ML Neural Nets - Supervised

NN, Perceptron, Multi-layer perceptron

Perceptron Algorithm

Based on gradient Descent

1. Init w, i.e. set it to some value

2. Given data instance x_n compute output $\hat{y}_n = \text{Sign}(x_n w)$ where Sign() is an activation function

3. Update w as

gradient Descent Loss

Differentiate the hinge loss

$$w_{new} = w_{old} - \lambda \frac{dL_{Hinge}}{dw}$$

$$= w_{old} - \lambda y_n x_n \mathbb{I}[\hat{Y}_n \neq Y_n]$$

where $\lambda = 1$

$\mathbb{I}[\hat{Y}_n \neq y_n] = 1$ if $\hat{y}_n \neq y_n$

$\mathbb{I}[\hat{Y}_n \neq y_n] = 0$ if $\hat{y}_n = y_n$

4. Repeat until convergance

Stochastic Gradient Descent (SGD)

Approximate the GD gradient using just one data instance

before:

$$\frac{dL(w)}{dw} = \frac{1}{N}\sum_{n=1}^{N}\frac{dl(f_w(x_n), y_n)}{dw}$$

with SDG

$$\frac{dL(w)}{dw} \approx \frac{dl(f_w(x_t), y_t)}{dw}$$

$$\Rightarrow w^{(T+1)} = w^{(T)} - \lambda \frac{dl(f_w(x_t), y_t)}{dw}$$

where T = current iteration

$(x_T, y_T) = (x_n, y_n)$ @ current iteration

Mini-batch

The method actually used in ML sits somewhere in between called Mini-batching

MB updates w/ approximate descent of B data instances

$$\{(x_b, y_b), \forall b, 1 \leq b \leq B\}_T, \text{ where } B < N$$

$$\Rightarrow w^{(T+1)} = w^{(T)} - \lambda \frac{1}{B} \sum_{b=1}^{B} \frac{dl(f_w(x_b), y_b)}{dw}$$

~~dda~~ update the model using the loss gradient from this batch

Perceptron Convergence Theorem (4)

if a solution exists, i.e. linearly sep

the perc algo will find it in a finite number of iterations

~~Limitations of Perceptron is linear seperation~~

Summary of Perceptron

1. it is a single layer NN
2. Only works for binary classification
3. Perceptron will fail for non-linearly seperable classes
4. Optimization of Model paras based on gradient Descent