EECS 545 - Homework 2

Ding Ding

February 22, 2016

1. Support Vector Machine

1.

a). " $\Rightarrow$ "

For $\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i$

$t^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i \quad \Rightarrow \quad \xi_i \geq \max\{0, \; 1 - t^{(i)}(w^T x^{(i)} + b)\}$

$\xi_i \geq 0$

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

$$\underbrace{\min_{w,b} \max_{\lambda_i} L(w,b,\lambda) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i + \sum_{i=1}^{N} \lambda_i(\max\{0, 1 - t^{(i)}(w^T x^{(i)} + b)\} - \xi_i\}}$$

$$\frac{\partial L}{\partial \lambda} = 0 = \max\{0, 1 - t^{(i)}(w^T x^{(i)} + b)\} - \xi_i$$

$$\Rightarrow \quad \xi_i = \max\{0, 1 - t^{(i)}(w^T x^{(i)} + b)\}$$

$\Rightarrow$ The problem is equivalent to $\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \max\{0, 1 - t^{(i)}(w^T x^{(i)} + b)\}$

" $\Leftarrow$ " let $\xi_i = \max(0, 1 - t^{(i)}(w^T x^{(i)} + b)) \Rightarrow \begin{cases} \xi_i \geq 0 \\ \xi_i \geq 1 - t^{(i)}(w^T x^{(i)} + b) \end{cases}$

$\therefore$ this unconstricted problem can be writen as

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i$$

$$\text{s.t.} \begin{cases} \xi_i \geq 0 \\ \xi_i \geq 1 - t^{(i)}(w^T x^{(i)} + b) \end{cases}$$

(b). The Hyperplane is $\{x, \; 1 - t^{(i)}((w^*)^T x + b^*) = 0\}$.

we have for $x^{(i)}$ in training data st

$$x^{(i)} = x^{(0)} + r \cdot \frac{w}{\|w\|} \qquad \text{for } x^{(0)} \in H \quad r \neq 0$$

$$\Rightarrow 1 - t^{(i)}((w^*)^T x + b^*) = 1 - t^{(i)}(w^{*T}(x^{(0)} + r\frac{w}{\|w\|}) + b^*)$$

$$= 1 - t^{(i)}(w^{*T} x^{(0)} + b) - t^{(i)}(w^{*T} r \frac{w}{\|w\|})$$

$$= -t^{(i)} r \|w\|$$

$$\Rightarrow \quad r = \frac{|t^{(i)}((w^*)^T x + b^*) - 1|}{\|w\|}$$

since $\xi_i > 0$, $\xi_i \neq 0$,

For $\mathcal{L} = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i + \sum_{i=1}^{N}\alpha_i\xi_i + \sum_{i=1}^{N}\beta_i(1 - t^{(i)}(w^T x^{(i)} + b - \xi_i)$

due to KKT condition,

$\xi_i > 0, \quad \alpha_i = 0,$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0.$$

$\alpha_i + \beta_i = C \Rightarrow \beta_i \neq 0,$ due to KKT, $1 - t^{(i)}(w^{*T}x^{(i)} + b) = \xi_i$

$\Rightarrow r = + \frac{\xi_i^*}{\|w^*\|}$ which is proportional to ξ_i^*.

(c). For $t^{(i)}(w^T x^{(i)} + b) \neq 1,$

we have

$$\nabla_w E(w,b) = w - C\sum_{i=1}^{N} t^{(i)}x^{(i)} \mathbb{1}_{\{t^{(i)}(w^T x^{(i)} + b) < 1\}}$$

$$\frac{\partial}{\partial b}E(w,b) = -C\sum_{i=1}^{N} t^{(i)} \mathbb{1}_{\{t^{(i)}(w^T x^{(i)} + b) < 1\}}$$

For $\overset{\text{all } (i)}{\forall}\, t^{(i)}(w^T x^{(i)} + b) = 1,$

$$\nabla_w^- E(w,b) = \lim_{w \to w^{*-}} w - \frac{1 - w^{(i)}(w^T x^{(i)} + b)}{w - w^*} = w - C\sum_{i=1}^{N} t^{(i)}x^{(i)} \qquad \frac{\partial E(w,b)^-}{\partial b} = \lim_{w \to w^{*-}} \frac{1 - w^{(i)}(w^T x^{(i)} + b)}{b - b^*} = -C\sum_{i=1}^{N} t^{(i)}.$$

$$\nabla_w^+ E(w,b) = \lim_{w \to w^{*+}} w - \frac{0 - 0}{w - w^*} = w. \qquad \frac{\partial E(w,b)^+}{\partial b} = \lim_{w \to w^{*+}} \frac{1 - w^{(i)}(w^T x^{(i)} + b)}{b - b^*} = 0$$

