Homework 3 Part 1 - Solutions

Problem 1

Suppose you are performing classification using the k-NN algorithm. Would you be able to apply regularization with lasso or ridge regularization? Why or why not?

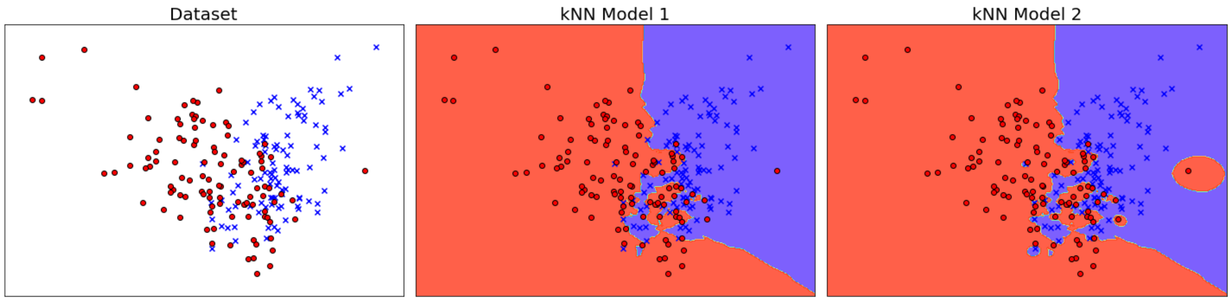
Regularization is introduced to the objective function as a penalty term on the model's parameters. k-NN is a non-parametric algorithm. A point is classified purely based on the training data for a given distance metric and talling scheme. Therefore feature selection using the Lasso regularizer is not feasible for k-NN.

Problem 2

Consider the following two-dimensional dataset with two classes (shown with "circles" and "crosses"), and its decision surface performance as computed using k-Nearest Neighbor (kNN) algorithm with $k = 3$. Describe in words the major differences you observe in the two decision surfaces, and provide a discussion about the voting system utilized to obtain each respective decision surface. Justify your answer.

```
In [2]: from IPython.display import Image
        Image('figures/kNN-performances.png',width=900)
```

Out[2]:



Both decision surfaces correspond to the performance of a k-NN algorithm, this is visibly clear because the edges of the decision surface are not smooth for regions of overlap.

Moreover, the difference between performance 1 and 2 are for points that stand on its own, for example the class circle/orange on the far center right, as a couple of crosses/blue samples in the overlapping region. The performance illustrated in 2 corresponds to k-NN with a weighted distance and the performance in 1 correspond to k-NN with standard distance metric.

Problem 3

Recall that the Support Vector Machine objective function is

$$C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|w\|^2$$

subject to the constraints

$$t_n y(x_n) \geq 1 - \xi_n, n = 1, \ldots, N \quad (1)$$
$$\xi_n \geq 0, n = 1, \ldots, N \quad (2)$$

Answer the following questions:

1. **From the training set, which points are used to make predictions during the test stage? Explain your reasoning.**
2. **C is a parameter that is set by the user. Describe the relationship between values of C and the resulting SVM decision surface, performance and number of support vectors.**

1. During training, each sample x_n will have an associated Lagrange multiplier $a_n \geq 0$. Sample points that are correctly classified (away from the margin) will have an associated slack variable $\xi_n = 0$ and the Lagrange multiplier $a_n = 0$.

All other points will have $a_n > 0$:

- If $0 < a_n < C$, then $\mu_n > 0$ (because $a_n = C - \mu_n$) thus $\xi_n = 0$. So x_n is a support vector (exactly on the margin).
- If $a_n = C$, then all corresponding samples x_n will have $\mu_n = 0$ and $\xi_n > 0$. So x_n is either correctly classified but inside the margin ($0 < \xi_n < 1$) or misclassified ($\xi_{\geq}1$).

In summary, samples x_n that are support vectors, inside the margin or misclassified are used to make inferences during the test stage.

1. The parameter C controls the trade-off between slack variable and the margin. Decreasing C will soften the margin and increase the number of support vectors.

Problem 4

List similarities and differences, any assumptions, computational complexity and convergence criteria for the following classifiers: the Perceptron, Logistic Regression, Fisher's LDA and Support Vector Machine (SVM).

As classifiers, all listed algorithms are linear discriminative classifiers.

