AI6102 Machine Learning Methodologies & Applications

Chen Yongquan (G2002341D)

Nanyang Technological University

Assignment 1

Question 1

There are two approaches to multi-class classification using logistic regression. The first approach is the one-vs-rest (OvR) scheme while the other is multinomial logistic regression.

One-vs-rest

For the OvR scheme with regularization, we still use the cross-entropy loss defined in logistic regression for binary classification, defined as:

$$\ell(h(\boldsymbol{x};\boldsymbol{w}),y) = -\sum_{i=1}^{N}\left[y_i \ln\big(h(\boldsymbol{x}_i;\boldsymbol{w})\big) - (1-y_i)\ln\big(1-h(\boldsymbol{x}_i;\boldsymbol{w})\big)\right] + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

Where N is the number of training samples per batch while h is the logistic function.

The gradient descent update rule for binary classification is then defined as:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \rho\left(\sum_{i=1}^{N}\big(y_i - h(\boldsymbol{x}_i;\boldsymbol{w})\big)\boldsymbol{x}_i - \lambda\boldsymbol{w}\right)$$

Where t is the training iteration and ρ is the learning rate.

For a multi-class classification problem of C classes, $\{0, 1, \cdots, C-1\}$, we will then train C binary classifiers separately for all classes, each predicting the probability $P(y = c|\boldsymbol{x})$ that $\boldsymbol{x}$ belongs to the class c or not. Thus, we have C classifiers that give the predicted value for:

$$P(y = c|\boldsymbol{x}) = \frac{\exp(-\boldsymbol{w}_c^T\boldsymbol{x})}{1+\exp(-\boldsymbol{w}_c^T\boldsymbol{x})}$$

Thus, for the OvR scheme, given an input $\boldsymbol{x}$, we use C classifiers to separately predict the probability that $\boldsymbol{x}$ belongs to a class c. The predicted class is then the class with the highest probability out of all the classes:

$$\hat{y} = \underset{c}{\operatorname{argmax}}\big(P(y = c|\boldsymbol{x})\big)$$

Such an approach can result in probabilities for all classifiers summing up to more than 1.0. For single label cases, scikit-learn normalizes the probabilities for all classes such that they sum up to 1.0.

Multinomial Logistic Regression

The second approach for multi-class classification uses the softmax function, $g(\boldsymbol{w}^T\boldsymbol{x})$, instead of the logistic function, $h(\boldsymbol{w}^T\boldsymbol{x})$, for generating the multi-class probability distribution. The softmax function is a generalization of the logistic function for more than 2 classes, and is defined as:

$$g(\boldsymbol{z}_i) = \frac{\exp(\boldsymbol{z}_i)}{\sum_{c=0}^{C-1}\big(\exp(\boldsymbol{z}_c)\big)}$$

Where $z_i = -\boldsymbol{w}_c^T\boldsymbol{x}$ and the probabilities for all classes sum to 1.

If we fix the weight for class 0 to 0 and learn $C-1$ weights $\boldsymbol{w}_c$ for classes $c \in \{1, \cdots, C-1\}$, then we can represent the predicted conditional probabilities in parametric form as:

$$\text{For } c > 0: \quad P(y = c|\boldsymbol{x}) = \frac{\exp(-\boldsymbol{w}_c^T\boldsymbol{x})}{1 + \sum_{c=1}^{C-1}\exp(-\boldsymbol{w}_c^T\boldsymbol{x})}$$
$$\text{For } c = 0: \quad P(y = 0|\boldsymbol{x}) = \frac{1}{1 + \sum_{c=1}^{C-1}\exp(-\boldsymbol{w}_c^T\boldsymbol{x})}$$

Because we do not implement $\boldsymbol{w}_c$ as a learnable parameter, we can assume its weight is 0 and $\exp(-\boldsymbol{w}_0^T\boldsymbol{x}) = 1$, giving us the 1 in the numerator and denominator in the second equation. Also, since the probabilities for the softmax sum to 1, we can get the conditional probability for class zero from the difference between 1 and the sums of the probabilities for all other classes:

$$\sum_{c=0}^{C-1} P(y = c|\boldsymbol{x}) = 1$$
$$P(y = 0|\boldsymbol{x}) = 1 - \sum_{c=1}^{C-1} P(y = c|\boldsymbol{x})$$