For $\exists$ some i s.t $t^{(i)}(w^T x^{(i)} + b) = 1$

while other $j \neq i \in [1, N]$ s.t $t^{(j)}(w^T x^{(j)} + b) \neq 1,$

we have

$$\nabla_w^- E(w,b) = w - C\sum_{i=1}^{N} t^{(i)}x^{(i)} \mathbb{1}_{\{1 - t^{(i)}(w^T x^{(i)} + b) \geq 0\}}$$

$$\nabla_w^+ E(w,b) = w - C\sum_{j \neq i} t^{(j)}x^{(j)} \mathbb{1}_{\{1 - t^{(j)}(w^T x^{(j)} + b) > 0\}}$$

$$\frac{\partial^- E(w,b)}{\partial b} = -C\sum_{i=1}^{N} t^{(i)}$$

$$\frac{\partial^+ E(w,b)}{\partial b} = -C\sum_{j \neq i} t^{(j)}.$$

(d)

```
import numpy as np
%matplotlib inline
from matplotlib import pyplot as plt
#import data
traindata=np.loadtxt("digits_training_data.csv",delimiter=",")
trainlabel=np.loadtxt("digits_training_labels.csv",delimiter=",")
#)1 (d)
c=3
ita=0.001
for i in range(trainlabel.size):
    if trainlabel[i]==4:
        trainlabel[i]=1
    else:
        trainlabel[i]=-1
w1=np.zeros(traindata[1,:].size)
b1=0
N1=np.arange(100)
accurate1=np.zeros(100)
m=1
for j in range(1,101):
    wgrab1=w1
    alpha=np.float(ita)/(1+j*(np.float(ita)))
    for k in range(trainlabel.size):
        wgrab1=wgrab1-c*((trainlabel[k]*traindata[k,:]) if trainlabel[k]*(np.dot(traindata[k,:],w1.T)+b1)<1 else 0)
        bgrab1=-c*(trainlabel[k] if trainlabel[k]*(np.dot(traindata[k,:],w1.T)+b1)<1 else 0)
    w1=(w1-alpha*wgrab1)
    b1=(b1-alpha*bgrab1)
    a=0
    for k in range(trainlabel.size):
        if trainlabel[k]*(np.dot(traindata[k,:],w1.T)+b1)>=1:
            a=a+1
        else:
            a=a
    accurate1[j-1]=np.float(a)/np.float(trainlabel.size)
plt.savefig('plot1')
```

We get the plot as:

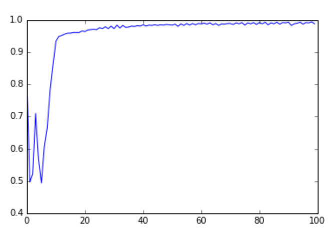

Figure 1: Iteration VS Accuracy

(f)
```
#(f)
w2=np.zeros(traindata[1,:].size)
b2=0
N=np.arange(100)
accurate2=np.zeros(100)
m=1
for j in range(1,101):
    alpha=np.float(ita)/(1+j*(np.float(ita)))
    for k in np.random.permutation(trainlabel.size):
        wgrab2=w2/(trainlabel.size)
        wgrab2=wgrab2-c*((trainlabel[k]*traindata[k,:]) if trainlabel[k]*(np.dot(traindata[k,:],w2.T
)+b2)<1 else 0)
        bgrab2=-c*(trainlabel[k] if trainlabel[k]*(np.dot(traindata[k,:],w2.T)+b2)<1 else 0)
        w2=(w2-alpha*wgrab2)
        b2=(b2-alpha*bgrab2)
    a=0
    for u in range(trainlabel.size):
        if trainlabel[u]*(np.dot(traindata[u,:],w2.T)+b2)>=1:
            a=a+1
        else:
            a=a
    accurate2[j-1]=np.float(a)/np.float(trainlabel.size)
plt.axis([0,100,0.3,1.3])
plt.plot(N,accurate2,label='Batch Gradient Descent')
plt.plot(N,accurate1,label='Stochastic Gradient Descent')
plt.legend()
plt.savefig('plot2')
```

We get the plot as:

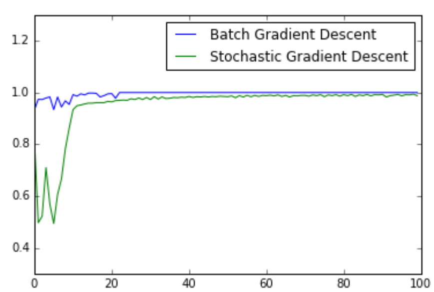

Figure 2: Iteration VS Accuracy

(g) We can conclude from the plot that Stochastic Gradient Descent convergent fast than Batch Gradient Descent. Also, we can see that for each iteration, w only update once with Batch Gradient Descent while w update N times in one iteration with Stochastic Gradient Descent.

e). Similarly as c, we have

For $1 - t^{(i)}(w^T x^{(i)} + b) \neq 0$

$$\nabla_w E^{(i)}(w,b) = \frac{1}{N} w - C t^{(i)} x^{(i)} \mathbb{1}_{\{1 - t^{(i)}(w^T x^{(i)} + b) > 0\}}$$

$$\frac{\partial E^{(i)}(w,b)}{\partial b} = -C t^{(i)} \mathbb{1}_{\{1 - t^{(i)}(w^T x^{(i)} + b) > 0\}}$$

