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

Instructions

Zitong Wang k5j0b Jingyuan Hu z3o0b

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 Regression

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.

Answer:

For Polynomial kernel, we found:

Training error: 0.183

Validation error: 0.170

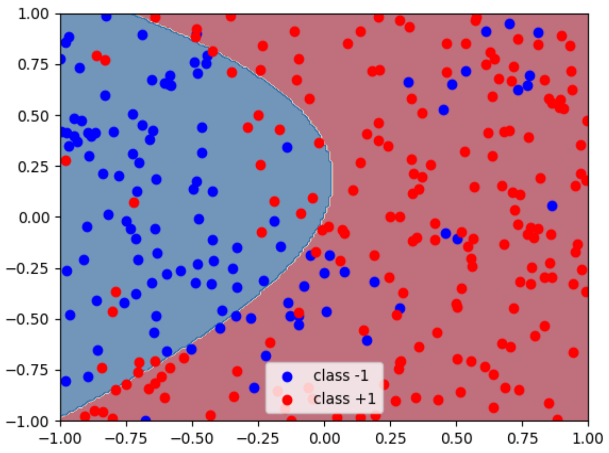
For RBF kernel, we found:

Training error: 0.127

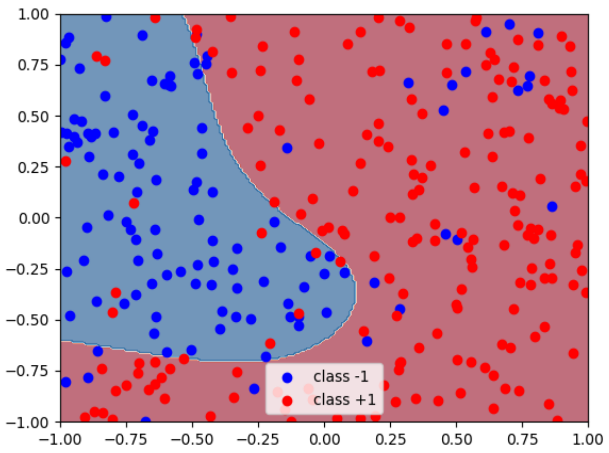
Validation error: 0.090

Plots:

Below is for Polynomial Kernel:



Below is for RBF Kernel:



For code submission, please refer to file "logReg.py" in repository "code".

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.

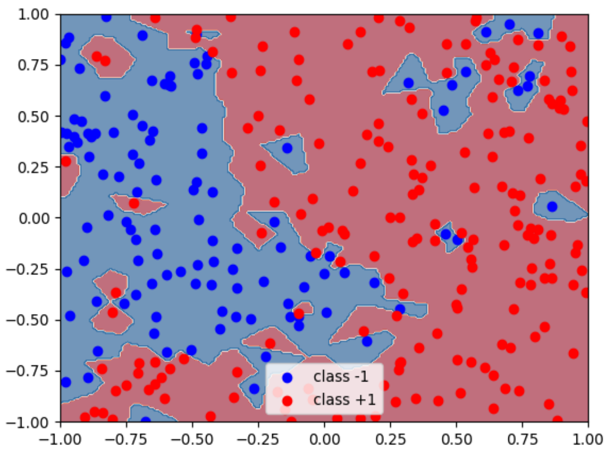
Note: on the job you might choose to use a tool like scikit-learn's `GridSearchCV` to implement the grid search, but here we are asking you to implement it yourself by looping over the hyperparameter values.

Answer:

By testing different values of σ and λ we found that:

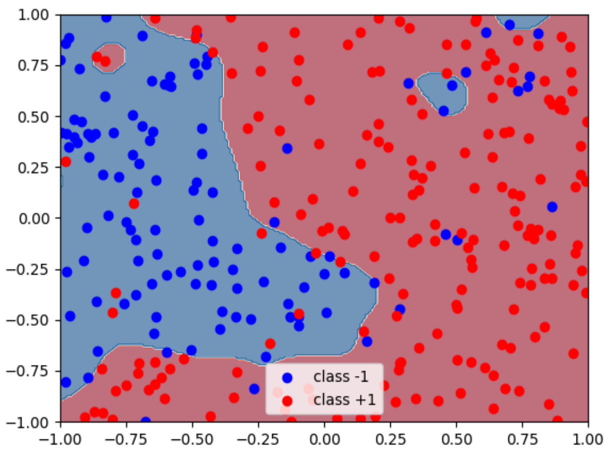
For $\sigma = 10^{-2}$ and $\lambda = 10^{-4}$ we found the lowest training error = 0.000

Plot below:



For $\sigma = 10^{-1}$ and $\lambda = 1$ we found the lowest validation error = 0.120

Plot below:



For code submission, please refer to file "logReg.py" in repository "code".

