1 More Regression & Classification (Samy)

1.1 Optimal Classification & Regression

1. Let $g$ be the Bayes Classifier and $f$ be any other rule. Since $R(f) - R(g) = E[P(Y \neq f(X)|X) - P(Y \neq g(X)|X)]$, it is sufficient to show that $R = P(Y \neq f(X)) - P(Y \neq g(X)) \geq 0$.

$$R = f(X)P(Y \neq 1|X) + (1 - f(X))P(Y = 1|X) - (g(X)P(Y \neq 1|X) + (1 - g(X))P(Y = 1|X))$$

$$= 2(E[Y|X] - 1/2)(g(X) - f(X)) \geq 0$$

The last step follows by noting that $E[Y|X] = P(Y = 1|X)$. The inequality follows by noting that $g(X) \geq 0$ if $E[Y|X] \geq 1/2$.

2. Let $g(X) = E[Y|X]$ and $f : \mathcal{X} \rightarrow \mathcal{Y}$ be any other rule.

$$E_{XY}[(f(X) - Y)^2] = E_{XY}[(f(X) - g(X))^2 + (g(X) - Y)^2 + 2(f(X) - g(X))(g(X) - Y)]$$

$$= E_{X}[(f(X) - g(X))^2] + E_{XY}[(g(X) - Y)^2] + 2E_{X}[(f(X) - g(X))E_{Y}[(g(X) - Y)|X]]$$

$$\geq E_{XY}[(g(X) - Y)^2]$$

The last step follows by noting that $E_{Y}[(g(X) - Y)|X] = g(X) - E_{Y}[Y|X] = 0$ and that $E_{X}[(f(X) - g(X))^2] \geq 0$.

1.2 Support Vector Regression

1. By introducing slack variables $s_i$ s.t. $s_i = |y_i - f(x_i)| - \epsilon$ if $|y_i - f(x_i)| > \epsilon$ and 0 otherwise, we have the following problem

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|w\|^2 + c \sum_{i=1}^{n} s_i \\
\text{subject to} & \quad s_i \geq 0 \quad i = 1, \ldots, n \\
& \quad |y_i - x_i^\top w| \leq s_i + \epsilon \quad i = 1, \ldots, n
\end{align*}$$

By writing $s \in \mathbb{R}^n$ and denoting $\geq, \leq$ to denote elementwise inequalities we obtain the following quadratic program.

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} w^\top w + c1^\top s \\
\text{subject to} & \quad s \geq 0 \\
& \quad -s - c1 \leq y - Xw \leq s + c1
\end{align*}$$

In the Lagrangian, we use $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}^n$ to denote the dual variables for the three inequality constraints above.

$$\mathcal{L}(w, s, \lambda_1, \lambda_2, \lambda_3) = \frac{1}{2} w^\top w + c1^\top s - \lambda_1^\top s + \lambda_2^\top (y - Xw - s - c1) + \lambda_3^\top (-s - c1 - y + Xw)$$

$$= \frac{1}{2} w^\top w + [X^\top (\lambda_3 - \lambda_2)]^\top w + (c1 - \lambda_1 - \lambda_2 - \lambda_3)^\top s + (\lambda_2 - \lambda_3)^\top y - \epsilon(\lambda_2 + \lambda_3)^\top 1$$
We may derive the dual via the KKT Conditions. First compute the derivatives w.r.t to the primal variables.

\[ \nabla_w \mathcal{L} = w + X^\top (\lambda_3 - \lambda_2) := 0 \]
\[ \nabla_s \mathcal{L} = c1 - \lambda_1 - \lambda_2 - \lambda_3 := 0 \]

Write \( \mu = \lambda_2 - \lambda_3 \). By setting the above to 0 and observing dual feasibility gives us,

\[ \lambda_1 \geq 0 \implies \lambda_2 + \lambda_3 \leq c1 \implies 2\lambda_2 - \mu \leq c1 \]
\[ \lambda_3 \geq 0 \implies \lambda_2 \geq \mu \]