- Logistic Regression is a probabilistic discriminative classifier, which means that it leverages the posterior probability as a confidence in the decision. In addition, it utilizes an unconstrained, smooth objective. The resulting decision surfaces is a collection of singly convex regions for each class.
- The Perceptron algorithm only converges if the classes are linearly separable.
- Fisher's LDA assumes classes are Gaussian-distributed, which may result in a poor separation.
- SVMs utilize the concept of a margin. Soft-margin SVM promotes generalization by introducing penalty variables (slack variables). SVM is the most robust and complex algorithm. It utilizes a constrained objective function which results in a computational complexity $O(N^3)$, where N is the number of samples.

Problem 5

Suppose you have a binary (2-class) classification problem. Answer the following questions:

1. **Describe how the logistic regression and the SVM find a decision surface to classify the two classes.**
2. **When would you choose SVM over Logistic Regression and vice-versa? Justify your answers.**

1. Logistic regression focuses on maximizing the probability of the data. The farther the data lies from the separating hyperplane (on the correct side), the better. An SVM tries to find the separating hyperplane that maximizes the distance of the closest points to the margin (the support vectors). If a point is not a support vector, it does not affect the placement of the hyperplane.
2. As always, it depends on the problem. Logistic Regression: (1) provides calibrated probabilities that can be interpreted as confidence in a decision; (2) utilizes an unconstrained, smooth objective; and, (3) can be straightforwardly used within Bayesian models. SVMs: (1) do not penalize examples for which the correct decision is made with sufficient confidence. this may be better for generalization; and, (2) have a nice dual form, giving sparse solutions when using the kernel trick, which provides better scalability.

Problem 6

Suppose you have the following training labels associated with binary classifier:

$$t = \{1, 1, 0, 0, 1, 0, 0, 1, 1, 0\}$$

Suppose you trained a logistic regression classifier to produce a confidence of target given a sample. For the above data points, your classifier produced the following confidence values:

$$c = \{0.85, 0.9, 0.1, 0.05, 0.7, 0.2, 0.1, 1.0, 0.8, 0.9\}$$

the corresponding weighted sum $z = \mathbf{w}^T\mathbf{x} + w_0$ are:

$$z = \{4, 5, -5, -6, 1, -1.5, -1, -1, 4.5, 5.5\}$$

1. **Create a table with several entries (at least 10) that computes TPR and FPR for a list of possible thresholds for the confidence c, δ.**
2. **From the table you computed in 1, draw the associated ROC curve.**
3. **Which threshold would you use to achieve a FPR $\leq 20\%$?**
4. **Suppose you decided your threshold point of operation is ≥ 0.8. What is the resulting confusion matrix?**

1. Let's first organize the data in decreasing order of their confidence:

c	t
0.05	0
0.1	0
0.1	0
0.2	0
0.7	1
0.8	1
0.85	1
0.9	1
0.9	0
1.0	1

Recall that $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$ and $\text{FPR} = \frac{\text{FP}}{\text{FP+TN}}$.

The threshold table is as follows:

δ	TP	TN	FP	FN	TPR	FPR

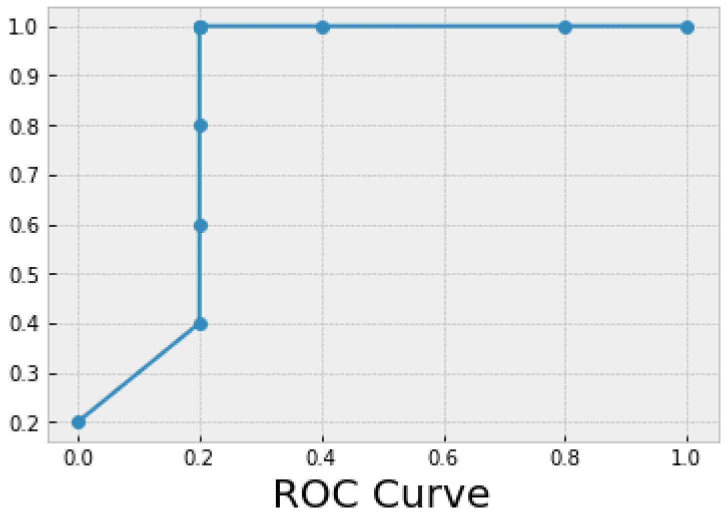
δ	TP	TN	FP	FN	TPR	FPR
0	5	0	5	0	$\frac{5}{5}$	$\frac{5}{5}$
0.1	5	1	4	0	$\frac{5}{5}$	$\frac{4}{5}$
0.2	5	3	2	0	$\frac{5}{5}$	$\frac{2}{5}$
0.5	5	4	1	0	$\frac{5}{5}$	$\frac{1}{5}$
0.6	5	4	1	0	$\frac{5}{5}$	$\frac{1}{5}$
0.7	5	4	1	0	$\frac{5}{5}$	$\frac{1}{5}$
0.8	4	4	1	1	$\frac{4}{5}$	$\frac{1}{5}$
0.85	3	4	1	2	$\frac{3}{5}$	$\frac{1}{5}$
0.9	2	4	1	3	$\frac{2}{5}$	$\frac{1}{5}$
1.0	1	5	0	4	$\frac{1}{5}$	0

