EECS 545 – Machine Learning - Homework #3

Changhan (Aaron) Wang

Due: 11:00pm 02/22/2016

Homework Policy: Working in groups is fine, but each member must submit their own writeup. Please write the members of your group on your solutions. There is no strict limit to the size of the group but we may find it a bit suspicious if there are more than 4 to a team. **For coding problems, please include your code and report your results (values, plots, etc.)** in your PDF submission. You will lose points if your experimental results are only accessible through rerunning your code. Homework will be submitted via Gradescope (https://gradescope.com/).

1) **Support Vector Machine (40 points).**

Recall that maximizing the soft-margin in SVM is equivalent to the following minimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & t^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \qquad (i = 1, \ldots, N) \end{aligned} \tag{1}$$

Equivalently, we can solve the following unconstrained minimization problem:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\max\left(0, 1 - t^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b)\right) \tag{2}$$

(a) Prove that minimization problem (1) and (2) are equivalent.

Solution:
For any optimal solution ξ_i^* in (1), one of the equivalences $t^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) = 1 - \xi_i^*$ and $\xi_i^* = 0$ must hold. Otherwise, we can let $\xi_i = 1 - t^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b)$ (or $\xi_i = 0$) to get a lower objective function value, which is a contradiction. Hence it is safe to use stricter constraints $\xi_i = \max\left(0, 1 - t^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b)\right)$ and plug them into the objective function to get (2).

(b) Let $(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*)$ be the solution of minimization problem (1). Show that if $\xi_i^* > 0$, then the distance from the training data point $\mathbf{x}^{(i)}$ to the margin hyperplane $t^{(i)}((\mathbf{w}^*)^T\mathbf{x} + b^*) = 1$ is proportional to ξ_i^*.

Solution:
Using the analysis above, we have $\xi_i^* = 1 - t^{(i)}(\mathbf{w}^{*T}\mathbf{x}^{(i)} + b) > 0$. If we write $\mathbf{x}^{(i)}$ as $\tilde{\mathbf{x}} + r\mathbf{w}^*/\|\mathbf{w}^*\|_2$, where $\tilde{\mathbf{x}}$ lies in the hyperplane, we have that

$$\begin{aligned}\xi_i^* &= 1 - t^{(i)}(\mathbf{w}^{*T}\mathbf{x}^{(i)} + b) \\ &= 1 - t^{(i)}(\mathbf{w}^{*T}\tilde{\mathbf{x}} + b) - rt^{(i)}\frac{\mathbf{w}^{*T}\mathbf{w}^*}{\|\mathbf{w}^*\|_2} \\ &= -rt^{(i)}\|\mathbf{w}^*\|_2\end{aligned}$$

So the distance between $\mathbf{x}^{(i)}$ and the margin hyperplane is

$$|r| = \frac{\xi_i^*}{\|\mathbf{w}\|_2}$$

(c) The error function in minimization problem (2) is

$$E(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\max\Big(0, 1 - t^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b)\Big)$$

Find its derivatives: $\nabla_{\mathbf{w}}E(\mathbf{w}, b)$ and $\frac{\partial}{\partial b}E(\mathbf{w}, b)$. Where the derivative is undefined, use a subderivative.

Solution:

$$\nabla_{\mathbf{w}}E(\mathbf{w}, b) = \mathbf{w} - C\sum_{i=1}^{N}\mathbf{I}\Big[t^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) < 1\Big]\,t^{(i)}\mathbf{x}^{(i)}$$

$$\frac{\partial}{\partial b}E(\mathbf{w}, b) = -C\sum_{i=1}^{N}\mathbf{I}\Big[t^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) < 1\Big]\,t^{(i)}$$

(d) Implement the soft-margin SVM using batch gradient descent. Here is the pseudo code:

