Deniz Aytemiz
denizaytemiz@vt.edu
no: 906470552

ECE 5424/CS 5824 - Advanced Machine Learning - Problem Set 2

Question 1.1:

Question 1:

$$\min_{\lambda, \xi} \|\lambda\|^2 + C \sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad \left.\begin{aligned} y_i(\lambda x_i + \lambda_0) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \end{aligned}\right\} \text{ for } i = 1, \dots, n$$

checking the constraints in above optimization problem.

$$\xi_i \geq \underbrace{\boxed{1 - y_i(\lambda x_i + \lambda_0)}}_{\text{lower bond on } \xi_i}$$

$$\xi_i \geq 0$$

hence $\xi_i = (1 - y_i(\lambda x_i + \lambda_0))_+ = \max(0, 1 - y_i(\lambda x_i + \lambda_0))$

$$\xi_i = \begin{cases} 1 - y_i(\lambda x_i + \lambda_0) & \text{if } \quad 1 - y_i(\lambda x_i + \lambda_0) > 0 \\ 0 & \text{otherwise} \end{cases}$$

say $t = y_i(\lambda x_i + \lambda_0)$, $\ell_{hinge}(t)$ is

$$\ell_{hinge}(y_i(\lambda x_i + \lambda_0)) = \begin{cases} 1 - y_i(\lambda x_i + \lambda_0) & \text{if } y_i(\lambda x_i + \lambda_0) < 1 \\ 0 & \text{otherwise} \end{cases}$$

CS Scanned with CamScanner

Hence, $\xi_i = \ell_{hinge}(y_i(\lambda x_i + \lambda_0))$ can be written, and it automatically includes the constraints $y_i(\lambda x_i + \lambda_0) \geq 1 - \xi_i$ & $\xi_i \geq 0$ for $i = 1, \dots, n$ in the primal problem. We can directly plug in this term to objective function and drop the constraints, hence we end up with:

$$\min_{\lambda} \|\lambda\|^2 + C \sum_{i=1}^{n} \ell_{hinge}(y_i(\lambda x_i + \lambda_0))$$

CS Scanned with CamScanner

Deniz Aytemiz
denizaytemiz@vt.edu
no: 906470552

Question 1.2:

Dual formulation for soft-margin SVM:

$$\min_{\lambda,\lambda_0} \frac{1}{2}\|\lambda\|_2^2 + C\sum_{i=1}^{n} \xi_i$$

s.t.

$$\left.\begin{array}{l} 1-y_i(\lambda^T x_i+\lambda_0)-\xi_i < 0 \\ -\xi_i < 0 \end{array}\right\} \text{ for } i=1,\dots,n$$

we know $\mathcal{L}(\alpha,x) = f(x) + \sum_{i=1}^{n} \alpha_i . g_i(x)$, here

$$\mathcal{L}(\alpha,\beta,[\lambda\ \lambda_0]) = \frac{1}{2}\|\lambda\|_2^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i . (1-y_i(\lambda^T x_i+\lambda_0)-\xi_i)$$

$$+ \sum_{i=1}^{n}\beta_i . -\xi_i \quad , \quad \alpha_i, \beta_i \geq 0$$

additional dual variable

$$\Theta_D(\alpha,\beta) = \min_{\lambda,\lambda_0} \mathcal{L}([\lambda,\lambda_0],\alpha,\beta)$$

to find min of Lagrangian we can set the gradient to 0.

$$\frac{\nabla\mathcal{L}}{\nabla\lambda} = \lambda + \sum_{i=1}^{n} -\alpha_i . y_i . x_i = 0 \implies \lambda = \sum_{i=1}^{n}\alpha_i . y_i . x_i$$

$$\frac{\partial L}{\partial\lambda_0} = \sum_{i=1}^{n}\alpha_i . -y_i = 0 \implies \underbrace{\sum_{i=1}^{n}\alpha_i . y_i = 0}$$

λ, λ_0 optimal values defining Θ_D.

what if we evaluate α such that $\sum_{i=1}^{n}\alpha_i . y_i \neq 0$

if $\sum_{i=1}^{n}\alpha_i . y_i > 0$, we can take $\lambda_0 \to \infty$ here $\Theta_D \to -\infty$

since the goal of α is to maximize dual objective we will make sure the condition $\sum_{i=1}^{n}\alpha_i . y_i = 0$ is met.