1. The associated ROC curve is:

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('bmh')

tpr = np.array([1, 1, 1, 1, 1, 1, 4/5, 3/5, 2/5, 1/5])
fpr = np.array([1, 4/5, 2/5, 1/5, 1/5, 1/5, 1/5, 1/5, 1/5, 0])

plt.plot(fpr, tpr, '-o')
plt.xlabel('FPR', size=20)
plt.xlabel('TPR', size=20)
plt.xlabel('ROC Curve', size=20);
```



1. A threshold $0.5 \leq \delta \leq 0.7$ will retrieve 20% FPR and TPR=100%.

1. If the threshold is $\delta = 0.8$, that is, prediction is 1 if $c \geq \delta$, the resulting confusion matrix is:

	Predicted +	**Predicted -**
True +	TP = 4	FN=1
True -	FP= 1	TN=4

Problem 7

Suppose you would like to use PCA to reduce dimensionality of your data prior to classification. Is PCA always an effective dimensionality reduction technique to be used in conjunction with classification? Why or why not?

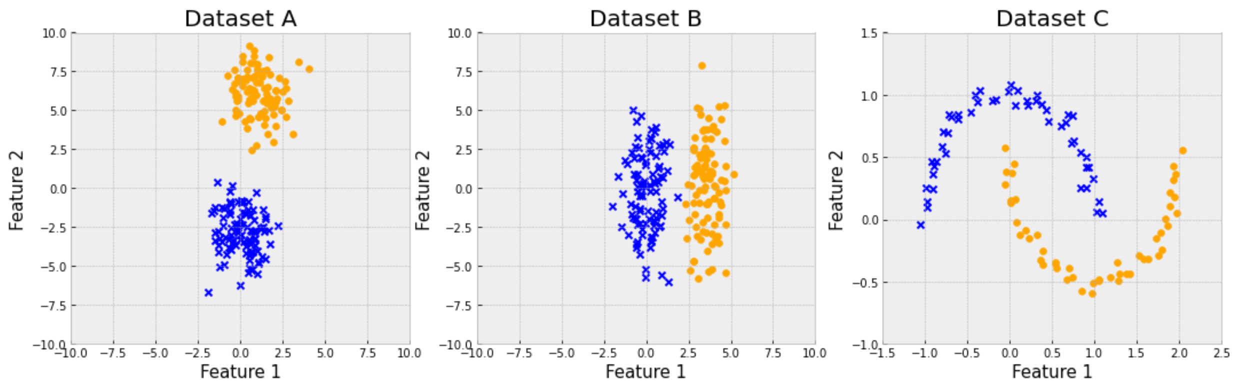
No. PCA is an unsupervised algorithm for performing dimensionality reduction. It finds a lower-dimensional space that maximizes explained variance but it will not necessarily maximize class separability, which may pose a problem to a subsequent classifier.

Problem 8

Consider the following three two-dimensional datasets containing two classes (depicted as "circles" and "crosses").

In [3]:
```
Image('figures/2-D-datasets.png',width=800)
```

Out[3]:



Suppose you would like to apply Principal Component Analysis (PCA) to reduce the dimensionality of each of these datasets from 2-D to 1-D where the two clusters remain separated in the projection space. For each dataset (A, B and C), address each of the following two questions:

1. **Will PCA be effective at keeping the two clusters separated in the 1-D projection? Why or why not? If yes, state what characteristics of the dataset allow PCA to be**

effective. If no, state what characteristics of the dataset cause PCA to fail.

2. **Can you think of another dimensionality reduction technique that would be successful at reducing the dimensionality for this dataset while maintaining (or increase) class separability? State the other method and describe why it would be successful.**

1. PCA will not be effective at keeping the two clusters separated in the 1-D projection for datasets B and C. For dataset B, the directions of maximum variance corresponds to the direction parallel to the canonical y-axis (as it represents a wider range of values, even after normalization). On the other hand, dataset C cannot be represented with a linear subspace where classes are linearly separated, so PCA will fail. For dataset A, PCA will be effective at keeping the classes separable because the principal component with largest variance corresponds to the canonical y-axis.
2. For dataset B, instead of PCA, LDA would be a preferred dimensionality reduction technique as it projects the data to a subspace that maximizes class separability (which, in this case, will be the parallel to the canonical x-axis). For dataset C, LDA would still not perform well. The best approach to this problem is to perform kernel PCA, where the data is first transformed into a higher dimensional space and then we find a linear subspace projection (standard PCA).

Problem 9

