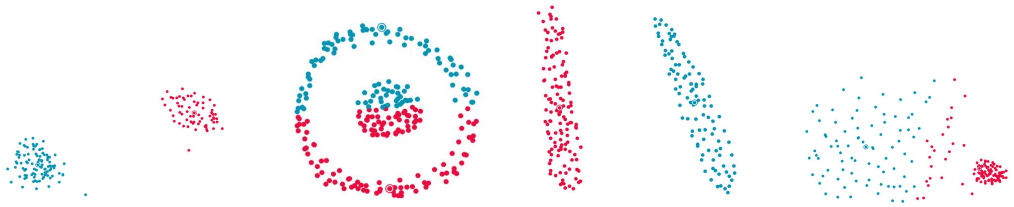
Homework 2
Introduction to Machine Learning
Arya Jalali
98105665

1 Clustering and K-means

1.1 Intuitive Understanding



(a) (b) (c) (d)

Figure 1: (a) Two small blobs (linearly separable) (b) Two circles (c) Two arrows (linearly separable) (d) Dense pocket with noise

From the photos we can infer, that datasets that are linearly separable can be meaningfully separated. This stands for two of our datasets(Two small blobs and Two arrows), but when we try to separate two classes that cannot be linearly separated, we get clusters that don't make much sense. This is because at it's core, K-means is a linear classifier. We can prove this by showing the decision boundary is a hyperplane

$$\|X - C_1\|^2 = \|X - C_2\|^2$$

$$\|X\|^2 + \|C_1\|^2 - 2C_1.X = \|X\|^2 + \|C_2\|^2 - 2C_2.X$$

$$2(C_1 - C_2).X + \|C_2\|^2 - \|C_1\|^2 = 0$$

$$(C_1 - C_2).X + \frac{\|C_2\|^2 - \|C_1\|^2}{2} = 0$$

$$W.X + C = 0$$

Feature scaling can't really help us with the circle dataset, because it cannot change the non-linear nature of data. Feature scaling can however give us a better result for the Two arrows dataset, because of the scaling difference between the x and y axis. Feature scaling may or may not help with the Dense pocket dataset; after normalization the dense pocket will still be compact, but the scattered points would form a dense pocket themselves, this two pockets might get tangled, and still not be linearly separable. We can't really be sure.

1.2 Finding proper value of K

- Inertia is not a normalized metric: we just know that lower values are better and zero is optimal. But in very high-dimensional spaces, Euclidean distances tend to become inflated (this is an instance of the so-called "curse of dimensionality"). Running a dimensionality reduction algorithm such as Principal component analysis (PCA) prior to k-means clustering can alleviate this problem and speed up the computations.Silhouette analysis doesn't suffer from "curse of dimensionality" because of it's normalized coefficient.

- $k = 5$ seems to be the best pick because of two reasons. Firstly, all of our clusters have an above average silhouette score. Secondly, there isn't much variation in their sizes, and are about the same size.

1.3 Applications of Clustering

1.3.1 Semi-supervised learning

The continuity, or smoothness, assumption indicates that close-together datapoints are likely to have the same label.

Similarly, the cluster assumption indicates that, in a classification problem, data tends to be organized into high-density clusters, and that datapoints of the same cluster are likely to have the same label. Therefore, a decision boundary should not lie in areas of densely packed datapoints; rather, it should lie in-between high-density regions, separating them into discrete clusters.

