COMPSCI 671D Fall 2019
Homework 3

Vinayak Gupta
vg101

1 Kernels (25 points)

Suppose we have two valid kernels k_1, k_2, prove/disprove that the following kernels are valid.

(a) (3 points)

$$k(x, z) = \alpha k_1(x, z) + \beta k_2(x, z), \text{ for } \alpha, \beta \geq 0$$

Solution Based on Mercer's Theorem, we have $k_1(x, z) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x)\phi_i(z)$, similar form for $k_2(x, z)$ can be obtained. Therefore,

$$k(x, z) = \alpha \sum_{i=1}^{\infty} \lambda_i \phi_i(x)\phi_i(z) + \beta \sum_{j=1}^{\infty} \eta_j \psi_j(x)\psi_j(z)$$

$$= \sum_{i=1}^{\infty} \alpha\lambda_i \phi_i(x)\phi_i(z) + \beta\eta_i \psi_i(x)\psi_i(z)$$

$$=< \mathbf{\Upsilon}(x), \mathbf{\Upsilon}(z) >_{H_{new}}$$

where $\Upsilon(x) = [..., \sqrt{\alpha\lambda_i}\phi_i(x), \sqrt{\beta\eta_i}\psi_i(x), ...]$

This means that $k(x, z)$ can be expressed as an inner product where the components of the inner products itself contain feature maps of valids kernels and is **VALID**, also $\alpha, \beta \geq 0$

(b) (3 points)

$$k(x, z) = k_1(x, z)k_2(x, z)$$

Solution Based on Mercer's Theorem, we have $k_1(x, z) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x)\phi_i(z)$, similar form for $k_2(x, z)$ can be obtained. Therefore,

$$k(x, z) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x)\phi_i(z) \sum_{j=1}^{\infty} \eta_j \psi_j(x)\psi_j(z)$$

$$= \sum_{i,j} \lambda_i \phi_i(x)\phi_i(z)\eta_j \psi_j(x)\psi_j(z)$$

$$= \sum_{i,j} \lambda_i \eta_j [\phi_i(x)\psi_j(x)][\phi_i(z)\psi_j(z)]$$

$$=< \boldsymbol{\Upsilon}(x), \boldsymbol{\Upsilon}(z) >_{H_{new}}$$

where $\Upsilon(x) = [..., \sqrt{\lambda_i \eta_j}\phi_i(x)\psi_j(x), ...]$

This means that $k(x, z)$ can be expressed as an inner product where the components of the inner products itself contain feature maps of valids kernels and is **VALID**

(c) (3 points)

$$k(x, z) = f(x)f(z) \text{ for } f : \mathcal{X} \to \mathbb{R}$$

Solution We need to show the following two propoerties:

1) Symmetric:

$$k(x, z) = f(x)f(z)$$
$$= f(z)f(x) => k(z, x)$$

where the last line follows from the commutative property of multiplication.

2) k should give rise to a positive semi-definite "Gram matrix"

Given any dataset $x_i, i = 1, ..., N$, define vector $f = (f(x_1), f(x_2), ..., f(x_N))^T$

$$K = ff^T$$

then K is Positive Semi definite as $\forall v$

$$v^T K v = v^T f f^T v$$

$$= (v^T f)^2 \geq 0$$

Since, both the properties are satisfied, the kernel is **VALID**

(d) (3 points)

$$k(x, z) = f(k_1(x, z)), \text{ where } f(x) \text{ is a polynomial with positive coefficients.}$$

Solution Any polynomial can be wriiten generally as:

$$f(k_1(x, z)) = a_n(k_1(x, z))^n + a_{n-1}(k_1(x, z))^n + ... + a_0$$

It is given that all coefficients are positive, also this polynomial is nothing but sum of products of kernels with positive coefficients.
From 1(b), any product of valid kernels is a kernel. Also from 1(a), any sum of product of valid kernels with positive coefficients is a valid kernel. Therefore the above kernel $k(x, z)$ is a **VALID** kernel.

(e) (3 points)

$$k(x,z) = \exp(-\gamma\|x-z\|_2^2)$$

Solution First let us show that $k_2(x,z) = \exp(k_1(x,z))$ is a valid kernel. By Taylor expansion,

$$exp(k_1(x,z)) = \lim_{i\to\infty}(1 + k_1(x,z) + ... + \frac{k_1(x,z)^i}{i!})$$

This again is a polynomial function with positive coefficients, and it follows from 1(d) that this $k_2(x,z)$ is a valid Kernel function.

Now, $k(x,z) = \exp(-\gamma\|x-z\|_2^2)$, lets consider first the case when γ is ≥ 0 :

$$k(x,z) = \exp(-\gamma\|x-z\|_2^2) = \exp(-\gamma(\|x\|_2^2 - \|z\|_2^2 + 2x^Tz))$$
$$= \exp(-\gamma\|x\|_2^2)\exp(-\gamma\|z\|_2^2)\exp(\gamma 2x^Tz)$$

We know f(x)f(z) is a valid kernel from 1(c), and we showed above $exp(k_1(x,z))$ is a valid kernel. And by 1(b), product of two valid kernel is a valid kernel.

However, this fails when $\gamma < 0$ as can be shown by the following counter example:
Suppose the dataset has 2 points. The gram matrix defined by these points is:

$$\begin{bmatrix} k(x,x) & k(x,z) \\ k(z,x) & k(z,z) \end{bmatrix} = \begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix}$$

Now $\alpha = k(x,z) = \exp(-\gamma\|x-z\|_2^2) > 1$ as the value inside the exponent is always positive. For a kernel to be valid, the Gram Matrix defined by it should be Positive Semi-definite, which means the determinant must be ≥ 0. However, the determinant of the above matrix is, $1-(\alpha)^2$ and since $\alpha > 1$, the determinant is negative. This means that when $\gamma < 0$, then not a valid kernel.