Consider the data matrix $\mathbf{X}$ of size $3 \times N$, where N is the number of samples. The covariance matrix $\mathbf{K}$, of size 3×3 has 3 eigenvectors $v_1 = [-0.99, 0.09, 0]^T$, $v_2 = [0, 0, 1]^T$ and $v_3 = [-0.09, -0.99, 0]^T$ with eigenvalues $\lambda_1 = 0.98$, $\lambda_2 = 0.5$ and $\lambda_3 = 1.98$, respectively. Answer the following questions:

1. **What linear transformation would you use to uncorrelate the data $\mathbf{X}$? Provide a numerical solution and justify your answer.**
2. **Use Principal Component Analysis (PCA) to project the 3-dimensional space to a 2-dimensional space. Define the linear transformation (using a numerical answer).**
3. **What is the amount of explained variance of this 2-D projection? Show your work.**
4. **Let $\mathbf{Y}$ be the data (linear) transformation obtained by principal component transform of $\mathbf{X}$ onto a 2-dimensional space. What is resulting covariance matrix of transformed data $\mathbf{Y}$? Use a numerical answer and justify your answer.**

1. We can use PCA to uncorrelate the data by performing a change of basis linear

transformation, using the eigenbasis of the covariance matrix.

The transformation can be written as

$$Y = AX$$

where

$$A = \begin{bmatrix} v_3 & v_1 & v_2 \end{bmatrix}^T = \begin{bmatrix} -0.09 & -0.99 & 0 \\ -0.99 & 0.09 & 0 \\ 0 & 0 & 1 \end{bmatrix}^T$$

1. PCA will project the data X into a 2-D space where the two main directions correspond to the orthonormal eigenvectors of the covariance matrix of X with the largest eigenvectors, in particular, v_3 and v_1. The 2-D transformation can be written as

$$Y = AX$$

where

$$A = \begin{bmatrix} v_3 & v_1 \end{bmatrix}^T = \begin{bmatrix} -0.09 & -0.99 \\ -0.99 & 0.09 \\ 0 & 0 \end{bmatrix}^T$$

1. The amount of explained variance corresponds to the sum of the eigenvalues of the directions of projection:

$$\frac{1.98 + 0.98}{1.98 + 0.98 + 0.5} \approx 0.8555 \Rightarrow 85.55\%$$

1. The covariance of the PCA projection is a diagonal matrix (uncorrelated features), and the elements along the diagonal corresponds to the eigenvalues of the eigenvectors of projection:

$$\mathrm{cov}(Y) = \begin{bmatrix} 1.98 & 0 \\ 0 & 0.98 \end{bmatrix}$$

Problem 10

What are the differences and similarities between Fisher's Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA) as dimensionality reduction approaches? When would you prefer LDA over PCA or vice-versa?

PCA is an unsupervised algorithm for finding a basis (set of vectors) to (1) uncorrelate the data, by preserving the dimensionality, and (2) dimensionality reduction.

As an unsupervised algorithm, PCA does not use class labels to find such basis but rather it finds the data-centric vectors that explain the most variance in the data. (A similar derivation of PCA also shows that the PCA basis can be found by minimizing the projection error.)

If the data variance direction of projection is not the one that maximizes class separability than PCA will not be able to preserve class separability.

An alternative dimensionality reduction technique is LDA, a supervised algorithm to find a basis whose projection maximize class separability. LDA assumes class Gaussianity and can only project up to $C - 1$ dimensions, where C is the number of classes in the data set.