For $1 - t^{(i)}(w^T x^{(i)} + b) = 0$,

$$\nabla_w E^{(i)}(w,b)^- = \frac{1}{N} w - C t^{(i)} x^{(i)}$$

$$\nabla_w E^{(i)}(w,b)^+ = \frac{w}{N}$$

$$\frac{\partial E^{(i)}(w,b)^-}{\partial b} = -C t^{(i)}$$

$$\frac{\partial E^{(i)}(w,b)^+}{\partial b} = 0$$

h. $\min \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i$

s.t $t_i(w^T x_i + b) \geq 1 - \xi_i$

$\xi_i \geq 0.$

$$\Rightarrow \mathcal{L} = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \alpha_i (1 - \xi_i - t_i(w^T x_i + b)) - \sum_{i=1}^{n} \beta_i \xi_i$$

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = w - \sum_{i=1}^{n} \alpha_i t_i x_i = 0 \\ \frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{n} \alpha_i t_i = 0 \\ \frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \beta_i = 0. \end{cases} \Rightarrow \begin{cases} w = \sum_{i=1}^{n} \alpha_i t_i x_i \\ \sum_{i=1}^{n} \alpha_i t_i = 0 \\ C = \alpha_i + \beta_i \end{cases}$$

$$\Rightarrow \mathcal{L} = \frac{1}{2}\|\sum_i \alpha_i t_i x_i\|^2 + \sum_{i=1}^{n} (C - \alpha_i - \beta_i)\xi_i + \sum_{i=1}^{n} \alpha_i \left(1 - t_i\left(\left(\sum_{i=1}^{n} \alpha_i t_i x_i\right)^T x_i + b\right)\right)$$

$$= \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j x_i^T x_j + 0 + \sum_{i=1}^{n} \alpha_i + \sum_{i=1}^{n} \left(-\alpha_i t_i \left(\sum_{j=1}^{n} \alpha_j t_j x_j\right)^T x_i\right) - \underbrace{b \sum_{i=1}^{n} t_i \alpha_i}_{0}$$

$$= \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j x_i^T x_j - \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \alpha_j t_j x_j^T\right)(\alpha_i t_i x_i) + \sum_{i=1}^{n} \alpha_i$$

$$= -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i$$

I

$\Rightarrow$ The dual problem is

$$\mathcal{L}_D = -\frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i$$

s.t $\sum \alpha_i t_i = 0$

$0 \leq \alpha_i \leq C$

Since kenel represent the inner product of X_i, X_j,

We change $X^{(i)T}X^{(j)}$ into $k(x^{(i)}, x^{(j)})$

$$\Rightarrow \mathcal{L}_D(\alpha_i, \beta_i) = -\frac{1}{2}\sum_{i,j=1}^{N} \alpha_i \alpha_j t^{(i)} t^{(j)} k(x^{(i)}, x^{(j)}) + \sum_i \alpha_i$$

$$\text{s.t } \sum \alpha_i t_i = 0$$

$$0 \leq \alpha_i \leq C.$$

(i) First, let C=3,find an appropriate gamma to do soft-margin SVM with rbf kernel.

```
import numpy as np
%matplotlib inline
from matplotlib import pyplot as plt
#(i)
import sklearn
testdata=np.loadtxt("digits_test_data.csv",delimiter=",")
testlabel=np.loadtxt("digits_test_labels.csv",delimiter=",")
for i in range(testlabel.size):
    if testlabel[i]==4:
        testlabel[i]=1
    else:
        testlabel[i]=-1
gamma=np.logspace(-9.10,13,num=50)
accurate2=np.zeros(50)
for i in range(0,50):
    clf=sklearn.svm.SVC(kernel="rbf",gamma=gamma[i],C=3)
    clf.fit(traindata,trainlabel)
    pre=clf.predict(testdata)
    a=0
    for u in range(testlabel.size):
        if testlabel[u]==pre[u]:
            a=a+1
        else:
            a=a
    accurate2[i]=np.float(a)/np.float(testlabel.size)
np.argmax(accurate2)
gamma[6]
```

we get gamma=4.0375921650671599e-07.

```
import matplotlib.cm as cm
clf=svm.SVC(kernel="rbf",gamma=4.0375921650671599e-07,C=3)
clf.fit(traindata,trainlabel)
pret=clf.predict(traindata)
a=0
for u in range(trainlabel.size):
    if trainlabel[u]==pret[u]:
        a=a+1
    else:
        a=a
accuratet=np.float(a)/np.float(trainlabel.size)
pre=clf.predict(testdata)
a=0
for u in range(testlabel.size):
    if testlabel[u]==pre[u]:
        a=a+1
    else:
        a=a
accurate=np.float(a)/np.float(testlabel.size)
print accuratet,accurate
```

We get training accuracy=1 and test accuracy=0.986.