Algorithm 1: SVM Batch Gradient Descent

```
$\mathbf{w}^* \leftarrow \mathbf{0}$ ;
$b^* \leftarrow 0$ ;
for $j$=1 to NumIterations do
    $\mathbf{w}_{grad} \leftarrow \nabla_{\mathbf{w}}E(\mathbf{w}^*, b^*)$ ;
    $b_{grad} \leftarrow \frac{\partial}{\partial b}E(\mathbf{w}^*, b^*)$ ;
    $\mathbf{w}^* \leftarrow \mathbf{w}^* - \alpha(j)\ \mathbf{w}_{grad}$ ;
    $b^* \leftarrow b^* - \alpha(j)\ b_{grad}$ ;
end
return $\mathbf{w}^*$
```

The learning rate for the j-th iteration is defined as:

$$\alpha(j) = \frac{\eta_0}{1 + j \cdot \eta_0}$$

Set η_0 to 0.001 and the slack cost C to 3. Show the iteration-versus-accuracy (training accuracy) plot. The training and test data/labels are provided in `digits_training_data.csv`, `digits_training_labels.csv`, `digits_test_data.csv` and `digits_test_labels.csv`.

Solution:
(Refer to (f).)

(e) Let

$$E^{(i)}(\mathbf{w}, b) = \frac{1}{2N}\|\mathbf{w}\|^2 + C\max\left(0, 1 - t^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b)\right)$$

then

$$E(\mathbf{w}, b) = \sum_{i=1}^{N} E^{(i)}(\mathbf{w}, b)$$

Find the derivatives $\nabla_{\mathbf{w}} E^{(i)}(\mathbf{w}, b)$ and $\frac{\partial}{\partial b} E^{(i)}(\mathbf{w}, b)$. Once again, use a subderivative if the derivative is undefined.

Solution:

$$\nabla_{\mathbf{w}} E^{(i)}(\mathbf{w}, b) = \frac{1}{N}\mathbf{w} - C\mathbf{I}\left[t^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) < 1\right] t^{(i)}\mathbf{x}^{(i)}$$
$$\frac{\partial}{\partial b} E^{(i)}(\mathbf{w}, b) = -C\mathbf{I}\left[t^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) < 1\right] t^{(i)}$$

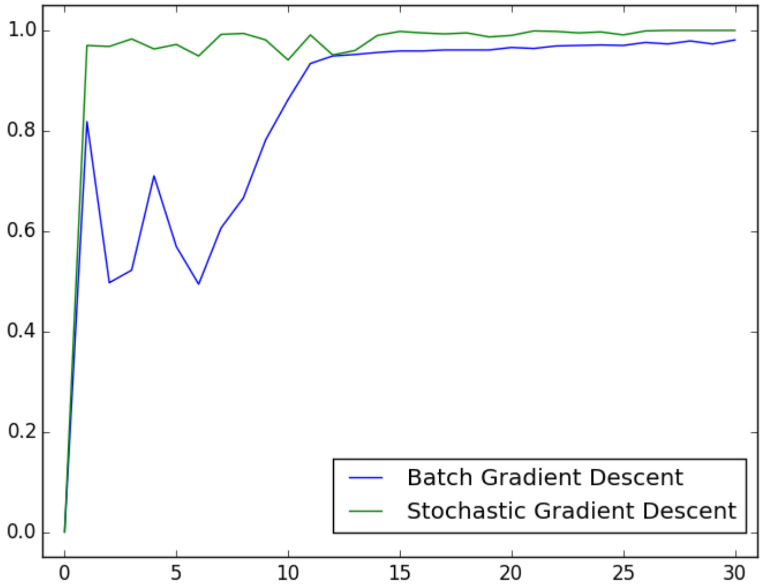
(f) Implement the soft-margin SVM using stochastic gradient descent. Here is the pseudo-code:

Algorithm 2: SVM Stochastic Gradient Descent

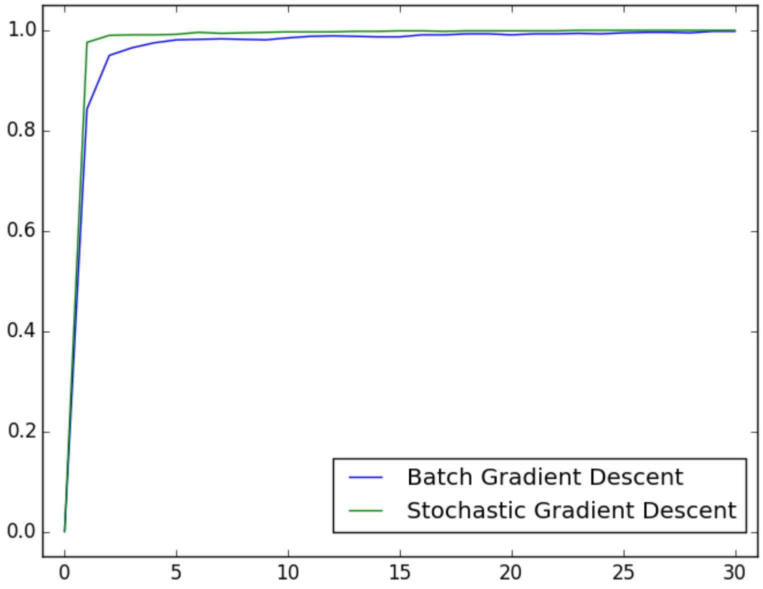
```
$\mathbf{w}^* \leftarrow \mathbf{0}$ ;
$b^* \leftarrow 0$ ;
for $j$=1 to NumIterations do
    for $i$ = Random Permutation of 1 to $N$ do
        $\mathbf{w}_{grad} \leftarrow \nabla_{\mathbf{w}} E^{(i)}(\mathbf{w}^*, b^*)$ ;
        $b_{grad} \leftarrow \frac{\partial}{\partial b} E^{(i)}(\mathbf{w}^*, b^*)$ ;
        $\mathbf{w}^* \leftarrow \mathbf{w}^* - \alpha(j)\ \mathbf{w}_{grad}$ ;
        $b^* \leftarrow b^* - \alpha(j)\ b_{grad}$ ;
    end
end
return $\mathbf{w}^*$
```

Use the same $\alpha(\cdot)$, η_0 and C in (c). Be sure to use a new random permutation of the indices of the inner loop for each iteration of the outer loop. Show the iteration-versus-accuracy (outer iteration and training accuracy) curve **in the same plot** as that for batch gradient descent. The training and test data/labels are provided in `digits_training_data.csv`, `digits_training_labels.csv`, `digits_test_data.csv` and `digits_test_labels.csv`.

Solution:
With unnormalized data:



With normalized data:



(g) What can you conclude about the convergence rate of stochastic gradient descent versus batch gradient descent? How did you make this conclusion?

Solution:
Stochastic gradient descent converges faster. It can be observed from the plot (the slope becomes zero after convergence.)

(h) Show the Lagrangian function for minimization problem (1) and derive the dual problem. Your result should have only dual variables in the objective function as well as the constraints. How can you kernelize the soft-margin SVM based on this dual problem?

Solution:
Let $C_0 = CN$. The Lagrangian is given by

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2}||\mathbf{w}||^2 + \frac{C_0}{N}\sum_i \xi_i - \sum_{i=1}^{n} \alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^{n} \beta_i \xi_i$$

From KKT conditions, we get

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0$$
$$\frac{\partial L}{\partial b} = \sum_i \alpha_i y_i = 0$$
$$\frac{\partial L}{\partial \xi_i} = \frac{C_0}{N} - \alpha_i - \beta_i = 0$$

Plug in $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$, $\sum_i \alpha_i y_i = 0$ and $\alpha_i + \beta_i = \frac{C_0}{N}$, we get the dual problem

$$\begin{aligned} \max_{\alpha,\beta} \quad & -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \sum_i \alpha_i \\ \text{subject to} \quad & \sum_i \alpha_i y_i = 0 \\ & \forall i, \alpha_i + \beta_i = \frac{C_0}{N} \\ & \forall i, \alpha_i \geq 0, \beta_i \geq 0 \end{aligned} \tag{3}$$

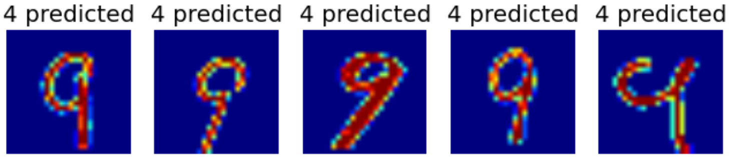
Moreover, we can eliminate the dual variable β and get