CS Scanned with CamScanner

Deniz Aytemiz
denizaytemiz@vt.edu
no: 906470552

for $\lambda^* = \sum_{i=1}^{n} \alpha_i y_i x_i$, $\lambda_0^* = 0$

$$\Theta_D(\alpha) = \mathcal{L}([\lambda^*, \lambda_0^*], \alpha, \beta)$$

$$= \frac{1}{2}\|\lambda^*\|_2^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i(1 - y_i(\lambda^{*T}x_i) - \xi_i) - \sum_{i=1}^{n}\beta_i \xi_i$$

$$= \frac{1}{2}\|\lambda^*\|_2^2 + \lambda^{*T}\sum_{i=1}^{n}(-\alpha_i y_i x_i) + \sum_{i=1}^{n}\alpha_i - \xi_i + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\beta_i \xi_i$$

$$= -\frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{n}\alpha_i\alpha_k y_i y_k x_i^T x_k + \sum_{i=1}^{n}\alpha_i + \underbrace{\sum_{i=1}^{n}\alpha_i - \xi_i - \sum_{i=1}^{n}\beta_i \xi_i}_{\sum_{i=1}^{n}(-\alpha_i - \beta_i)\xi_i}$$

$\max_\alpha \Theta_D(\alpha)$ s.t $\alpha \geq 0$, $\sum_{i=1}^{n}\alpha_i y_i = 0$

$\beta \geq 0$

& $0 \leq \alpha_i \leq C$
$0 \leq \beta_i \leq C$

since the problem is maximization & $-\beta_i \xi_i$ will be negative it is best to choose $\beta_i = 0$.

here we can write:

$$\max_\alpha \Theta_D(\alpha) = \max_\alpha\left[-\frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{n}\alpha_i\alpha_k y_i y_k x_i^T x_k + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i \xi_i\right]$$

where $0 \leq \alpha_i \leq C$ &
$\sum_{i=1}^{n}\alpha_i y_i = 0$

Scanned with CamScanner

Deniz Aytemiz
denizaytemiz@vt.edu
no: 906470552

Question 2: Programming Assignment

Task 1:

```
## STUDENT: YOUR CODE STARTS HERE
# Task: Append '1' to the beginning of each vector.
# Hint: You can use data_features.toarray() to transform data_features into a numpy array
# The output should be a numpy array named data_mat
data_features = data_features.toarray()
n = data_features.shape[0] #n=3000
m = data_features.shape[1] #m=4500
data_mat = np.zeros((n, m+1))


for i in range(n):
  # copy the elements of the existing array into the data_mat's row, starting at index 1
  data_mat[i][1:] = data_features[i]

  # insert the value 1 at the beginning of the data_mat's first index
  data_mat[i][0] = 1


## STUDENT: CODE ENDS
print ('The updated size: ',data_mat.shape)
```

```
The updated size:  (3000, 4501)
```

The size of data_mat is (3000,4501)

Deniz Aytemiz
denizaytemiz@vt.edu
no: 906470552

Task 2:
Derivation of gradient of loss function with respect to weight array:

$$\tilde{\Theta}_t = \underset{\tilde{\Theta}}{\mathrm{argmin}}\, L(\tilde{\Theta}) = \underset{\tilde{\Theta}}{\mathrm{argmin}} \underbrace{\sum_{i=1}^{n} \ln\left(1+e^{-y_i \tilde{\Theta}_t^T \tilde{x}_i}\right)}$$