We can also start by define the model as a set of logits:

$$\ln\frac{P(y = 1|\boldsymbol{x})}{P(y = 0|\boldsymbol{x})} = -\boldsymbol{w}_1^T\boldsymbol{x}$$
$$\vdots$$
$$\ln\frac{P(y = C-1|\boldsymbol{x})}{P(y = 0|\boldsymbol{x})} = -\boldsymbol{w}_{C-1}^T\boldsymbol{x}$$

This set of logits then gives us the equations:

$$P(y = 1|\boldsymbol{x}) = P(y = 0|\boldsymbol{x})\exp(-\boldsymbol{w}_1^T\boldsymbol{x})$$
$$\vdots$$
$$P(y = C-1|\boldsymbol{x}) = P(y = 0|\boldsymbol{x})\exp(-\boldsymbol{w}_{C-1}^T\boldsymbol{x})$$

Since all the conditional probabilities sum to 1, we have the equation:

$$P(y=0|\boldsymbol{x}) = 1 - \sum_{c=1}^{C-1} P(y=c|\boldsymbol{x}) = 1 - \sum_{c=1}^{C-1} \big(P(y=0|\boldsymbol{x})\exp(-\boldsymbol{w}_c^T\boldsymbol{x})\big)$$
$$= 1 - P(y=0|\boldsymbol{x}) \sum_{c=1}^{C-1} \exp(-\boldsymbol{w}_c^T\boldsymbol{x})$$

Then:

$$1 = P(y=0|\boldsymbol{x}) + P(y=0|\boldsymbol{x}) \sum_{c=1}^{C-1} \exp(-\boldsymbol{w}_c^T\boldsymbol{x})$$
$$1 = P(y=0|\boldsymbol{x}) + \left(1 + \sum_{c=1}^{C-1} \exp(-\boldsymbol{w}_c^T\boldsymbol{x})\right)$$
$$P(y=0|\boldsymbol{x}) = \frac{1}{1 + \sum_{c=1}^{C-1} \exp(-\boldsymbol{w}_c^T\boldsymbol{x})}$$

Thus, substituting into the set of logits we defined at the beginning, we arrive at the same parametric form for multinomial logistic regression:

$$P(y=c|\boldsymbol{x}) = \frac{\exp(-\boldsymbol{w}_c^T\boldsymbol{x})}{1 + \sum_{c=1}^{C-1} \exp(-\boldsymbol{w}_c^T\boldsymbol{x})}$$

We can also represent the parametric form more compactly as:

$$P(y|\boldsymbol{x};\boldsymbol{w}) = \prod_{c=0}^{C-1} [g(\boldsymbol{x};\boldsymbol{w}_c)]^{y_c} = \sum_{c=0}^{C-1} y_c \ln\big(g(\boldsymbol{x};\boldsymbol{w}_c)\big)$$

Thus, given a set of N training samples, in order to maximize the likelihood of $P(y_i|\boldsymbol{x}_i;\boldsymbol{w}_c)$ with respect to $\boldsymbol{w}_c$, where $i \in \{1,\cdots,N\}$, we have to find a solution for $\boldsymbol{w}_c$**:**

$$\widehat{\boldsymbol{w}_c} = \underset{\boldsymbol{w}_c}{\operatorname{argmax}} \prod_{i=1}^{N} P(y_i|\boldsymbol{x}_i;\boldsymbol{w}_c) = \underset{\boldsymbol{w}_c}{\operatorname{argmax}} \sum_{i=1}^{N} \sum_{c=0}^{C-1} \big(y_i \ln\big(g(\boldsymbol{x}_i;\boldsymbol{w}_c)\big)\big)$$

Then, the cross-entropy loss for a batch of N training samples using softmax function for prediction is given by:

$$\ell(g(\boldsymbol{x};\boldsymbol{w}), y_i) = -\sum_{i=1}^{N} \sum_{c=0}^{C-1} y_i \ln\big(g(\boldsymbol{x}_i;\boldsymbol{w}_c)\big)$$

