Homework 2, Problems 1 and 2

Name: Shashank Singh[1] [2]

10-715 Advanced Introduction to Machine Learning

Due: Wednesday October 15, 2014

1 More Regression and Classification (Samy)

1.1 Optimal Classification & Regression

1. Note that, $g(x) = 1_{\{\mathbb{P}[Y=1|X=x]\}}$, and hence, if $f(x) \neq g(x)$, then

$$\mathbb{P}_{XY}[Y \neq g(X)] \leq 1/2 \leq \mathbb{P}_{XY}[Y = g(X)] \leq \mathbb{P}_{XY}[Y \neq f(X).$$

Since clearly $\mathbb{P}_{XY}(Y \neq g(X), f(x) = g(X)) = \mathbb{P}_{XY}(Y \neq f(X), f(x) = g(X))$,

$$\begin{aligned} R(g) &= \mathbb{P}_{XY}(Y \neq g(X), f(x) = g(X)) + \mathbb{P}_{XY}(Y \neq g(X), f(x) \neq g(X)) \\ &\leq \mathbb{P}_{XY}(Y \neq f(X), f(x) = g(X)) + \mathbb{P}_{XY}(Y \neq f(X), f(x) \neq g(X)) = R(f). \quad \blacksquare \end{aligned}$$

2. For any $x \in \mathcal{X}$, since the objective is smooth and convex, for any $f : \mathcal{X} \to \mathcal{Y}$ minimizing $\mathbb{E}_Y[(Y - f(X))^2 | X = x]$, under certain regularity conditions on f and the joint distribution of (X, Y),

$$0 = \frac{d}{df(x)} \mathbb{E}_Y[(Y - f(X))^2 | X = x] = 2\mathbb{E}_Y[f(X) - Y | X = x] = 2f(x) - \mathbb{E}_Y[Y | X = x]$$

and hence $f(x) = \mathbb{E}_Y[Y|X = x]$. Hence, the Bayes estimator g minimizes the risk pointwise (i.e., for all X, and so it minimizes the expected risk. $\blacksquare$

1.2 Support Vector Regression

1. Let $\xi \in \mathbb{R}^n$ defined by $\xi_i := l_\varepsilon(w^T x_i - y_i)$. Then, the optimization problem can be written

$$\min_{w \in \mathbb{R}^m, \xi \in \mathbb{R}^n} \frac{1}{2}\|w\|_2^2 + c\sum_{i=1}^n \xi_i, \quad \text{s.t.} \quad 0 \leq \xi_i, \xi_i + \varepsilon - w^T x_i + y_i, \xi_i + \varepsilon + w^T x_i - y_i \quad \text{for} \quad i \in [n],$$

The Lagrangian is

$$L(w, \xi, \alpha, \beta, \gamma) = \frac{1}{2}\|w\|_2^2 + \sum_{i=1}^n c\xi_i - \alpha_i \xi_i - \beta_i(\xi_i + \varepsilon - w^T x_i + y_i) - \gamma_i(\xi_i + \varepsilon + w^T x_i - y_i)$$

If $\hat{w}$ and $\hat{\xi}$ minimize the Lagrangian, then

$$0 = \nabla_w L(w, \hat{\xi}, \alpha, \beta, \gamma)\big|_{w=\hat{w}} = \hat{w} + \sum_{i=1}^n (\beta_i - \gamma_i) x_i$$

[1] sss1@andrew.cmu.edu

[2] Machine Learning Department & Department of Statistics

and

$$0 = \nabla_\xi L(w, \xi, \alpha, \beta)\big|_{\xi=\hat{\xi}} = c1_n - \alpha - \beta - \gamma,$$

where 1_n denotes the n-dimensional vector of all ones vector. Hence, $\hat{w} = \sum_{i=1}^n (\gamma_i - \beta_i)x_i$ and, cancelling several terms,

$$\begin{aligned} L(\hat{w}, \hat{\xi}, \alpha, \beta) &= \frac{1}{2}\left\|\sum_{i=1}^n (\beta_i - \gamma_i)x_i\right\|_2^2 - \sum_{i=1}^n \beta_i(\varepsilon - w^T x_i + y_i) + \gamma_i(\varepsilon + w^T x_i - y_i) \\ &= -\frac{1}{2}\left\|\sum_{i=1}^n (\beta_i - \gamma_i)x_i\right\|_2^2 + \sum_{i=1}^n y_i(\gamma_i - \beta_i) - \varepsilon(\gamma_i + \beta_i) \\ &= -\frac{1}{2}(\gamma - \beta)^T G(\gamma - \beta) + y^T(\gamma - \beta) + \varepsilon(\gamma + \beta)^T 1_n, \end{aligned}$$

where G denotes the Gram matrix (so $G_{i,j} = x_i^T x_j$). The dual problem is then

$$\max_{0 \le \beta_i, \gamma_i \le c} -\frac{1}{2}(\gamma - \beta)^T G(\gamma - \beta) + y^T(\gamma - \beta) + \varepsilon(\gamma + \beta)^T 1_n.$$