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \sum_i \alpha_i \\ \text{subject to} \quad & \sum_i \alpha_i y_i = 0 \\ & \forall i, 0 \leq \alpha_i \leq \frac{C_0}{N} \end{aligned} \tag{4}$$

We notice that the objective function in (4) is related to $\mathbf{x}_i$s only in the form of inner products between $\mathbf{x}_i$s. For kernelization, we can replace those inner products with kernel function values.

(i) Apply the soft-margin SVM (with RBF kernel) to handwritten digit classification. The training and test data/labels are provided in `digits_training_data.csv`, `digits_training_labels.csv`, `digits_test_data.csv` and `digits_test_labels.csv`. Report the training and test accuracy, and show 5 of the misclassified test images (if fewer than 5, show all; label them with your predictions). You can use the scikit-learn (or equivalent) implementation of the kernelized SVM in this question. You are free to select the parameters (for RBF kernel and regularization), and please report your parameters.

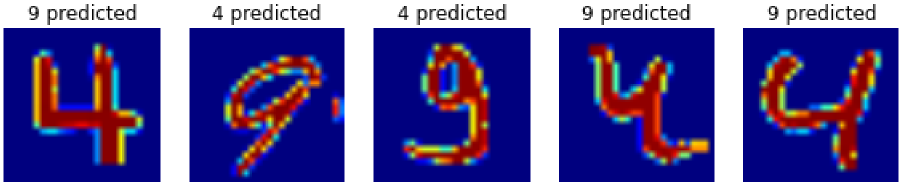
Solution:
With default parameters in Scikit-learn, we get 99.5% training accuracy and 95.4% test accuracy on normalized data (100% training accuracy and 46.8% test accuracy on unnormalized data). The following are 5 misclassified test images (for the model on normalized data):



Your results can be different from the above depending on data normalization and the kernel/regularization parameters you used.

(j) Implement linear discriminant analysis (LDA) using the same data in (i). Report the training and test accuracy, and show 5 of the misclassified test images (if fewer than 5, show all; label them with your predictions). Is there any significant difference between LDA and SVM (with RBF kernel)? Using implementation from libraries is NOT allowed in this question. You can use pseudoinverse during computation.

Solution:
We get 99.4% training accuracy and 90.2% test accuracy. The following are 5 misclassified test images:



2) **Open Kaggle Challenge (20 points).** You can use any algorithm and design any features to perform classification on the handwritten digit dataset. Please refer to `https://inclass.kaggle.com/c/handwritten-digit-classification` for details. This problem will be graded separately based on your performance on the private leaderboard.

3) **Constructing Kernels (20 points).**

(a) Let $\mathbf{u}, \mathbf{v}$ be vectors of dimension d. What feature map ϕ does the kernel

$$k(\mathbf{u}, \mathbf{v}) = (\langle \mathbf{u}, \mathbf{v} \rangle + 1)^4$$

correspond to? In other words, specify the function $\phi(\cdot)$ so that $k(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u})^\top \phi(\mathbf{v})$ for all $\mathbf{u}, \mathbf{v}$. Please show the expression for $d = 3$ and describe how to extend it to arbitrary dimension d.

Solution:

$$
\begin{aligned}
& k(\mathbf{u},\mathbf{v}) \\
&= \left(\sum_{i=1}^{d} u_i v_i + 1\right)^4 \\
&= \left(\sum_{i=1}^{d} u_i v_i\right)^4 + 4\left(\sum_{i=1}^{d} u_i v_i\right)^3 + 6\left(\sum_{i=1}^{d} u_i v_i\right)^2 + 4\sum_{i=1}^{d} u_i v_i + 1 \\
&= \left(\sum_{i=1}^{d} u_i^4 v_i^4 + 4\sum_{\substack{i,j=1\\ i\neq j}}^{d} u_i^3 v_i^3 u_j v_j + 6\sum_{\substack{i,j=1\\ i\neq j}}^{d} u_i^2 v_i^2 u_j^2 v_j^2 + 12\sum_{\substack{i,j,k=1\\ i\neq j\neq k}}^{d} u_i^2 v_i^2 u_j v_j u_k v_k + 24\sum_{\substack{i,j,k,p=1\\ i\neq j\neq k\neq p}}^{d} u_i v_i u_j v_j u_k v_k u_p v_p\right) \\
&+ 4\left(\sum_{i=1}^{d} u_i^3 v_i^3 + 3\sum_{\substack{i,j=1\\ i\neq j}}^{d} u_i^2 v_i^2 u_j v_j + 6\sum_{\substack{i,j,k=1\\ i\neq j\neq k}}^{d} u_i v_i u_j v_j u_k v_k\right) + 6\left(\sum_{i=1}^{d} u_i^2 v_i^2 + 2\sum_{\substack{i,j=1\\ i\neq j}}^{d} u_i v_i u_j v_j\right) + 4\sum_{i=1}^{d} u_i v_i + 1 \\
&= [u_1^4, \ldots, u_d^4, 2u_1^3 u_2, \ldots, 2u_{d-1}^3 u_d, \sqrt{6}u_1^2 u_2^2, \ldots, \sqrt{6}u_{d-1}^2 u_d^2, 2\sqrt{3}u_1^2 u_2 u_3, \ldots, 2\sqrt{3}u_{d-2}^2 u_{d-1} u_d, \\
&\quad 2\sqrt{6}u_1 u_2 u_3 u_4, \ldots, 2\sqrt{6}u_{d-3d-2}^u u_{d-1} u_d, 2u_1^3, \ldots, 2u_d^3, 2\sqrt{3}u_1^2 u_2, \ldots, 2\sqrt{3}u_{d-1}^2 u_d, \\
&\quad 2\sqrt{6}u_1 u_2 u_3, \ldots, 2\sqrt{6}u_{d-2} u_{d-1} u_d, \sqrt{6}u_1^2, \ldots, \sqrt{6}u_d^2, 2\sqrt{3}u_1 u_2, \ldots 2\sqrt{3}u_{d-1} u_d, 2u_1, \ldots, 2u_d, 1]\cdot \\
&\quad [v_1^4, \ldots, v_d^4, 2v_1^3 v_2, \ldots, 2v_{d-1}^3 v_d, \sqrt{6}v_1^2 v_2^2, \ldots, \sqrt{6}v_{d-1}^2 v_d^2, 2\sqrt{3}v_1^2 v_2 v_3, \ldots, 2\sqrt{3}v_{d-2}^2 v_{d-1} v_d, \\
&\quad 2\sqrt{6}v_1 v_2 v_3 v_4, \ldots, 2\sqrt{6}v_{d-3d-2}^v v_{d-1} v_d, 2v_1^3, \ldots, 2v_d^3, 2\sqrt{3}v_1^2 v_2, \ldots, 2\sqrt{3}v_{d-1}^2 v_d, \\
&\quad 2\sqrt{6}v_1 v_2 v_3, \ldots, 2\sqrt{6}v_{d-2} v_{d-1} v_d, \sqrt{6}v_1^2, \ldots, \sqrt{6}v_d^2, 2\sqrt{3}v_1 v_2, \ldots 2\sqrt{3}v_{d-1} v_d, 2v_1, \ldots, 2v_d, 1] \\
&= \phi(\mathbf{u}) \cdot \phi(\mathbf{v})
\end{aligned}
$$

Note that we should remove the term

$$24\sum_{\substack{i,j,k,p=1\\ i\neq j\neq k\neq p}}^{d} u_i v_i u_j v_j u_k v_k u_p v_p$$

when $d = 3$.

(b) Let k_1, k_2 be positive-definite kernel functions over $\mathbb{R}^D \times \mathbb{R}^D$, let $a \in \mathbb{R}^+$ be a positive real number, let $f : \mathbb{R}^D \to \mathbb{R}$ be a real-valued function and let $p : \mathbb{R} \to \mathbb{R}$ be a polynomial with *positive* coefficients. For each of the functions k below, state whether it is necessarily a positive-definite kernel. If you think it is, prove it; if you think it is not, give a counterexample.