Accordingly, we have the following dual QP

\[ \begin{align*}
\text{maximize} & \quad -\frac{1}{2} \mu^\top XX^\top \mu + (y + \epsilon 1)^\top \mu - 2\epsilon \lambda_2^\top 1 \\
\text{subject to} & \quad 2\lambda_2 \leq \mu + c1 \\
& \quad \lambda_2 \geq \mu \\
& \quad \lambda_2 \geq 0
\end{align*} \]

2. To Kernelize the algorithm, we replace \( XX^\top \) via a kernel matrix \( K = (k(x_i, x_j))_{ij} \in \mathbb{R}^{n \times n} \) and solve,

\[ \begin{align*}
\text{maximize} & \quad -\frac{1}{2} \mu^\top K \mu + (y + \epsilon 1)^\top \mu - 2\epsilon \lambda_2^\top 1 \\
\text{subject to} & \quad 2\lambda_2 \leq \mu + c1 \\
& \quad \lambda_2 \geq \mu \\
& \quad \lambda_2 \geq 0
\end{align*} \]

The prediction at a new point \( x_* \) is \( \hat{f}(x_*) = \sum_{i=1}^{n} \mu_i k(x_i, x_*) \).

To see this, the prediction from the primal solution \( w \) is, \( \hat{f}(x_*) = w^\top x_* \). If we solve the dual problem, then \( \hat{f}(x) = \mu^\top X x_* \). In the kernelized version, denote the mapping of \( x_* \) by \( \phi(x_*) \) and the mapping of the training data by \( \Phi \). Then \( \hat{f}(x_*) = \mu^\top \Phi \phi(x_*) \) which can be computed using just the inner products as \( \hat{f}(x_*) = \sum_{i=1}^{n} \mu_i k(x_i, x_*) \)

3. In classification, the support vectors are those points whose inequality constraints are active and are used in computing the prediction. Here, similarly they are the points for which \( s_i > 0 \) in the primal problem and hence by complementary slackness \( \lambda_{1i} = 0 \iff 2\lambda_2i = \mu_i + c \) in the dual problem. Geometrically, these are the points that lie outside an \( \epsilon \)-tube of the estimated function.

4. This is our implementation.

```matlab
function [params, svs] = dualSVMRegression(K, y, c, eps)

N = size(y, 1);

cvx_begin
    variables muu(N) lambda2(N)
    maximize( -0.5* muu' * K * muu + (y + eps)'*muu - 2*eps * sum(lambda2) );
    subject to
        2*lambda2 <= muu + c;
        lambda2 >= muu;
        lambda2 >= 0;
    cvx_end

    params.muu = muu;
```
params.lambda2 = lambda2;
svs = abs(2*lambda2 - muu -c) < 1e-5;
end

function K = rbfKernel(X, Y, h)
    D = dist2(X, Y);
    K = exp(-D/(2*h^2));
end

function preds = dualSVMPredict(params, k)
    preds = params.muu' * k;
end

2 Expectation Maximization (Samy)

2.1 EM Basics

The solutions are straightforward. Please see the slides.

2.2 Pólya Mixture Model

2.2.1 Model

Conditioned on $z_i = k, m_i$ the distribution of $x$ corresponds to a Dirichlet Multinomial with parameters $m_i, \alpha_k$. Its mass function and the logarithm is

$$p_{dm}(x_i; \alpha_k) = \frac{\Gamma(A_k)}{\Gamma(m_i + A_k)} \prod_{s=1}^{V} \frac{\Gamma(x_i^{(s)} + \alpha_k^{(s)})}{\Gamma(\alpha_k^{(s)})}$$

$$\log p_{dm}(x_i; \alpha_k) = \log \Gamma(A_k) - \log \Gamma(m_i + A_k) + \sum_{s=1}^{V} \left( \log \Gamma(x_i^{(s)} + \alpha_k^{(s)}) - \log \Gamma(\alpha_k^{(s)}) \right)$$

where $A_k = \sum_{s=1}^{V} \alpha_k^{(s)}$ and $m_i = \sum_{s=1}^{V} x_i^{(s)}$. 
Let $\Theta = \theta, \alpha_1, \ldots, \alpha_K$. The likelihood and log likelihood of the data $D = (x_i)_{i=1}^n$ is given by,

$$p(D|\Theta) = \prod_{i=1}^n \sum_{k=1}^K p(x_i, z_k; \Theta)$$

$$\ell(\Theta|D) = \sum_{i=1}^n \log \sum_{k=1}^K p(x_i, z_k; \Theta)$$

where, $p(x_i, z_k; \Theta) = p(z_k; \Theta)p(x_i|z_k; \Theta) = \theta(k)p_d(x_i; \alpha_k)$