In order to perform gradient descent, we need the first derivative of the loss function with respect to parameter $\boldsymbol{w}_c$:

$$\boldsymbol{w}_{c,t+1} = \boldsymbol{w}_{c,t} - \rho \frac{\partial E(\boldsymbol{w}_c)}{\partial \boldsymbol{w}_c}$$

$$\frac{\partial E(\boldsymbol{w}_c)}{\partial \boldsymbol{w}_c} = \frac{\partial\left(-\sum_{i=1}^{N}\sum_{c=0}^{C-1}\left(y_i \ln\left(g(\boldsymbol{x}_i;\boldsymbol{w}_c)\right)\right)\right)}{\partial \boldsymbol{w}_c}$$

$$= -\sum_{i=1}^{N}\sum_{c=0}^{C-1}\left(\frac{\partial y_i \ln\left(g(\boldsymbol{x}_i;\boldsymbol{w}_c)\right)}{\partial \boldsymbol{w}_c}\right) \quad (1)$$

$$\frac{\partial y_i \ln\left(g(\boldsymbol{x}_i;\boldsymbol{w}_c)\right)}{\partial \boldsymbol{w}_c} = y_i\left(\frac{1}{g(\boldsymbol{x}_i;\boldsymbol{w}_c)}\right)\left(\frac{\partial g(\boldsymbol{x}_i;\boldsymbol{w}_c)}{\partial \boldsymbol{w}_c}\right) \quad (2)$$

Let $z_{id} = -\boldsymbol{w}_d^T \boldsymbol{x}_i$, then when $c = d$:

$$\frac{\partial g(z_{id})}{\partial z_{ic}} = \frac{\partial \frac{\exp(z_{ic})}{\sum_{j=0}^{C-1}\exp(z_{ij})}}{\partial z_{ic}}$$

$$= \frac{\left(\sum_{j=0}^{C-1}\exp(z_{ij})\right)\left(\exp(z_{ic})\right) - \left(\exp(z_{ic})\right)\left(\exp(z_{ic})\right)}{\left(\sum_{j=0}^{C-1}\exp(z_{ij})\right)^2}$$

$$= \frac{\left(\exp(z_{ic})\right)\left(\sum_{j=0}^{C-1}\exp(z_{ij}) - \exp(z_{ic})\right)}{\left(\sum_{j=0}^{C-1}\exp(z_{ij})\right)^2}$$

$$= \left(\frac{\exp(z_{ic})}{\sum_{j=0}^{C-1}\exp(z_{ij})}\right)\left(\frac{\sum_{j=0}^{C-1}\exp(z_{ij}) - \exp(z_{ic})}{\sum_{j=0}^{C-1}\exp(z_{ij})}\right)$$

$$= \left(\frac{\exp(z_{ic})}{\sum_{j=0}^{C-1}\exp(z_{ij})}\right)\left(1 - \frac{\exp(z_{ic})}{\sum_{j=0}^{C-1}\exp(z_{ij})}\right)$$

$$= g(\boldsymbol{x}_i;\boldsymbol{w}_c)\left(1 - g(\boldsymbol{x}_i;\boldsymbol{w}_c)\right) \quad (3)$$

When $c \neq d$:

$$\frac{\partial g(z_{id})}{\partial z_{ic}} = \frac{0 - \left(\exp(z_{ic})\right)\left(\exp(z_{id})\right)}{\left(\sum_{j=0}^{C-1}\exp(z_{ij})\right)^2} = -\frac{\exp(z_{ic})}{\sum_{j=0}^{C-1}\exp(z_{ij})}\frac{\exp(z_{id})}{\sum_{j=0}^{C-1}\exp(z_{ij})}$$

$$= -g(\boldsymbol{x}_i;\boldsymbol{w}_c)g(\boldsymbol{x}_i;\boldsymbol{w}_d) \quad (4)$$

Then, by chain rule:

$$\frac{\partial g(z_{id})}{\partial \boldsymbol{w}_c} = \frac{\partial g(z_{id})}{\partial z_{ic}} \cdot \frac{\partial z_{ic}}{\partial \boldsymbol{w}_c} = \frac{\partial g(z_{id})}{\partial z_{ic}} \frac{\partial(-\boldsymbol{w}_c^T \boldsymbol{x}_i)}{\partial \boldsymbol{w}_c} = \frac{\partial g(z_{ic})}{\partial z_{ic}} (-\boldsymbol{x}_i)$$

