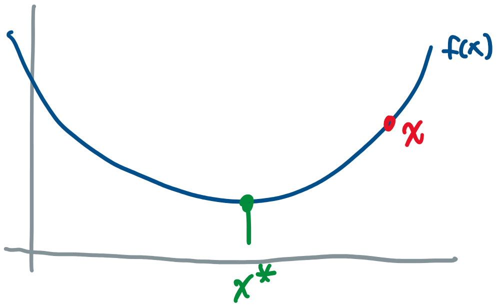
Gradient Descent

Thursday, January 23, 2020 13:26

Let's look at 1-D case: $f(x)$



if $x > x^* \iff f'(x) > 0$
if $x < x^* \iff f'(x) < 0$
if $x = x^* \iff f'(x) = 0$

Algorithm:

① Initialize x to some value

② if $f'(x)$ is close to ϕ, then done ✓

else if $f'(x) > 0$, then $x_{i+1} = x_i - \eta$

else if $f'(x) < 0$, then $x_{i+1} = x_i + \eta$

step size

if step size too small: algorithm too slow

too large: osillation, no convergene.

typically: step size ↓ over iterations.

for $d > 1$:

some direction to minimize $f(\underline{x})$

$$f(\underline{x}_1) = f(\underline{x}_0 + \eta \cdot \underline{v})$$

$$= f(\underline{x}_0) + \eta \underline{v}^T \nabla f(\underline{x}_0) + \cancel{\ldots\ldots}$$

ignore

$$\approx f(\underline{x}_0) + \eta \underline{v}^T \nabla f(\underline{x}_0)$$

learning rate

$$\Rightarrow \quad \underline{v}^* = \underset{\|v\|=1}{\operatorname{argmin}} \; f(\underline{x}_0) + \eta \underline{v}^T \nabla f(\underline{x}_0)$$

not minimizable

minimize this

$$\underline{v}^* = \frac{-\nabla f(\underline{x}_0)}{\|\nabla f(\underline{x}_0)\|}$$

Updating Weights:

adaptive "learning rate"

Req. 1: $w(t+1) = w(t) + \eta \cdot v_t$

Reg. 1:

$$\underline{w}(t+1) = \underline{w}(t) + \eta_t \cdot \underline{v}_t$$

$$\eta_t = \eta \cdot \| \underbrace{\nabla E_{in}}_{\nabla f(\underline{x}_t)} \|$$

$$\eta_t \underline{v}_t = -\eta \nabla f(x_t)$$

Reg 2: Initilization of weights

Recall in perceptron we initialize to $\emptyset$

⚠ to avoid symmetry, we initialize weights to random values (gaussian)

$$\underline{w}(t=0) = normal(0, I)$$

Reg. 3: Termination

▷ could be # of iterations

▷ could be length of gradient : $\|\nabla E_{in}\| < \delta_1$

—— error : $E_{in} < \delta_2$

—— change of error : $|\Delta E_{in}| < \delta_3$

Summary:

① Initialize weights at $t=0$ w/ $\underline{w}(t=0)$

② for $t = 1, 2, \ldots$

③ Compute $\nabla E_{in}(\underline{w}(t))$

④ Set direction $\underline{v}$ to $\frac{-\nabla E}{\|\nabla E\|}$

⑤ Update weights

$$\underline{w}(t+1) = \underline{w}(t) + \eta \cdot \underline{v}(t)$$

⑥ Iterate until weights are satisfactory.

Gradient Descent in Logistic Regression

Recall logistic regression cost function:

$$E_{in} = \frac{1}{N} \cdot \sum_{n=1}^{N} \underbrace{\log\left(1 + e^{-y_n \cdot \underline{w}^T \underline{x}_n}\right)}_{e_n}$$

$$\nabla E_{in} = \frac{\partial E_{in}}{\partial \underline{W}} = \frac{1}{N} \sum_{n=1}^{N} \left(\frac{\partial E_{in}}{\partial \underline{w}} \right)$$

$$= \frac{1}{N} \sum_{n=1}^{N} \frac{1}{\log\left(1 + e^{-y_n \underline{w}^T \underline{x}}\right)} \cdot \left(e^{-y_n \underline{w}^T \underline{x}}\right)\left(-y_n \underline{x}_n\right)$$

$$= \frac{1}{N} \sum_{n=1}^{N} \frac{y_n x_n}{1 + e^{y_n \underline{w}^T \underline{x}_n}}$$

ex. linear regression solve vs. GD.

$$f(\underline{x}) \equiv E_{in}(\underline{w}) = \frac{1}{N} \sum_{n=1}^{N} \left(\underline{w}^T x_n - y_n \right)^2$$

Recall closed form sol: $X^T X \underline{w}^* = X^T \underline{y}$

$$\underline{w}^* = X^T \underline{y} \left(X^T X\right)^{-1}$$

computation complexity: $O(N \cdot d^2 + d^3)$ operations

alternatively we could use gradient descent.

$$\underline{w}(t+1) = \underline{w}(t) - \eta \cdot \frac{2}{N} \sum_{n=1}^{N} \left(\underline{w}^T(t) \underline{x}_n - \underline{y}_n \right) \cdot \underline{x}_n$$

$$\underline{w}(t+1) = \underline{w}(t) - \eta \cdot \underbrace{\frac{2}{N}\sum_{n=1}^{N}\left(\underline{w}^T(t)\underline{x}_n - \underline{y}_n\right)\cdot \underline{x}_n}$$

Complexity: $O(N \times d \times \underset{\uparrow}{\underline{I}})$ operations

iterations

$\Rightarrow$ gradient descent is faster if $\underline{d}$ is large

$\Rightarrow$ gradient descent is not good for large N

(gradient is expensive for large N)