↓ we want to find $\tilde{\Theta}$ such that this summation is minimized.

$$\frac{\nabla L}{\nabla \tilde{\Theta}} = \frac{\nabla}{\nabla \tilde{\Theta}}\left(\sum_{i=1}^{n} \ln\left(1+e^{-y_i \tilde{\Theta}_t^T \tilde{x}_i}\right)\right)$$

$$\frac{\nabla L}{\nabla \tilde{\Theta}} = \sum_{i=1}^{n}\left(\frac{\nabla}{\nabla \tilde{\Theta}}\left(\ln\left(1+e^{-y_i \tilde{\Theta}_t^T \tilde{x}_i}\right)\right)\right)$$

first, $\dfrac{\partial L}{\partial \Theta_0} = \displaystyle\sum_{i=1}^{n} \frac{\nabla}{\nabla \Theta_0}\left(\ln\left(1+e^{-y_i \tilde{\Theta}_t^T \tilde{x}_i}\right)\right)$

say $u = 1 + e^{-y_i \tilde{\Theta}_t^T \tilde{x}_i}$

$$\frac{\partial\left(\ln\left(1+e^{y_i \tilde{\Theta}_t^T \tilde{x}_i}\right)\right)}{\partial \Theta_0} = \frac{\partial \ln(u)}{\partial u} \cdot \frac{\partial u}{\partial \Theta_0} \quad \text{by chain rule}$$

we know $\dfrac{\partial \ln(u)}{\partial u} = \dfrac{1}{u}$,

$$\frac{\partial u}{\partial \Theta_0} = \frac{\partial\left(1+e^{-y_i \tilde{\Theta}_t^T \tilde{x}_i}\right)}{\partial \Theta_0} = \frac{\partial\left(e^{-y_i \tilde{\Theta}_t^T \tilde{x}_i}\right)}{\partial \Theta_0}, \quad \text{say } -y_i \tilde{\Theta}_t^T \tilde{x}_i = z$$

$$\frac{\partial\left(e^{-y_i \Theta x}\right)}{\partial \Theta_0} = \frac{\partial e^{(z)}}{\partial z} \cdot \frac{\partial z}{\partial \Theta_0}, \quad \text{where } \frac{\partial e^{(z)}}{\partial z} = e^{z}$$

Deniz Aytemiz
denizaytemiz@vt.edu
no: 906470552

$$\frac{\partial z}{\partial \theta_0} = \frac{\partial(-y_i \tilde{\theta}_t^T \tilde{x}_i)}{\partial \theta_0},$$

since $-y_i \tilde{\theta}_t^T \tilde{x}_i$ has the form of : $y_i(\theta_0 . 1 + \theta_1 x_1 + \dots \theta_d x_d)$

where $\tilde{x} = [1 \ x_1 \ x_2 \ \dots \ x_d]^T$, $\tilde{\theta} = [\theta_0 \ \theta_1 \dots \theta_d]^T$

$$\frac{\partial(-y_i \tilde{\theta}_t^T \tilde{x}_i)}{\partial \theta_0} = -y_i,$$ finally substituting all findings :

$$\frac{\partial L}{\partial \theta_0} = \sum_{i=1}^{n} \left(\frac{\partial \ln(u)}{\partial u} \cdot \frac{\partial u}{\partial \theta_0} \right), \quad u = 1 + e^{-y_i \tilde{\theta}_t^T \tilde{x}_i}$$

$$= \frac{1}{1+e^{-y_i \theta_t^T \tilde{x}_i}} \cdot \frac{\partial u}{\partial \theta_0}, \quad \frac{\partial u}{\partial \theta_0} = \frac{\partial(1+e^{-y_i \tilde{\theta}_t^T x_i})}{\partial \theta_0} = \frac{\partial(e^{-y_i \theta_t^T x})}{\partial \theta_0}$$

