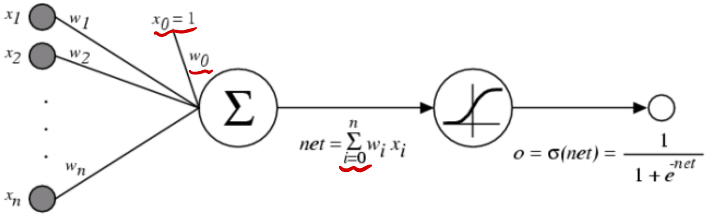
Neural Network & Reinforcement Learning

田鹏超

tianpch@shanghaitech.edu.cn

Neural Network



$\sigma(x)$ is the sigmoid function

$$\frac{1}{1+e^{-x}}$$

Nice property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

MLE and MSE (Lecture21， p8)

$y = f(x) + \varepsilon$, $\varepsilon \sim N(0, \sigma^2)$

$\Rightarrow y \sim N(f(x), \sigma^2)$

$$
\begin{aligned}
\mathrm{l}(\boldsymbol{w}, \sigma) &= \ln \prod_{i=1}^{N} p(t|x_i, \boldsymbol{w}, \sigma) \\
&= \sum_{i=1}^{N} \ln p(t|x_i, \boldsymbol{w}, \sigma) \\
&= \sum_{i=1}^{N} \ln \left[\frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(t_i - f(x_i, \boldsymbol{w}))^2}{2\sigma^2}) \right] \\
&= -\frac{1}{2\sigma^2} \sum_{i=1}^{N} (t_i - f(x_i, \boldsymbol{w}))^2 - \sum_{i=1}^{N} \ln \left(\sqrt{2\pi}\sigma \right)
\end{aligned}
$$

Thus, to maximize $\mathrm{l}(\boldsymbol{w}, \sigma)$ is equal to minimize $\sum_{i=1}^{N} (t_i - f(x_i, \boldsymbol{w}))^2$, which is the sum-of-squares error (or we can convert into MSE).

MAP and regularized MSE (Lecture21， p9)

More, if we assume that the polynomial coefficients $\boldsymbol{w}$ is distributed as the Gaussian distribution of the form

$$p(\boldsymbol{w}|\alpha) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \alpha\boldsymbol{I})$$

We can write the poster probability function as

$$p(\theta|\mathcal{D}) = \prod_{i=1}^{N} p(\boldsymbol{w}|t_i) = \prod_{i=1}^{N} \frac{p(t_i|x_i, \boldsymbol{w}, \sigma)p(\boldsymbol{w}|\alpha)}{p(t_i|x_i, \sigma)}$$

Then

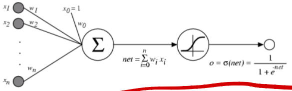
$$\text{maximize } \ln p(\theta|\mathcal{D}) \equiv \text{maximize } \ln\left[\prod_{i=1}^{N} p(t_i|x_i, \boldsymbol{w}, \sigma)p(\boldsymbol{w}|\alpha)\right]$$

Denote D is the dimension of $\boldsymbol{w}$,

$$\begin{aligned}
\ln\left[\prod_{i=1}^{N} p(t_i|x_i, \boldsymbol{w}, \sigma)p(\boldsymbol{w}|\alpha)\right] &= \ln\prod_{i=1}^{N} p(t_i|x_i, \boldsymbol{w}, \sigma) + \ln p(\boldsymbol{w}|\alpha) \\
&= -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(t_i - f(x_i, \boldsymbol{w}))^2 - \sum_{i=1}^{N}\ln\left(\sqrt{2\pi}\sigma\right) \\
&+ \ln\left[\frac{1}{(2\pi)^{D/2}}\frac{1}{|\alpha\boldsymbol{I}|^{1/2}}\exp\left(-\frac{1}{2}\boldsymbol{w}^T(\alpha\boldsymbol{I})^{-1}\boldsymbol{w}\right)\right] \\
&= -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(t_i - f(x_i, \boldsymbol{w}))^2 - \frac{1}{2\alpha}\boldsymbol{w}^T\boldsymbol{w} + const
\end{aligned}$$

Backpropagation (MLE v.s. MAP)

Error Gradient for a Sigmoid Unit



$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\left(\frac{1}{2}\sum_{d\in D}(t_d - o_d)^2\right) \\
&= \frac{1}{2}\sum_d \frac{\partial}{\partial w_i}(t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d 2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d) \\
&= \sum_d (t_d - o_d)\left(-\frac{\partial o_d}{\partial w_i}\right) \\
&= -\sum_d (t_d - o_d)\frac{\partial o_d}{\partial net_d}\frac{\partial net_d}{\partial w_i}
\end{aligned}
$$

$$\left(\frac{1}{2}\sum_{d\in D}(t_d - o_d)^2 + \lambda \|w\|_2^2\right)$$

But we know:

$$\frac{\partial o_d}{\partial net_d} = \frac{\partial \sigma(net_d)}{\partial net_d} = o_d(1 - o_d)$$

$$\frac{\partial net_d}{\partial w_i} = \frac{\partial(\vec{w}\cdot\vec{x}_d)}{\partial w_i} = x_{i,d}$$

So:

$$\frac{\partial E}{\partial w_i} = -\sum_{d\in D}(t_d - o_d)o_d(1 - o_d)x_{i,d}$$

x_d = input

t_d = target output

o_d = observed unit output

w_i = weight i

Batch GD, SGD, Mini-batch GD

Batch mode Gradient Descent:
Do until satisfied

size $|D|$

1. Compute the gradient $\nabla E_D[\vec{w}]$
2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_D[\vec{w}]$

Incremental mode Gradient Descent:
Do until satisfied

- For each training example d in D
  1. Compute the gradient $\nabla E_d[\vec{w}]$

     only one observation
  2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_d[\vec{w}]$

$$E_D[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

$$E_d[\vec{w}] \equiv \frac{1}{2} (t_d - o_d)^2$$

Incremental Gradient Descent can approximate *Batch Gradient Descent* arbitrarily closely if η made small enough

with size b

$$E_D[\vec{w}] = \frac{1}{2} \sum_{\substack{d \in B \\ B \subset D}} (t_d - O_d)^2$$

Reinforcement Learning

Value function & Q-Learning

基于此假设

MDP

$P(s_{t+1} | s_t, s_{t-1}, \cdots)$

$= P(s_{t+1} | s_t)$

Define new function, closely related to V*

$$V^*(s) = E[r(s, \pi^*(s))] + \gamma E_{s'|\pi^*(s)}[V^*(s')]$$

V*(s) is the expected discounted reward of following the optimal policy from time 0 onward.

$$Q(s,a) = E[r(s,a)] + \gamma E_{s'|a}[V^*(s')]$$

Q(s,a) is the expected discounted reward of first doing action a and then following the optimal policy from the next step onward.

If agent knows Q(s,a), it can choose optimal action without knowing $P(s_{t+1}|s_t,a)$!

$$\pi^*(s) = \arg\max_a Q(s,a) \qquad V^*(s) = \max_a Q(s,a)$$

Just chose the action that maximizes the Q value

And, it can <u>learn</u> Q without knowing $P(s_{t+1}|s_t,a)$

using something very much like the dynamic programming algorithm we used to compute V*.