Therefore, if $\gamma \geq 0$, then a **VALID** kernel , else **INVALID**.

(f) (5 points) Show that the **rbf** kernel for some fixed $\gamma \in \mathbb{R}^+$ corresponds to the inner product of some feature map ϕ in infinite dimensional space, that is,

$$k(x,z) = \exp(-\gamma\|x-z\|_2^2) = \phi(x)^T\phi(z)$$

where $x,z \in \mathcal{X}, \quad \phi : \mathcal{X} \to \mathcal{X}^\infty$

Solution From 1(e), $k(x,z) = \exp(-\gamma\|x\|_2^2)\exp(-\gamma\|z\|_2^2)\exp(\gamma 2x^Tz)$, also assume that $x,z \in \mathcal{R}$, W.L.O.G we can use Taylor expansion on the last term:

$$= k(x,z) = \exp(-\gamma\|x\|_2^2)\exp(-\gamma\|z\|_2^2)\left[1 + \gamma 2x^Tz + \frac{(\gamma 2x^Tz)^2}{2!} + ...\right]$$

$$= k(x,z) = \exp(-\gamma\|x\|_2^2)\exp(-\gamma\|z\|_2^2)\left[1 + \sqrt{\gamma 2}x\sqrt{\gamma 2}z + \sqrt{\frac{(\gamma 2)^2}{2!}}x^2\sqrt{\frac{(\gamma 2)^2}{2!}}z^2 + ...\right]$$

$$= \phi(x)^T\phi(z)$$

where $\phi(x) = \exp(-\gamma\|x\|_2^2)[1, \sqrt{\gamma 2}x, \sqrt{\frac{(\gamma 2)^2}{2!}}x^2, ...]$
Therefore, rbf kernel can be written as inner product of some feature map ϕ in infinite dimensional space.

(g) (5 points) Suppose we have a dataset $\{x_i, y_i\}_{i=1}^n, x_i \in \mathbb{R}^p, y_i \in \{-1, 1\}$ with no conflicting labels, where a conflicting label means $\exists i, j \quad s.t. \quad x_i = x_j, y_i \neq y_j$. Show that a SVM classifer with the **rbf** kernel can always achieve 0 training error.

Solution An SVM classifier can be written as :

$$f(x^{new}) = \sum_j \lambda_j^* x^{new(j)} + \lambda_0^*$$

$$= \sum_i \alpha_i^* y_i < \phi(x_i), \phi(x^{new}) >_{H^k} + \lambda_0^*$$

$$= \sum_i \alpha_i^* y_i k(x_i, x^{new}) + \lambda_0^*$$

We have an rbf kernel, suppose that $\gamma \to \infty$, then the above SVM classifier can be reduced in the following way while prediciting a point in the training dataset (as we need to show training error is 0):

$$f(x_j) = \sum_{i \neq j} \alpha_i^* y_i k(x_i, x_j) + \lambda_0^* + \alpha_j^* y_j k(x_j, x_j)$$

As $\gamma \to \infty$ the value of rbf kernel $k(x_i, x_j)$ is 0 when $x_i \neq x_j$ and is equal to 1 when $x_i = x_j$ by simple limit properties.

When $x_i \neq x_j, -\gamma\|x_i - x_j\|_2^2 \to -\infty, \exp(-\gamma\|x - z\|_2^2) \to 0$

When $x_i = x_j, -\gamma\|x_i - x_j\|_2^2 \to 0, \exp(-\gamma\|x - z\|_2^2) \to 1$
Therefore, the above SVM classifier reduces to:

$$f(x_j) = \lambda_0^* + \alpha_j^* y_j$$

We also know that out SVM problem solves the following equation:

$$maxL(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i,k} \alpha_i \alpha_k y_i y_k k(x_i, x_k)$$

s.t. $\alpha_i \geq 0$, $i = 1, ..., n$ and $\sum_{i=1}^n \alpha_i y_i = 0$

Again, we can reduce the equation to the following equation:

$$maxL(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_i {\alpha_i}^2$$

as ${y_i}^2 = 1$. Since, this is a constrained optimization problem, we can construct a Lagrangian for it:

$$L'(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_i {\alpha_i}^2 + a[\sum_{i=1}^{n} \alpha_i y_i]$$

From Lagrangian Stationarity condition, we get $\frac{\partial L'}{\partial \alpha_i} = 1 - \alpha_i + a y_i = 0 \ \forall i$

Therefore, $a = \frac{1-\alpha_i}{y_i}$, select two positive points, i and j, then $\frac{1-\alpha_i}{y_i} = \frac{1-\alpha_j}{y_j}$ as a is constant(dual variable) across all i's. Since, $y_i = y_j$, for positive points $\alpha_i = \alpha_j = \alpha_+$. Similarly for negative points, it is $\alpha_i = \alpha_j = \alpha_-$. If we select a positive point i and a negative point j, then again using the equation $\frac{1-\alpha_i}{y_i} = \frac{1-\alpha_j}{y_j}$, we can get $\alpha_i + \alpha_j = 2$ which means $\alpha_+ + \alpha_- = 2$.