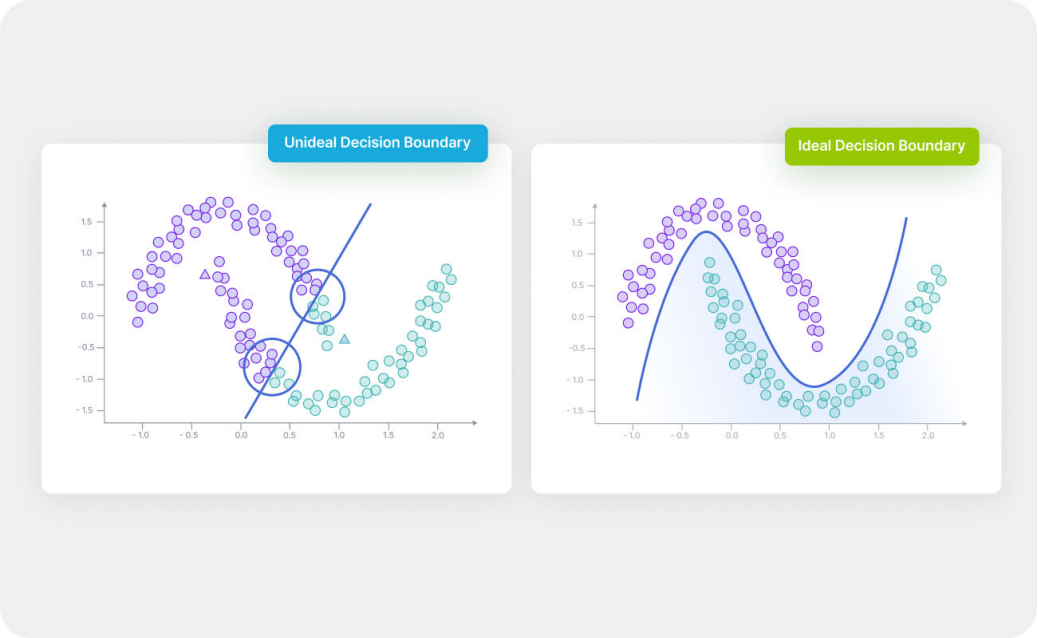


Figure 2: Unideal Decision Boundary vs. Ideal Decision Boundary (Picture taken from v7labs.com)

1.3.2 Active learning

Much like its counterpart, active learning tries to find patterns by clustering data together and establishing a relationship between features, and then verifying these relationships with labeled data. This technique, however, can query a user for information about anomalous data, and get a better understanding of data than semi-supervised learning which can only work with data that is given to us at the beginning of training.

2 Whitening Using PCA

We know from linear algebra the covariance matrix of X can be constructed from $\frac{1}{n}X^TX$ (covariance matrix should express the relation between our features, not our data, so X^TX is the correct formulation.). in addition we can project our data onto principle components t_j with a simple matrix multiplication $T = XW$, where columns of W are orthonormal eigenvectors of X^TX.

The intuition behind the matrix multiplication is that we want to project our data points onto our new basis, because our new basis is orthonormal, we can find the coefficinet of our projection along a vector in our basis by calculating the dot product of our original data sample with said vector.

$$W = \begin{bmatrix} | & \dots & | \\ \frac{w_1}{\sqrt{\lambda_1}} & \dots & \frac{w_m}{\sqrt{\lambda_m}} \\ | & \dots & | \end{bmatrix}$$

Calculating the covariance matrix of XW we get

$$cov(XW) \propto W^T X^T XW$$

$$X^T XW = \begin{bmatrix} | & \dots & | \\ X^T X \frac{w_1}{\sqrt{\lambda_1}} & \dots & X^T X \frac{w_m}{\sqrt{\lambda_m}} \\ | & \dots & | \end{bmatrix} = \begin{bmatrix} | & \dots & | \\ \sqrt{\lambda_1} w_1 & \dots & \sqrt{\lambda_m} w_m \\ | & \dots & | \end{bmatrix}$$

$$W^T \begin{bmatrix} | & \dots & | \\ \sqrt{\lambda_1} w_1 & \dots & \sqrt{\lambda_m} w_m \\ | & \dots & | \end{bmatrix}$$

$$= \begin{bmatrix} w_1.w_1 & w_1.w_2 & w_1.w_3 & \cdots & \cdots & \cdots & \cdots & w_1.w_m \\ w_2.w_1 & w_2.w_2 & w_2.w_3 & \ddots & & & & \vdots \\ w_3.w_1 & w_3.w_2 & w_3.w_3 & w_3.w_4 & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & \vdots & & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \vdots \\ w_m.w_1 & \cdots & \cdots & \cdots & \cdots & \cdots & w_{m-1}.w_m & w_m.w_m \end{bmatrix} = I$$

The last equality comes from w_i's being orthonormal.

