Day 22, Oct-24, 2024.

→ Foundations of Deep Learning

Understanding Some Probability Related Concepts & terms

Flipping Coins (Loglikelihood function)

Using Input (with $\log p(y|\pi)$)

Decision Boundary & Non-linear Decision Boundary

Adaptability - Learning $\pi(x)$ or function.

$$\log p(y|\pi) \Rightarrow$$

$$\log \left[\prod_{i=1}^{n} \pi^{\mathbb{1}(y_i=1)} (1-\pi)^{\mathbb{1}(y_i=0)} \right]$$

$$\Rightarrow \sum_{i=1}^{n} \mathbb{I}(y_i=1) \log \pi + \mathbb{I}(y_i=0) \log(1-\pi)$$

log-likelihood function for a series of Coin flips.

Flipping Coins

- Imagine n flips of a coin with probability π of coming up hea
- Loglikelihood function (independent bernoulli trials)

$$\log p(y|\pi) = \log \left[\prod_{i=1}^{n} \pi^{\mathbb{I}(y_i=1)} (1-\pi)^{\mathbb{I}(y_i=0)} \right]$$

$$= \sum_{i=1}^{n} \mathbb{I}(y_i = 1) \log \pi + \mathbb{I}(y_i = 0) \log(1-\pi$$

where we use the shorthand $y = (y_1, \ldots, y_n)$.

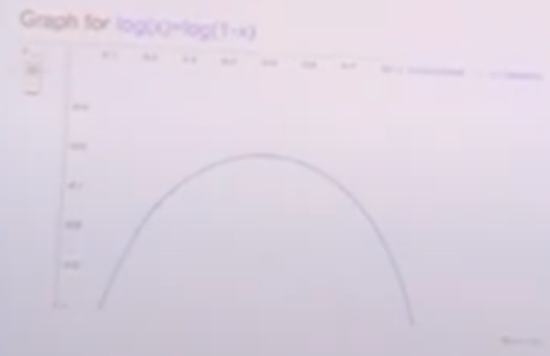


Figure: Logliklihoods over π when we see one head and one tail. Seems likely that $\pi \approx 0.5$.

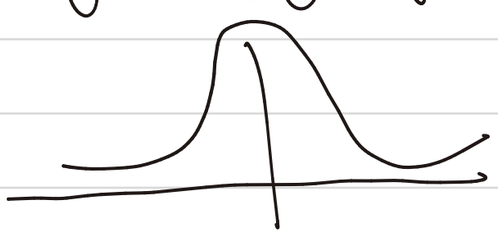
" It is used to estimate the probability of getting heads (π) based on the observed outcomes (y) of the coin flips."

$$\log p(y|\pi) = \log \left[\prod_{i=1}^{n} \pi^{\mathbb{I}(y_i=1)} (1-\pi)^{y_i=0} \right]$$

$$\Rightarrow \sum_{i=1}^{n} \pi\,(y_i=1) \log \pi + \pi\,(y_i=0) \log(1-\pi)$$

where we use the shorthand $y = (y_1, y_2 \cdots y_n)$ $(3,7) \rightarrow$ (Head, tail)

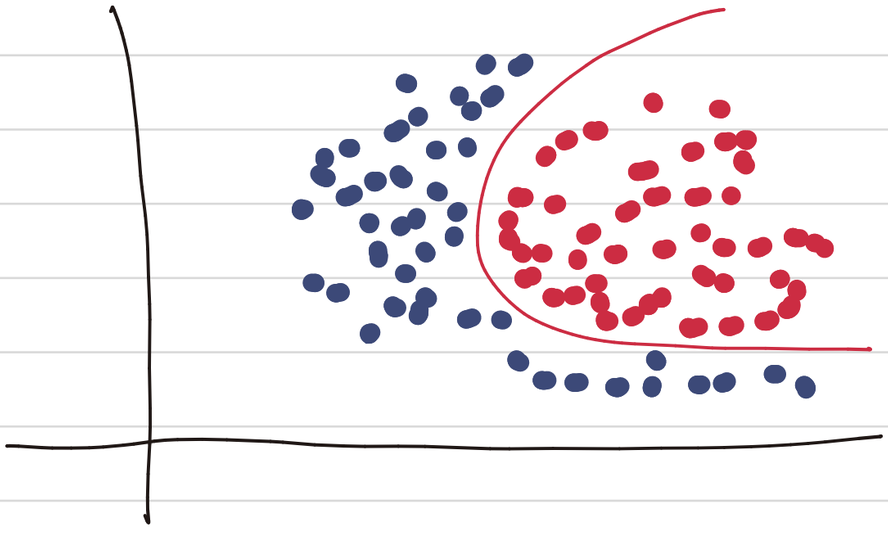
if we use $(0,1) \rightarrow$ (Head, tail) then

