Machine Learning Jiecao Chen ID:4746311
HOMEWORK #2

**Question 1**

**a.** Given $X$, take $y = 1$ and $W_2$ such that $W_2^T X = 0$, for any $0 \leq t \leq 1$:

$$E(tW_1 + (1-t)W_2) = (1 - \hat{y}(tW_1)) \tag{1}$$

$$tE(W_1)+(1-t)E(W_2) = tE(W_1)+(1-t)(1-\frac{1}{1+exp(-0)})^2 = tE(W_1)+(1-t)(\frac{1}{2})^2 \tag{2}$$

now let $W_1^T X = -N$ where $N$ is a very large number, we have

$$(1) \approx 1$$

and

$$(2) \approx t + \frac{1-t}{4}$$

Take $t = 0.5$ gives

$$(1) > (2)$$

This example shows that $E(W)$ is not necessarily a convex function of $W$

**b.** Take $y = 1$, we have

$$E(W) = log(1 + exp(-W^T X)) = log(1 + exp(-\sum_{i=1}^{N} w_i x_i)) \tag{3}$$

To show (3) a convex function of $W$, we can simply show the corresponding Hessian matrix is positive semi-definite:

Denote $k = 1 + exp(-W^T X)$

$$\frac{\partial E(W)}{\partial w_i} = \frac{-x_i exp(-W^T X)}{k} \tag{4}$$

$$H_{ij} = \frac{\partial^2 E(W)}{\partial w_i \partial w_j} = exp(-W^T X)\frac{x_i x_j}{k^2} \tag{5}$$

which means the Hessian matrix can be represented as

$$H = exp(-W^T X) A^T A$$

where

$$A = (w_1 \ w_2 \ \dots \ w_N)$$

thus $H$ is positive semi-deinite, we proved what we need.

**c.** This could be directly seen when we note that no matter $y = 1$ or $y = 0$, we have

$$E(W) = log(1 + exp(-W^T X))$$

We can definte the loss function as

$$loss(W) = \sum_{i=1}^{N} log(1 + exp(-W^T X_i))$$

it's a logistics regression problem.

## Question 2

**a.** it's trivial to see that

$$k = \sum_{j=1}^{m} w_j k_j$$

is symmetric
when $k_j$ is symmetric for all $1 \leq j \leq m$.
Given any vector $x$

$$x^T k x = \sum_{j=1}^{m} w_j x^T k_j x \tag{6}$$

as for any $j$, $w_j \geq 0$, $k_j$ is positive semi-definite, thus

$$w_j x^T k_j x \geq 0$$

we have

$$(6) \geq 0$$

Which implies $k$ is positive semi-definite, thus is a valid kernel.

**b.** It's trival to see that $K$ is symmetric let $M^{-1} = diag(e^{x_1^2}, e^{x_2^2}, \ldots, e^{x_n^2})$ then we have

$$(M^{-1})^T K M^{-1} = H = A^T A$$

where $H_{ij} = e^{2x_i x_j}$ and $A = (e^{\sqrt{2}x_1}\ e^{\sqrt{2}x_2}\ \ldots\ e^{\sqrt{2}x_n})$ which implies that $K = M^T A^T A M$, thus $K$ is positive semi-definite. So $K$ is a valid kernel.

## Question 3

### Fisher's Linear Discriminant.

*Idea.* The main idea of **Fisher's Linear Discriminant** is to project the features from a high-dimention space to a low-dimention space when maximize the distinctions between different classes. And Fisher set the criterion as

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

the value of $J(w)$ indicates the distance between different classes. To get the optimal $w$, we just need to select the $D'$ largest eigenvalues of $S_W^{-1}S_B$ where $D'$ is the dimention of the projected feature space. In the code **diagFisher**, we simply set $S_w = I$.

*Experiment results.* In our implementatin, the dimention of projected feature space $D'$ was set as 3 for data "Iris.csv" and 9 for "Wine.csv", and in the projected space, **Gaussian Generative Model** will be used to do classification

==========

```
$ python diagFisher.py 'Iris.csv' 10

Data:  Iris.csv
Error rate for cross_validation: 0.0333333333333

$ python Fisher.py 'Iris.csv' 10

Data:  Iris.csv
Error rate for cross_validation: 0.02

$ python diagFisher.py 'Wine.csv' 10

Data:  Wine.csv
Error rate for cross_validation: 0.0588235294118


$ python Fisher.py 'Wine.csv' 10

Data:  Wine.csv
Error rate for cross_validation: 0.0470588235294
```

==========

we can see that both models are quite accurate, and the performence of *Fisher.py* is slightly better.

**Least squares linear discriminant.**

**Idea.** This model is quite simple and we just need to minimize the following criterion

$$E(W) = \frac{1}{2}Tr\big\{(Y - XW)^T(Y - XW)\big\}$$

The optimal solution would be

$$W = (X^TX)^{-1}X^TY$$

where $X$ is the matrix of features and $Y$ is the vector of labels.