Now we need to prove the mean of our data is zero. Note that we can first normalize our data to have 0 mean, we then show this property is preserved after our transformation.

$$T = \begin{bmatrix} | & \dots & | \\ t_1 & \dots & t_m \\ | & \dots & | \end{bmatrix}$$

$$t_i = Xw_i = w_1^{(i)}x_1 + \cdots + w_p^{(i)}x_p$$

$$E[t_i] = E[Xw_i] = w_1^{(i)}E[x_1] + \cdots + w_p^{(i)}E[x_p] = 0$$

Which shows the sum of each column of our new data matrix is 0, therefore the mean of our new matrix is 0 and it's covariance matrix is I. Note that we can scale the covariance matrix however we want.

3 Linear Regression

3.1 Lagrange Multipliers

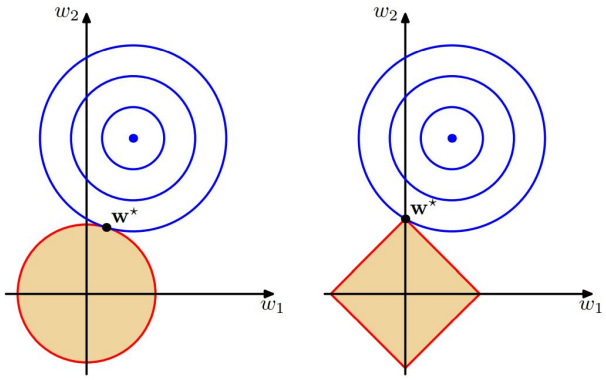


Figure 3: Plot of the contours of the unregularized error function (blue) along with constraint region. Optimum value for the parameter vector w is denoted by w^*. The lasso gives a sparse solution in which $w_1^* = 0$

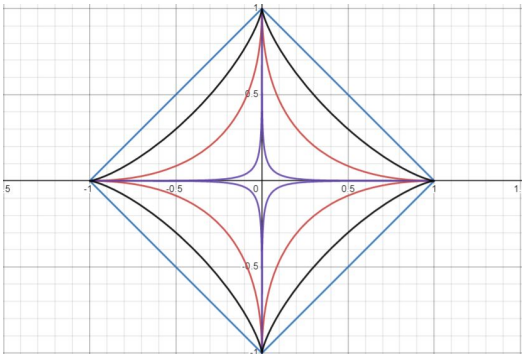


Figure 4: Plot of unit ball for different values of p. p = 1 (blue), p = 0.75 (black), p = 0.5 (red), p = 0.25 (purple)

It is evident from the figure above, that decreasing the value of p gives us a more compact unit ball, which leads to more sparse solutions (contours

of the unregularized error function can sink deeper to get to w^* compared to lasso norm).

The convexity of the unit ball of L_1 norm is much more important and convenient for us than the scant improvement we get from using smaller values of p. The reason is the vast amount of convex optimization algorithms that can achieve an optimal value briskly.

All that remains is for us to prove that unit ball is not convex for $p < 1$

Let S_p be the set of all values of w that are inside the unit ball and e_i the vector with a 1 in the ith coordinate and 0's elsewhere.

$$e_1 \in S_p, e_2 \in S_p$$

If S_p is convex then $te_1 + (1-t)e_2$ should be in the set for all values of $0 \leq t \leq 1$. We let $t = 0.5$

$$\frac{1}{2}^p + \frac{1}{2}^p > 1 \equiv \frac{1}{2}^p > \frac{1}{2}$$

The last inequality stands for all $0 < p < 1$, which tells us that the middle point of our two vectors is not part of the set, therefore S_p is non-convex.