Therefore, for all positive points $\alpha_i = \alpha_+$

for all negative points $\alpha_i = \alpha_-$

Also, $\alpha_+ + \alpha_- = 2$

By Primal feasability, $\sum_{i=1}^{n} \alpha_i y_i = 0$, Suppose there are N_+ positive points and N_- negative points, then

$$N_+ + N_- = n$$
$$\sum_{i=1}^{N_+} \alpha_+ = \sum_{i=1}^{N_-} \alpha_-$$
$$N_+ \alpha_+ = (n - N_+)\alpha_-$$
$$\alpha_+ = \frac{n - N_+}{N_+}\alpha_-$$

Plugging into the equation $\alpha_+ + \alpha_- = 2$, we get $\alpha_- = \frac{2N_+}{n}$ and $\alpha_+ = \frac{2N_-}{n}$. Note that these alphas are always positive, we can also calculate the bias. Selecting a positive point, we get:

$$f(x_j) = \lambda_0^* + \alpha_j^* y_j$$
$$1 = \lambda_0^* + \alpha_+ 1$$
$$\lambda_0^* = 1 - \alpha_+$$

Now, if we make a prediction of positive point, we get the following:

$$sign(f(x_j)) = sign(\lambda_0^* + \alpha_j^* y_j) = sign(1 - \alpha_+ + \alpha_+)$$
$$sign(f(x_j)) = sign(1) = +$$

Similarly for a negative point, we get:

$$sign(f(x_j)) = sign(1 - \alpha_+ - \alpha_-)$$
$$sign(f(x_j)) = sign(1 - (\alpha_+ + \alpha_-) = sign(-1) = -$$

This means our SVM classifier gives **0 TRAINING ERROR**.

2 Statistical Learning Theory (25 points)

(a) (7 points) Prove that the VC-dimension of affine classifiers (half spaces) in $\mathbb{R}^d$ is $d+1$.

Solution Any affine classifier can be written as $\text{sign}(w^T x + b), b \in R$. Let us first compute the lower bound of the VC-dimension of affine classifier.

We can show that any affine classifier in d dimension can shatter at least d+1 points. Suppose the value of dependant variable for d+1 points is $y = (y_1, y_2, ..., y_{d+1})$ and y_i can take either +1 or -1 value. We can define a set of d+1 points X in the following way:

$$X = \begin{bmatrix} 1 & 0 & 0 & .... & 0 \\ 1 & 1 & 0 & .... & 0 \\ 1 & 0 & 1 & .... & 0 \\ .. & .. & .. & .... & .. \\ 1 & 0 & 0 & .... & 1 \end{bmatrix}$$

This is a $d+1 \times d+1$ matrix where the first column allows for bias b. Since the determinant of X is not equal to 0, X is invertible. Therefore, we can select w s.t. $w = X^{-1}y$, this implies $sign(Xw) = y$. Therefore, we can always shatter d+1 points by selecting the points in the above mentioned way and choosing the value of w as $X^{-1}y$ for any pre-specified y.

Hence, VC-dimension $\geq d+1$

Now, lets calculate the upper bound. We can show that an affine classifier in d dimension cannot shatter more than d+1 points. Suppos, we have d+2 points in a d+1 dimensional space (consider bias a feature). Since, for a d dimension, d vectors are linearly independent. If there are more than d vectors, then they are linearly independent. Therefore, d+2 points in d+1 space are linearly dependent. This means we can write for some $x_j = \sum_{i \neq j} a_i x_i$ and not all a_i are 0.

$y_j = \text{sign}(w^T x_j) = \text{sign}(\sum_{i \neq j} a_i w^T x_i)$

We can select $y_i = \text{sign}(a_i) = \text{sign }(w^T x_i)$. If $a_i = 0$, we can select y_i to be 1 or -1. The above equation is then always positive as:

$\text{sign}(a_i) = \text{sign }(w^T x_i)$, so $\text{sign}(\sum_{i \neq j} a_i w^T x_i) > 0$. If we select y_j = -1 then, there exists an assignment of y for the d+2 points such that an affince classifier in d dimensions cannot shatter it.

Hence, VC-dimension $\leq d+1$

This means that VC-dimension is $\mathbf{d+1}$

(b) (5 points) Find the VC-d of SVM with polynomial kernel $k(x,z) = (x^T z + 1)^d, x, z \in \mathbb{R}^p$.

Solution SVM classifier are just affine functions as we can see below:

$$f(x^{new}) = \sum_j \lambda_j^* x^{new(j)} + \lambda_0^*$$

$$= \sum_i \alpha_i^* y_i < \phi(x_i), \phi(x^{new}) >_{H^k} + \lambda_0^*$$

$$= \sum_i \alpha_i^* y_i k(x_i, x^{new}) + \lambda_0^*$$

The first equation says that any SVM classifier is an affine classifier. Second equation shows that it is an affine classifier in the dimension of ϕ as well and 3rd equation is just equivalent of 2nd equation. We know that the kernel here being taken is $k(x,z) = (x^T z + 1)^d$, which is a polynomial kernel. All we need to do is calculate the dimension of the feature space and then we can use the result proved in 2(a) to calculate the VC-dimension of the classifier. It is given the dimension of x and z is p, we can rewrite the kernel as $k(x,z) = (x^T z)^d$ by adding another dimension to x and z that contains the value 1 always. So now the dimension of x and z is p+1. Note that adding a dimension which is always constant and then applying stars and bars problem allows you to get all the polynomial degrees upto d in the feature space.

Using stars and bars combinatorics methodology, we can easily calculate the dimension of the feature space. There would be $\binom{p+1+d-1}{d}$ terms in the feature space (which is the dimension of feature space). Here the stars are represented by the polynomial number we wish to achieve and the bars are represented by the features of x. Every feature can be assigned 0 or more stars and the sum of all stars in the bins/features(created by bars) must add up to the polynomial degree d.

Therefore the feature space is of dimension $\binom{p+d}{d}$, but the last term of the feature space is a constant and is in the same dimension as the bias. So we can effectively say that the feature space is in dimension $\binom{p+d}{d} - 1$, and from 2a we know that VC-dimension of affine classifiers (half spaces) in $\mathbb{R}^d$ is $d+1$. Hence, the VC-dimension is $\binom{\mathbf{p+d}}{\mathbf{d}}$

(c) (3 points) Find the VC-dimension of SVM with **rbf** kernel $k(x,z) = \exp(-\gamma \|x - z\|_2^2)$, for some fixed $\gamma \in \mathbb{R}$, where $x, z \in \mathbb{R}^p$ and prove your result.

Solution Again, we know that SVM classifiers are just affine functions. Similar to 2(b), we need to find the dimension of the feature space for the rbf kernel. In 1(f), we showed that feature space of rbf kernel is an **infinite** dimension feature space.