$$= \frac{1}{1+e^{-y_i \theta_t^T \tilde{x}_i}} \cdot \frac{\partial(e^{-y_i \tilde{\theta}_t^T \tilde{x}_i})}{\partial \theta_0}, \quad \frac{\partial(e^{-y_i \tilde{\theta}_t^T \tilde{x}_i})}{\partial \theta_0} = e^z \cdot \frac{\partial z}{\partial \theta_0}$$

where $z = -y_i \tilde{\theta}_t^T \tilde{x}_i$

$$= \frac{1}{1+e^{-y_i \theta_t^T \tilde{x}_i}} \cdot e^{(-y_i \theta_t^T \tilde{x}_i)} \cdot \frac{\partial z}{\partial \theta_0}, \quad \text{where } \frac{\partial z}{\partial \theta_0} = -y_i$$

hence

$$\frac{\partial L}{\partial \theta_0} = \sum_{i=1}^{n} \left(\frac{-y_i \cdot e^{(-y_i \theta_t^T \tilde{x}_i)}}{1+e^{-y_i \theta_t^T \tilde{x}_i}} \right),$$ for other θ_t's in vector $\theta = [\theta_0 \ \theta_1 \ \dots \ \theta_d]$

only $\frac{\partial z}{\partial \theta_t}$ term will change and be equal to $-y_i . x_t^{(i)}$

i.e, $$\frac{\partial L}{\partial \theta_1} = \sum_{i=1}^{n} \left(\frac{-y_i . x_1^{(i)} . e^{(-y_i \theta_t^T \tilde{x}_i)}}{1+e^{-y_i \theta_t^T \tilde{x}_i}} \right)$$

CS Scanned with CamScanner

Deniz Aytemiz
denizaytemiz@vt.edu
no: 906470552

Therefore for more compact expression with vectors we can write

$$\frac{\nabla L}{\nabla \tilde{\Theta}_t} = \sum_{i=1}^{n} \left(\frac{-y_i \cdot \tilde{x}_i \cdot e^{(-y_i \tilde{\theta}_t^T \tilde{x}_i)}}{1 + e^{-y_i \tilde{\theta}_t^T \tilde{x}_i}} \right), \text{ where } x^{(i)} = [\underbrace{1}_{(x_0)} \; x_1 \; x_2 \dots x_d].$$

$$\downarrow$$

$$\left[\frac{\nabla L}{\nabla \Theta_0}, \frac{\nabla L}{\nabla \Theta_1}, \dots, \frac{\nabla L}{\nabla \Theta_D} \right]$$

as the final form.

code:

```
# STUDENT: PRINT THE OUTPUT AND COPY IT TO THE SOLUTION FILE
my_weights = np.ones(data_mat.shape[1]) # a weight of all 1s
derivative = weight_derivative(my_weights,train_data,train_labels)
#len(my_weights)

#print(derivative[0][4500])
print (derivative[:10])
```

```
[ 1.23415752e+03 -4.13993755e-08  1.00000000e+00  9.99993856e-01
  1.99987630e+00  9.99859072e-01  9.52574127e-01  3.59772256e+01
  2.99996572e+00 -1.38879439e-11]
```

First 10 elements of the derivative array

Deniz Aytemiz
denizaytemiz@vt.edu
no: 906470552

Task 3:
initial weights = [1 … 1], all 1 vector
step size = 0.01
tolerance = 5

```
Here are the final weights after convergence:
[-1.57105795  1.06197822  0.44064367 ... -2.00719313  1.
  0.52556552]
```

Task 4:

```
## STUDENT: CODE STARTS HERE
## Pull out the parameters (theta_0, theta) of the logistic regression model
theta0 = final_weights[0]
#theta = [θ1 θ2 ... θd]
#theta[1] = θ2, theta[2] = θ3,
theta = final_weights[1:4501]
## STUDENT: CODE ENDS HERE
print ('y intercept: ',theta0)
print ('theta1 and theta2: ',theta[1],theta[2])

y intercept:  -1.5710579458497245
theta1 and theta2:  0.44064366763718950.3361974703293962
```