(i) $k(\mathbf{x},\mathbf{z}) = k_1(\mathbf{x},\mathbf{z}) + k_2(\mathbf{x},\mathbf{z})$

Solution:
Kernel. The sum of 2 positive semidefinite matrices is a positive semidefinite matrix: $\forall \mathbf{u}, \mathbf{u}^T G_1 \mathbf{u} \geq 0, \mathbf{u}^T G_2 \mathbf{u} \geq 0$ since k_1, k_2 are kernels. This implies $\forall \mathbf{u}, \mathbf{u}^T G \mathbf{u} = \mathbf{u}^T G_1 \mathbf{u} + \mathbf{u}^T G_2 \mathbf{u} \geq 0$.

(ii) $k(\mathbf{x},\mathbf{z}) = k_1(\mathbf{x},\mathbf{z}) - k_2(\mathbf{x},\mathbf{z})$

Solution:
Not a kernel. Counterexample: let $k_2(.,.) = 2k_1(.,.)$ (we are using (iii) here to claim $2k_1$ is a kernel). Then we have $\forall \mathbf{u}, \mathbf{u}^T G \mathbf{u} = \mathbf{u}^T (G1 - 2G1)\mathbf{u} = -\mathbf{u}^T G_1 \mathbf{u} \leq 0$.

(iii) $k(\mathbf{x}, \mathbf{z}) = ak_1(\mathbf{x}, \mathbf{z})$

Solution:
Kernel. Multiplying a PSD by a non-negative integer produces a PSD. $\forall \mathbf{u}, \mathbf{u}^T G_1 \mathbf{u} \geq 0$, which implies $\forall \mathbf{u} a \mathbf{u}^T G_1 \mathbf{u} \geq 0$.

(iv) $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z})k_2(\mathbf{x}, \mathbf{z})$

Solution:
Kernel. k_1 is a kernel, thus $\exists \phi^{(1)}, k_1(\mathbf{x}, \mathbf{z}) = \phi^{(1)}(\mathbf{x})^T \phi^{(1)}(\mathbf{z}) = \sum_i \phi_i^{(1)}(\mathbf{x})\phi_i^{(1)}(\mathbf{z})$. Similarly, k_2 is a kernel, thus $\exists \phi^{(2)}, k_2(\mathbf{x}, \mathbf{z}) = \phi^{(2)}(\mathbf{x})^T \phi^{(2)}(\mathbf{z}) = \sum_i \phi_i^{(2)}(\mathbf{x})\phi_i^{(2)}(\mathbf{z})$.

$$\begin{aligned}
k(\mathbf{x}, \mathbf{z}) &= k_1(\mathbf{x}, \mathbf{z})k_2(\mathbf{x}, \mathbf{z}) \\
&= \sum_i \phi_i^{(1)}(\mathbf{x})\phi_i^{(1)}(\mathbf{z}) \sum_j \phi_j^{(2)}(\mathbf{x})\phi_j^{(2)}(\mathbf{z}) \\
&= \sum_i \sum_j \phi_i^{(1)}(\mathbf{x})\phi_i^{(1)}(\mathbf{z})\phi_j^{(2)}(\mathbf{x})\phi_j^{(2)}(\mathbf{z}) \\
&= \sum_i \sum_j \left(\phi_i^{(1)}(\mathbf{x})\phi_j^{(2)}(\mathbf{x})\right) \left(\phi_i^{(1)}(\mathbf{z})\phi_j^{(2)}(\mathbf{z})\right) \\
&= \sum_{(i,j)} \psi_{i,j}(\mathbf{x})\psi_{i,j}(\mathbf{z})
\end{aligned}$$

Where the last equality holds because we can define $\psi_{i,j}(\mathbf{x}) = \phi_i^{(1)}(\mathbf{x})\phi_j^{(2)}(\mathbf{x})$. Therefore, we can define $\psi(\mathbf{x})$ to be a vector containing all $\psi_{i,j}(\mathbf{x})$ for all possible pairs (i, j). And we can rewrite:

$$k(\mathbf{x}, \mathbf{z}) = \psi(\mathbf{x})^T \psi(\mathbf{z})$$

Since k can be written in this form, it is a kernel.