3.2 Ridge Regression (optional)

We are going to make a series of assumptions, and see how under those assumptions the ridge regression loss comes up naturally.
If we want to maximize the MAP estimation of our weights based on our data, we can write

$$Max(P(w|y,x)) \equiv maxP(y|x,w) \times P(w)$$

If $P(y|w,x)$ follows a normal distribution with $w^T x$ mean and σ^2 variance, and $P(w)$ follows a normal distribution with zero mean and b variance, we can rewrite the above equation as

$$Max(P(w|y,x)) \equiv Max(\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-||y-Xw||^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi b}} e^{\frac{-||w||^2}{2b}})$$

Maximizing the above equation is equivalent to minimizing it's negative log.

$$Min(\frac{1}{2\sigma^2}||y - Xw||^2 + \frac{1}{2b}||w||^2) \equiv Min(||y - Xw||^2 + \frac{\sigma^2}{b}||w||^2)$$

Replacing $\frac{\sigma^2}{b}$ with λ gives our desired loss.

Note that we got rid of constant coefficients $\frac{1}{\sqrt{2\pi\sigma^2}}$ and $\frac{1}{\sqrt{2\pi b}}$, because they had no effect on the global minima or maxima.

Taking the derivative with respect to the vector w and setting it to 0 we have

$$\frac{\partial}{\partial w}(Xw - y)^T(Xw - y) + \lambda w^T w = \frac{\partial}{\partial w} w^T X^T X w - w^T X^T y - y^T X w + \lambda w^T w$$

Note that $w^T X^T y$ is a scalar, and we have $w^T X^T y = y^T X w$

$$\frac{\partial L(w)}{\partial w} = \frac{\partial}{\partial w} w^T X^T X w - 2y^T X^T w + \lambda w^T w$$

To make sense of the above equation, we need to use the following matrix calculus identities

$$\begin{cases} \frac{\partial}{\partial w} w^T w = 2w^T \\ \frac{\partial}{\partial w} a^T w = a \\ \frac{\partial}{\partial w} w^T A w = w^T(A + A^T) \end{cases}$$

$$\frac{\partial L(w)}{\partial w} = 2w^T(X^T X) - 2y^T X^T + 2\lambda w^T$$

Setting it to zero gives us the optimal value for w

$$w^T(X^T X + \lambda I) = y^T X^T \equiv (X^T X + \lambda I)w = X^T y \rightarrow w^* = (X^T X + \lambda I)^{-1} X^T y$$

4 Generalization Error

4.1 Dimensionality Reduction

The curse of dimensionality occurs because the sample density decreases exponentially with the increase of the dimensionality. When we keep adding features without increasing the number of training samples as well, the dimensionality of the feature space grows and becomes sparser and sparser.

Due to this sparsity, it becomes much easier to find a "perfect" solution for the machine learning model which highly likely leads to overfitting.

Overfitting happens when the model corresponds too closely to a particular set of data and doesn't generalize well. An overfitted model would work too well on the training dataset so that it fails on future data and makes the prediction unreliable.

4.2 Data Augmentation

4.2.1 Translation

Translation just involves moving the image along the X or Y direction (or both). In the following example, we assume that the image has a black background beyond its boundary, and are translated appropriately. This method of augmentation is very useful as most objects can be located at almost anywhere in the image. This forces your convolutional neural network to look everywhere.

Figure 5: Translation of the ball can help our model look at different positions of the picture and become more robust

4.2.2 Gaussian Noise

Over-fitting usually happens when your neural network tries to learn high frequency features (patterns that occur a lot) that may not be useful. Gaussian noise, which has zero mean, essentially has data points in all frequencies, effectively distorting the high frequency features. This also means that lower frequency components (usually, your intended data) are also distorted, but your neural network can learn to look past that. Adding just the right amount of noise can enhance the learning capability.

4.2.3 Backtranslation

A sentence is translated in one language and then a new sentence is translated again in the original language. So, different sentences are created.