In this question y intercept is -1.57 is theta0, and theta array is in form = [$\theta 1$ $\theta 2$ $\theta 3$ … $\theta 4500$], so theta array consists of the weights, not bias. Hence when the code returns theta[1] it returns $\theta 2$ and for theta[2] it returns $\theta 3$. If question wanted to it to print $\theta 1, \theta 2$ then the print statement should be updated as print(theta[0],theta[1]) but since the template is given such I haven't changed it.

Deniz Aytemiz
denizaytemiz@vt.edu
no: 906470552

Task 5:

The prediction is done such that given data point if it's probability of having label +1 is bigger then 0.5, we predict that points label as +1 otherwise we say it is -1.

```
# STUDENT: copy the output of this section to the solution file

## Get predictions on training and test data
preds_train = model_predict(train_data,final_weights)
preds_test = model_predict(test_data,final_weights)

## Compute errors
errs_train = np.sum((preds_train > 0.0) != (train_labels > 0.0))
errs_test = np.sum((preds_test > 0.0) != (test_labels > 0.0))

print ("Training error: ", float(errs_train)/len(train_labels))
print ("Test error: ", float(errs_test)/len(test_labels))
```

```
Training error:  0.0076
Test error:  0.18
```

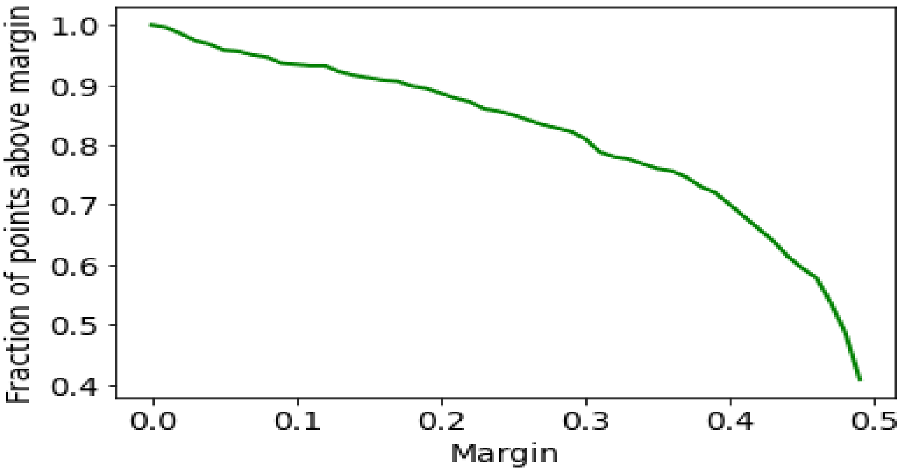
Task 6:
code:

```
def margin_counts(feature_matrix, weights, gamma):
## Return number of points for which Pr(y=1) lies in [0, 0.5 - gamma) or (0.5 + gamma, 1]

    # Input:
    # feature_matrix: numpy array of size n by d+1, where n is the number of data points, and d+1 is the
    #                 note we have included the dummy feature as the first column of the feature_matrix
    # weights: weight vector to start with, a numpy vector of dimension d+1
    # gamma: the margin value
    # Output:
    # number of points for which Pr(y=1) lies in [0, 0.5 - gamma) or (0.5 + gamma, 1]

    ## STUDENT: YOUR CODE HERE

    #y_pred = np.zeros(feature_matrix.shape[0])
    j = 0
    for i in range(feature_matrix.shape[0]):
      z = np.dot(weights, feature_matrix[i])
      prob_1 = 1/(1+np.exp(-z))
      if prob_1 >= 0.5+ gamma :
        j = j + 1
      if prob_1 < 0.5 - gamma:
        j = j + 1

    return j

    ## STUDENT: CODE ENDS
```

given the data points xi, total number of points that lie in interval P(Y=1|X=xi) >= 0.5 + gamma and P(Y=1|X=xi) < 0.5 - gamma are returned.