We know that VC-d of affine classifiers in $\mathbb{R}^d$ is $d+1$. Hence, the VC-dimension is ∞

(d) (10 points) Suppose we have a dataset $\{\mathbf{x}_i, y_i\}_{i=1}^n$ with $\mathbf{x}_i \in \mathcal{X} = \{0,1\}^p, y_i \in \mathcal{Y} = \{0,1\}$. Consider constructing a function $f : \mathcal{X} \to \mathcal{Y}$ to predict y from $\mathbf{x}$.

(i) If we set the function class $\mathcal{F}$ to be the conjunction of at most p literals, derive an upper bound of the true risk $R^{\text{true}}(f)$ for any $f \in \mathcal{F}$ at confidence level $1-\delta$. Represent your upper bound with the empirical risk $R^{\text{emp}}(f)$, δ and other quantities given in the problem. (Note: A literal of a boolean variable x is either x or $\neg x$. A conjunction of k variables is $x_1 \wedge x_2 \cdots \wedge x_k$. Assume we do not take the conjunction of positive and negative literals of the same variable, that is $x \wedge \neg x$.)

Solution Function class $\mathcal{F}$ is a conjunction of at most p literals, it is a **FINITE** class of functions. We know from Occham's Razor bound (Hoeffding + Union bound) that for a finite F, $\forall\ \delta > 0$ with probability at least 1 - δ :

$$\forall f \in \mathcal{F}, R^{true}(f) \leq R^{emp}(f) + \sqrt{\frac{logM + log\frac{1}{\delta}}{2n}}$$

The only unknown in the above equation is M, which is just the number of functions in the function class $\mathcal{F}$. This is just a Combinatorics question:

Number of functions s.t. $\mathcal{F}$ is a conjunction of 0 literals: $\mathbf{\binom{p}{0}2^0}$

Number of functions s.t. $\mathcal{F}$ is a conjunction of i literals: $\mathbf{\binom{p}{i}2^i}$

Number of functions s.t. $\mathcal{F}$ is a conjunction of p literals: $\mathbf{\binom{p}{p}2^p}$

Therefore, the total number of functions is: $\mathbf{\binom{p}{0}2^0}$ + $\mathbf{\binom{p}{1}2^1}$ +.. $\mathbf{\binom{p}{i}2^i}$+ .. + $\mathbf{\binom{p}{p}2^p}$

M $= \sum_{i=0}^{p} \binom{p}{i} 2^i = (1+2)^p = 3^p$ where this M includes a constant baseline function when no literals are included.

Therefore, the bound reduces to the following:

$$\forall f \in \mathcal{F}, R^{true}(f) \leq R^{emp}(f) + \sqrt{\frac{log3^p + log\frac{1}{\delta}}{2n}}$$

$$\forall f \in \mathcal{F}, R^{true}(f) \leq R^{emp}(f) + \sqrt{\frac{plog3 + log\frac{1}{\delta}}{2n}}$$

(ii) What would the upper bound be if we set the function class $\mathcal{F}$ to be the conjunction of at most p' literals, where $p' \leq p$.

Solution Again, it is a **FINITE** class of functions, but this time it is at most p' literals, where $p' \leq p$.

Number of functions s.t. $\mathcal{F}$ is a conjunction of 0 literals: $\mathbf{\binom{p}{0}2^0}$

Number of functions s.t. $\mathcal{F}$ is a conjunction of i literals: $\mathbf{\binom{p}{i}2^i}$

Number of functions s.t. $\mathcal{F}$ is a conjunction of p' literals: $\mathbf{\binom{p}{p'}2^{p'}}$

Therefore, the total number of functions is: $\mathbf{\binom{p}{0}2^0}$ + $\mathbf{\binom{p}{1}2^1}$ +.. $\mathbf{\binom{p}{i}2^i}$+ .. + $\mathbf{\binom{p}{p'}2^{p'}}$

M $= \sum_{i=0}^{p'} \binom{p}{i} 2^i$ and the bound reduces to:

$$\forall f \in \mathcal{F}, R^{true}(f) \leq R^{emp}(f) + \sqrt{\frac{log\sum_{i=0}^{p'} \binom{p}{i} 2^i + log\frac{1}{\delta}}{2n}}$$

3 Ridge Regression (20 points)

Given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$, suppose the data is independently and identically generated as follows:

$$y_i = \mathbf{w}^T \mathbf{x}_i + \epsilon_i, \text{ where } \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \frac{1}{\lambda}\mathbf{I}) \text{ and noise } \epsilon_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

We can stack our data in a $n \times p$ matrix $\mathbf{X}$, and our labels in a $n \times 1$ vector $\mathbf{y}$.

(a) (4 points) Please give an expression for the likelihood $\Pr(\mathbf{y}|\mathbf{X}, \mathbf{w})$.
Solution We know that $E(\epsilon_i) = 0$, also in Matrix form it is:

$$y = Xw + \epsilon$$
$$\epsilon = y - Xw$$

$$E(\epsilon) = 0 => E(y - Xw) = 0$$
$$E(y) = E(Xw)$$

Now, taking condition expectation w.r.t W and w :

$$E(y|Xw) = E(Xw|Xw)$$
$$E(y|Xw) = XwE(1)$$

since $E(Xy|X) = XE(y|X)$, also $E(1) = 1$:

$$E(y|Xw) = Xw$$

So, we have the mean of $y|Xw$, now we calculate the variance:

$$Var(\epsilon) = I$$

Now , we will use Law of Expected Iteration (LIE) which says the following:

$$Var(\epsilon) = Var(\epsilon|Xw) + E(Var(\epsilon|Xw))$$
$$Var(\epsilon|Xw) = 0$$
$$Var(\epsilon) = E(Var(\epsilon|Xw))$$