We can write this as a quadratic program as

$$\min_{\theta \in [0,c]^{2n}} \frac{1}{2}\theta^T Q\theta + z^T\theta.$$

where $z = (\varepsilon - y_1, \ldots, \varepsilon - y_n, \varepsilon + y_1, \ldots, \varepsilon + y_n)$ and $Q = \begin{bmatrix} G & -G \\ -G & G \end{bmatrix}$.

2. Note that the quadratic program derived in the previous part depends on $x_1, \ldots, x_n$ only through the Gram matrix G. Hence, we can replace G with a kernel matrix of our choice to kernelize the algorithm. After solving the quadratic program for β and γ, we can compute the prediction for a new input x_* as $\hat{f}(x_*) = \sum_{i=1}^n (\gamma_i - \beta_i)k(x_*, x_i)$.

3. Support vectors are those x_i for which $\gamma_i - \beta_i \neq 0$.

4. See Figure 1.

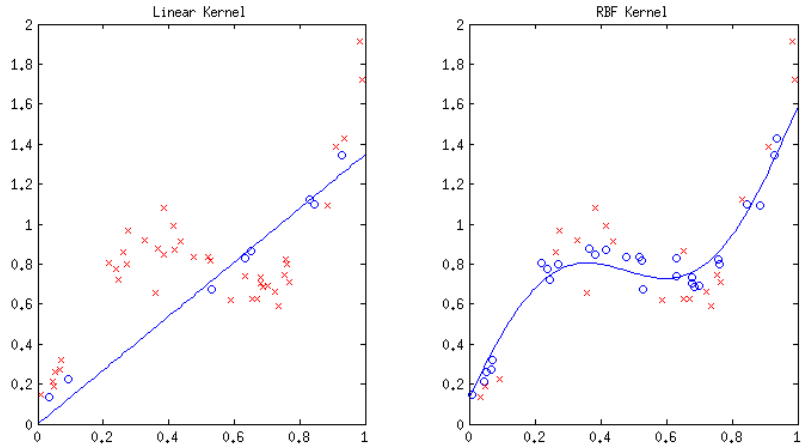

Figure 1: $\hat{f}$ as predicted by support vector regression using linear and RBF kernels. Support vectors are indicated by crosses.

Code

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

  n = length(y);
  z = [eps - y', eps + y'];
  Q = [K -K; -K K];

  theta = quadprog(Q, z, [], [], [], [], zeros(2*n,1), c*ones(2*n,1));

  params.beta = theta(1:n);
  params.gamma = theta((n + 1):end);
  svs = abs(params.beta - params.gamma) > 0.01;

end

function preds = dualSVMPredict(params, ks)
  preds = (params.beta - params.gamma)'*ks;
end

function K = rbfKernel(X1, X2, h)