Substituting back into (2), (3) and (4) back into (1):

$$\begin{aligned}
\frac{\partial E(\boldsymbol{w}_c)}{\partial \boldsymbol{w}_c} &= -\sum_{i=1}^{N}\sum_{c=0}^{C-1}\left(\frac{\partial y_i \ln\big(g(z_{ic})\big)}{\partial \boldsymbol{w}_c}\right) \\
&= -\sum_{i=1}^{N}\sum_{c=0}^{C-1}\left(y_i\left(\frac{1}{g(z_{ic})}\right)\left(\frac{\partial g(z_{ic})}{\partial \boldsymbol{w}_c}\right)\right) \\
&= -\sum_{i=1}^{N}\left(\left(\frac{y_i}{g(z_{ic})}\right)\left(\frac{\partial g(z_{ic})}{\partial \boldsymbol{w}_c}\right) + \sum_{d\neq c}^{C-1}\left(\frac{y_i}{g(z_{ic})}\frac{\partial g(z_{id})}{\partial \boldsymbol{w}_c}\right)\right) \\
&= -\sum_{i=1}^{N}\left(\left(\frac{y_i}{g(z_{ic})}\right)\left(\frac{\partial g(z_{ic})}{\partial z_{ic}}\right) + \sum_{d\neq c}^{C-1}\left(\frac{y_i}{g(z_{ic})}\frac{\partial g(z_{id})}{\partial z_{ic}}\right)\right)\cdot\frac{\partial z_{ic}}{\partial \boldsymbol{w}_c} \\
&= -\sum_{i=1}^{N}\left(\left(\frac{y_i g(z_{ic})\big(1-g(z_{ic})\big)}{g(z_{ic})}\right) + \sum_{d\neq c}^{C-1}\left(\frac{-y_i g(z_{ic}) g(z_{id})}{g(z_{ic})}\right)\right)(-\boldsymbol{x}_i) \\
&= -\sum_{i=1}^{N}\left(\big(y_i\big(1-g(z_{ic})\big)\big) + \sum_{d\neq c}^{C-1}\big(-y_i g(z_{id})\big)\right)(-\boldsymbol{x}_i) \\
&= -\sum_{i=1}^{N}\left(y_i - y_i g(z_{ic}) - \sum_{d\neq c}^{C-1}\big(y_i g(z_{id})\big)\right)(-\boldsymbol{x}_i) \\
&= -\sum_{i=1}^{N}\left(y_i - \sum_{c=0}^{C-1}\big(y_i g(z_{ic})\big)\right)(-\boldsymbol{x}_i) \\
&= -\sum_{i=1}^{N}\left(y_i - g(z_{ic})\sum_{c=0}^{C-1} y_i\right)(-\boldsymbol{x}_i) \\
&= -\sum_{i=1}^{N}\big(g(\boldsymbol{x}_i;\boldsymbol{w}_c) - y_i\big)\,(\boldsymbol{x}_i)
\end{aligned}$$

Finally, substituting everything back into the gradient descent rule:

$$\boldsymbol{w}_{t+1,c} = \boldsymbol{w}_{t,c} + \rho \sum_{i=1}^{N} (g(\boldsymbol{x}_i; \boldsymbol{w}_c) - y_i)\, \boldsymbol{x}_i$$

With L2 regularization, the loss function is augmented by a regularization parameter as shown:

$$\ell(g(\boldsymbol{x}; \boldsymbol{w}), y_i) = -\sum_{i=1}^{N} \left[\sum_{c=0}^{C-1} y_i \ln\big(g(\boldsymbol{x}_i; \boldsymbol{w}_c)\big) \right] + \frac{\lambda}{2} \|\boldsymbol{w}_c\|_2^2$$

Consequently, the partial derivative for the loss function with respect to $\boldsymbol{w}$ becomes:

$$\frac{\partial E(\boldsymbol{w}_c)}{\partial \boldsymbol{w}_c} = -\sum_{i=1}^{N} (g(\boldsymbol{x}_i; \boldsymbol{w}_c) - y_i)\, (\boldsymbol{x}_i) + \lambda \boldsymbol{w}_c$$

And the gradient descent rule:

$$\boldsymbol{w}_{t+1,c} = \boldsymbol{w}_{t,c} + \rho \left[\sum_{i=1}^{N} (g(\boldsymbol{x}_i; \boldsymbol{w}_c) - y_i)\, \boldsymbol{x}_i - \lambda \boldsymbol{w}_c \right]$$