Assuming all $\epsilon_i|Xw$ have the same variance (homoskedastic and not autocorrelated)(assumptions of linear regression), then the above equation reduces to:

$$Var(\epsilon) = Var(\epsilon|Xw)$$
$$Var(y - Xw|Xw) = I$$
$$Var(y|Xw) + Var(Xw|Xw) - 2Cov(y|Xw, Xw|Xw) = I$$

$Var(X|X) = 0$, and similarly Covariance as:

$$Cov(y|Xw, Xw|Xw) = E(yXw|Xw) - E(y|Xw)E(Xw|Xw) = XwE(y|Xw) - XwE(y|Xw) = 0$$

$$Var(y|Xw) = I$$

So, $Y|Xw \sim N(Xw, I)$ as ϵ and w are normally distributed and we are doing conditional probability given X and w. The probability from Multivariate normal distribution tells us the following:

$$Pr(\mathbf{y}|\mathbf{X},\mathbf{w}) = \frac{1}{(2\pi)^{p/2}|I|^{1/2}} exp(-\frac{1}{2}(y - Xw)^T I^{-1}(y - Xw))$$
$$= \frac{1}{(2\pi)^{p/2}} exp(-\frac{1}{2}(y - Xw)^T(y - Xw))$$

(b) (8 points) Please give the Maximum A Posteriori (MAP) expression for **w**.

Hint: first show the following:

$$\Pr(\mathbf{w}|\mathbf{y},\mathbf{X},\lambda) \propto \exp\left\{-\frac{1}{2}\left[\mathbf{w}^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} - 2\mathbf{w}^T\mathbf{X}\mathbf{y}\right]\right\}$$
$$\Pr(\mathbf{w}|\mathbf{y},\mathbf{X},\lambda) \propto \exp\left\{-\frac{1}{2}(\mathbf{w} - (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y})^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})(\mathbf{w} - (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y})\right\}.$$

Solution By Bayes Rule we have:

$$Pr(w|y, X, \lambda) \propto Pr(y|X, w, \lambda)Pr(w)$$
$$posterior \propto likelihood\ .\ prior$$

We already calculated the likelihood in 3(a), we need to calculate the prior now. Given that $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \frac{1}{\lambda}\mathbf{I})$, we can calculate the probabiility :

$$Pr(\mathbf{w}) = \frac{1}{(2\pi)^{p/2}|\frac{1}{\lambda}I|^{1/2}} exp(-\frac{1}{2}(w - 0)^T(\frac{1}{\lambda}I)^{-1}(w - 0))$$
$$Pr(\mathbf{w}) = \frac{1}{(2\pi)^{p/2}|\frac{1}{\lambda}I|^{1/2}} exp(-\frac{1}{2}(w)^T\lambda I(w))$$

Plugging these values, we get the following:

$$Pr(w|y, X, \lambda) \propto \frac{1}{(2\pi)^{p/2}} exp(-\frac{1}{2}(y - Xw)^T(y - Xw))\frac{1}{(2\pi)^{p/2}|\frac{1}{\lambda}I|^{1/2}} exp(-\frac{1}{2}(w)^T\lambda I(w))$$

$$Pr(w|y, X, \lambda) \propto exp(-\frac{1}{2}(y - Xw)^T(y - Xw))exp(-\frac{1}{2}(w)^T\lambda I(w))$$

since terms outside exp do not contain any w term

$$Pr(w|y, X, \lambda) \propto exp(-\frac{1}{2}((y - Xw)^T(y - Xw) + w^T\lambda Iw))$$
$$Pr(w|y, X, \lambda) \propto exp(-\frac{1}{2}((y^T - w^TX^T)(y - Xw) + w^T\lambda Iw))$$
$$Pr(w|y, X, \lambda) \propto exp(-\frac{1}{2}(y^Ty - w^TX^Ty - y^TXw + w^TX^TXw + w^T\lambda Iw))$$

We can take out the $y^T y$ since it does not contain w, so the equation becomes:

$$Pr(w|y, X, \lambda) \propto exp(-\frac{1}{2}(-w^T X^T y - y^T X w + w^T X^T X w + w^T \lambda I w))$$

Note that $w^T X^T y$ and $y^T X w$ are just scalar matrix (1×1) and are transpose of each other so we can directly add them:

$$Pr(w|y, X, \lambda) \propto exp(-\frac{1}{2}(-2w^T X^T y + w^T X^T X w + w^T \lambda I w))$$

$$Pr(w|y, X, \lambda) \propto exp(-\frac{1}{2}(w^T (X^T X + \lambda I) w - 2w^T X^T y))$$

Also, applying log on both sides , we know that:

$$log\, posterior \propto log\, likelihood\; +\; log\, prior$$

Maximum A Posteriori (MAP) says the following:

$$max\, log\, posterior \propto min(-log\, likelihood\; -\; log\, prior)$$

$$max\; log\; Pr(w|y, X, \lambda) \propto min(-log\; exp(-\frac{1}{2}(w^T (X^T X + \lambda I) w - 2w^T X^T y))$$

$$\propto min(\frac{1}{2}(w^T (X^T X + \lambda I) w - 2w^T X^T y))$$

$$\propto min(w^T (X^T X + \lambda I) w - 2w^T X^T y)$$

The **w** at which the above equation minimizes :

$$\nabla(w^T (X^T X + \lambda I) w - 2w^T X^T y) = 2(X^T X + \lambda I) w - 2X^T y = 0$$

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

The above w gives the Maximum A Posteriori (MAP). Note that the second derivative is positive semi-definite as λ is positive (since variance is always positive). The second derivative is $(X^T X + \lambda I)^T$ which is the same as $(X^T X + \lambda I)$ (positive semi-definite) . $X^T X$ is always positive semi definite.