  n1 = length(X1);
  n2 = length(X2);
  K = zeros(n1,n2);
  for i=1:n1
    for j=1:n2
      K(i,j) = exp(-(X1(i) - X2(j))^2/(2*h^2));
    end
  end
end
```

2 Expectation Maximization (Samy)

2.1 EM Basics

1. For any probability density q, since $P(y, D|\theta^t) = P(D|\theta^t)P(y|D, \theta^t)$,

$$\begin{aligned}\log P(D|\theta^t) &= \int q(y) \log P(D|\theta^t)\, dy = \int q(y) \log \frac{P(y, D|\theta^t) q(y)}{P(y|D, \theta^t q(y)}\, dy \\ &= \int q(y) \log P(y, D|\theta^t)\, dy + \int q(y) \log \frac{1}{q(y)}\, dy + \int q(y) \log \frac{q(y)}{P(y|D, \theta^t)}\, dy \\ &= \int q(y) \log P(y, D|\theta^t)\, dy + H(q) + D_{KL}(q(y) \| P(y|D, \theta^t)),\end{aligned}$$

   where H denotes Shannon entropy and D_{KL} denotes KL-divergence. Since H and D_{KL} are nonnegative, in the case $q(y) = P(y|D, \theta^t)$,

$$\log P(D|\theta^t) \geq \int P(y|D, \theta^t) \log P(y, D|\theta^t)\, dy = Q(\theta|\theta^t),$$

   where $Q(\theta|\theta^t)$ is the expression we maximize in the M-step. ■

2. Since $\theta^{t+1} := \text{argmax}_\theta\, Q(\theta|\theta^t)$ and $D_{KL}(p||q) \geq 0$ with equality if $p = q$,

$$\begin{aligned}\log P(D|\theta^t) &= Q(\theta^t|\theta^t) + H(P(y|D, \theta^t)) + D_{KL}(P(y|D, \theta^t) \| P(y|D, \theta^t)) \\ &\leq Q(\theta^{t+1}|\theta^t) + H(P(y|D, \theta^t)) + D_{KL}(P(y|D, \theta^{t+1}) \| P(y|D, \theta^t)) \\ &= \log P(D|\theta^{t+1}). \quad \blacksquare\end{aligned}$$

2.2 Pólya Mixture Model

2.2.1 Model

As shown in HW 1, for any labels $z_1, \ldots, z_n$ the log-joint probability is

$$\begin{aligned}&\ell(x_1, \ldots, x_n, z_1, \ldots, z_n, \theta, \alpha_1, \ldots, \alpha_K) \\ &= \sum_{j=1}^{K} (\theta_0^{(j)} - 1) \log \theta^{(j)} - \lambda \|\alpha_j\|^2 + \sum_{i=1}^{n} 1_{\{z_i = j\}} \left(\log \theta^{(j)} + \log p_{dm}(x_i; \alpha_j) \right) + C(\theta_0, \lambda),\end{aligned}$$

where C is constant in x_i's, z_i's, θ, and α_j's. I didn't have time to write out the derivation.

In the E-step, we predict $z_1, \ldots, z_n$ using the previous value of θ and $\alpha_1, \ldots, \alpha_k$. In the M-step, we update our estimates of θ and $\alpha_1, \ldots, \alpha_k$ using the labels $z_1, \ldots, z_n$. Both prediction of $z_1, \ldots, z_n$ and estimation of θ and $\alpha 1, \ldots, \alpha_K$ are performed as in Homework 1. Our prediction rule is the MAP estimate

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

where $\hat{\theta}$ and $\hat{\alpha}_1, \ldots, \hat{\alpha}_K$ are as predicted by our EM procedure.

2.2.2 Experiment

```
>> q22

theta =

    0.3003
    0.4532
    0.2465

alpha =

    0.3658    0.2736    0.3919    0.2659    0.3302
    0.2915    0.2626    0.3893    0.2735    0.3887
    0.4732    0.3347    0.2665    0.5032    0.4318

preds =

     1
     1
     3
     1
     2
Accuracy: 99.1300
```

Code

```
function [theta, alpha] = trainPMM(XTrain, K, theta0, lambda, thetaInit, alphaInit);

  theta = thetaInit;
  alpha = alphaInit;

  for iter = 1:10
    Z = predictPMM(XTrain, theta, alpha);
    [theta, alpha] = trainPDA(XTrain, Z, theta0, lambda);
  end
end

function [theta, alpha] = trainPDA(X, y, theta0, lambda)

  % Prelims
  K = numel(unique(y));
  V = size(X, 2);
  numData = size(X, 1);

  % MAP for theta
  table = tabulate(y);
  adjustedFreqs = table(:,2) + theta0 - 1;
  theta = adjustedFreqs/sum(adjustedFreqs);

  % MAP for alpha
  alpha = zeros(K, V);
  parfor k = 1:K
    Xk = X(y == k, :);
    alpha(k, :) = newtonRaphsonPDA(Xk, lambda);
  end
end

function [alpha_k] = newtonRaphsonPDA(Xk, lambda)

  % Prelims
  numNRIters = 10; % Just use 5 iterations of NR
  nk = size(Xk, 1); % number of training data in this class
  m = sum(Xk, 2); % number of words in each documents
  initPt = sum(Xk); initPt = initPt/sum(initPt); % Initialization

  % Now perform Newton's
  alpha_k = initPt; % alphak in the current iteration
  for nrIter = 1:numNRIters
    % Compute the following
```

