HW5-2 report

I use the read_csv function to read the dataset

```
def read_csv(path,len,label=False):
    with open(path, newline='') as csvfile:
      rows = csv.reader(csvfile)
      if not label:
        arr=np.zeros((len,784))
      else:
        arr=np.zeros(len)
      for i,row in enumerate(rows):
        arr[i]=np.asarray(row)
    return arr
```

```
x_train=read_csv('X_train.csv',5000)
y_train=read_csv('Y_train.csv',5000,True)
x_test=read_csv('X_test.csv',2500)
y_test=read_csv('Y_test.csv',2500,True)
```

- part1

a. code with detailed explanations

The svm parameter “-t” is control the kernel type. 0 is linear, 1 is polynomial, 2 is RBF.

```
params={'Linear: ':'0','Polynomial: ':'1','RBF':'2'}
acc=[]
for key,val in params.items():
    prob=svm_problem(y_train,x_train)
    param=svm_parameter('-t '+val)
    model=svm_train(prob,param)
    p_label,p_acc,p_val=svm_predict(y_test,x_test,model)
    acc.append(p_acc)

print('\nAccuracy')
i=0
for key in params:
    print(key,acc[i])
    i+=1
```

The formula of different kernel functions:

$$\text{Linear kernel: } k(x,y) = < x, y >$$

$$\text{Polynomial kernel: } k(x,y) = (< x, y > + c)^d$$

$$\text{Gaussian Radial Basis Function kernel (RBF): } k(x,y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$$

b. experiments settings and results

all using the default parameter

```
Accuracy
Linear:  (95.08, 0.1404, 0.931149802516624)
Polynomial:  (34.68, 2.6212, 0.14887572191533946)
RBF (95.32000000000001, 0.1492, 0.9271864783823697)
```

The RBF has the best accuracy 95.32%

- part2

a. code with detailed explanations

I do the grid search for finding parameters of RBF kernel because the RBF kernel has the best performance in part1. And I do the 3-folds cross validation by setting svm parameter “-v 3”

```
C=gamma=['0.5','0.3','0.1','1','3','5']
best_acc=0
for c in C:
    for g in gamma:
        prob=svm_problem(y_train,x_train)
        param=svm_parameter('-v 3 -t 2 -c '+c+' -g '+g)
        acc=svm_train(prob,param)
        print(acc)
        if acc>best_acc:
            best_acc=acc
            optimal_c=c
            optimal_g=g

print('The parameter C is: %s\nThe parameter gamma is: %s\nThe accuracy is: %f' % (optimal_c,optimal_g,best_acc))
```

b. experiments settings and results

```
The parameter C is: 5
The parameter gamma is: 0.1
The accuracy is: 91.660000
```

- part3

a. code with detailed explanations

If the programmer wants to use the user-defined kernel in libsvm, the programmer needs to create the precomputed kernel. This precomputed kernel is linear+RBF. The function in image shows how to create the precomputed kernel.

The output of precomputed kernel must include sample serial number as first column. Hence, I use the np.hstack to combine the serial nuber and kernel.

```
def kernel(Xn,Xm,gamma=2e-3):
    linear=Xn@Xm.T
    RBF=np.exp(-gamma*cdist(Xn,Xm,'sqeuclidean'))
    precomputed_kernel=linear+RBF
    precomputed_kernel=np.hstack((np.arange(1,len(Xn)+1).reshape(-1,1),precomputed_kernel))
    return precomputed_kernel
```

After I had created the precomputed kernel, I can use it to do the svm train and predict. In other word, the svm parameter need to set “-t 4”. It means use the precomputed kernel.

```
precomputed_kernel=kernel(x_train,x_train)
prob=svm_problem(y_train,precomputed_kernel,isKernel=True)
param=svm_parameter('-t 4')
model=svm_train(prob,param)

precomputed_kernel=kernel(x_test,x_train)
p_label,p_acc,p_val=svm_predict(y_test,precomputed_kernel,model)
print('\n\n',p_acc)
```

b. experiments settings and results

```
Accuracy = 95.08% (2377/2500) (classification)
```

- **observations and discussion**
  I try the different user-defined kernel function (Polynomial + RBF)
  From part1 and this experiment, I find the polynomial kernel function may not suitable in this case.

  Linear + RBF

```
Accuracy = 95.08% (2377/2500) (classification)
```

  Polynomial + RBF

```
Accuracy = 88.88% (2222/2500) (classification)
```