(c) (4 points) We can also obtain an estimate of **w** by minimizing the L^2 loss with L^2 regularisation.

$$\mathbf{w}^* = \underset{\mathbf{w} \in \mathbb{R}^p}{\arg\min} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2, \tag{1}$$

where λ is a hyperparameter that controls the amount of regularization.
Give the expression for **w** that minimizes the above objective function.

Solution To minimize, take the first derivative w.r.t w and set to 0:

$$\nabla(\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2) = 0$$

$$-2X^T (y - Xw) + 2\lambda I w = 0$$

$$-X^T y + X^T X w + \lambda I w = 0$$
$$(X^T X + \lambda I)w = X^T y$$
$$w^* = (X^T X + \lambda I)^{-1} X^T y$$

We also need to check whether second derivative is positive semidefinite:

$$(X^T X + \lambda I)^T$$
$$= X^T X + \lambda I$$

As shown in 3(b) this is always positive semi-definite so, the above first derivative finds the minimum. Therefore the answer is:

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

(d) (1 point) From your answer in (b) and (c) what are the dual interpretations of ridge regression?

Solution (b) gives the **Generative** interpretation of regression using the concept of priors.

(c) gives the **Frequentist** interpretation of ridge regression.

(e) (3 points) Suppose our generative model is wrong: **w** is not distributed with mean **0**. Does adding $\lambda\|\mathbf{w}\|^2$ as the regularization term introduce additional bias? If so, (i) How may you preprocess and/or modify the objective function to reduce this additional bias? Assume you know the correct mean for the generative process for **w**.

Solution YES, an additional bias is introduced if we add $\lambda\|\mathbf{w}\|^2$ as the regularization term and the mean is not 0. If we look at the prior if the mean is not 0 and is rather $\mathbf{w_0}$, then the prior changes to the following:

$$Pr(\mathbf{w}) = \frac{1}{(2\pi)^{p/2}|\frac{1}{\lambda}I|^{1/2}} exp(-\frac{1}{2}(w - w_0)^T \lambda I (w - w_0))$$

And the Posterior changes to:

$$Pr(w|y, X, \lambda) \propto exp(-\frac{1}{2}((y - Xw)^T (y - Xw) + (w - w_0)^T \lambda I (w - w_0)))$$

It essentially reduces to the following:

$$\mathbf{w}^* = \underset{\mathbf{w} \in \mathbb{R}^p}{\arg\min} \|\mathbf{y} - \mathbf{Xw}\|_2^2 + \lambda\|\mathbf{w} - \mathbf{w_0}\|_2^2, \quad (2)$$

(ii) If we do not know the correct mean, but that we know **w** is not distributed with mean **0**, why should we add any regularization?

Solution We should add regularization because we would like to reduce the variance even though the bias may increase. This is the **bias variance tradeoff** and it has been observed that low variance produced by regularization provides superior MSE performance. Moreover, we can use cross validation on λ to find an acceptable value of bias and variance. However, if we do not include regularization, then even though our model would be unbiased it would be highly susceptible to changes in data and will usually overfit data (high variance).

4 Catch Mice with Clustering (15 points)

Suppose you were given m mouse tracks, each track is of duration p, i.e. $\{\mathbf{m}_i\}_{i=1}^{m}, \mathbf{m}_i \in \mathbb{R}^{p\times 2}$. We are asked to place K mouse traps to catch some mice. A possible solution would be to optimize the placement of mouse traps to minimize the distances between trap locations and all of the locations where a mouse has been spotted so far:

$$\min_{\mathbf{c}_1,\dots,\mathbf{c}_K} \sum_{i=1}^{N} \min_{k\in\{1,\dots,K\}} \|\mathbf{x}_i - \mathbf{c}_k\|_1,$$

where $\{\mathbf{c}_k\}_{k=1}^{K}, \mathbf{c}_k \in \mathbb{R}^2$ are cluster centers representing the locations of mouse traps, and $\{\mathbf{x}_i\}_{i=1}^{N=m\times p}$, $\mathbf{x}_i \in \mathbb{R}^2$ are data points representing the locations of mice, and $\|\cdot\|_1$ is the L^1 norm, also known as the taxicab norm or Manhattan norm.

(a) (1 point) How is the above objective function different from k-means' objective taught in class?

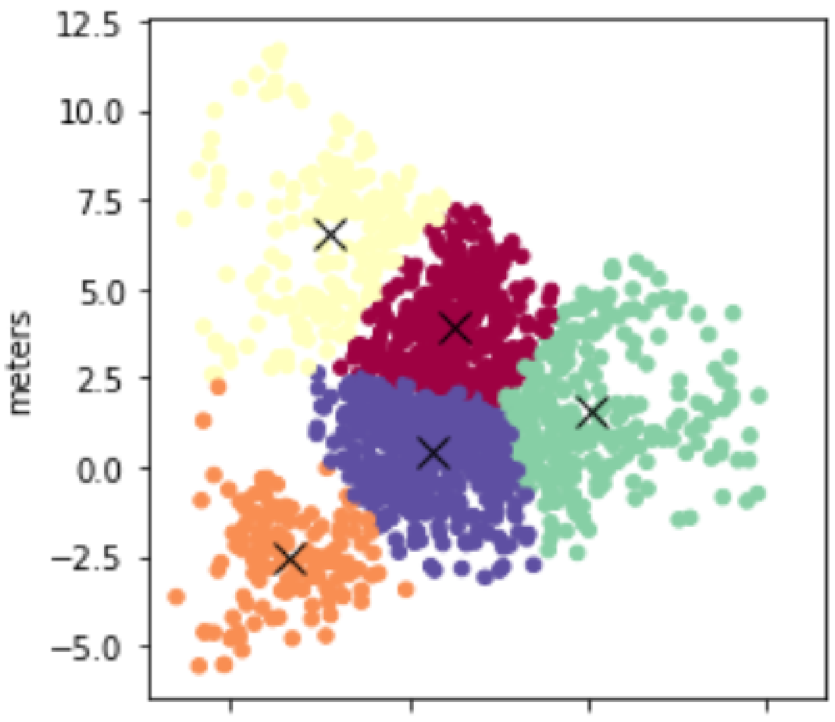
Solution The above objective function minimzes L^1 norm distance instead of L^2 norm distance. Also, the cluster update at every iteration is the median of the points in the cluster rather than mean.