Plot 5 mistakes picture:

```python
diff=np.abs(testlabel-pre)
data = np.genfromtxt("digits_test_data.csv", delimiter=',')
N=np.zeros(7)
i=0
for k in range(0,500):
    if diff[k]!=0:
        N[i]=k
        i=i+1
    else:
        i=i
plt.imshow(data[N[1]].reshape((26,26)), interpolation="nearest", cmap=cm.Greys_r)
plt.savefig('wrong1')
plt.imshow(data[N[2]].reshape((26,26)), interpolation="nearest", cmap=cm.Greys_r)
plt.savefig('wrong2')
plt.imshow(data[N[3]].reshape((26,26)), interpolation="nearest", cmap=cm.Greys_r)
plt.savefig('wrong3')
plt.imshow(data[N[4]].reshape((26,26)), interpolation="nearest", cmap=cm.Greys_r)
plt.savefig('wrong4')
plt.imshow(data[N[5]].reshape((26,26)), interpolation="nearest", cmap=cm.Greys_r)
plt.savefig('wrong5')
```

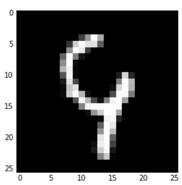

Figure 3: No.122(see 9 as 4)

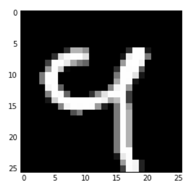

Figure 4: No.163(see 9 as 4)

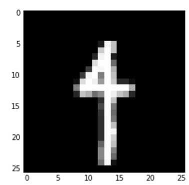

Figure 5: No.165(see 4 as 9)

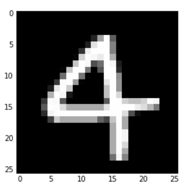

Figure 6: No.209(see 4 as 9)

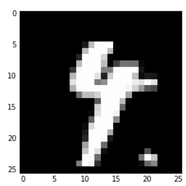

Figure 7: No.329(see 4 as 9)

(j) Apply LDA:

```
#(j)
import sklearn
testdata=np.loadtxt("digits_test_data.csv",delimiter=",")
testlabel=np.loadtxt("digits_test_labels.csv",delimiter=",")
for i in range(testlabel.size):
    if testlabel[i]==4:
        testlabel[i]=1
    else:
        testlabel[i]=-1
#import data
traindata=np.loadtxt("digits_training_data.csv",delimiter=",")
trainlabel=np.loadtxt("digits_training_labels.csv",delimiter=",")
for i in range(trainlabel.size):
    if trainlabel[i]==4:
        trainlabel[i]=1
    else:
        trainlabel[i]=-1
#get pi
pi1=(trainlabel==1).sum()
pi2=1000-pi1
pi11=np.float(pi1)/np.float(trainlabel.size)
pi22=1-pi11
mu1=np.zeros((traindata[1,:].size,1))
mu2=np.zeros((traindata[1,:].size,1))
sigma1=np.zeros((traindata[:,1].size,traindata[1,:].size))
for i in range(traindata[1,:].size):
    for j in range(trainlabel.size):
        if trainlabel[j]==1:
            mu1[i,0]=mu1[i,0]+(traindata[j,i])
        else:
            mu2[i,0]=mu2[i,0]+(traindata[j,i])
    mu1[i,0]=mu1[i,0]/pi1
    mu2[i,0]=mu2[i,0]/pi2
for j in range(trainlabel.size):
    if trainlabel[j]==1:
        sigma1[j:j+1,:]=traindata[j:j+1,:]-mu1.T
    else:
        sigma1[j:j+1,:]=traindata[j:j+1,:]-mu2.T
sigma=np.dot(sigma1.T,sigma1)/trainlabel.size
gamma1 = -0.5*mu1.T.dot(np.linalg.pinv(sigma)).dot(mu1)+np.log(pi11)
gamma2 = -0.5*mu2.T.dot(np.linalg.pinv(sigma)).dot(mu2)+np.log(pi22)
beta1=np.linalg.pinv(sigma).dot(mu1)
beta2=np.linalg.pinv(sigma).dot(mu2)
y=np.zeros(trainlabel.size)
ytest=np.zeros(testlabel.size)
yita1=np.zeros(trainlabel.size)
yita2=np.zeros(trainlabel.size)
yitat1=np.zeros(testlabel.size)
yitat2=np.zeros(testlabel.size)
for i in range(trainlabel.size):
    yita1[i]=np.exp(np.dot(beta1.T,traindata[i,:])+gamma1)
    yita2[i]=np.exp(np.dot(beta2.T,traindata[i,:])+gamma2)
for i in range(testlabel.size):
```

```
    yitat1[i]=np.exp(np.dot(beta1.T,testdata[i,:])+gamma1)
    yitat2[i]=np.exp(np.dot(beta2.T,testdata[i,:])+gamma2)
a=0
b=0
for i in range(trainlabel.size):
    p1=np.float(yita1[i])/np.float(yita1[i]+yita2[i])
    p2=np.float(yita2[i])/np.float(yita1[i]+yita2[i])
    if p1>p2:
        y[i]=1
        if y[i]==trainlabel[i]:
            a=a+1
        else:
            a=a
    else:
        y[i]=-1
        if y[i]==trainlabel[i]:
            a=a+1
        else:
            a=a
#test data
for i in range(testlabel.size):
    p1=yitat1[i]/np.float(yitat1[i]+yitat2[i])
    p2=yitat2[i]/np.float(yitat1[i]+yitat2[i])
    if p1>p2:
        ytest[i]=1
        if ytest[i]==testlabel[i]:
            b=b+1
        else:
            b=b
    else:
        ytest[i]=-1
        if ytest[i]==testlabel[i]:
            b=b+1
        else:
            b=b
accuratey=np.float(a)/np.float(trainlabel.size)
accurateyt=np.float(b)/np.float(testlabel.size)
print accuratey
print accurateyt
```