Accordingly, the joint probability of the parameters and the data and its log are given by,

$$p(D, \Theta) = p(\theta) \prod_{k=1}^K p(\alpha_k) p(D|\Theta)$$

$$\ell(\Theta, D) = \sum_{k=1}^K (\theta_0(k) - 1) \log \theta(k) + \sum_{k=1}^K -\lambda \|\alpha_k\|^2 + \ell(\Theta|D) + C(\theta_0, \lambda)$$

The log joint probability can be bounded via $\ell(\Theta, D) \geq \ell_b(\Theta; D)$ where,

$$\ell_b(\Theta; D) = \sum_{k=1}^K (\theta_0(k) - 1) \log \theta(k) - \lambda \sum_{k=1}^K \|\alpha_k\|^2 + \sum_{i=1}^n \sum_{k=1}^K R(z_k|x_i) \log \left( \frac{p(x_i, z_k; \Theta)}{R(z_k|x_i)} \right) + C(\theta_0, \lambda)$$

Here $R(z_k|x_i)$ is any distribution on the $z_k$'s. In E step we set it to $R(z_k|x_i) = p(z_k|x_i; \Theta)$ where the class posterior probabilities are computed using the current estimates for $\Theta$. Let $S_k = \sum_{i=1}^n R(z_k|x_i)$.

In the M-step we maximize the above w.r.t $\Theta$. We can write $\ell_b(\Theta; D) = \ell_0(\theta) + \sum_{k=1}^K \ell_k(\alpha_k) + C_1$ where $C_1$ is a constant that does not affect the optimization and

$$\ell_0(\theta) = \sum_{k=1}^K (\theta_0(k) - 1 + S_k) \log \theta(k)$$

$$\ell_k(\alpha_k) = -\lambda \|\alpha_k\|^2 + \sum_{i=1}^n R(z_k|x_i) \left( \log \Gamma(A_k) - \log \Gamma(m_i + A_k) + \sum_{s=1}^V \log \Gamma(x_i^{(s)} + \alpha_k^{(s)}) - \log \Gamma(x_i^{(s)}) \right)$$

$$= -\lambda \|\alpha_k\|^2 + S_k \log \Gamma(A_k) - \sum_{i=1}^n R(z_k|x_i) \log \Gamma(m_i + A_k) + \sum_{i=1}^n R(z_k|x_i) \sum_{s=1}^V \log \Gamma(x_i^{(s)} + \alpha_k^{(s)}) - S_k \sum_{s=1}^V \log \Gamma(\alpha_k^{(s)})$$

To optimize w.r.t $\theta$ we write out the Lagrangian and obtain the derivatives. The maximum can be found to be,

$$\hat{\theta}(k) := \frac{S_k + \theta_0(k) - 1}{n + \sum_j \theta_0(j) - K}$$

To maximize w.r.t $\alpha_k$ we use a Newton scheme as before. The first and second derivatives of $\ell_k$ are,

$$\frac{\partial \ell_k}{\partial \alpha_k^{(s)}} = -2\lambda \alpha_k^{(s)} + S_k \Psi'(A_k) - \sum_{i=1}^n R(z_k|x_i) \Psi'(m_i + A_k) + \sum_{i=1}^n R(z_k|x_i) \Psi'(x_i^{(s)} + \alpha_k^{(s)}) - S_k \Psi'(\alpha_k^{(s)})$$

$$\frac{\partial^2 \ell_k}{\partial \alpha_k^{(s)}} = -2\lambda + S_k \Psi''(A_k) - \sum_{i=1}^n R(z_k|x_i) \Psi''(m_i + A_k) + \sum_{i=1}^n R(z_k|x_i) \Psi''(x_i^{(s)} + \alpha_k^{(s)}) - S_k \Psi''(\alpha_k^{(s)})$$