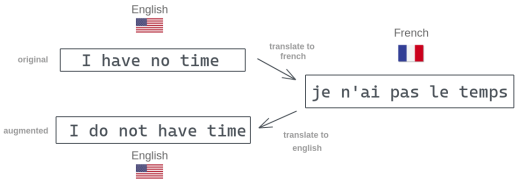


Figure 6: Creating new (but related) sentences from our original data

5 Kernels

5.1 Feature Space

Using $x.y = \|x\| \, \|y\| \, cos(\angle x, y)$ and expanding the expression we have

$$(1 + \frac{x.y}{\|x\| \, \|y\|})^2 = 1 + \frac{x.y}{\|x\| \, \|y\|} + \frac{(x.y)^2}{\|x\|^2 \, \|y\|^2}$$

For each part of the above equation, we can find a specific feature space, and then concatenate them at the end to get $\phi(x)$

$$\phi_1(x) = (1)$$

$$\phi_2(x) = (\frac{\sqrt{2}x_1}{\|x\|}, \ldots, \frac{\sqrt{2}x_n}{\|x\|}),$$

$$\phi_3(x) = (\frac{x_1^2}{\|x\|^2}, \frac{x_2^2}{\|x\|^2}, \ldots, \frac{x_n^2}{\|x\|^2}, \frac{\sqrt{2}x_1x_2}{\|x\|^2}, \ldots, \frac{\sqrt{2}x_1x_n}{\|x\|^2}, \frac{\sqrt{2}x_2x_3}{\|x\|^2}, \ldots, \frac{\sqrt{2}x_{n-1}x_n}{\|x\|^2})$$

Combining all three we get

$$\phi(x) = (1, \phi_2(x), \phi_3(x))$$

5.2 Kernel Matrix

5.2.1

We can fix m, and generalize our proof to $l > m$ using induction.

If k is a valid kernel, then

$$\exists \phi(X) \; s.t \; k(x, y) = \phi(x).\phi(y)$$

We will use the notation $\phi(x_i) = s_i$ for the rest of our proof

$$v^T \begin{bmatrix} s_1.s_1 & s_1.s_2 & s_1.s_3 & \cdots & \cdots & \cdots & \cdots & s_1.s_m \\ s_2.s_1 & s_2.s_2 & s_2.s_3 & \ddots & & & & \vdots \\ s_3.s_1 & s_3.s_2 & s_3.s_3 & s_3.s_4 & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & \vdots & & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \vdots \\ s_m.s_1 & \cdots & \cdots & \cdots & \cdots & \cdots & s_{m-1}.s_m & s_m.s_m \end{bmatrix} v$$

$$= \sum_k \sum_i v_k v_i (x_k.x_i) = (v_1x_1 + v_2x_2 + \cdots + v_mx_m).(v_1x_1 + v_2x_2 + \cdots + v_mx_m)$$

$$= \|v_1x_1 + v_2x_2 + \cdots + v_mx_m\| \therefore v^T K v \geq 0$$

The last equation is the condition for Semi positive definiteness.

5.2.2

Every real symmetric matrix can be diagonalized, so we can represent the kernel matrix (for a fixed m) like below

$$K = PDP^T$$

Where D and P are diagonal and orthogonal respectively.

$$K = \begin{bmatrix} | & \dots & | \\ p_1 & \dots & p_m \\ | & \dots & | \end{bmatrix} \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_m \end{bmatrix} \begin{bmatrix} — & p_1^T & — \\ \dots & \dots & \dots \\ — & p_m^T & — \end{bmatrix}$$

$$\begin{bmatrix} | & \dots & | \\ d_1p_1 & \dots & d_mp_m \\ | & \dots & | \end{bmatrix} \begin{bmatrix} — & p_1^T & — \\ \dots & \dots & \dots \\ — & p_m^T & — \end{bmatrix} = d_1p_1p_1^T + \cdots + d_mp_mp_m^T = M$$

$$M_{ij} = d_1p_i^{(1)}p_j^{(1)} + \cdots + d_mp_i^{(m)}p_j^{(m)} = \phi(x_i).\phi(x_j)$$

$$\phi(x_i) = (\sqrt{d_1}p_i^{(1)}, \sqrt{d_2}p_i^{(2)}, \dots, \sqrt{d_m}p_i^{(m)})$$

$$\phi(x_j) = (\sqrt{d_1}p_j^{(1)}, \sqrt{d_2}p_j^{(2)}, \dots, \sqrt{d_m}p_j^{(m)})$$