(v) $k(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})f(\mathbf{z})$

Solution:
Kernel. Just let $\psi(\mathbf{x}) = f(\mathbf{x})$, and since $f(\mathbf{x})$ is a scalar, we have $k(\mathbf{x}, \mathbf{z}) = \psi(\mathbf{x})^T \psi(\mathbf{z})$ and we are done.

(vi) $k(\mathbf{x}, \mathbf{z}) = p(k_1(\mathbf{x}, \mathbf{z}))$

Solution:
Kernel. By combining (i) sum, (iii) scalar product, (iv) powers, (v) constant term, we see that any polynomial of a kernel k_1 will again be a kernel.

(vii) Prove that the Gaussian Kernel $k(\mathbf{x}, \mathbf{z}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}\right)$ can be expressed as $\phi(\mathbf{x})^T \phi(\mathbf{z})$, where $\phi(\cdot)$ is an infinite-dimensional vector. (Hint: using power series)

Solution:

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= \exp(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}) \\ &= \exp(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}) \cdot \exp(-\frac{\|\mathbf{z}\|^2}{2\sigma^2}) \cdot \exp(\frac{\mathbf{x}^T\mathbf{z}}{\sigma^2}) \\ &= \exp(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}) \cdot \exp(-\frac{\|\mathbf{z}\|^2}{2\sigma^2}) \cdot \sum_{n=0}^{\infty} \frac{1}{n!} \left(\frac{\mathbf{x}^T\mathbf{z}}{\sigma^2} \right)^n \end{aligned}$$

where

$$\begin{aligned} \frac{(\mathbf{x}^T\mathbf{z})^n}{n!\sigma^{2n}} &= \frac{1}{n!\sigma^{2n}} \sum_{k_1+\cdots+k_d=n} \left(\begin{array}{c} n \\ k_1, \cdots, k_d \end{array} \right) \Pi_{i=1}^d (x_i z_i)^{k_i} \\ &= \left(\frac{\sqrt{\left(\begin{array}{c} n \\ k_1^{(1)}, \cdots, k_d^{(1)} \end{array} \right)}}{\sigma^n \sqrt{n!}} \Pi_{i=1}^d x_i^{k_i^{(1)}}, \cdots, \frac{\sqrt{\left(\begin{array}{c} n \\ k_1^{(m_n)}, \cdots, k_d^{(m_n)} \end{array} \right)}}{\sigma^n \sqrt{n!}} \Pi_{i=1}^d x_i^{k_i^{(m_n)}} \right) \\ &\cdot \left(\frac{\sqrt{\left(\begin{array}{c} n \\ k_1^{(1)}, \cdots, k_d^{(1)} \end{array} \right)}}{\sigma^n \sqrt{n!}} \Pi_{i=1}^d z_i^{k_i^{(1)}}, \cdots, \frac{\sqrt{\left(\begin{array}{c} n \\ k_1^{(m_n)}, \cdots, k_d^{(m_n)} \end{array} \right)}}{\sigma^n \sqrt{n!}} \Pi_{i=1}^d z_i^{k_i^{(m_n)}} \right)^T \\ &= \tilde{\boldsymbol{\phi}}_n(\mathbf{x})^T \tilde{\boldsymbol{\phi}}_n(\mathbf{z}) \end{aligned}$$

And we have

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= \exp(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}) \cdot \exp(-\frac{\|\mathbf{z}\|^2}{2\sigma^2}) \cdot \sum_{n=0}^{\infty} \frac{1}{n!} \left(\frac{\mathbf{x}^T\mathbf{z}}{\sigma^2} \right)^n \\ &= \left(\exp(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}) \sum_{n=0}^{\infty} \tilde{\boldsymbol{\phi}}_n(\mathbf{x}) \right)^T \left(\exp(-\frac{\|\mathbf{z}\|^2}{2\sigma^2}) \sum_{n=0}^{\infty} \tilde{\boldsymbol{\phi}}_n(\mathbf{z}) \right) \\ &= \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{z}) \end{aligned}$$