$$\frac{\partial^2 \ell_k}{\partial \alpha_k^{(s)} \partial \alpha_k^{(t)}} = S_k \Psi''(A_k) - \sum_{i=1}^n R(z_k|x_i) \Psi''(m_i + A_k)$$
The Newton step update is given by $\alpha_{k}^{new} \leftarrow \alpha_{k}^{old} - H_{k}^{-1}g_{k}$ where $g_{k}^{(s)} = \frac{\partial f_{k}}{\partial \alpha_{k}^{(s)}}$ and $H_{k}^{(s,t)} = \frac{\partial^{2} f_{k}}{\partial \alpha_{k}^{(s)} \partial \alpha_{k}^{(t)}}$. As before, we can use the Sherman Morrison formula to compute the Newton step in $\tilde{O}(V)$ time. We can write $H_{k} = D_{k} + z_{11}^{\top}$ where, 

$$D_{k}^{(s,s)} = -2\lambda + \sum_{i=1}^{n} R(z_{k}|x_{i})\Psi'(x_{i}^{(s)} + \alpha_{k}^{(s)}) - S_{k}\Psi'(\alpha_{k}^{(s)})$$

and then, $[H_{k}^{-1}g_{k}]^{(i)} = g_{k}^{(i)}/D^{(i,i)} - \sum_{j}g_{k}^{(j)/D^{(j,j)}}(1/D^{(i,i)})$.

To summarize, our learning algorithm is as follows:

- Initialize: $t = 0$, Set $\theta[0], \alpha_{k}[0]$ to reasonable values.
- Repeat until convergence:
  - $t = t + 1$
  - E-step
    - Compute $R(z_{k}|x_{i}) = p(z_{k}|x_{i}; \Theta[t]) = \frac{\theta[t]^{(k)} p_{dm}(x_{i}; \alpha_{k}[t])}{\sum_{j=1}^{K} \theta[t]^{(j)} / p_{dm}(x_{i}; \alpha_{j}[t])}$
    - Compute $S_{k} = \sum_{i=1}^{n} R(z_{k}|x_{i})$.
  - M-step
    - Set $\hat{\theta}^{(k)} := \frac{S_{k} + \theta_{0}^{(k)} - 1}{n + \sum_{j} \theta_{0}^{(j)} - K}$
    - Maximize w.r.t the $\alpha_{k}$’s as outlined above.

Finally, to obtain the prediction at a new point we choose the class that maximizes the posterior $p(z|x) = \frac{p(x|z)p(z)}{p(x)}$,

$$z_{*} = \arg\max_{k \in \{1,\ldots,K\}} p(z = k|x_{*}) = \arg\max_{k \in \{1,\ldots,K\}} p(x_{*}|z = k)p(z = k) = \arg\max_{k \in \{1,\ldots,K\}} p_{dm}(x_{*}; \alpha_{k})\theta^{(k)}$$

### 2.2.2 Experiment

Our Implementation is as follows,

```matlab
function [theta, alpha] = trainPMM(X, K, theta0, lambda, thetaInit, alphaInit)
% X is an nxV matrix, y is an nx1 vector
% This function returns
% theta: a Kx1 vector indicating the class probabilities
% alpha: a KxV matrix
%
% Prelims
V = size(X, 2);
numData = size(X, 1);
numEMIters = 10;
```


% Perform EM
theta = thetaInit;
alpha = alphaInit;
for emIter = 1:numEMIters
    fprintf('EM Iter: %d\n', emIter);
    [theta, alpha] = emPMM(X, K, theta0, lambda, theta, alpha);
end

% This function performs EM
function [theta, alpha] = emPMM(X, K, theta0, lambda, thetaPrev, alphaPrev)

    % prelims
    n = size(X, 1);
    V = size(X, 2);

