i - y_i)^2}{2\sigma_i^2} \right] + \sum_{j=1}^d \lambda |w_j| \\
f(w) &= \text{constant} + \left\| \frac{Xw - Y}{2\Sigma} \right\|_1 + \sum_{j=1}^d \lambda |w_j|
\end{aligned}$$

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

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

where  $\Gamma$  is the “gamma” function (which is always non-negative).

$$\begin{aligned}
f(w) &= - \sum_{i=1}^n \left[ \log \left( \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{(\nu\pi)}\Gamma(\frac{\nu}{2})} \left( 1 + \frac{(w^T x_i - y_i)^2}{\nu} \right)^{-\frac{\nu+1}{2}} \right) \right] - \log \left[ \frac{\sqrt{\lambda}}{\sqrt{2\pi}} \exp \left( -\sum_{j=1}^d \lambda \frac{w_j^2}{2} \right) \right] \\
f(w) &= - \sum_{i=1}^n \left[ \log \left( \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{(\nu\pi)}\Gamma(\frac{\nu}{2})} \right) + \log \left( \left( 1 + \frac{(w^T x_i - y_i)^2}{\nu} \right)^{-\frac{\nu+1}{2}} \right) \right] - \log \left( \frac{\sqrt{\lambda}}{\sqrt{2\pi}} \right) - \log \left[ \exp \left( -\sum_{j=1}^d \lambda \frac{w_j^2}{2} \right) \right] \\
f(w) &= - \sum_{i=1}^n \left[ \text{constant} + \log \left( \left( 1 + \frac{(w^T x_i - y_i)^2}{\nu} \right)^{-\frac{\nu+1}{2}} \right) \right] - \text{constant} + \sum_{j=1}^d \lambda \frac{w_j^2}{2} \\
f(w) &= \text{constant} - \sum_{i=1}^n \left[ \log \left( \left( 1 + \frac{(w^T x_i - y_i)^2}{\nu} \right)^{-\frac{\nu+1}{2}} \right) \right] + \frac{\lambda}{2} \|w\|_2^2 \\
f(w) &= \text{constant} - \log \left( \left( 1 + \frac{1}{\nu} \|Xw - Y\|_2^2 \right)^{-\frac{\nu+1}{2}} \right) + \frac{\lambda}{2} \|w\|_2^2
\end{aligned}$$

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

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

(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.)

$$f(w) = - \sum_{i=1}^n \log \left[ \frac{\exp(y_i w^T x_i) \exp(-\exp(w^T x_i))}{y_i!} \right] - \log(k)$$

$$\begin{aligned}
f(w) &= - \sum_{i=1}^n \left[ \log \left( \frac{1}{y_i!} \right) + \log \left( \exp(y_i w^T x_i) \right) + \log \left( \exp(-\exp(w^T x_i)) \right) \right] - \log(k) \\
f(w) &= - \sum_{i=1}^n \log \left( \frac{1}{y_i!} \right) - \sum_{i=1}^n \log \left( \exp(y_i w^T x_i) \right) - \sum_{i=1}^n \log \left( \exp(-\exp(w^T x_i)) \right) - \log(k) \\
f(w) &= - \left\| \log \left( \frac{1}{y!} \right) \right\|_1 - \sum_{i=1}^n y_i w^T x_i + \sum_{i=1}^n \exp(w^T x_i) - \log(k) \\
f(w) &= - \left\| \log \left( \frac{1}{y!} \right) \right\|_1 - y^T X w + \left\| \exp(X w) \right\|_1 - \log(k)
\end{aligned}$$

### 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?

The first principal component is  $(\frac{-2}{\sqrt{5}}, \frac{1}{\sqrt{5}})$

2. What is the reconstruction loss (L2 norm squared) of the point  $(-3, 2.5)$ ? (Show your work.) Using the first principal component we transform the point  $x = (-3, 2.5)$ ,

$$[-3 \quad 2.5] \times \begin{bmatrix} -\frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix} = \frac{8.5}{\sqrt{5}}$$