Depending on the convention we use for the softmax function,

$$\text{Maximum convention:} \quad g(z_i) = \frac{\exp(\boldsymbol{w}_i^T \boldsymbol{x})}{\sum_{c=0}^{C-1} \big(\exp(\boldsymbol{w}_c^T \boldsymbol{x})\big)}$$

$$\text{Minimum convention:} \quad g(z_i) = \frac{\exp(-\boldsymbol{w}_i^T \boldsymbol{x})}{\sum_{c=0}^{C-1} \big(\exp(-\boldsymbol{w}_c^T \boldsymbol{x})\big)}$$

Where $z_{ic} = \pm \boldsymbol{w}_c^T \boldsymbol{x}_i$, the resulting partial derivative would also follow the change in sign, $\frac{\partial z_{ic}}{\partial \boldsymbol{w}_c} = \pm \boldsymbol{x}_i$. Thus, if we follow the positive convention, the update rule would then be as such:

$$\boldsymbol{w}_{t+1,c} = \boldsymbol{w}_{t,c} + \rho \left[\sum_{i=1}^{N} \big(y_i - g(\boldsymbol{x}_i; \boldsymbol{w}_c)\big)\, \boldsymbol{x}_i - \lambda \boldsymbol{w}_c \right]$$

Question 2

1.

This is the code for importing the training and test data and setting up the basic environment.

```
from sklearn.datasets import load_svmlight_files
from sklearn.model_selection import KFold
from sklearn.svm import LinearSVC
from sklearn import datasets
import numpy as np
import pandas as pd
import sklearn as sk
np.set_printoptions(linewidth = 140)

C = [0.01, 0.1 , 1.0, 10, 100]
train_X, train_y, test_X, test_y  =
sk.datasets.load_svmlight_files(['a5a', 'a5a.t'])
```

This is the code used for performing the 3-fold cross-validation and for creating the LinearSVC instance:

```
def kfLSVC(X, y, C = 1, n_splits = 3, max_iter = 1000):
  svc = sk.svm.LinearSVC(C = C, max_iter = max_iter)
  kf = sk.model_selection.KFold(n_splits)
  scores = []
  for cv_train, cv_test in kf.split(X):

    svc.fit(X[cv_train], y[cv_train])
    scores.append(svc.score(X[cv_test], y[cv_test]))
  return svc, scores

svc_set = []
scores = []
for C_i in C:
  output = kfLSVC(train_X, train_y, C_i)
  svc_set.append(output[0])
  scores.append(output[1])
  print("C: %-5s Scores: [%-20s%20s%20s], Average: %-20s n_iter_: %s" %
  (str(C_i)+',',
  str(output[1][0])+',', str(output[1][1])+',', str(output[1][2]),
  str(np.average(output[1]))+',', output[0].n_iter_))

print('')

for i, svc in enumerate(svc_set):
  print("C: " + str(C[i]) + " TScore: " + str(svc.score(test_X, test_y)))
```

2.

The 3-fold cross-validation results of the set of 5 candidate values of C on the a5a training set is as shown below:

	Accuracy of linear SVMs
$C = 0.01$	0.839569691
$C = 0.1$	0.842531961
$C = 1$	0.839101964
$C = 10$	0.839101964
$C = 100$	0.797318366

The results above were obtained with default parameters for LinearSVC and the KFold model except for the values of C. We note that the maximum number of iterations was reached by the classifier when C was set to 10 and 100. The default maximum number of iterations is 100.

3.

Using C = 0.1, which gave the best mean accuracy in the 3-fold cross validation, we train a LinearSVC model using the entire training set. The code used is as shown below:

```
idx = 1
final_svc, final_svc_score = kfLSVC(train_X, train_y, C[idx])
print("C: %-5s Scores: [%-20s%20s%20s], Average: %-20s n_iter_: %s" %
      (str(C[idx])+',',
       str(final_svc_score[0])+',',
       str(final_svc_score[1])+',',
       str(final_svc_score[2]),
       str(np.average(final_svc_score))+',',
       final_svc.n_iter_))
print("C: %-5s TScore:  %-20s" %
      (str(C[idx])+',',
       final_svc.score(test_X, test_y)))
```

The test results obtained from this classifier is as shown:

	Accuracy of linear SVMs
$C = 0.1$	0.8451065131755077

We note that by setting max iterations to 100000, we can observe convergence in the classifiers for when $C = 10$ and $C = 100$.