We get that training accuray=0.997, test accuracy=0.894;

Plot 5 mistakes picture:

```
diff=np.abs(testlabel-ytest)
data = np.genfromtxt("digits_test_data.csv", delimiter=',')
N=np.zeros(53)
i=0
for k in range(0,500):
    if diff[k]!=0:
        N[i]=k
        i=i+1
    else:
        i=i
plt.imshow(data[N[2]].reshape((26,26)), interpolation="nearest", cmap=cm.Greys_r)
```

```
plt.savefig('wrong11')
plt.imshow(data[N[5]].reshape((26,26)), interpolation="nearest", cmap=cm.Greys_r)
plt.savefig('wrong12')
plt.imshow(data[N[6]].reshape((26,26)), interpolation="nearest", cmap=cm.Greys_r)
plt.savefig('wrong13')
plt.imshow(data[N[7]].reshape((26,26)), interpolation="nearest", cmap=cm.Greys_r)
plt.savefig('wrong14')
plt.imshow(data[N[10]].reshape((26,26)), interpolation="nearest", cmap=cm.Greys_r)
plt.savefig('wrong15')
```

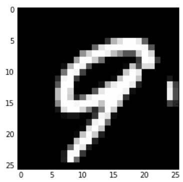

Figure 8: No.21(see 9 as 4)

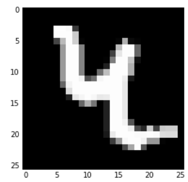

Figure 9: No.34(see 4 as 9)

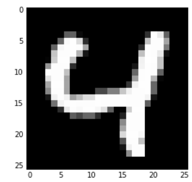

Figure 10: No.43(see 4 as 9)

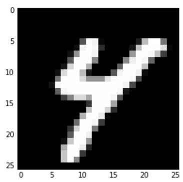

Figure 11: No.50(see 4 as 9)

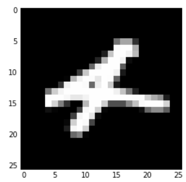

Figure 12: No.89(see 4 as 9)

There is little different between LDA and SVM.SVM perform better in this case.

2. Open Kaggle Challenge

```
import numpy as np
%matplotlib inline
from matplotlib import pyplot as plt
from sklearn import svm
from matplotlib import pyplot
import matplotlib as mpl
trainLabels = np.loadtxt('trainingLabels.gz', dtype=np.uint8, delimiter=',')
trainData = np.loadtxt('trainingData.gz', dtype=np.uint8, delimiter=',')
testData = np.loadtxt('testData.gz', dtype=np.uint8, delimiter=',')
clf=svm.SVC(kernel="rbf",gamma=4.0375921650671599e-07,C=3)
clf.fit(trainData,trainLabels)
pre=clf.predict(testData)
import csv
fl = open('test_result.csv', 'w')
writer = csv.writer(fl,lineterminator='\n')
#writer.writerow(['id', 'Strength']) #if needed
for values in pre:
    writer.writerow([values])
fl.close()
```

3. Constructing Kernels

3).

a). $k(u,v) = (\langle u,v\rangle + 1)^4$

$= \langle u,v\rangle^4 + 4\langle u,v\rangle^3 + 6\langle u,v\rangle^2 + 4\langle u,v\rangle + 1$

For $\langle u,v\rangle^4$, since $d=3$ $u=(u_1,u_2,u_3)$, $v=(v_1,v_2,v_3)$

we have

$$\langle u,v\rangle^4 = \left(\sum_{i=1}^{3} u_i v_i\right)^4$$

$$= \sum_{\substack{j_1,j_2,j_3 \\ \sum_k j_k = 4}} \binom{4}{j_1,j_2,j_3} (u_1v_1)^{j_1} (u_2v_2)^{j_2} (u_3v_3)^{j_3}$$