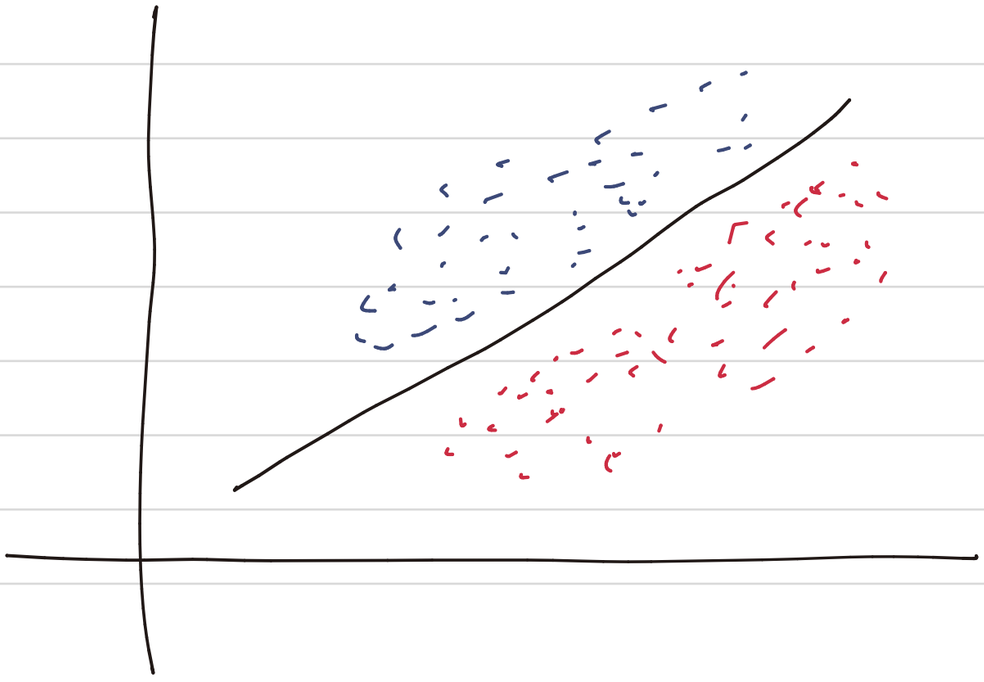


② Using Input Data

→ What if the probability head depend on Input data?

→ In the ANN, our model predict using input features. So, in the same way some input x_i we have probability $\pi(x_i)$ of heads?





→ So the model prediction depend upon the characteristics of data.

→ Using data to influence the probability of an event introduce the idea of conditional probability where the probability of an event is not fixed but depends on additional info/ motion (input - data)

Binomial Distribution:

$$P(X=K) = C(n,k) * p^{K} * (1-p)^{n-K}$$

Each trial has two possible outcomes

n → total number of trials

K ⇒ number of successes

Where $P(X=K)$

$$C(n,K) = \frac{n!}{K! * (n-K)!}$$

$$\log p(y|\pi) \Rightarrow \log\left[\prod\left(\pi^{I(y_i=1)} * (1-\pi)^{I(y_i=0)}\right)\right] \quad \text{— eqn (1)}$$

