Video 18

Monday, October 26, 2020 7:19 PM

Binary classification w/ logistic regression

$z = \vec{w}^T \vec{x} + b$

$y = \sigma(z)$ (will be between 0 and 1) where $\sigma(z) = \frac{1}{1+e^{-z}}$ (logistic function.)

$y \geq 1/2 \longrightarrow$ positive

$y < 1/2 \longrightarrow$ negative.

Loss (cross-entropy) : $\mathcal{L}_{CE} = -t \log y - (1-t)\log(1-y)$

(target $t \in \{0,1\}$; y: model's prediction)

Problem : when $z \ll 0$ (model is very confident that $\vec{x}$ is a negative example).

$z \longrightarrow -\infty$

$e^{-z} \longrightarrow +\infty$

$y = \sigma(z) = \frac{1}{1+e^{-z}} \longrightarrow \frac{1}{\infty} \longrightarrow 0$ (σ: logistic function)

$-t\log y = -\log y$ tries to compute $\log(0)$ ($-\infty$)

Pb: Hard-to-find subtle bugs because we're trying to compute log(0).

Solution : combine the loss function (cross-entropy) with the activation function (logistic function).

Logistic-cross-entropy function :

$$\mathcal{L}_{LCE}(z,t) = \mathcal{L}_{CE}(\sigma(z),t) = -t\log(\sigma(z)) - (1-t)\log(1-\sigma(z))$$

($z = \vec{w}^T\vec{x} + b$ rather than model's output $y = \sigma(z)$; target $t \in \{0,1\}$)

$$= -t\log\left(\frac{1}{1+e^{-z}}\right) - (1-t)\log\left(1 - \frac{1}{1+e^{-z}}\right)$$

$\log\left(\frac{A}{B}\right) = \log A - \log B$

$$= -t\left(\log(1) - \log(1+e^{-z})\right) - (1-t)\log\left(\frac{1+e^{-z}-1}{1+e^{-z}}\right)$$

($\log(1) = 0$)

$$= +t\log(1+e^{-z}) - (1-t)\log\left(\frac{e^{-z}}{1+e^{-z}}\right)$$

($\times e^{z}$ numerator and denominator)

$$= t\log(1+e^{-z}) - (1-t)\log\left(\frac{1}{e^{z}+1}\right)$$

$$= t\log(1+e^{-z}) - (1-t)\left(\log(1) - \log(e^{z}+1)\right)$$

($\log(1) = 0$)

Combined loss and activation function (cross-entropy, logistic):

$$\mathcal{L}_{LCE}(z,t) = t\log(1+e^{-z}) + (1-t)\log(1+e^{z})$$

($\neq 0$, $\neq 0$)

More numerically stable.

NumPy Implementation

$E = t * \text{np.logaddexp}(0, -z) + (1-t) * \text{np.logaddexp}(0, z)$

(E: combined loss and activation function)

np.logaddexp: function included in NumPy which computes $\text{np.logaddexp}(a,b) = \log(e^a + e^b)$

np here stands for NumPy

```
import numpy as np
```

Gradient descent

Update rule : $w \longleftarrow w - \alpha \frac{\partial \mathcal{L}}{\partial w}$

(α: learning rate (hyperparameter))

For simplicity, assume that w is a scalar but using vectorized notation, we could derive a similar result.

($\mathcal{L}_{LCE}$ combines loss and activation functions)

$$\frac{\partial \mathcal{L}(z,t)}{\partial w} = \frac{\partial \mathcal{L}}{\partial z} \cdot \frac{\partial z}{\partial w}$$ using chain rule.

(per definition of z) $\frac{\partial z}{\partial w} = \frac{\partial(wx+b)}{\partial w} = x$

$$\frac{\partial \mathcal{L}_{LCE}(z,t)}{\partial z} = \frac{\partial\left(t\log(1+e^{-z}) + (1-t)\log(1+e^{z})\right)}{\partial z}$$

$$= \frac{\partial\left(t\log(1+e^{-z})\right)}{\partial z} + \frac{\partial\left((1-t)\log(1+e^{z})\right)}{\partial z}$$

$$= t\frac{\partial \log(1+e^{-z})}{\partial z} + (1-t)\frac{\partial \log(1+e^{z})}{\partial z}$$

($(\log(u))' = \frac{u'}{u}$)

$$= t\frac{\frac{\partial(1+e^{-z})}{\partial z}}{1+e^{-z}} + (1-t)\frac{\frac{\partial(1+e^{z})}{\partial z}}{1+e^{z}}$$

($(e^u)' = u'e^u$: $(-1)e^{-z}$; $(1)e^{z}$)

$$= t\frac{-e^{-z}}{1+e^{-z}} + (1-t)\frac{e^{z}}{1+e^{z}}$$

($\times e^{-z}$ numerator and denominator)

$$= -t\left(\frac{1-1+e^{-z}}{1+e^{-z}}\right) + (1-t)\frac{1}{e^{-z}+1}$$

($\frac{1}{e^{-z}+1} = y = \sigma(z)$; logistic function $y = \sigma(z) = \frac{1}{1+e^{-z}}$)

$$= -t\left(\frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}}\right) + (1-t)y$$

($\frac{1+e^{-z}}{1+e^{-z}} = 1$)

$$= -t\left(1 - \frac{1}{1+e^{-z}}\right) + (1-t)y$$

($\frac{1}{1+e^{-z}} = y$)

$$= -t(1-y) + (1-t)y$$

$$= -t + ty + y - ty$$

$$\frac{\partial \mathcal{L}_{LCE}(z,t)}{\partial z} = y - t$$

$$\frac{\partial \mathcal{L}_{LCE}(z,t)}{\partial w} = \frac{\partial \mathcal{L}_{LCE}(z,t)}{\partial z}\frac{\partial z}{\partial w}$$

$$= (y-t)x$$

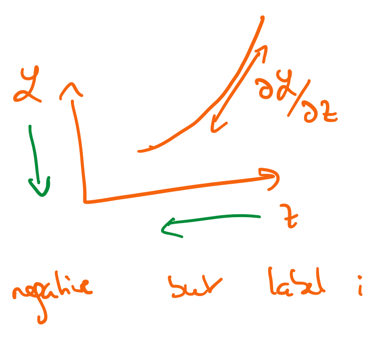
Update rule for binary linear classifier :
with
- a logistic activation function
- a cross-entropy loss function

$$w \longleftarrow w - \alpha(y-t)x$$

If model predicts positive but label is negative $(t=0)$, $y > t$

$y > t \Rightarrow y - t > 0$

$\Rightarrow \frac{\partial \mathcal{L}}{\partial z} > 0$ is consistent w/ the intuition that to decrease the loss (improve the prediction) we need to decrease z



If model predicts negative but label is positive $(t=1)$, $y < t$

$y < t \Rightarrow y - t < 0$

$\Rightarrow \frac{\partial \mathcal{L}}{\partial z} < 0$ corresponds to our intuition that we should increase z to decrease the loss