    % E-step
    % First obtain the class log likelihoods
    classLogLs = zeros(n, K);
    for k = 1:K
        classLogLs(:, k) = classLogLikelihoods(X, alphaPrev(k, :));
    end
    % Add the prior to obtain the joint
    classLogJoints = bsxfun(@plus, classLogLs, log(thetaPrev'));
    shiftClassLogJoints = ...
    bsxfun(@minus, classLogJoints, max(classLogJoints, [], 2));
    shiftLogJoints = exp(shiftClassLogJoints);
    R = bsxfun(@rdivide, shiftLogJoints, sum(shiftLogJoints, 2));
    S = sum(R);

    % M-step
    % First theta
    theta = theta0 + S' -1;
    theta = theta / sum(theta);
    % Then alpha
    alpha = zeros(K, V);
    for k = 1:K
        % Iterate through each class and obtain the alpha_k's
        alpha(k, :) = newtonRaphsonPMM(X, R(:,k), S(k), lambda);
    end
end

% This function implements Newton's Method.
function [alphak] = newtonRaphsonPMM(X, Rk, Sk, lambda)
numNRIters = 10; % Just use 10 iterations of NR
V = size(X, 2); % size of vocabulary
n = size(X, 1); % number of training data in this class
m = sum(X, 2); % number of words in each documents

% Set up initializations
initPt = sum( bsxfun(@times, X, Rk) );
initPt = initPt / sum(initPt);

nrProgress = zeros(numNRIters, 1);
alphak = initPt; % alphak in the current iteration
for nrIter = 1:numNRIters
    % Compute the following
    Ak = sum(alphak);
    XplusAlpha = bsxfun(@plus, X, alphak);
    % The gradient
    g = Sk * psi(Ak) - Rk' * psi(m + Ak) + Rk' * psi(XplusAlpha) ...
        - Sk * psi(alphak) - 2 * lambda * alphak;
    % The value z ( see solutions)
    z = Sk * psi(1, Ak) - Rk' * psi(1, m + Ak);
    % The diagonal of the Hessian
    D = Rk' * psi(1, XplusAlpha) - Sk * psi(1, alphak) - 2*lambda;
    % Newton's step update
    Hinv = g./D - (1./D) * sum(g./D) / (1/z + sum(1./D));
    alphak = alphak - 1*Hinv;

    % DEBUG
    nrProgress(nrIter) = Rk' * classLogLikelihoods(X, alphak);
end

function logP = classLogJointProb(X, alphak, lambda)
% Computes the log joint probability for one class (ignoring the constants).
logL = classLogLikelihoods(X, alphak);
logP = sum(logL) - lambda * norm(alphak)^2;
end

function logL = classLogLikelihoods(X, alphak)
% X is an nxV matrix, alphak is the class Dirichlet parameter. logL is a nx1
% vector with the log likelihood of each point

Ak = sum(alphak);
V = size(X, 2); % size of vocabulary
n = size(X, 1); % number of training data in this class
m = sum(X, 2); % number of words in each documents
XplusAlpha = bsxfun(@plus, X, alphak);
% Compute the log likelihood
logL = gammaln(Ak) - gammaln(m + Ak) + ...
    sum(gammaln(XplusAlpha), 2) - sum( gammaln(alphak) );

end

function [preds, classLogJoints] = predictPMM(X, theta, alpha)
% X is an nxV matrix. theta, alpha are the learned parameters.
% preds (nx1) is the predictions for X
% post (nxK) is the posterior for each class

% prelims
n = size(X, 1);
V = size(X, 2);
K = numel(theta);

% First obtain the class log likelihoods
classLogLs = zeros(n, K);
for k = 1:K
    classLogLs(:, k) = classLogLikelihoods(X, alpha(k, :));
end
% Add the prior to obtain the joint
classLogJoints = bsxfun(@plus, classLogLs, log(theta'));

% Finally obtain the predictions
[~, preds] = max(classLogJoints, [], 2);

end

3  Kernels and RKHS (Veeru)

3.1  Image similarity functions

1. Let $d$ denote the number of possible $16 \times 16$ pixel patches. As each pixel can take 256 values, $d = 256^{16 \times 16}$. Define feature map $\phi$ from the space of arbitrary rectangular pictures to $\{0, 1\}^d$ by setting 1 in a position if the corresponding patch is present in the picture, 0 otherwise. It is easy to see that $k_1(x, x') = \langle \phi(x), \phi(x') \rangle$ for this $\phi$.