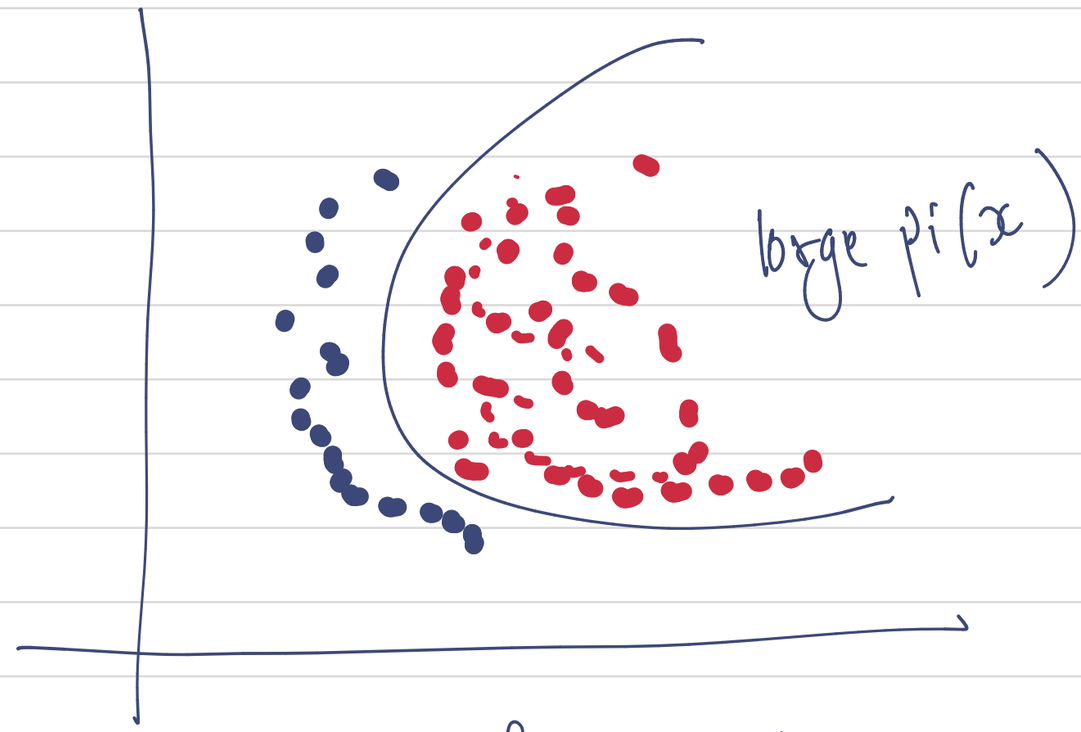
$$\Rightarrow \sum_{i=1}^{n}\left(\pi(y_i=1) * \log\pi + \pi(y_i=0) * \log(1-\pi)\right)$$

$$\log p(y/x) \Rightarrow \sum_{i=1}^{n} \pi(y_i=1)\log\pi(x) + \pi(y_i=0)\log(1-\pi(x)) \quad \text{— eqn (2)}$$

Now Instead of Having Same probability for every person I have probability that depends upon each person.

Before we did maximum likelihood for π now we need to learn some function $\pi_i(x)$ that explains the data well.

The only difference between them eqn (i) and eqn (ii) is Pi(x) π of x.



Learning $\pi(x)$: figuring out the best way to calculate the probability of Heads based on the Input data.

(1) Log-likelihood of flipping a coin is calculated, estimating π (probability of heads)

(2) probability of Heads Depends on input data (x_i), shown as classification.

(3) Log-likelihood is adapted to include input data, learning $\pi(x)$.

Flipping Coins

- Imagine n flips of a coin with probability π of coming up he
- Loglikelihood function (independent bernoulli trials)

$$\log p(y|\pi) = \log\left[\prod_{i=1}^{n} \pi^{\mathbb{I}(y_i=1)} (1-\pi)^{\mathbb{I}(y_i=0)}\right]$$

$$= \sum_{i=1}^{n} \mathbb{I}(y_i = 1) \log \pi + \mathbb{I}(y_i = 0) \log(1 - \pi$$

where we use the shorthand $y = (y_1, \ldots, y_n)$.

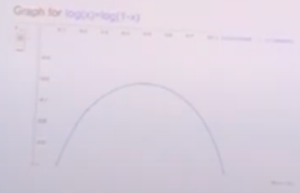


Figure: Logliklihoods over π when we see one head and one tail. Seems likely that $\pi \approx 0.5$.

Using Input Data

- What if the probability depended on input data?
- Depending on some input x_i, we have probability $\pi(x_i)$ of heads

Figure: Example of a classification problem when x_i are two dimensional, heads are green, and tails are red.