Question 3

The optimization problem statement for ridge regression is given by:

$$\widehat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \left[\left(\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2 \right) + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 \right]$$

Applying the kernel trick to map $\boldsymbol{x}$ to a higher dimension to solve non-linear regression problems by finding a linear function in the higher dimension:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)$$

$$\widehat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \left[\left(\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \varphi(\boldsymbol{x}_i) - y_i)^2 \right) + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 \right]$$

Then, to find a closed-form solution, we have to solve for:

$$\frac{\partial \left[\left(\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \varphi(\boldsymbol{x}_i) - y_i)^2 \right) + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 \right]}{\partial x} = 0$$

$$\frac{\partial \left[\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \varphi(\boldsymbol{x}_i) - y_i)^2 \right]}{\partial x} + \frac{\partial \left[\frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 \right]}{\partial x} = 0$$

$$\frac{\partial \left[\frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \varphi(\boldsymbol{x}_i) - y_i)^2 \right]}{\partial x} + \lambda \boldsymbol{w} = 0$$

$$\sum_{i=1}^{N} (\boldsymbol{w}^T \varphi(\boldsymbol{x}_i) - y_i) \varphi(\boldsymbol{x}_i) + \lambda \boldsymbol{w} = 0$$

$$\sum_{i=1}^{N} \left(\boldsymbol{w}^T \varphi(\boldsymbol{x}_i) \right) \varphi(\boldsymbol{x}_i) - \sum_{i=1}^{N} y_i \varphi(\boldsymbol{x}_i) + \lambda \boldsymbol{w} = 0$$

$$\sum_{i=1}^{N} \varphi(\boldsymbol{x}_i) \left(\boldsymbol{w}^T \varphi(\boldsymbol{x}_i) \right) - \sum_{i=1}^{N} y_i \varphi(\boldsymbol{x}_i) + \lambda \boldsymbol{w} = 0$$

$$\left(\sum_{i=1}^{N} (\varphi(\boldsymbol{x}_i) \varphi(\boldsymbol{x}_i)^T) \right) \boldsymbol{w} - \sum_{i=1}^{N} y_i \varphi(\boldsymbol{x}_i) + \lambda \boldsymbol{w} = 0$$

Let $\boldsymbol{\Phi}$ represent the matrix for $\varphi(\boldsymbol{x}_i)$ for N data samples of m dimension as below:

$\boldsymbol{\Phi}$

$\varphi(\boldsymbol{x}_{01})$	$\varphi(\boldsymbol{x}_{02})$	$\cdots$	$\varphi(\boldsymbol{x}_{0N})$
$\varphi(\boldsymbol{x}_{11})$	$\varphi(\boldsymbol{x}_{12})$	$\cdots$	$\varphi(\boldsymbol{x}_{1N})$
$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\varphi(\boldsymbol{x}_{m1})$	$\varphi(\boldsymbol{x}_{m2})$	$\cdots$	$\varphi(\boldsymbol{x}_{mN})$

Then:

$$\sum_{i=1}^{N}(\varphi(\boldsymbol{x}_i)\varphi(\boldsymbol{x}_i)^T) = \boldsymbol{\Phi}\boldsymbol{\Phi}^T$$

Substituting into our intermediate solution previously:

$$\boldsymbol{\Phi}\boldsymbol{\Phi}^T\boldsymbol{w} - \sum_{i=1}^{N} y_i\varphi(\boldsymbol{x}_i) + \lambda\boldsymbol{w} = 0$$

$$\boldsymbol{\Phi}\boldsymbol{\Phi}^T\boldsymbol{w} - \boldsymbol{\Phi}\mathrm{y} + \lambda\boldsymbol{w} = 0$$

$$\boldsymbol{\Phi}\boldsymbol{\Phi}^T\boldsymbol{w} + \lambda\boldsymbol{w} = \boldsymbol{\Phi}\mathrm{y}$$

$$(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \lambda\boldsymbol{I})\boldsymbol{w} = \boldsymbol{\Phi}\mathrm{y}$$

$$\boldsymbol{w} = (\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}\mathrm{y}$$

$$\boldsymbol{w} = \boldsymbol{\Phi}^{\boldsymbol{T}}(\boldsymbol{\Phi}^{\boldsymbol{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\mathrm{y}$$

Let

$$\boldsymbol{K} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)\cdot\varphi(\boldsymbol{x}_j)$$

Then,

$$\boldsymbol{w} = \boldsymbol{\Phi}^{\boldsymbol{T}}(\boldsymbol{K} + \lambda\boldsymbol{I})^{-1}\mathrm{y}$$