2. Let $A, B$ denote a two patches with all 0’s and all 1’s respectively. Let $x_1, x_2, x_3$ be three pictures with $x_1 = A, x_2 = B, x_3 = [AB]$ ($A, B$ horizontally concatenated. Then the Gram matrix is $K$ is not positive semi-definite.

3.2  Positive definiteness of Gaussian Kernel

1. Let $x_1, x_2, x_n$ be arbitrary points in $\mathbb{R}^d$. Let $K_1, K_2$ be the Gram matrices of $k_1, k_2$ for these points. Then the Gram matrix of $k$ is $\alpha K_1 + \beta K_2$ which is $\succeq 0$ because $K_1, K_2 \succeq 0$ and $\alpha, \beta \geq 0$.

2. Let $K_1, K_2$ be the Gram matrices of $k_1, k_2$. Then their element-wise product $K = K_1 \circ K_2$ is the Gram matrix of $k$. Let $U, V$ be independent zero-mean Gaussian random variables with covariance matrices
Then the covariance matrix of $U \circ V$ is $K_1 \circ K_2$ as its $ij$th element is
\[
\mathbb{E}[U_iV_jU_jV_j] = \mathbb{E}[U_iU_j]\mathbb{E}[V_iV_j] = (K_1)_{ij}(K_2)_{ij}
\]
which means $K \succeq 0$.

3. Let the partial sums in the Taylor expansion of $\exp(-\delta||x||^2)$ be
\[
k_m = \sum_{i=1}^{m} \frac{k^i}{i!}, \text{ so that } \exp(k) = \lim_{m \to \infty} k_m.
\]
k_m is a valid kernel for $m \in \mathbb{N}$. Let $x_i, i \in [n]$ be $n$ arbitrary points in $\mathbb{R}^d$. Let $u \in \mathbb{R}^n$. Let $\epsilon > 0$. \(\exists m_0 \in \mathbb{N} \ni m \geq m_0 \Rightarrow |k(x_i, x_j) - k_m(x_i, x_j)| < \epsilon\). Let $m \geq m_0$ and let $K, K_m$ be the Gram matrices of $k, k_m$ respectively for $x_1, \cdots, x_n$.
\[
|u^T K u - u_m^T u| = \left| \sum_{i,j} u_iu_j(K - K_m)_{ij} \right| \leq \epsilon u^T u \Rightarrow u^T K u \geq u_m^T u - \epsilon u^T u \geq -\epsilon u^T u
\]
Thus, given any $u \in \mathbb{R}^n$, and $\epsilon > 0$, we can show $u^T K u \geq -\epsilon u^T u$, which means $u^T K u \geq 0$. This shows that $K \succeq 0$ and hence $\exp(k)$ is a positive definite kernel.

4. Let $\psi(x) = \exp(-\delta||x||^2)$. Write
\[
k(x, x') = \exp(-\delta||x - x'||^2) = \psi(x)\psi(x')\exp(\delta(x, x'))
\]
Let $k_1(x, x') = \psi(x)\psi(x')$. Then for arbitrary points $x_1, x_2, \cdots, x_n$, the Gram matrix constructed from $k_1$ would be the outer product
\[
[\psi(x_1), \cdots, \psi(x_n)][\psi(x_1), \cdots, \psi(x_n)]^T
\]
which is $\succeq 0$, and hence $k_1$ is a positive definite kernel. Now using parts 3 and 2 of this subproblem, $k$ is a positive definite kernel.

5. Let $k_1 = \exp(-k)$ and let $x, y$ be two distinct points. We will show that $k_1^2(x, y) > k_1(x, x)k_1(y, y)$ which means $k_1$ is not a positive definite kernel.
\[
k_1^2(x, y) > k_1(x, x)k_1(y, y)
\]
\[
\Leftrightarrow e^{-2k(x, y)} > e^{-k(x, x)}e^{-k(y, y)}
\]
\[
\Leftrightarrow -2k(x, y) > -k(x, x) - k(y, y)
\]
\[
\Leftrightarrow \|k(x, .) - k(y, .)\|^2 > 0
\]
which is true.