Using Input Data

- What if the probability depended on input data?
- Depending on some input x_i, we have probability $\pi(x_i)$ of heads

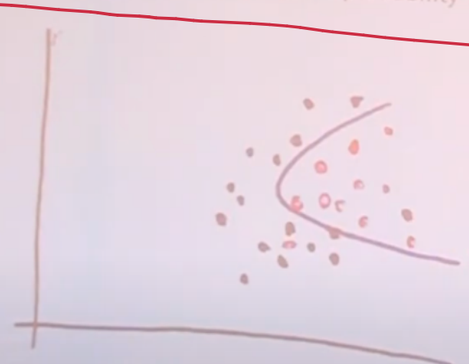


Figure: Example with a nonlinear boundary

Using Input Data

- Can adapt previous loglikelihood to this setting

$$\log p(y|x) = \sum_{i=1}^{n} \mathbb{I}(y_i = 1) \log \pi(x) + \mathbb{I}(y_i = 0) \log(1 - \pi(x))$$

- Need to learn $\pi(x)$, just like we learned π before

Gradient Descent of Logistic Regression

① We need derivative of Sigmoid function

② Partial Derivatives of weights w_j w.r.t. Cost $J(\vec{w}, b)$

③ Gradient with respect to 'w_j' & Gradient w.r.t 'b'

Gradient descent

cost

$$J(\vec{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)\right]$$

repeat {

$j = 1 \ldots n$

$$w_j = w_j - \alpha\frac{\partial}{\partial w_j}J(\vec{w}, b)$$

$$\frac{\partial}{\partial w_j}J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})x_j^{(i)}$$

$$b = b - \alpha\frac{\partial}{\partial b}J(\vec{w}, b)$$

$$\frac{\partial}{\partial b}J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})$$

}

① $f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$

② $J(\vec{w},b) = \frac{-1}{m}\sum_{i=1}^{m}\left[y^{(i)}\cdot\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right)+(1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)\right]$

③ Gradient with Respect to w_j.

$$\frac{\partial}{\partial w_j}J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}-y^{(i)}\right)x_j^{(i)}$$

where $f_{\vec{w},b}(\vec{x}^{(i)}-y^{(i)}) \rightarrow$ difference between (Actual - prediction)

(7) Gradient with respect to b.

$$\frac{\partial}{\partial b} J(\vec{w}, b) \Rightarrow \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)} - y^{(i)} \right)$$

(8) So, update Rule -

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$$

$$w_j := w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)} - y^{(i)} \right) x_j^{(i)}$$

$$b := b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b)$$

$$b := b - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)} - y^{(i)} \right)$$

Derivative of Sigmoid function

① $$f_{\vec{w},b}(\vec{x}^{(i)}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}^{(i)}+b)}}$$

② $$f(z) = \frac{1}{1+e^{-z}} \quad \text{where, } z = \vec{w}\cdot\vec{x}^{(i)} + b$$

So, $\frac{d}{dz} f(z) = f(z)(1-f(z))$ — eqn (I)

③ Derivative of z w.r.t. w_j → $z = \vec{w}\cdot\vec{x}^{(i)} + b$ w.r.t w_j

$$z = w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_j x_j^{(i)} + \cdots + b$$

Derivative of z w.r.t. to w_j Simply $\left[\frac{\partial z}{\partial w_j} = x_j^{(i)}\right]$ — eqn (II)

where z is linear Combination of the weights and features.

(i) $\frac{\partial}{\partial w_j} f_{\vec{w},b}(\vec{x}^{(i)}) \Rightarrow \frac{d}{dz} f(z) \cdot \frac{\partial z}{\partial w_j}$ ⟶ eqn (iii)

Substituting eqn (i) and eqn (ii) — in eqn (iii)

$$\frac{\partial}{\partial w_j} f_{\vec{w},b}(\vec{x}^{(i)}) \Rightarrow f(z)(1 - f(z)) \cdot x_j^{(i)}$$

But $f(z)(1-f(z))$ ⟶ eqn (v) has $f(z)$

and $f(z)$ can be written as

$$f(z) \Rightarrow f_{\vec{w},b}(\vec{x}^{(i)})$$

So, the eqn (iii) becomes,

$$\frac{\partial}{\partial w_j} f_{\vec{w},b}(\vec{x}^{(i)}) \Rightarrow f_{\vec{w},b}(\vec{x}^{(i)})\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right) \cdot x_j^{(i)}$$

which becomes eqn (iv)

eqn (iv) helps update the weight w_j during each step of gradient descent.

Since, Gradient descent is optimization algorithm that aim to find the minima and minimize the cost function and optimize the model parameters to reduce the prediction error.