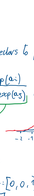
Machine Learning

Generalization for C>2
Multiclass - Classification

Softmax maps vectors to probability simplex.

$$\text{softmax}(a_i) = \frac{\exp(a_i)}{\sum_j \exp(a_j)}$$



$\exp(a_i)$: Higher a_i get large probability
Lower a_i get scaled down

→ ensure everything sum up to 1

$f(x) = \text{softmax}(\theta^T x_i)$ → Logits

$c=3$ $\quad$ $\text{deg}=[0,1,0]$ $\quad$ $\text{graph}=[0,0,1]$

Probability distribution with all the mass on a single class
$f(x)$ encodes the probability distribution over all the classes
$f(x)_c$ represent the probability for the single class c

$$P(Y|f(x)) = \prod_c^C [f(x)_c]^{Y_c}$$

Since Y is one-hot vector only $f(x)_c$ is going to contribute in the product

$Y = [0,1,0]$

$$P(Y|f(x)) = f(x)_Y$$

$f(x)_1^{Y_1} \cdot f(x)_2^{Y_2} \cdot f(x)_3^{Y_3}$

$1 \cdot f(x)_2 \cdot 1$

$f(x)_2$, true class

$$L = \prod_i^n P(Y^i|f(x^i))$$

$$\log L = \sum_i^n \log P(Y^i|f(x^i))$$

$$= \sum_i^n \log f(x^i)_{Y^i}$$

$$NLL = -\sum_i^n \log f(x^i)_{Y^i}$$

Cross Entropy

$0 \cdot f(x)_1 + 1 \cdot f(x)_2 + 0 \cdot f(x)_3$

expect order to match in the priority order of single class in class vector

$$= -\sum_i^n \sum_j^c Y_j^i \log f(x^i)_j$$

$f(x)^T$: $\begin{bmatrix} 0.1 & 0.2 & 0.7 \end{bmatrix}$

Y: cat
Y: graph

Y: cat

$$= -\left(\underbrace{1(-1)}_{\text{cat}} + \underbrace{0(-0.69)}_{\text{dog}} + \underbrace{0(-0.15)}_{\text{graph}}\right)$$

$= 1$

Y: graph

$$= -\left(0(-1) + 0(-.69) + 1(-0.15)\right)$$

$= 0.15$

x_1 $\quad Y_1 = \text{cat}$ $\quad [1,0,0]$ $\quad f(x_1) = [0.1, 0.2, 0.7]$

x_2 $\quad Y_2 = \text{dog}$ $\quad [0,1,0]$ $\quad f(x_2) = [0.01, 0.98, 0.01]$

$$CE(\theta) = -\sum_i^n \sum_j^c Y_j^i \log f(x^i)_j$$

$$= Y_1 \quad -\left(Y_1^1 \log f(x^1)_1 + Y_2^1 \log f(x^1)_2 + Y_3^1 \log f(x^1)_3\right.$$

$$Y_2 \quad \left.Y_1^2 \log f(x^2)_1 + Y_2^2 \log f(x^2)_2 + Y_3^2 \log f(x^2)_3\right)$$

$$= -\left(\begin{matrix} 1(-0.91) \\ 0(-4.6) \end{matrix} + \begin{matrix} 0(-1.6) \\ 1(-0.02) \end{matrix} + \begin{matrix} 0(-0.91) \\ 0(-1.6) \end{matrix}\right)$$

$\log(f(x^1)) = [-0.91, -1.6, -0.91]$

$\log(f(x^2)) = [-4.6, -0.02, -1.6]$

$$= -\left(-0.91 - 0.02\right) = 0.93$$

Maximise the probability of the true class at the expense of the other categories.

Softmax : unstable $\quad : -\log\left(\frac{\exp P_i}{\sum_j \exp P_j}\right)$

$-P_i + \log \text{sumexp}(P)$

$\log(\sum_i \exp P_i)$

$\log(\sum \exp(P_i - c)) + c$

allows us moving with our optimization by adding a small constant c
if you set $c = \max(P)$ ensure numerical instabilities will never occur

$-\frac{1}{2n}(\text{Loss})$

use for differentiation of square differences
Normalize by number of samples in the dataset

$f(x) = \text{softmax}(\theta^T x_i)$ → Multiclass Logistic Regression

best set of parameters determined by minimizing $CE(\theta)$

Entropy $H(P) = -\sum_i^c P_i \log P_i$ $\quad$ measure in certainty of distribution P

Cross Entropy $CE(P,q) = -\sum_i^c P_i \log q_i$ $\quad$ measure uncertainty of distribution P when we encode it with q

KL Divergence $KL(P||q) = +\sum_i^c P_i \log \frac{P_i}{q_i}$ $\quad$ how much information is lost when you use q instead of P

$CE(P,q) = \underbrace{H(P)}_{\text{uncertainty of true distribution}} + \underbrace{KL(P||q)}_{\text{additional cost of using } q \text{ instead of } P}$

$f(x)_i$ logits cannot be considered as real probabilities.