$$\phi_4(u) = \Big(\binom{4}{0,0,4}^{1/2} u_3^4,\ \binom{4}{4,0,0}^{1/2} u_1^4,\ \binom{4}{0,4,0}^{1/2} u_2^4,\ \binom{4}{1,0,3}^{1/2} u_3^3 u_1,\ \binom{4}{1,3,0}^{1/2} u_1 u_2^3,\ \binom{4}{0,1,3}^{1/2} u_2 u_3^3,\ \binom{4}{0,3,1}^{1/2} u_2^3 u_3,\ \binom{4}{3,0,1}^{1/2} u_1^3 u_3,$$
$$\binom{4}{3,1,0}^{1/2} u_1^3 u_2,\ \binom{4}{2,1,1}^{1/2} u_1^2 u_2 u_3,\ \binom{4}{1,2,1}^{1/2} u_1 u_2^2 u_3,\ \binom{4}{1,1,2}^{1/2} u_1 u_2 u_3^2,\ \binom{4}{2,2,0}^{1/2} u_2^2 u_3^2,\ \binom{4}{2,0,2}^{1/2} u_1^2 u_3^2,\ \binom{4}{0,2,2}^{1/2} u_2^2 u_3^2\Big)$$

$$= (u_1^4, u_2^4, u_3^4, 2u_1^3u_3, 2u_1^3u_2, 2u_2^3u_1, 2u_2^3u_3, 2u_3^3u_1, 2u_3^3u_2, 2\sqrt{3}u_1^2u_2u_3, 2\sqrt{3}u_1u_2^2u_3, 2\sqrt{3}u_1u_2u_3^2, \sqrt{6}u_1^2u_2^2, \sqrt{6}u_1^2u_3^2, \sqrt{6}u_2^2u_3^2)$$

$$\langle u,v\rangle^3 = \left(\sum_{i=1}^{3} u_i v_i\right)^3 = \sum_{\substack{j_1,j_2,j_3 \\ \sum_k j_k = 3}} \binom{3}{j_1,j_2,j_3} (u_1v_1)^{j_1} (u_2v_2)^{j_2} (u_3v_3)^{j_3}$$

$$\phi_3(u) = \Big(\binom{3}{0,0,3}^{1/2} u_3^3,\ \binom{3}{0,3,0}^{1/2} u_2^3,\ \binom{3}{3,0,0}^{1/2} u_1^3,\ \binom{3}{2,1,0}^{1/2} u_1^2 u_2,\ \binom{3}{2,0,1}^{1/2} u_1^2 u_3,\ \binom{3}{0,2,1} u_2^2 u_3,\ \binom{3}{0,1,2} u_2 u_3^2,\ \binom{3}{1,0,2} u_1 u_3^2,$$
$$\binom{3}{1,2,0}^{1/2} u_1 u_2^2,\ \binom{3}{1,1,1}^{1/2} u_1 u_2 u_3\Big)$$

$$= (u_1^3, u_2^3, u_3^3, \sqrt{3}u_1^2u_2, \sqrt{3}u_1^2u_3, \sqrt{3}u_2^2u_3, \sqrt{3}u_2^2u_1, \sqrt{3}u_3^2u_1, \sqrt{3}u_3^2u_2, \sqrt{6}u_1u_2u_3)$$

$$\langle u,v\rangle^2 = \left(\sum_{i=1}^{3} u_i v_i\right)^2$$

$$= \sum_{\substack{j_1,j_2,j_3 \\ \sum_k j_k = 2}} \binom{2}{j_1,j_2,j_3} (u_1v_1)^{j_1} (u_2v_2)^{j_2} (u_3v_3)^{j_3}$$

$$\phi_2(u) = \Big(\binom{2}{2,0,0}^{1/2} u_1^2,\ \binom{2}{0,2,0} u_2^2,\ \binom{2}{0,0,2} u_3^2,\ \binom{2}{1,1,0} u_1u_2,\ \binom{2}{1,0,1} u_1u_3,\ \binom{2}{0,1,1} u_2u_3\Big)$$

$$= (u_1^2, u_2^2, u_3^2, \sqrt{2}u_1u_2, \sqrt{2}u_2u_3, \sqrt{2}u_1u_3)$$

$$\langle u,v\rangle = \sum_{i=1}^{3} u_i v_i$$

$$\phi_1(u) = (u_1, u_2, u_3)$$

Then, we have $\phi(u) = (\phi_4(u), 2\phi_3(u), \sqrt{6}\phi_2(u), 2\phi_1(u), 1)$

$\phi(v) = (\phi_4(v), \phi_3(v), \phi_2(v), \phi_1(v), 1)$

which satisfy $k(u,v) = \phi(u)^T \phi(v)$

For any d dimension u, & v, we can still get $\phi_4(u)$, $\phi_3(u)$, $\phi_2(u)$, $\phi_1(u)$, 1.

