CPSC 340 Assignment t (due November 27 ATE)

1 PCA Generalizations

1.1 Robust PCA

The function *example_RPCA* loads 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 of the first 50 frames (pausing for a tenth of a second on each frame):[1]

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,

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

and it has recently been proposed as a more effective model for background subtraction. Write a new function, *robustPCA*, that uses a smooth approximation to the absolute value to implement robust PCA. Hand in your code.

Hint: most of the work has been done for you in the function *PCA_gradient*. This function implements an alternating minimization approach to minimizing the 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).

[1] If any of you are still having trouble with PyPlot on Windows, I found these instructions worked on multiple computers: `https://stackoverflow.com/questions/46399480/julia-runtime-error-when-using-pyplot`

1.2 L1-Regularized and Binary Latent-Factor Models

We have a matrix X, where we have observed a subset of its individual elements. Let $\mathcal{R}$ be the set of indices (i, j) where we have observed the element x_{ij}. We want to build a model that predicts the missing entries, so we use a latent-factor model with an L1-regularizer on the coefficients W and a separate L2-regularizer on the coefficients Z,

$$f(Z, W) = \frac{1}{2} \sum_{(i,j)\in\mathcal{R}} \left[(w_j^T z_i - x_{ij})^2\right] + \lambda_W \sum_{j=1}^{d} \left[\|w_j\|_1\right] + \frac{\lambda_Z}{2} \sum_{i=1}^{n} \left[\|z_i\|^2\right],$$

where the regularization parameters satisfy $\lambda_W > 0$ and $\lambda_Z > 0$.

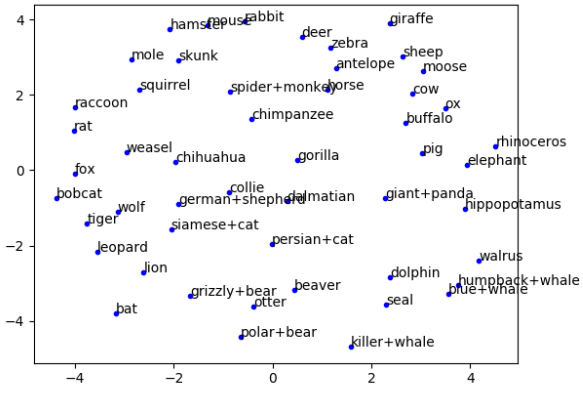
1. What is the effect of λ_W on the sparsity of the parameters W and Z?
2. What is the effect of λ_Z on the sparsity of W and Z?
3. What is the effect of λ_Z on the two parts of the fundamental trade-off in machine learning?
4. What is the effect of k on the two parts?
5. Would either of answers to the *previous two questions* change if $\lambda_W = 0$?
6. Suppose each element of the matrix X is either $+1$ or -1 and our goal is to build a model that makes the sign of $w_j^T z_i$ match the sign of x_{ij}. Write down a (continuous) objective function that would be more suitable.

2 Multi-Dimensional Scaling

The function *example_MDS* loads the animals dataset and then applies gradient dsecent to minimize the following multi-dimensional scaling (MDS) objective (starting from the PCA solution):

$$f(Z) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} (\|z_i - z_j\| - \|x_i - x_j\|)^2. \tag{1}$$

The result of applying MDS is shown below.



Although this visualization isn't perfect (with "gorilla" being placed close to the dogs and "otter" being placed close to two types of bears), this visualization does organize the animals in a mostly-logical way.

2.1 ISOMAP

Euclidean distances between very different animals are unlikely to be particularly meaningful. However, since related animals tend to share similar traits we might expect the animals to live on a low-dimensional manifold. This suggests that ISOMAP may give a better visualization. Make a new function *ISOMAP* that computes the approximate geodesic distance (shortest path through a graph where the edges are only between nodes that are k-nearest neighbour) between each pair of points, and then fits a standard MDS model (1) using gradient descent. Hand in your code and the plot of the result when using the 3-nearest neighbours.

Hint: the function *dijskstra* (in *misc.jl*) can be used to compute the shortest (weighted) distance between two points in a weighted graph. This function requires an n by n matrix giving the weights on each edge (use ∞ as the weight for absent edges). Note that ISOMAP uses an undirected graph, while the k-nearest neighbour graph might be asymmetric. You can use the usual heuristics to turn this into an undirected graph of including an edge i to j if i is a KNN of j or if j is a KNN of i. (Also, be careful not to include the point itself in the KNN list).

2.2 ISOMAP with Disconnected Graph

An issue with measuring distances on graphs is that the graph may not be connected. For example, if you run your ISOMAP code with 2-nearest neighbours then some of the distances are infinite. One heuristic to address this is to set these infinite distances to the maximum distance in the graph (i.e., the maximum geodesic distance between any two points that are connected), which will encourage non-connected points to be far apart. Modify your ISOMAP function to implement this heuristic. Hand in your code and the plot of the result when using the 2-nearest neighbours.

3 Neural Networks

The file `example_nnet.jl` runs a stochastic gradient method to train a neural network on the *basisData* dataset from a previous assignment. However, in its current form it doesn't fit the data very well. Modify the training procedure to improve the performance of the neural network. Hand in your plot after changing the code to have better performance, and list the changes you made.

Hint: there are many possible strategies you could take to improve performance. Below are some suggestions, but note that the some will be more effective than others:

- Changing the network structure (*nHidden* is a vector giving the number of hidden units in each layer).
- Changing the training procedure (you can change the stochastic gradient step-size, use mini-batches, run it for more iterations, add momentum, switch to *findMin*, and so on).
- Transform the data by standardizing the feautres, standardizing the targets, and so on.
- Add regularization (L2-regularization, L1-regularization, dropout, and so on).
- Add bias variables within the hidden layers.
- Change the loss function or the non-linearities (right now it uses squared error and tanh to introduce non-linearity).

4 Very-Short Answer Questions

1. Is the NMF loss function convex? What is an optimization method you could use to try to minimize it?
2. Why is it difficult for a standard collaborative filtering model to make good predictions for new items?
3. If a dataset in 3-dimensions is confined to a plane, is an algorithm MDS with $k = 2$ guaranteed to find a representation with an error of zero? What about PCA?
4. What is the difference between Euclidean distance and geodesic distance?
5. Why is a sigmoid used as a "squashing" function in neural networks?
6. Assuming we could globally minimize the neural network objectve, how does the depth of a neural network affect the fundamental trade-off?
7. List 3 forms of regularization we use to prevent overfitting in neural networks.
8. Assuming we could globally minimize the neural network objectve, how would the width of the convolutions in a convolutational neural network affect the fundamental trade-off?

We're looking for short and concise 1-sentence answers, not long and complicated answers. Also, there is roughly 1-2 questions per lecture.