$P(Y=i|x) = f(x)_i$ unless this happens!

How calibrate if a model is calibrated.

1 pick a validation set
2 split $[0,1]$ into m bins $[1/m]$

B_m : number of samples whose confidence falls into bin m

P_m : average confidence for each bin

a_m : average accuracy for each bin

Expected Calibration Error (ECE) $= \sum_m \frac{B_m}{n} |a_m - P_m|$

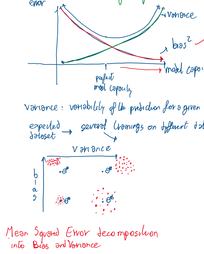
Hyperplanes

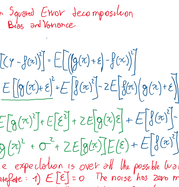
N dimensional space → N-1 dimensional flat subspace

2D → 1D line
3D → 2D plane

1 can only separate classes that are linearly separable
2 distance from closest data sample and hyperplane is called margin

Bias - Variance

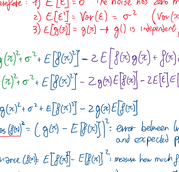




difference between the expected prediction and the correct one

Variance : variability of the prediction for a given dataset

expected dataset → several trainings on different data partitions



θ^* obtained by minimizing Expected Risk

$\theta'\theta^2$ - obtained by minimizing the Empirical Risk

Mean Squared Error decomposition into Bias and Variance

$$E[(y - f(x))^2] = E[(f(x) + \varepsilon - f(x))^2]$$

$$= E[(f(x) - \varepsilon)^2] + E[2\varepsilon(f(x) - 2E[f(x)]\varepsilon)]$$

$$= E[f(x)^2] + E[\varepsilon^2] + 2E[f(x)\varepsilon] + E[2\varepsilon^2] - 2E[f(x)(f(x) + \varepsilon)]$$

$$= f(x)^2 + \sigma^2 + 2E[f(x)]E[\varepsilon] + E[2\varepsilon^2] - 2E[f(x)^2] - 2E[f(x)\varepsilon]$$

The expectation is over all the possible training datasets.
Therefore : 1) $E[\varepsilon] = 0$ The noise has zero mean
2) $E[\varepsilon^2] = Var(\varepsilon) = \sigma^2$ $\quad (Var(x) = E[x^2] - E[x]^2)$
3) $E[f(x)\varepsilon] = f(x) \cdot E[\varepsilon]$ → $f(x)$ is independent from the dataset of noise

$$= f(x)^2 + \sigma^2 + E[f(x)^2] - 2E[f(x)g(x) + f(x)\varepsilon]$$

$$= f(x)^2 + \sigma^2 + E[f(x)^2] - 2g(x)E[f(x)] - 2E[\varepsilon]E[f(x)]$$

Because $f(x)$ and ε are independent : $E[f(x) \cdot \varepsilon] = E[\varepsilon]E[f(x)]$

$$= g(x)^2 + \sigma^2 + E[f(x)^2] - 2g(x)E[f(x)]$$

$$\text{Bias}(f(x))^2 = (g(x) - E[f(x)])^2$$

Error between true function and expected prediction

Variance $(f(x)) = E[(f(x) - E[f(x)])^2]$: measure how much $f(x)$ varies for different training datasets

$$= g(x)^2 + \sigma^2 + Var(f(x)) + E[f(x)]^2 - 2g(x)E[f(x)]$$

$$= \text{Bias}(f(x))^2 + Var(f(x)) + \sigma^2 \text{ Irreducible error}$$

Ridge regression

where P is the total number of parameters

$$L(\theta) = \frac{1}{N}\sum_i^N (Y_i - f(x_i))^2 + \alpha\sum_j^P \theta_j^2$$

Not the bias term

$\alpha = 0$
$\alpha = 10$
$\alpha = 10000$

Lasso Regression

$$L(\theta) = \frac{1}{N}\sum_i^N (Y_i - f(x_i))^2 + \alpha\sum_j^P |\theta_j|$$

$\theta_1 + \theta_1 x_1 + \theta_2 x_2 \ldots + \theta_p x_p$

Elastic Net

$$L(\theta) = \frac{1}{N}\sum_i^N (Y_i - f(x_i))^2 + \alpha_1\sum_j^P |\theta_j| + \alpha_2\sum_j^P \theta_j^2$$