We managed to find a feature mapping $\phi(x)$ that corresponds to our kernel; proving that k is a valid kernel.

6 Feature Expansion

6.1

6.1.1

Let us denote the center of the line (center of red poins) c, we put forward the following transform

$$\phi_1(x) = |x - c|$$

After transforming the data using the above function, because the distance of the red points is lower than that of the blue points, they can be linearly separated.

6.1.2

Because of the radius difference of two circles, we can make use of their norms

$$\phi_2(x) = x_1^2 + x_2^2 = \|x\|$$

6.1.3

If we perform the ϕ_2 transform on our data, our transformed data will be like the first line. We can again transform this data using $\phi_1(x)$ to get the final linearly separable data (here we choose c to be the radius of the second circle).

$$\phi_3(x) = \phi_1(\phi_2(x))$$

6.2

Expanding the equation gives us

$$x_1^2 + a^2 - 2x_1a + x_2^2 + b^2 - 2bx_2 = r^2 \equiv (-2a, -2b, 1, 1).(x_1, x_2, x_1^2, x_2^2) = r^2 - (a^2 + b^2)$$

$$\begin{cases} f(x) = 1 \; if \; \Theta.(x_1, x_2, x_1^2, x_2^2) \leq r^2 - (a^2 + b^2) \\ f(x) = 0 \; if \; \Theta.(x_1, x_2, x_1^2, x_2^2) > r^2 - (a^2 + b^2) \end{cases}$$

Which defines a linear separator.

7 SVM Decision Boundaries

- (a) : RBF kernel, $\gamma = 10$, $C = 1$
- (b) : Linear kernel, $C = 1$
- (c) : Linear kernel, $C = 0.1$
- (d) : RBF kernel, $\gamma = 0.1$, $C = 15$
- (e) : RBF kernel, $\gamma = 1$, $C = 3$
- (f) : Linear kernel, $C = 10$

Linearity comes from the decision boundaries being straight lines.

C acts a regularization parameter. For larger values of C, SVM tries to increase its accuracy and minimizing its bias at the cost of a smaller margin; This can explain our answer for linear kernels.

The γ parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors.

The behavior of the model is very sensitive to the γ parameter. If γ is too large, the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with C will be able to prevent overfitting.

When γ is very small, the model is too constrained and cannot capture the complexity or "shape" of the data. The region of influence of any selected support vector would include the whole training set. The resulting model will behave similarly to a linear model.

Correlating γ with the complexity of our model, we can see which plot is for which γ.

8 Monte Carlo Cross Validation

8.1

If n_t is proportional to the size of the training set, overfitting might occur and we might end up with a high and low variance and bias respectively. If

we take n_t to be small, our model cannot learn the distribution of our data, and might lack complexity, resulting in a low variance and high bias.

8.2

Each method has its own advantages and disadvantages. Under k-fold cross validation, each point gets tested exactly once, which seems fair. However, k-fold cross-validation only explores a few of the possible ways that your data could have been partitioned. Monte Carlo lets you explore somewhat more possible partitions, though you're unlikely to get all of them–there are $\binom{100}{50} \approx 10^{28}$ possible ways to 50/50 split a 100 data point set.

If you're attempting to do inference (i.e., statistically compare two algorithms), averaging the results of a k-fold cross validation run gets you a (nearly) unbiased estimate of the algorithm's performance, but with high variance. Since you can, in principle, run it for as long as you want/can afford, Monte Carlo cross validation can give you a less variable, but more biased estimate.