```
    Ak = sum(alpha_k);
    XplusAlpha = bsxfun(@plus, Xk, alpha_k);
    % The gradient
    g = nk * psi(Ak) - sum(psi(m + Ak)) + sum( psi(XplusAlpha) ) ...
        - nk * psi(alpha_k) - 2 * lambda * alpha_k;
    % The value z (see solutions)
    z = nk * psi(1, Ak) - sum(psi(1, m + Ak));
    % The diagonal of the Hessian
    D = sum(psi(1, XplusAlpha)) - nk * psi(1, alpha_k) - 2*lambda;
    % Newton's step update
    Hinvg = g./D - (1./D) * sum(g./D) / (1/z + sum(1./D));
    alpha_k = alpha_k - 1*Hinvg;
  end
end

function [preds, classLogJoints] = predictPMM(X, theta, alpha)

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

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

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

end

function logL = classLogLikelihoods(X, alphak)

  % Prelims
  Ak = sum(alphak);
  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
```

Homework 2, Problems 3 and 4

Name: Shashank Singh[1] [2]

10-715 Advanced Introduction to Machine Learning

Due: Wednesday October 15, 2014

3 Kernels and RKHS (Veeru)

3.1 Image Similarity Functions

1. Consider a set I indexing the set of 16×16 pixel patches. Then, $k_1(x, x') = \sum_{i \in I} k_i$ where $k_i(x, x') = 0$ when either x or x' has no i^{th} patch or when their i^{th} patches are different, and $k_i(x, x') = 1$ otherwise. Since each k_i is trivially a positive definite kernel (specifically, a δ function), it follows from part 1 of section 3.2 below that k_1 is a positive definite kernel. ■

2. Consider three 16×32 images $A, B, C \in X$, where A is entirely black, B is black on the left half and white on the right half, and C is white on the left half and black on the right half. Then, the kernel matrix of k_2 evaluated on A, B, C is

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

   It can be checked that K has an eigenvalue of ≈ -0.41, and hence K is not positive semidefinite. ■

3.2 Positive definiteness of Gaussian Kernel

1. By Mercer's Theorem, it suffices to observe that, $\forall f \in L^2(\mathbb{R}^d)$,

$$\int_{\mathbb{R}^d} \int_{\mathbb{R}^d} (\alpha k_1 + \beta k_2)(x, y) f(x) f(y)\, dy\, dx$$
$$= \alpha \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} k_1(x, y) f(x) f(y)\, dy\, dx + \beta \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} k_2(x, y) f(x) f(y)\, dy\, dx \geq 0. \quad ■$$

2. Suppose $x_1, \ldots, x_n$ are in the domain of k_1 and k_2. Let $K_1, K_2 \in \mathbb{R}^{n \times n}$ denote the kernel matrices under k_1, k_2, respectively. Since K_1, K_2 are positive-semidefinite, they have positive-semidefinite square roots $A, B \in \mathbb{R}^{n \times n}$ with $A^2 = K_1$ and $B^2 = K_2$. Let $K := K_1 \circ K_2$ and note that, since K is the kernel under $k_1 k_2$, it suffices to show K is positive-semidefinite. Note that, for $i, j \in [n]$,

$$K_{i,j} = (K_1)_{i,j} (K_2)_{i,j} = \left(\sum_{k=1}^{n} A_{i,k} A_{j,k} \right) \left(\sum_{\ell=1}^{n} B_{i,\ell} B_{j,\ell} \right).$$

[1] sss1@andrew.cmu.edu

[2] Machine Learning Department & Department of Statistics

Thus, for any $y \in \mathbb{R}^n$,