$<u,v>^4 = (\sum_{i=1}^{d} u_i v_i)^4 = \sum_{\substack{j_1 \cdots j_d \\ \Sigma_k j_k = 4}} \binom{4}{j_1 \cdots j_d} (u_1 v_1)^{j_1} \cdots (u_d v_d)^{j_d}$

$\phi_4(u) = \left(\ldots, \binom{4}{j_1 \cdots j_d}^{1/2} u_1^{j_1} u_d^{j_d}, \cdots \right)_{\Sigma j_k = 4}$

similarly

$\phi_3(u) = \left(\ldots, \binom{3}{j_1 \cdots j_d}^{1/2} u_1^{j_1} \cdots u_d^{j_d}, \cdots \right)_{\Sigma_k j_k = 3}$

$\phi_2(u) = \left(\ldots, \binom{2}{j_1 \cdots j_d}^{1/2} u_1^{j_1} \cdots u_d^{j_d}, \cdots \right)_{\Sigma_k j_k = 2}$

$\phi_1(u) = (u_1, \cdots, u_d)$

Then we have $\phi(u) = (\phi_4(u), \phi_3(u), \phi_2(u), \phi_1(u), 1)$

$k(u,v) = \phi(u)^T \phi(v)$.

b). i). we already know that k_1, k_2 are positive-definite kernel functions,

$\therefore$ $\forall x \in \mathbb{R}^d, \neq 0$, we have for it's gram Matrix K_1^M, K_2^M, $x^T(K_1^M)x > 0$, $x^T(K_2^M)x > 0$.

$x^T(K^M)x = x^T(K_1^M + K_2^M)x > 0$ $\Rightarrow$ the gram Matrix of k, is positive-def

Set $k_1(x,z) = \phi_1^T(x)\phi_1(z)$

$k_2(x,z) = \phi_2^T(x)\phi_2(z)$

$k = \phi_1^T(x)\phi_1(z) + \phi_2^T(x)\phi_2(z) = <(\phi_1(x), \phi_2(x)), (\phi_1(z), \phi_2(z))>$

is also positive-definite kernel functions

ii). Counterexample:

Simply let $k_1(x,z) = k_2(x,z)$, we have

$k = k_1(x,z) - k_2(x,z) = 0$

$\therefore$ the gram matrix of k is a $D \times D$ matrix with all value be 0 which is not a positive-definite gram matrix.

$\Rightarrow$ k is not a positive definite kernel function.

iii) since $a \in \mathbb{R}^+$, $x^T(K_1^M)x > 0$ for the gram matrix of k_1

we have $x^T K^M x = x^T a K^M x = a(x^T K^M x) > 0$, which is also positive-definite matrix.

$k = <\sqrt{a}\phi_1(x), \sqrt{a}\phi_1(z)>$ is a positive-definite kernel function.

(iv). $k(x,z) = k_1(x,z)\,k_2(x,z)$

$$= \phi_1^T(x)\phi_1(z)\,\phi_2^T(x)\phi_2(z) = \sum_{i=1}^{D}\phi_{1i}(x)\phi_{1i}(z)\sum_{i=1}^{D}\phi_{2i}(x)\phi_{2i}(z) = \sum_{i,j=1}^{D}\phi_{1i}(x)\phi_{1i}(z)\phi_{2j}(x)\phi_{2j}(z)$$

$$= \sum_{i,j=1}^{D}\phi_{1i}(x)\phi_{2j}(x)\phi_{1i}(z)\phi_{2j}(z)$$

$\therefore k(x,z) = \langle(\phi_{11}(x)\phi_{21}(x), \phi_{11}(x)\phi_{22}(x), \cdots \phi_{11}(x)\phi_{2D}(x), \cdots, \phi_{1D}(x)\phi_{21}(x), \ldots, \phi_{1D}(x)\phi_{2D}(x)),$

$(\phi_{11}(z)\phi_{21}(z), \phi_{11}(z)\phi_{22}(z)\cdots, \phi_{11}(x)\phi_{2D}(z), \cdots, \phi_{1D}(z)\phi_{21}(z), \ldots, \phi_{1D}(z)\phi_{2D}(z))\rangle$

which is a positive-definite kernel function

(v). Counterexample:

set $f(x): \mathbb{R}^D \to 0$, which means for $\forall x \in \mathbb{R}^D$, we have $f(x)=0$.

$\Rightarrow$. $k(x,z) = f(x)f(z) = 0$ for $\forall x, z$,

the gram matrix of k is a matrix with all value $=0$. which is not positive-definite matrix.

$\Rightarrow$ k is not a positive-definite kernel function at this time.

(vi). $k(x,z) = p(k_1(x,z)) = \sum_{i=1}^{N} a_i (k_1(x,z))^i$ where $a_i > 0$

we only need to prove $k_1(x,z)^i$ is a positive-definit kernel, then due from i) & iii),

$k(x,z)$ is also a positive-definit kernel function.

as we proved in (iv), $k(x,z) = k_1(x,z)k_2(x,z)$ is also a postive-definit kernel function.

$k(x,z) = (k_1(x,z))^2$ also satisfy if let $k_2 = k_1$

similarly, $k(x,z) = (k_1(x,z))^i$ is positive-def for $\forall i \geq 1$

$\Rightarrow$. $k(x,z) = p(k_1(x,z))$ is positive-definit kernel function.