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?

Answer:

From the plot above, the smaller σ and λ values lead to a smaller training error (a model fits better) and it might also lead to overfitting. As discovered, smaller σ leads to a more complicated model while smaller λ leads to a model with smaller regularization.

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.

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

Answer: $\|Xw - y\|_1 + \frac{1}{2\sigma^2}\|w\|^2$

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.
Answer: $\frac{1}{2}(Xw - y)^T\Sigma^{-1}(Xw - y) + \lambda\|w\|_1$

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).
Answer: $\frac{\nu+1}{2}\sum_{i=1}^n \log(1 + \frac{(w^T x_i - y_i)^2}{\nu}) + \frac{\lambda}{2}\|w\|^2$

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.)
Answer: $\sum_{i=1}^{n} exp(w^T x_i) - y^T X w$

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? $w_1 = (\frac{2\sqrt{5}}{5}, \frac{-\sqrt{5}}{5})$
2. What is the reconstruction loss (L2 norm squared) of the point (-3, 2.5)? (Show your work.)
   Demean the point : $(-3, 2.5) \rightarrow (-3, 1.5)$, then $z = (-3, 1.5) \cdot w_1 = \frac{3\sqrt{5}}{2}$, the reconstruction is $z \cdot w_1 + (0, 1) = (-3, 2.5)$ and hence the reconstruction loss is 0
3. What is the reconstruction loss (L2 norm squared) of the point (-3, 2)? (Show your work.)
   Demean the point : $(-3, 2) \rightarrow (-3, 1)$, then $z = (-3, 1) \cdot w_1 = \frac{-7\sqrt{5}}{5}$, the reconstruction is $z \cdot w_1 + (0, 1) = (-2.8, 2.4)$ and hence the reconstruction loss is $(3 - 2.8)^2 + (2.4 - 2)^2 = 0.2$

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.

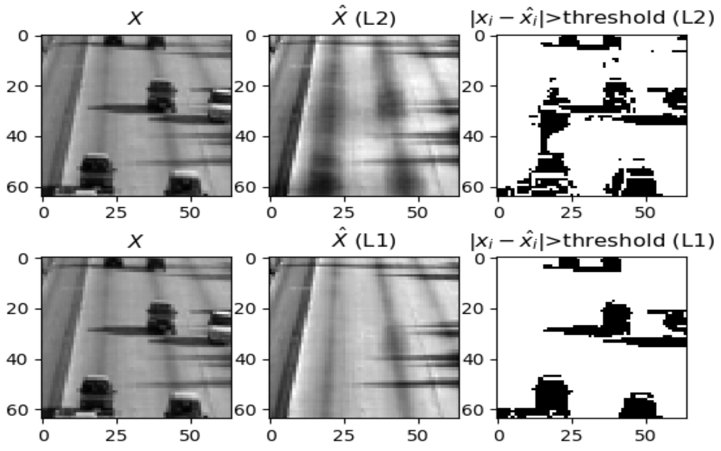
Hint: most of the work has been done for you in the class *pca.AlternativePCA*. This work implements an alternating minimization approach to minimizing the (L2) PCA objective (without enforcing orthogonality). This gradient-based approach to PCA can be modified to use a smooth approximation of the L1-norm. 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:

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

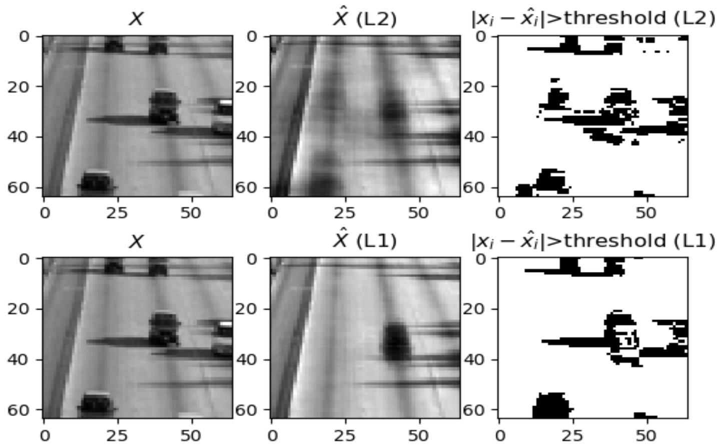
where ϵ controls the accuracy of the approximation (a typical value of ϵ is 0.0001).