### 3.3 Checking validity by Fourier transforms

1. Define $f : \mathbb{R} \to \mathbb{R}$ by $f(x) = \exp(-\delta x^2)$. Its Fourier transform $\hat{f} : \mathbb{R} \to \mathbb{R}$ can be looked up from Wikipedia:
\[
\hat{f}(a) = \int_{\mathbb{R}} e^{2\pi ax} e^{-\delta x^2 / 2} dx = \sqrt{\pi / \delta} e^{-\pi^2 a^2 / \delta} > 0
\]
The Fourier transform of $k'$ is
\[
\hat{k}'(w) = \int_{\mathbb{R}^d} e^{2\pi iw(x, x)} e^{-\delta||x||^2} dx = \prod_{i=1}^{d} \int_{\mathbb{R}} e^{2\pi iw_i x_i} e^{-\delta x_i^2} dx_i = \prod_{i=1}^{d} \hat{f}(w_i)
\]
which is positive and so $k'$ is positive definite.
2. Define \( f : \mathbb{R} \to \mathbb{R} \) by \( f(x) = \frac{1}{1 + x^2} \). Its Fourier transform \( \hat{f} : \mathbb{R} \to \mathbb{R} \) can be looked up from Wikipedia (characteristic function of univariate Cauchy distribution):

\[
\hat{f}(a) = \int_{\mathbb{R}} e^{2\pi i ax} \frac{1}{1 + x^2} \, dx = \pi e^{-2\pi |a|} > 0
\]

The Fourier transform of \( k' \) is

\[
\hat{k}'(w) = \int_{\mathbb{R}^d} e^{2\pi i \langle w, x \rangle} \prod_{i=1}^d \frac{1}{1 + x_i^2} \, dx = \int_{\mathbb{R}^d} \prod_{i=1}^d e^{2\pi i w_i x_i} \frac{1}{1 + x_i^2} \, dx = \prod_{i=1}^d \hat{f}(w_i)
\]

which is positive and so \( k' \) is positive definite.

3.4 RKHS from the eigen functions of the kernels integral operator

Let \( f \in \mathcal{H} \). Then \( f \) can be written as \( f = \sum_{j=1}^{\infty} a_j \phi_j \), for some reals \( a_j \).

\[
\langle f, k(x, \cdot) \rangle = \left( \sum_{j=1}^{\infty} a_j \phi_j, \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i \right) = \sum_{j=1}^{\infty} a_j \lambda_j \phi(x) / \lambda_j = \sum_{j=1}^{\infty} a_j \phi(x) = f(x)
\]

3.5 Optimizing over an RKHS

Let \( f^* \) be a minimizer. By Representer theorem, there exists \( \alpha_i \in \mathbb{R}, i \in [n] \) such that \( f^* = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot) \).

Let \( K \) denote the Gram matrix for the data points \( x_i, i \in [n] \). Note that \( f^*(x_i) = (K\alpha)_i \), and \( \|f^*\|^2 = \alpha^T K \alpha \).

\( f^* \) is an optimizer for the given problem

\[
\iff \alpha \text{ minimizes } \|y - K\alpha\|^2 + \lambda \alpha^T K \alpha \\
\iff K(K\alpha - y) + \lambda K\alpha = 0 \\
\iff (K + \lambda I)(K\alpha) = Ky \\
\iff K\alpha = (K + \lambda I)^{-1} Ky
\]

Notice that \( K + \lambda I \succ 0 \) and hence invertible because \( K \succeq 0 \) and \( \lambda I \succ 0 \). For the fitted values,

\[
\hat{y} = f^*(x) = K\alpha = (K + \lambda I)^{-1} Ky,
\]

which of the form \( \hat{y} = Sy \) with \( S = (K + \lambda I)^{-1}K \).

3.6 Some computational considerations for SVM

1. \( O(m^2) \)
2. \( O(md) \)
3. \( O(n \log d) \) where \( n \) is the number of random projections used.