$$\begin{aligned} y^T K y &= \sum_{i=1}^n \sum_{j=1}^n y_i y_j \left(\sum_{k=1}^n A_{i,k} A_{j,k} \right) \left(\sum_{\ell=1}^n B_{i,\ell} B_{j,\ell} \right) \\ &= \sum_{k=1}^n \sum_{\ell=1}^n \left(\sum_{i=1}^n y_i A_{i,k} B_{i,\ell} \right) \left(\sum_{j=1}^n y_j A_{j,k} B_{j,\ell} \right) \\ &= \sum_{k=1}^n \sum_{\ell=1}^n \left(\sum_{i=1}^n y_i A_{i,k} B_{i,\ell} \right)^2 \geq 0. \quad \blacksquare \end{aligned}$$

3. For $x_1, \ldots, x_n \in \mathbb{R}^d$, $y \in \mathbb{R}^n$, Taylor-expanding the exponential function,

$$\sum_{i=1}^n \sum_{j=1}^n \exp(k(x_i, x_j)) y_i y_j = \sum_{\ell=0}^\infty \sum_{i=1}^n \sum_{j=1}^n \frac{k^\ell(x_i, x_j)}{\ell!} y_i y_j \geq 0,$$

since k^ℓ is a kernel. $\blacksquare$

4. Any real inner product is a kernel, since, $\forall x_1, \ldots, x_n$ in the appropriate domain, $y \in \mathbb{R}^n$,

$$\sum_{i=1}^n \sum_{j=1}^n \langle x_i, x_j \rangle y_i y_j = \sum_{i=1}^n \left\langle y_i x_i, \sum_{j=1}^n y_j x_j \right\rangle = \left\langle \sum_{i=1}^n y_i x_i, \sum_{j=1}^n y_j x_j \right\rangle = \left\| \sum_{i=1}^n y_i x_i \right\|_{\langle \cdot, \cdot \rangle}^2 \geq 0.$$

Hence, since the function $(x_1, x_2) \mapsto \delta x_1^T x_2$ is an inner product on $\mathbb{R}^d$, by the previous part, the function $(x_1, x_2) \mapsto \exp(2\delta x_1^T x_2)$ is a kernel. Since the function $(x_1, x_2) \mapsto \exp(-\delta(\|x_1\|_2^2 + \|x_2\|_2^2))$ is also a kernel (I didn't have time to show this), their product, the Gaussian kernel, is a positive definite kernel. $\blacksquare$

5. Let $x, y \in \mathbb{R}^d$ with $x \neq y$. Since $k(x, \cdot) \neq k(y, \cdot)$, it follows that $2k(x, y) < k(x, x) + k(y, y)$, and hence $\exp(-(k(x, x) + k(y, y))) - \exp(-2k(x, y)) < 0$. This expression is the determinant of the 2×2 kernel matrix for x, y under $\exp(-k)$, and hence this matrix is not positive-semidefinite (a positive-semidefinite matrix clearly has nonnegative determinant). $\blacksquare$

3.3 Checking validity by Fourier transforms

1. For $\omega \in \mathbb{R}$, completing the square in the argument of exp,

$$\begin{aligned} \mathcal{F}[k'](\omega) &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} e^{-i\omega x} e^{-\delta x^2} \, dx \\ &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \exp\left(-(\delta x^2 + i\omega x) \right) \, dx \\ &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \exp\left(-\left(\sqrt{\delta} x + \frac{i\omega}{2\sqrt{\delta}} \right)^2 - \omega/(4\delta) \right) \, dx \\ &= \frac{\sqrt{1/\delta} \exp(-\omega/(4\delta))}{\sqrt{2\pi/\delta}} \int_{\mathbb{R}} \exp\left(-\frac{(x + i\omega/2)^2}{2/\delta} \right) \, dx = \frac{\exp(-\omega/(4\delta))}{\sqrt{\delta}} > 0, \end{aligned}$$

where we used the fact that the integral over $\mathbb{R}$ of the pdf of $\mathcal{N}(-i\omega/2, 1/\delta)$ is 1. $\blacksquare$

2. Define $\hat{k} : \mathbb{R} \to \mathbb{R}$ by $\hat{k}(\omega) = \sqrt{\pi/2}e^{-|\omega|}$, for all $\omega \in \mathbb{R}$. Then, $\forall x \in \mathbb{R}$,

$$\begin{aligned}\mathcal{F}\left[\hat{k}\right](x) = \frac{1}{2}\int_{\mathbb{R}} e^{-(ix\omega+|\omega|)}\,d\omega &= \frac{1}{2}\left(\int_0^\infty e^{-(1+ix)\omega}\,d\omega + \int_0^\infty e^{-(1-ix)\omega}\,d\omega\right) \\ &= \frac{1}{2}\left(\frac{1}{1+ix} + \frac{1}{1-ix}\right) = \frac{1}{1+x^2} = k'(x).\end{aligned}$$