(vii). $k(x,z) = \exp(-\frac{\|x-z\|^2}{2\sigma^2}) = \exp(-\frac{x^Tx}{2\sigma^2})\exp(\frac{x^Tz}{\sigma^2})\exp(-\frac{z^Tz}{2\sigma^2})$

it can be write as $k(x,z) = f(x)\exp(\frac{x^Tz}{\sigma^2})f(z)$ where $f(x) = \exp(-\frac{x^Tx}{2\sigma^2})$, $f(z) = \exp(-\frac{z^Tz}{2\sigma^2})$

$\exp(\frac{x^Tz}{\sigma^2}) = \sum_{n=1}^{\infty}\frac{(\frac{x^Tz}{\sigma^2})^n}{n!}$ by Taylor expansion.

Set $k_0(x,z) = x^Tz = \langle x, z\rangle$ which is a kernel function.

due from (iii) $\frac{k_0(x,z)}{\sigma^2}$ is a kernel function, due from (vi), $\sum_{n=1}^{\infty}\frac{(\frac{x^Tz}{\sigma^2})^n}{n!}$ is also a kernel function,

with infinit term, we have $\sum_{n=1}^{\infty}\frac{(\frac{x^Tz}{\sigma^2})^n}{n!}$ can be write as $\langle\phi(x), \phi(z)\rangle$ where $\phi(x), \phi(z)$ are infinite dimension

$\therefore$ $k(x,z) = f(x)\langle\phi(x), \phi(z)\rangle f(z) = \langle f(x)\phi(x), f(z)\phi(z)\rangle = \langle\phi'(x), \phi'(z)\rangle$. Q.E.D

4)

a).

$$P^{-1} - P^{-1}Q(R^{-1}+SP^{-1}Q)^{-1}SP^{-1} \qquad (*)$$

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

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

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

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

with $\hat{w} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T t$

set $P=\lambda I$, $Q=\Phi^T$, $R=I$ $S=\Phi$, we have

$$(\lambda I + \Phi^T\Phi)^{-1} = (\lambda I)^{-1}\Phi^T((I)^{-1} + \Phi(\lambda I)^{-1}\Phi^T)^{-1}(I)^{-1}(\Phi^T)^{-1}$$

$$(\lambda I + \Phi^T\Phi)^{-1} = \frac{1}{\lambda}\Phi^T(I + \frac{1}{\lambda}\Phi\Phi^T)^{-1}(\Phi^T)^{-1}$$

$$(\lambda I + \Phi^T\Phi)^{-1}\Phi^T = \Phi^T(\lambda I + \Phi\Phi^T)^{-1}$$

$\Rightarrow \hat{w} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T t = \Phi^T(\lambda I + \Phi\Phi^T)^{-1}t = \Phi^T a$ where $a = (\lambda I + \Phi\Phi^T)^{-1}t$

$\hat{f}(x) = w^T\phi(x) = (\Phi^T a)^T\phi(x) = a^T\Phi\phi(x) = a^T k(x)$ where $k(x) = \Phi\phi(x) = [k(x_1,x), \cdots, k(x_N,x)]^T$

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

$$= w^T\Phi^T\Phi w - 2t^T\Phi w + t^T t + \lambda w^T w$$

$$= a^T\Phi\Phi^T\Phi\Phi^T a - 2t^T\Phi\Phi^T a + t^T t + \lambda a^T\Phi\Phi^T a$$

$$= a^T K K a - 2t^T K a + t^T t + \lambda a^T K a$$

4. Kernelized Ridge Regression

(b)
```
import numpy as np
#import data
data=np.loadtxt("steel_composition_train.csv",delimiter=",", skiprows=1,usecols=(1,2,3,4,5,6,7,8,9))
X=data[:,:-1] #X is a feature set
Y=data[:,-1] #Y is an array with target value
#normalized
X=(X-np.mean(X,0))/np.std(X,0)
#get y with kernel
k0=np.dot(X,X.T)
I=np.identity(k0[:,1].size)
E=np.zeros(4)
for i in range(3):
    k=np.power((k0+1),i+2)
    a=np.dot(np.linalg.inv((I+k)),Y)
    Ypre=np.dot(a.T,k)
    E[i]=np.dot(Ypre,Ypre.T)-2*np.dot(Y.T,Ypre.T)+np.dot(Y.T,Y)+np.dot(Ypre,a)
k4=np.zeros((k0[:,1].size,k0[:,1].size))
for i in range(k0[:,1].size):
    for j in range(k0[:,1].size):
        k4[i,j]=np.exp(-np.power(np.linalg.norm(X[i,:]-X[j,:]),2)/2)
a=np.dot(np.linalg.inv((I+k4)),Y)
Ypre=np.dot(a.T,k4)
E[3]=np.dot(Ypre,Ypre.T)-2*np.dot(Y.T,Ypre.T)+np.dot(Y.T,Y)+np.dot(Ypre,a)
RMSE=np.sqrt(E/Y.size)
```

We get RMSE with four different kernel be:

kernel	poly with degree 2	poly with degree 3	poly with degree 4	Gaussian
RMSE	7.41955281	4.83144123	2.96214783	13.18899638