Now we calculate the error with the L2 norm squared of  $x$  and  $\hat{x}$

$$loss = \left\| \frac{8.5}{\sqrt{5}} \times \begin{bmatrix} -\frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix} - \begin{bmatrix} -3 \\ 2.5 \end{bmatrix} \right\|_2^2 = \left\| \begin{bmatrix} -\frac{17}{5} \\ \frac{8.5}{5} \end{bmatrix} - \begin{bmatrix} -3 \\ 2.5 \end{bmatrix} \right\|_2^2 = \left\| \begin{bmatrix} -\frac{2}{5} \\ \frac{-4}{5} \end{bmatrix} \right\|_2^2 = \frac{4}{25} + \frac{16}{25} = \frac{20}{25} = \boxed{0.8}$$

3. What is the reconstruction loss (L2 norm squared) of the point  $(-3, 2)$ ? (Show your work.) Using the first principal component we transform the point  $x = (-3, 2)$ ,

$$[-3 \quad 2] \times \begin{bmatrix} -\frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix} = \frac{8}{\sqrt{5}}$$

Now we calculate the error with the L2 norm squared of  $x$  and  $\hat{x}$

$$loss = \left\| \frac{8}{\sqrt{5}} \times \begin{bmatrix} \frac{-2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix} - \begin{bmatrix} -3 \\ 2.5 \end{bmatrix} \right\|_2^2 = \left\| \begin{bmatrix} \frac{-17}{5} \\ \frac{8}{5} \end{bmatrix} - \begin{bmatrix} -3 \\ 2.5 \end{bmatrix} \right\|_2^2 = \left\| \begin{bmatrix} \frac{-1}{5} \\ \frac{5}{5} \end{bmatrix} \right\|_2^2 = \frac{1}{25} + \frac{4}{25} = \frac{5}{25} = \boxed{0.2}$$

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

## 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  $\epsilon$  controls the accuracy of the approximation (a typical value of  $\epsilon$  is 0.0001).

```

class RobustPCA(AlternativePCA):
    pass
    def _fun_obj_z(self, z, w, X, k):
        n,d = X.shape
        Z = z.reshape(n,k)
        W = w.reshape(k,d)

        R = np.dot(Z,W) - X
        f = np.sum((R**2+0.0001)**0.5)
        g = np.dot(R/((R**2+0.0001)**0.5), W.transpose())
        return f, g.flatten()

    def _fun_obj_w(self, w, z, X, k):
        n,d = X.shape
        Z = z.reshape(n,k)
        W = w.reshape(k,d)

        R = np.dot(Z,W) - X
        f = np.sum((R**2+0.0001)**0.5)
        g = np.dot(Z.transpose(), R/((R**2+0.0001)**0.5))
        return f, g.flatten()