Answer:

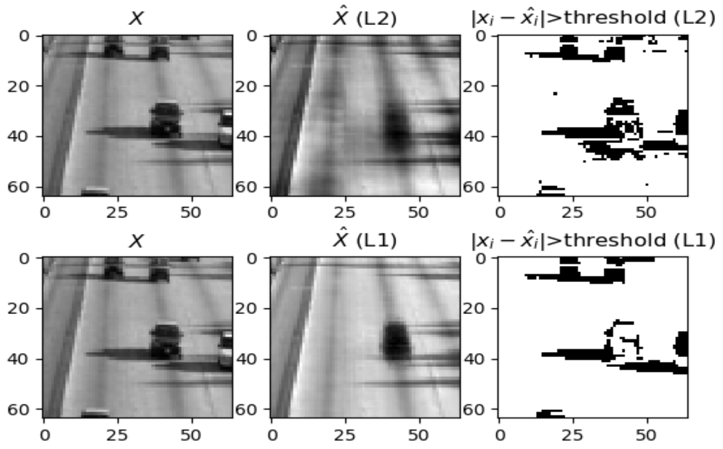
Plots after implementation:



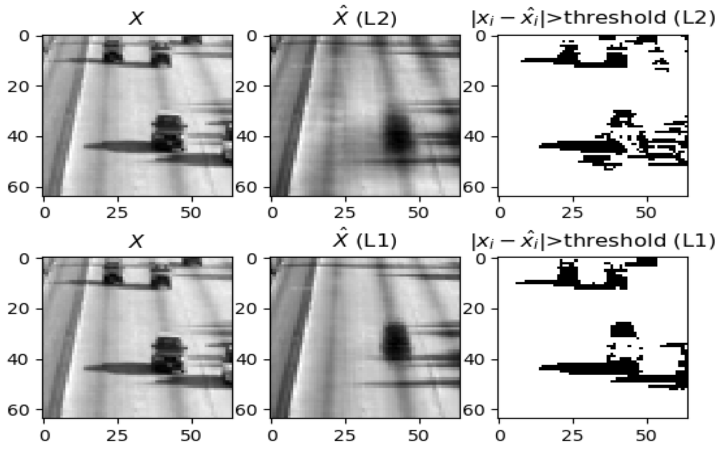
Iteration 0:



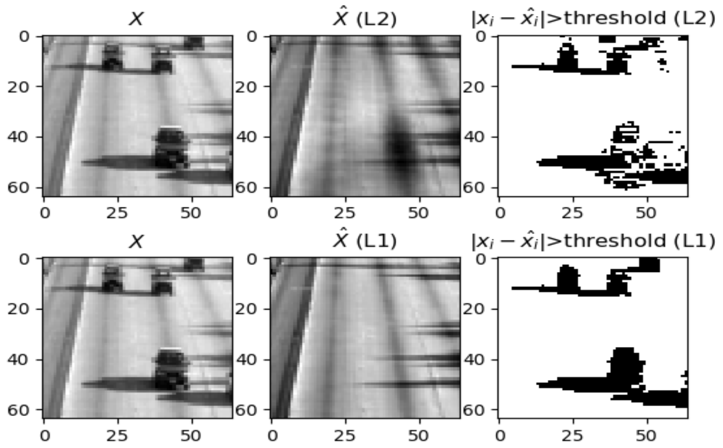
Iteration 1:



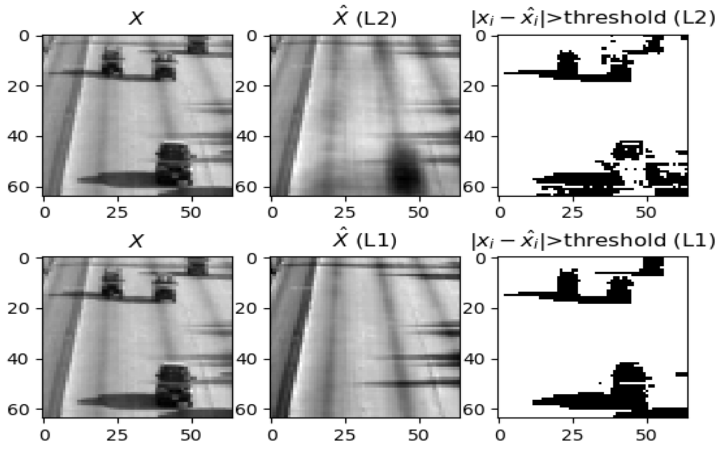
Iteration 2:



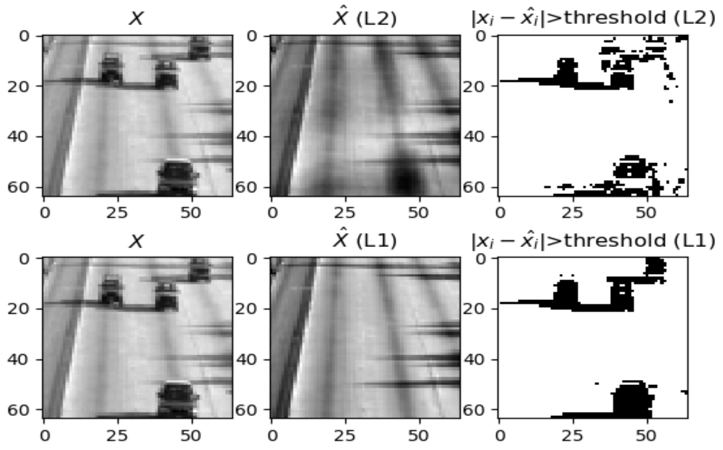
Iteration 3:


X
$\hat{X}$ (L2)
$|x_i - \hat{x}_i|$>threshold (L2)
X
$\hat{X}$ (L1)
$|x_i - \hat{x}_i|$>threshold (L1)

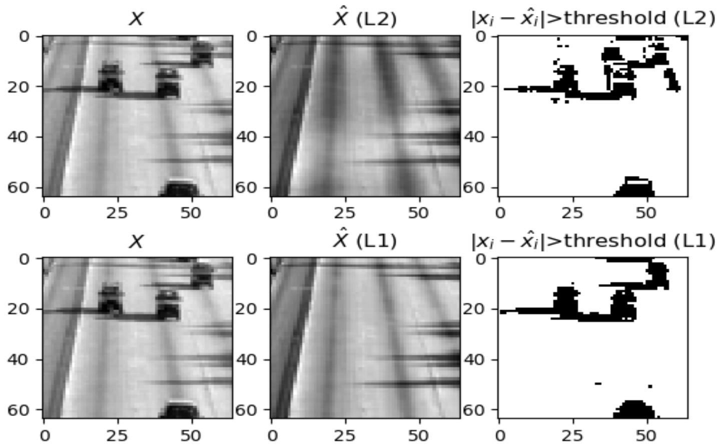
Iteration 4:


X
$\hat{X}$ (L2)
$|x_i - \hat{x}_i|$>threshold (L2)
X
$\hat{X}$ (L1)
$|x_i - \hat{x}_i|$>threshold (L1)

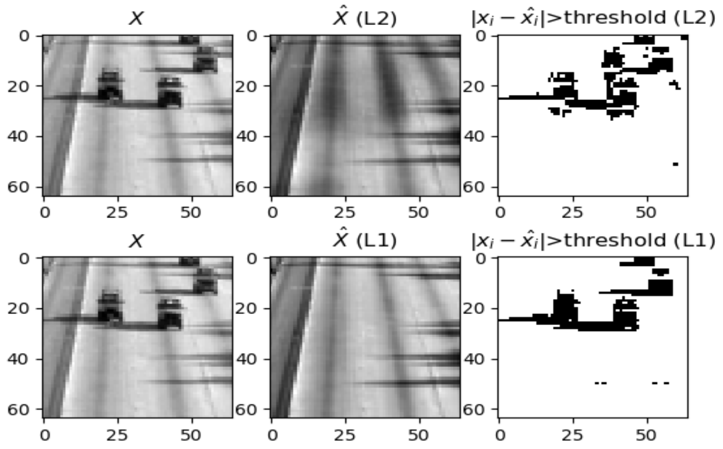
Iteration 5:


X
$\hat{X}$ (L2)
$|x_i - \hat{x}_i|$>threshold (L2)
X
$\hat{X}$ (L1)
$|x_i - \hat{x}_i|$>threshold (L1)