Deniz Aytemiz
denizaytemiz@vt.edu
no: 906470552

plot:



Task 7:

Copy the code and the output plot (i.e., visualization of the relationship between margin and error rate) to the solution file. What do you observe from the plot?

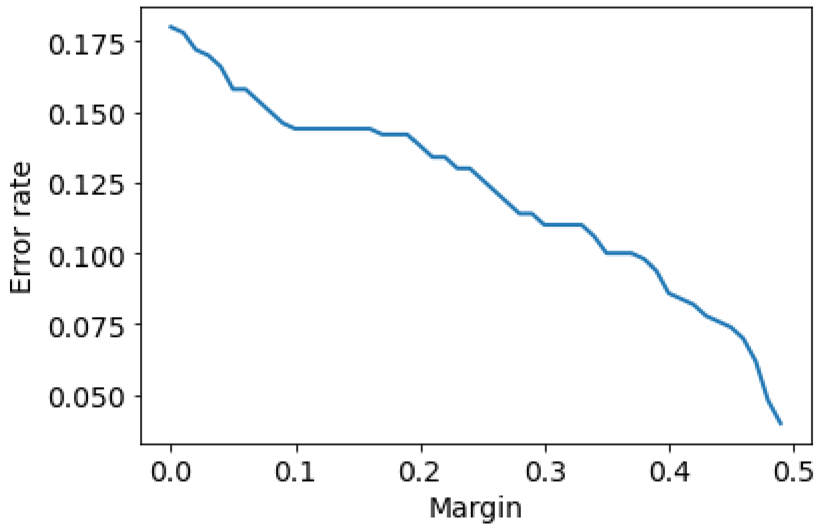
code:

```
def margin_errors(feature_matrix, labels, weights, gamma):
## Return error of predictions that lie in intervals [0, 0.5 - gamma) and (0.5 + gamma, 1]

    ## STUDENT: YOUR CODE HERE
    y_pred = np.zeros(feature_matrix.shape[0])
    for i in range(feature_matrix.shape[0]):
      z = np.dot(weights, feature_matrix[i])
      prob_1 = 1/(1+np.exp(-z))
      if prob_1 >= 0.5:
        y_pred[i] = 1
      else:
        y_pred[i] = -1

    #return y_pred

    error_counter = 0

    for i in range(feature_matrix.shape[0]):
      z = np.dot(weights, feature_matrix[i])
      prob_1 = 1/(1+np.exp(-z))
      if prob_1 >= 0.5 + gamma or prob_1 <= 0.5 - gamma:
        if y_pred[i] != labels[i]:
          error_counter = error_counter + 1

    #[0.5+gamma,1]
    #no_1 = # of points misclassified when y = 1 for the points P(Y=1|x) => 0.5 + gamma and y = 0 for th
    #no_1 = np.count_nonzero(error_margin)

    return error_counter/(feature_matrix.shape[0])

    ## STUDENT: YOUR CODE ENDS
```

Deniz Aytemiz
denizaytemiz@vt.edu
no: 906470552

In the function first prediction array is constructed with gamma = 0. Then these results compared with the classifications done when the gamma in model varies and the error is recorded. Plot indicates that best results are observed when gamma is 0. Hence the largest margin/buffer is most probably achieved when gamma is set to 0.

plot:



Deniz Aytemiz
denizaytemiz@vt.edu
no: 906470552

Task 8:

```
top 10 positive world
awful
deliciously
nicest
greatest
loved
perfected
lovely
beautifully
fantasy
world

top 10 negative world
poorly
disapppointment
badly
wasted
worth
disappointment
dog
maker
die
notable
```