```

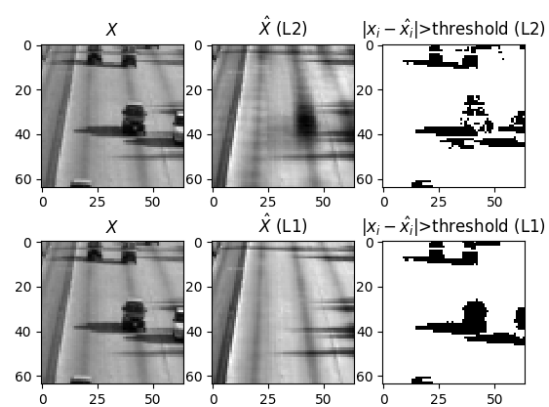
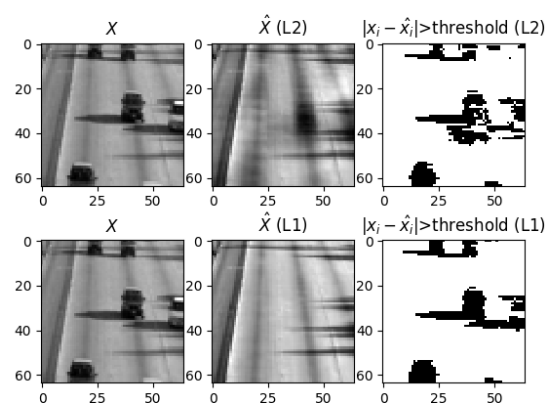
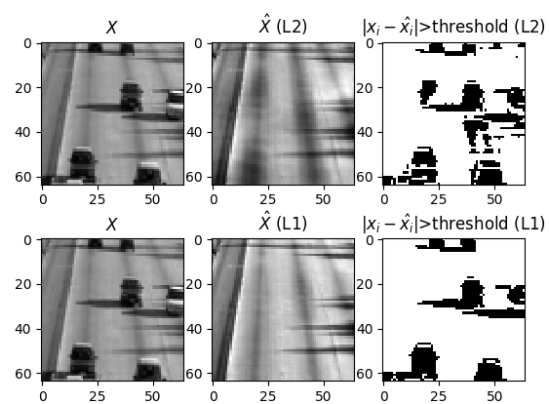
Losses of alternative pca

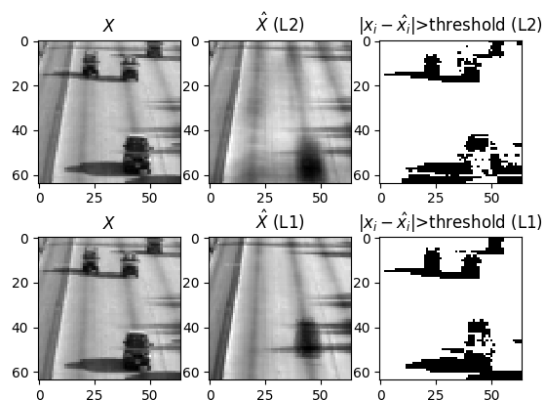
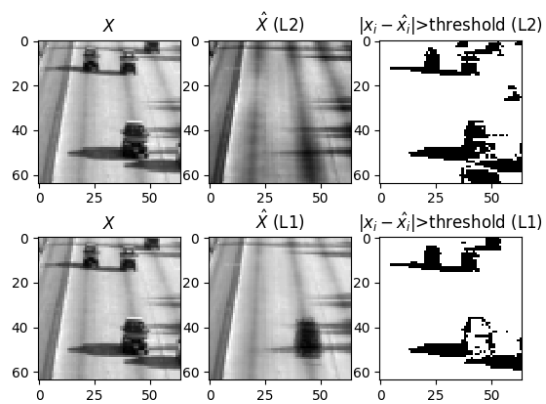
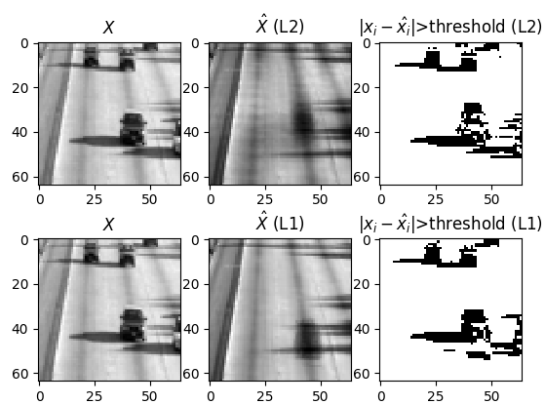
<i>Iteration</i>	<i>loss</i>
0	8020.7
1	7146.9
2	6940.3
3	6861.5
4	6826.9
5	6804.4
6	6785.7
7	6770.5
8	6764.5
9	6764.5