Since $\hat{k}$ is even and $\mathcal{F}^2$ is the time-reversing function, $\mathcal{F}[k'] = \mathcal{F}^2\left[\hat{k}\right] = \hat{k} > 0$. ■

3.4 RKHS from the eigenfunctions of the kernel's integral operator

Let $f \in \mathcal{H}_k, x \in X$. Since $\{\phi_i\}_{i=1}^\infty$ is a basis for $\mathcal{H}_k$, $\exists!\{f_i\}_{i=1}^\infty \in \mathbb{R}^{\mathbb{N}}$ such that $f = \sum_{i=1}^\infty f_i\phi_i$. Then,

$$f(x) = \sum_{i=1}^\infty f_i\phi_i(x) = \sum_{i=1}^\infty \frac{f_i\lambda_i\phi_i(x)}{\lambda_i} = \langle f, k(\cdot, x)\rangle. \quad ■$$

3.5 Optimizing over an RKHS

Define a subspace $\mathcal{H}_{||} \subseteq \mathcal{H}_k$ by $\mathcal{H}_{||} := \{\sum_{i=1}^n \alpha_i k(\cdot, x_i) : \alpha \in \mathbb{R}^n\}$. Since $\mathcal{H}_{||}$ is finite dimensional and hence closed, for any $f \in \mathcal{H}_k$, we can define $f_{||} \in \mathcal{H}_{||}$ as the unique projection of f onto $\mathcal{H}_{||}$, and define $f_\perp \in \mathcal{H}_k$ by $f_\perp := f - f_{||}$. Then, for each $i \in [n]$,

$$(y_i - f(x_i))^2 = (y_i - \langle f, k(\cdot, x_i)\rangle)^2 = (y_i - \langle f_{||}, k(\cdot, x_i)\rangle)^2 = (y_i - f_{||}(x_i))^2$$

and, by the Pythagorean Theorem, $\|f\|_{\mathcal{H}_k}^2 = \|f_{||}\|_{\mathcal{H}_k}^2 + \|f_\perp\|_{\mathcal{H}_k}^2$. Hence

$$\sum_{i=1}^n (y_i - f(x_i))^2 + \lambda\|f\|_{\mathcal{H}_k}^2 \geq \sum_{i=1}^n (y_i - f_{||}(x_i))^2 + \lambda\|f_{||}\|_{\mathcal{H}_k}^2,$$

and so any minimizer f_* of (2) has the form $f_* = \sum_{i=1}^n \alpha_i k(\cdot, x_i), \alpha \in \mathbb{R}^n$. So, (2) is equivalent to

$$\min_{\alpha\in\mathbb{R}^n} \sum_{i=1}^n \left(y_i - \sum_{j=1}^n \alpha_i k(x_j, x_i)\right)^2 + \lambda\left\|\sum_{i=1}^n \alpha_i k(\cdot, x_i)\right\|_{\mathcal{H}_k}.$$

By the reproducing property,

$$\left\|\sum_{i=1}^n \alpha_i k(\cdot, x_i)\right\|_{\mathcal{H}_k} = \sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j\langle k(\cdot, x_i), k(\cdot, x_j)\rangle = \sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j k(x_i, x_j) = \alpha^T K\alpha,$$

where K denotes the kernel matrix of k evaluated on $x_1, \ldots, x_n$. Also,

$$\sum_{i=1}^n \left(y_i - \sum_{j=1}^n \alpha_i k(x_j, x_i)\right)^2 = \|K\alpha - y\|_2^2.$$

Hence the problem can be rewritten as

$$\min_{\alpha \in \mathbb{R}^n} \|K\alpha - y\|_2^2 + \lambda \alpha^T K \alpha.$$

If $\hat{\alpha}$ minimizes this, then

$$0 = \nabla_\alpha \|K\alpha - y\|_2^2 + \lambda \alpha^T K \alpha\big|_{\alpha=\hat{\alpha}} = K(K\hat{\alpha} - y) + \lambda K\alpha = (K^2 + \lambda K)\hat{\alpha} - Ky.$$

As long as K is nonsingular, solving for $\hat{\alpha}$ gives and hence $\boxed{\hat{\alpha} = (K + \lambda I)^{-1} y}$ (noting that $K + \lambda I$ is nonsingular, since K has only nonnegative eigenvalues).

3.6 Some computational considerations for SVM

1. Naively implementing kernel SVM requires storing the $m \times m$ kernel matrix of $x_1, \ldots, x_m$. Hence, the space complexity is $O(m^2)$.
2. Since the decision function requires computing the kernel $k(x, x_i)$ for each support vector x_i, this takes $O(md)$ time.
3. The Fastfood approach proposed by Le et al. takes $O(m \log d)$ time and $O(n)$ storage.