Cole Beasley
Machine Learning Midterm
11/4/22

1 Regression + Regularization

1)

1) True
2) False
3) True
4) True
5) False
6) True
7) True

2) - As the points are non linear, we must first extend our points to a polynomial feature. In this case we will take x^3.

With our sample data expanded using a polynomial feature, we can learn β. Start with a random β, say all 0's and iterate as follows

$$\beta^{k+1} \leftarrow \beta^k + \eta \, \mathrm{grad}_\beta(\ell(\beta))$$

where η is the learning rate
and $\mathrm{grad}_\beta(\ell(\beta)) = \left(\frac{d\ell}{\beta_0}, \frac{d\ell}{\beta_1}, \ldots, \frac{d\ell}{\beta_D} \right)$

After learning the training set through iterations we can plot across a series of points (which have been transformed into our polynomial space) and compute their predictions through

$$y(x) = \beta_0 + \beta_1 x_1 + \ldots + \beta_D x_D$$

1 Regression + Regularization

2)

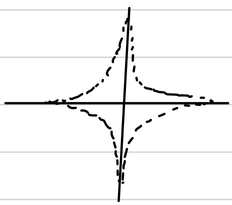
Pseudo code:

```
x = [-3.2, -2, 0, 2, 4.1, 4.6]
t = [-18.8704, -5.8, 2, -2.6, 4.2603, 8.6408]
β init = np.zeros(( len(x), 1))

current iter = 0
max iter = 100

poly = PolynomialFeature(3)
x tilde = poly.fit_transform(x)

while current iter < max_iter:
    current iter += 1
    for i in range(len(β)):
        β[i] += dl/dβ[i]
```

1 Regression + Regularization

3)

ℓ_p Ball for $p < 1$



ℓ_p Ball for $p > 2$

2 Neural Network

1)

1) False

2) False

3) False

4) False

5) True

2)

- Randomly set initial weights
- Forward propagate
  - take $x^{(i)}$ and run through neural net with initial weights
  - gives us $z_i^{(\ell)}, a_i^{(\ell)}$ for all neurons, save
  - Gives us $y(x^{(i)})$

- Get δ_{out} by $y(x^{(i)}) - t^{(i)}$

- Get all $\delta_i^{(\ell)}$ for all neurons in each layer by following

$$\delta_i^{(\ell-1)} = \sum_{j=1}^{N\ell} \delta_j^{(\ell)} w_{ji}^{(\ell)} \sigma'(a_i^{(\ell-1)})$$

- Gradient is then computed as: $\frac{d\ell}{dw_{ij}^{(\ell)}} = \delta_i^{(\ell)} z_j^{(\ell-1)}$

- Update weights using gradient

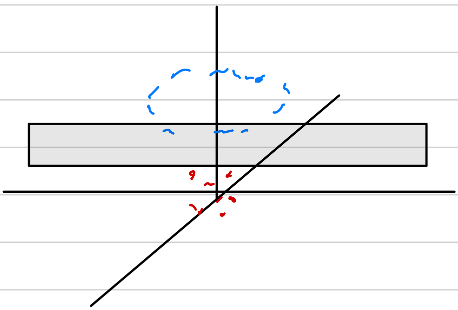
2 Neural Network

3) I would build the neural network for the circular data set as follows:

Use 2 hidden layers

Use activation function to output distance of point to origin

- use sigmoid to classify on output of first



3 Kernals

1) First define kernal

$$K_{ij} = \exp\left(-\frac{\|x - x^{(i)}\|^2}{\sigma}\right)$$

Then define lnda - start with all zeros

Learn lndas where for max iterations

$$\lambda^{k+1} \leftarrow \lambda^{k} + \left(\frac{eta}{n}\right)(f - K\lambda)$$

Gives us $y(x^{(j)}) = \sum_{i=1}^{n} \lambda_i \cdot \exp\left(-\frac{\| x^{j} - x^{i} \|^2}{\sigma}\right)$

Suedo code:

```
K= np.zeros(( len(data), len(data)))
sigma = .1
for i in range(len(data)):
    for j in range(len(data)):
        K[i,j] = np.exp((-norm(data[i] - data[j]**2)/sigma)
eta = 0.01
lnda = np.zeros( len(data))
maxiter = 100
current iter = 0
```

```
while   currentiter < maxiter:
    lmda = lmda + (eta/len(data)) · targets - matmult( K, lmda )
    current iter += 1


# Apply to points
prediction =  len (points)
for  j in range(len( points)):
    prediction[j]=0 p
    for i in range( len(data));
        prediction[j]+= prediction[j] · (
                    np.exp ((-norm(points[j] - data[i]··2)/sigma )
```

3 Kernals

2)

A kernal is valid if symetrical and positive semi definite

Is the following kernal symetrial and positive semidefinite?

$$k(x^{(i)}, x^{(j)}) = ((x^{(i)})^T (x^{(j)}) + c)^2$$

is symetric and semi definite?

$$z^T \left((x^{(i)})^T (x^{(j)}) + c\right)^2 z$$

$$= \sum_{i,j} z_i K_{ij} z_j$$

$$= \sum_{i,j} z_i ((x^i)^T (x^j) + c)((x^i)(x^j)^T + c) z_j$$

$$= \sum_K \left(\sum_i (z_i (x^{(i)})^2) + c\right) \sum_j (z_j x^{(j)2}) + c$$

$$= \geq 0$$

positive semidefinite + semitric
so valid kernal

3 Kernals

3)

$$y(x) = \sum_{i=1}^{N} \lambda_i \exp\left(-\frac{\|x - x^{(i)}\|^2}{\sigma}\right)$$

$\lambda = [1, 1, 1, 1, 1, 1, -1, -1, -1, -1]$

$\sigma = .5$