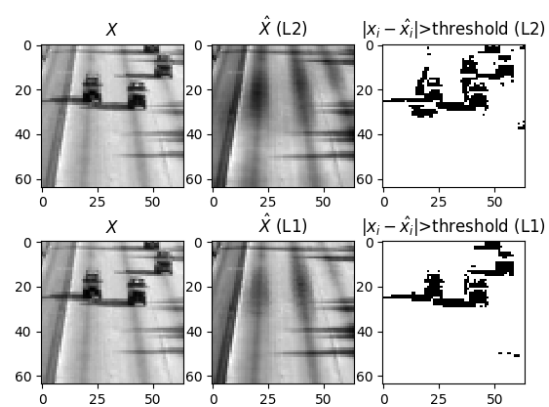
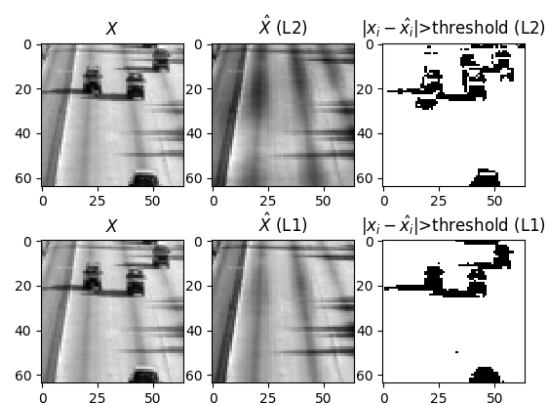
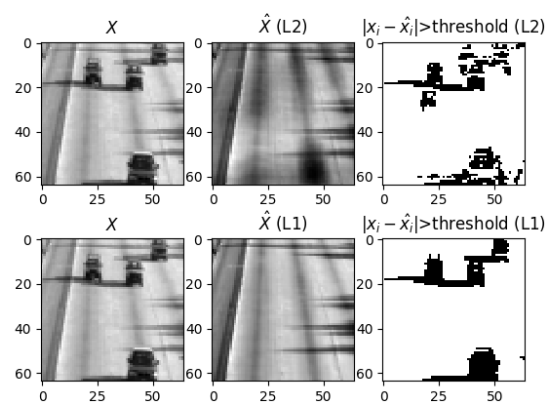
Losses of robsut pca

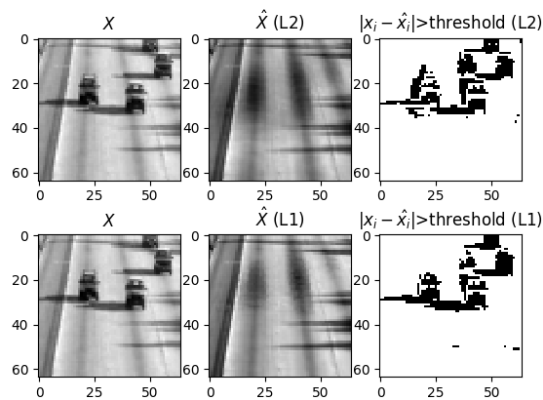
<i>Iteration</i>	<i>loss</i>
0	116386.2
1	96108.6
2	94137.2
3	93078.1
4	92427.2
5	91884.5
6	91449.0
7	91211.6
8	91068.3
9	91000.1











## 4.2 Reflection

Rubric: {reasoning:3}

1. Briefly explain why using the L1 loss might be more suitable for this task than L2.  
The L2 norm gives too much attention to data that is not
2. How does the number of video frames and the size of each frame relate to  $n$ ,  $d$ , and/or  $k$ ?  
 $n$  is the number of frames,  $d$  depends on the size of the image. And  $k$  depends on the whole dataset and how accurate do we want to represent our data, if we want a super accurate representation we will use more PCs than if we want a not so accurate representation.
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)?  
The threshold is the color that a pixel has to be considered as an object on the highway. So if we change it, it will either consider more things as an object or the highway (false positive) or make harder to find objects on the highway (false negatives).

## 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 stochastic 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 advantage of using global features, and what is advantage of using local features?