4) **Kernelized Ridge Regression (20 points).**

Recall that the error function for ridge regression (linear regression with L2 regularization) is:

$$E(\mathbf{w}) = (\Phi\mathbf{w} - \mathbf{t})^T(\Phi\mathbf{w} - \mathbf{t}) + \lambda\mathbf{w}^T\mathbf{w}$$

and its closed-form solution and model are:

$$\hat{\mathbf{w}} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T\mathbf{t} \text{ and } \hat{f}(\mathbf{x}) = \hat{\mathbf{w}}^T\phi(\mathbf{x}) = \mathbf{t}^T\Phi(\Phi^T\Phi + \lambda I)^{-1}\phi(\mathbf{x})$$

Now we want to kernelize ridge regression and allow non-linear models.

(a) Use the following matrix inverse lemma to derive the closed-form solution and model for kernelized ridge regression:

$$(P + QRS)^{-1} = P^{-1} - P^{-1}Q(R^{-1} + SP^{-1}Q)^{-1}SP^{-1}$$

where P is an $n \times n$ invertible matrix, R is a $k \times k$ invertible matrix, Q is an $n \times k$ matrix and S is a $k \times n$ matrix. Make sure that your kernelized model only depends on the feature vectors $\phi(\mathbf{x})$ through inner products with other feature vectors.

Solution:
Let $P = \lambda I$, $Q = \Phi^T$, $R = I$ and $S = \Phi$, then we have

$$\begin{aligned}
\hat{f}(\mathbf{x}) &= \mathbf{t}^T\Phi(\Phi^T\Phi + \lambda I)^{-1}\phi(\mathbf{x}) \\
&= \mathbf{t}^T\Phi\left(\frac{1}{\lambda}I - \frac{1}{\lambda^2}\Phi^T(I + \frac{1}{\lambda}\Phi\Phi^T)^{-1}\Phi)\right)\phi(\mathbf{x}) \\
&= \frac{1}{\lambda}\mathbf{t}^T\Phi\left(I - \Phi^T(\lambda I + K)^{-1}\Phi)\right)\phi(\mathbf{x}) \\
&= \frac{1}{\lambda}\mathbf{t}^T\left(\mathrm{k}(x) - K(\lambda I + K)^{-1}\mathrm{k}(x)\right) \\
&= \frac{1}{\lambda}\mathbf{t}^T\left((\lambda I + K)(\lambda I + K)^{-1}\mathrm{k}(x) - K(\lambda I + K)^{-1}\mathrm{k}(x)\right) \\
&= \mathbf{t}^T(\lambda I + K)^{-1}\mathrm{k}(x) \\
&= \mathrm{k}(x)^T(\lambda I + K)^{-1}\mathbf{t}
\end{aligned}$$

(b) Apply kernelized ridge regression to the steel ultimate tensile strength dataset. The training data and test data are provided in `steel_composition_train.csv` and `steel_composition_test.csv`, respectively. We recommend you to normalize the data before applying the models. Report the RMSE (Root Mean Square Error) of the models on the training data. Try (set $\lambda = 1$)

(i) Polynomial kernel $k(\mathbf{u}, \mathbf{v}) = (\langle \mathbf{u}, \mathbf{v}\rangle + 1)^2$

Solution:
Training RMSE = 7.14303912541

(ii) Polynomial kernel $k(\mathbf{u}, \mathbf{v}) = (\langle \mathbf{u}, \mathbf{v}\rangle + 1)^3$

Solution:
Training RMSE = 4.42103029781

(iii) Polynomial kernel $k(\mathbf{u}, \mathbf{v}) = (\langle \mathbf{u}, \mathbf{v}\rangle + 1)^4$

Solution:
Training RMSE = 2.54776712597

(iv) Gaussian kernel $k(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{2\sigma^2}\right)$ (set $\sigma = 1$)

Solution:
Training RMSE = 6.94063914936