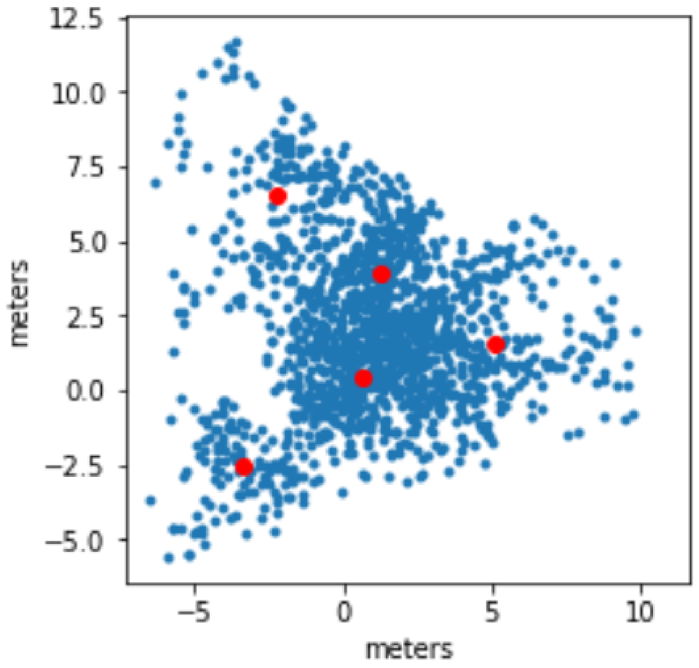
(b) (6 points) Implement k-means taught in class on the mice location data, sample .pynb code to load the data is provided.

Solution

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio
import random as random
import pandas as pd
from matplotlib import cm

tmp = sio.loadmat("mousetracks.mat")
tracks = {}
for trackno in range(30):
    tracks[trackno] = tmp["num%d"%(trackno)]

plt.close("all")
for trackno in range(30):
    plt.plot(tracks[(trackno)][:,0],tracks[(trackno)][:,1],'.')
plt.axis("square")
plt.xlabel("meters")
plt.ylabel("meters")
plt.show()

X = np.zeros([30*50,2])

for trackno in range(30):
    X[(trackno*50):((trackno+1)*50),:] = tracks[trackno]

def kmeans(X,K=5,maxiter=100):
    maxval=np.amax(X,axis=0)
    minval=np.amin(X,axis=0)
    df=pd.DataFrame(X)
    df['Center']=0
    # initialize cluster centers
    random.seed(10)
    C =[[random.uniform(minval[j],maxval[j]) for j in range(2)] for i in range(5)]
    for iter in range(maxiter):
        for i in range(1500):
            mindist= np.linalg.norm(X[i]-C[0])
            cent=1
            for j in range(1,5):
                if np.linalg.norm(X[i]-C[j])<mindist :
                    mindist= np.linalg.norm(X[i]-C[j])
                    cent= j+1
            df.iloc[i,2]= cent
        temp=df.groupby('Center')[[0,1]].mean()
        for k in range(K):
            C[k][0]=temp.iloc[k][0]
            C[k][1]=temp.iloc[k][1]
    plt.close("all")
    df.columns=['x','y','Center']
    df.plot.scatter(x='x', y='y', c='Center',colormap=cm.get_cmap('Spectral'))
    C=np.asarray(C)
    plt.plot(C[:,0],C[:,1],'x',color='black',markersize=10)
    plt.axis("square")
    plt.xlabel("meters")
    plt.ylabel("meters")
    plt.show()
    return C

C=kmeans(X)
plt.close("all")
plt.plot(X[:,0],X[:,1],'.')
#uncomment to plot your cluster centers
plt.plot(C[:,0],C[:,1],'ro')
plt.axis("square")
plt.xlabel("meters")
plt.ylabel("meters")
plt.show()
print(C)


def kmedians(X,K=5,maxiter=100):
    maxval=np.amax(X,axis=0)
    minval=np.amin(X,axis=0)
    df=pd.DataFrame(X)
    df['Center']=0
    # initialize cluster centers
    random.seed(10)
    C =[[random.uniform(minval[j],maxval[j]) for j in range(2)] for i in range(5)]
    for iter in range(maxiter):
        for i in range(1500):
            mindist= np.linalg.norm(X[i]-C[0],ord=1)
            cent=1
            for j in range(1,5):
                if np.linalg.norm(X[i]-C[j],ord=1)<mindist :
                    mindist= np.linalg.norm(X[i]-C[j],ord=1)
                    cent= j+1
            df.iloc[i,2]= cent
        temp=df.groupby('Center')[[0,1]].median()
        for k in range(K):
            C[k][0]=temp.iloc[k][0]
            C[k][1]=temp.iloc[k][1]
    plt.close("all")
    df.columns=['x','y','Center']
    df.plot.scatter(x='x', y='y', c='Center',colormap=cm.get_cmap('Spectral'))
    C=np.asarray(C)
    plt.plot(C[:,0],C[:,1],'x',color='black',markersize=10)
    plt.axis("square")
    plt.xlabel("meters")
    plt.ylabel("meters")
    plt.show()
    return C

C=kmedians(X)
plt.close("all")
plt.plot(X[:,0],X[:,1],'.')
#uncomment to plot your cluster centers
plt.plot(C[:,0],C[:,1],'ro')
plt.axis("square")
plt.xlabel("meters")
plt.ylabel("meters")
plt.show()
print(C)
```

The above code shows the following cluster and points assignments where the last figure shows the value of the cluster centers:





```
array([[ 1.27134993,  3.92589779],
       [-3.40256633, -2.52823357],
       [-2.20301519,  6.55295542],
       [ 5.0865041 ,  1.5759716 ],
       [ 0.64227373,  0.41183173]])
```

(c) (8 points) Using a similar derivation to k-means taught in class, derive the cluster assignment and cluster update steps that minimize the above objective function. Then implement this algorithm on the mouse location data. This algorithm is also known as k-medians.

Solution The above code in 4(b) implements the k-medians algorithm as well. The derivation for k-medians is shown below:

Suppose $c_1, c_2, ...c_k$ are the centers of the clusters and $x_1, x_2...., x_n$ are the data points, the cost can be defined as:

$$cost(c_1, ..., c_k) = \sum_i min_k(dist(x_i, c_k))$$

$$= \sum_k \sum_{i:x_i\, in\, cluster_k} dist(x_i, c_k)$$

$$cost(cluster_1, cluster_2, .., cluster_k, c_1, .., c_k) = \sum_k \sum_{i:x_i\, in\, cluster_k} dist(x_i, c_k)$$

Now, our **Input** is : Number of clusters K, randomly initialize centers c_k

Cluster assignment Step: Assign each point to the closest cluster center where distance is calculated according to the L1 norm:

$$min_{cluster_1,cluster_2,..,cluster_k} cost(cluster_1, cluster_2, .., cluster_k, c_1, .., c_k)$$

$$\sum_{i=1}^{N} \min_{k\in\{1,\dots,K\}} \|\mathbf{x}_i - \mathbf{c}_k\|_1,$$

Cluster Update Step: Change each cluster center to be the median of points (which can be shown by taking the derivative of the distance (L1 norm) of the points w.r.t clusters:

$$min_{c_1,c_2,..,c_k} cost(cluster_1, cluster_2, .., cluster_k, c_1, .., c_k)$$

We can show median of the points in cluster minimizes the L1 norm as follows:

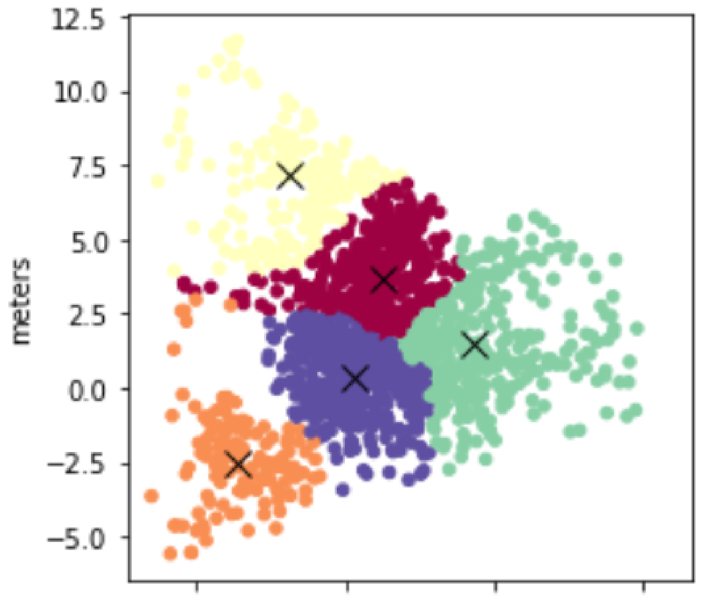
$$min_{c_k} \sum_{i:x_i\, in\, cluster_k} \|\mathbf{x}_i - \mathbf{c}_k\|_1$$

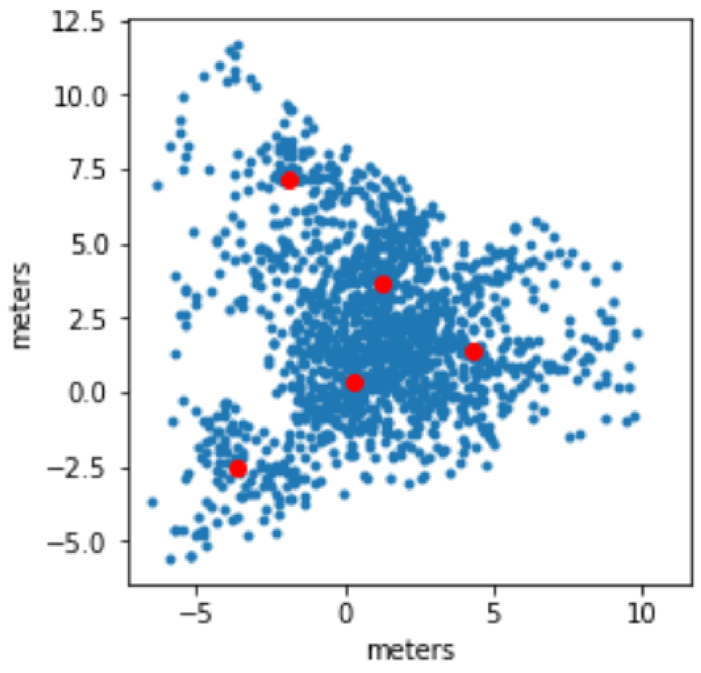
We will use the property that $\frac{d|x|}{dx} = sign(x)$, by using this property and differentiating the above equation w.r.t c_k we get:

$$\sum_{i:x_i\, in\, cluster_k} sign(x_i - c_k) = 0$$

This sum is equal to zero when the number of positive signs = number of negative signs which happens when c_k is the median of $x_i\, in\, cluster_k$. Therefore, cluster assignment is based on L1 norm distance and the cluster update step is based on the median of the points in the respective cluster.

The code for k-medians results in the following clusters and point assignments where the last figure shows the value of the cluster centers:





```
array([[ 1.22015006,  3.68835498],
       [-3.65998372, -2.51961279],
       [-1.90741962,  7.13354474],
       [ 4.29970085,  1.43838331],
       [ 0.32633366,  0.37123597]])
```