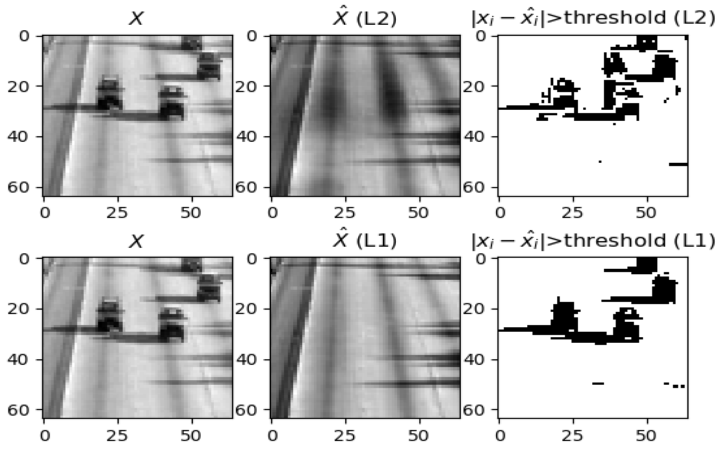
Iteration 6:


X
$\hat{X}$ (L2)
$|x_i - \hat{x}_i|$>threshold (L2)
X
$\hat{X}$ (L1)
$|x_i - \hat{x}_i|$>threshold (L1)

Iteration 7:


X
$\hat{X}$ (L2)
$|x_i - \hat{x}_i|$>threshold (L2)
X
$\hat{X}$ (L1)
$|x_i - \hat{x}_i|$>threshold (L1)

Iteration 8:


X
$\hat{X}$ (L2)
$|x_i - \hat{x}_i|$>threshold (L2)
X
$\hat{X}$ (L1)
$|x_i - \hat{x}_i|$>threshold (L1)

Iteration 9:

Iteration 0: Loss for alternative PCA (At each iteration):

Iteration 0, loss = 7850.8 Iteration 1, loss = 7002.0 Iteration 2, loss = 6853.1 Iteration 3, loss = 6800.7 Iteration 4, loss = 6775.6 Iteration 5, loss = 6767.9 Iteration 6, loss = 6767.9 Iteration 7, loss = 6767.8 Iteration 8, loss = 6767.8 Iteration 9, loss = 6767.7

Loss for Robust PCA (At each iteration):

Iteration 0, loss = 116894.6 Iteration 1, loss = 96735.6 Iteration 2, loss = 93990.9 Iteration 3, loss = 92705.1 Iteration 4, loss = 91849.2 Iteration 5, loss = 91409.6 Iteration 6, loss = 91191.8 Iteration 7, loss = 91083.7 Iteration 8, loss = 91013.1 Iteration 9, loss = 90968.9

From the perspective of loss value, Robust PCA has much more higher loss value than alternative PCA. From the plots we generated above, we can discover that car identification had improved in Robust PCA frame.

For code submission, please refer to file "pca.py" in "code" repository.

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)?

Answer:

1. Since L1 norm is more robust to the outliers, therefore it will ignore the outliers when reconstructing the picture.
2. Since the feature matrix has the dimension (n x d) therefore n determines the number of video frames and d determines the size of each frame. k is user-defined hyper-parameter and k has to be far less than d.
3. By changing the threshold, if we changed it higher, we would have lower false positive and higher false negatives; if we changed it lower, we would have higher false positive but lower 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?
   It will still be computationally more faster if we have less data compared to the number of features.
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?
   Each cluster doesn't have to convex now(still "convex" in some high dimension)
3. In the language of loss functions and regularization, what is the difference between MLE and MAP?
   MAP takes both likelihood and prior into consideration, and the prior is considered as regularization part, where in MLE we only have the loss function part but no regularizer.

4. What is the difference between a generative model and a discriminative model?
   Generative model focus on how the data was generated (often works well with lot of features), while discriminative model categorizes a given data (works better with large dataset or when naive assumptions aren't satisfied)

5. With PCA, is it possible for the loss to increase if k is increased? Briefly justify your answer.
   No, you could at least pick the W as before, so it won't be worse than before

6. What does "label switching" mean in the context of PCA?
   Switching the order of the factors(w_i) leads to non-uniqueness, but they still span the same space

7. Why doesn't it make sense to do PCA with $k > d$?
   PCA try to express high dimension data through low dimension, so when $k > d$ it will be meaningless. (It would exactly represent the model and hence no error)

8. In terms of the matrices associated with PCA (X, W, Z, $\hat{X}$), where would an "eigenface" be stored?
   Stored in W as a row

9. What is an advantage and a disadvantage of using stochatic gradient over SVD when doing PCA?
   Stochastic gradient is genearlly better for for large-scale problem (huge datasets, big X matrices), but does not enfore the uniqueness "constraints" and is sensitive to initialization (where SVD does not)

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) $\alpha^t = 1$.

    The ones that converge: $1/t$(but works badly in practise), $1/\sqrt{t}$

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?
    Global features: predict what is important to a generic user (e.g a new user); Local features: makes prediction personalized (e.g an old user)