1.

2. What is **Logistic Regression**?

Goal: train classifier that can make binary decision about the class of an input obs.

LR
- $y=1$: positive
- $y=0$: negative

⇒ Given test instance $x = [x_1, \ldots, x_f]$ ⇒ 1/0.

⇒ Model: calculate prob $P(y=1|x)$

Decision boundary as 0.5:

$$\text{Classify as } \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

(i). How is **Logistic Regression** similar to **Naive Bayes** and how is it different? In what circumstances would the former be preferable, and in what circumstances would the latter?

Similar:

1. classification methods ⇒ predict class Y for instance X.
2. Compute $P(Y|X)$ ⇒ predict class with highese prob.
   ⇒ probabilistic ML methods (prediction based on probs.)
3. Have params → maximise likelihood

Different:

NB	LR
Generative model	Discriminative model
⇒ maximise joint $P(x,y)$	⇒ model $P(y\|x)$ directly
assume condition (among features) independence	No independence assumption
Trying to learn what dogs & cats look like.	Trying to learn to "distinguish" between classes without actually learning much about them (e.g. all dogs are wearing collars and cats aren't)

Prefer:

Generally, LR outperform NB (NB can outperform LR when little data available)

NB: simpler, easier to implement, param estimated in closed-form.

LR: adopt iterative optimisation method ⇒ time consuming for large data.

(ii). What is "logistic"? What are we "regressing"?

We apply logistic function (sigmoid) σ to regression z.

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

$$z = \theta_0 + \theta_1 x_1 + \cdots + \theta_f x_f \quad (\theta_i : \text{params})$$

- Easy to calculate derivative ⇒ for gradient descent
- Has range $[0, 1]$ ⇒ estimate probability.

Regressing: log odds: $\log \frac{p}{1-p} = z$

2.

Bob tries to gather information about this year's apple harvest, and ran a search in his favourite online news outlet. He retrieved a number of articles, but found that a large portion of the retrieved articles are about the Apple laptops and computers -- and hence irrelevant to his search. He wants to build a logistic regression classifier, which uses the counts of selected words in the news articles to predict the class of the news article (fruit vs. computer).He built the following data set of 5 training instances and 1 test instance. Develop a logistic regression classifier to predict label $\hat{y} = 1$ (fruit) and $\hat{y} = 0$ (computer).

term occurences

ID	*apple*	*ibm*	*lemon*	*sun*	CLASS
TRAINING INSTANCES					
A	1	0	1	5	1 FRUIT
B	1	0	1	2	1 FRUIT
C	2	0	0	1	1 FRUIT
D	2	2	0	0	0 COMPUTER
E	1	2	1	7	0 COMPUTER
TEST INSTANCES					
T	1	2	1	5	?

For the moment, we assume that we already have an estimate of the model parameters, i.e., the weights of the 4 features (and the bias θ_0) is $\widehat{\theta} = [\theta_0, \theta_1, \theta_2, \theta_3, \theta_4] = [0.2, 0.3, -2.2, 3.3, -0.2]$.

(i). Explain the intuition behind the model parameters, and their meaning in relation to the features

Feature engineering:

- Choose terms (as attributes)
- Define word occurence counts as attribute values.

LR:

$$P(y=1 \mid \underset{\sim}{x}) = \frac{1}{1+e^{-z}} = \sigma(z), \quad z = \theta_0 + \theta_1 x_1 + \cdots + \theta_4 x_4$$

Params:

$\theta_1, \theta_2, \theta_3, \theta_4$ → importance of 4 features (terms) for predicting class 1 (fruit).

θ_0 → bias (intercept)

(ii). Predict the test label.

$$\hat{z} = \hat{\theta}_0 + \hat{\theta}_1 x_1 + \cdots + \hat{\theta}_4 x_4$$

$$= 0.2 + 0.3 - 2.2 \times 2 + 3.3 - 0.2 \times 5$$

$$= -1.6$$

$$\sigma(-1.6) = \frac{1}{1+e^{1.6}} = 0.17 \quad \text{(for fruit)}$$

$0.17 < 0.5 \Rightarrow$ Classify as computer

(iii). Recall the conditional likelihood objective (Optional)

$$-\log \mathcal{L}(\theta) = -\sum_{i=1}^{n} y_i \log\big(\sigma(x_i;\theta)\big) + (1-y_i)\log\big(1-\sigma(x_i;\theta)\big)$$

We want to make sure that the Loss (the negative log likelihood) our model, is lower when its prediction the correct label for test instance T, than when it's predicting a wrong label.

Compute the negative log-likelihood of the test instance (1) assuming that the true label y = 0 (computer), i.e., our classifier made a mistake; and (2) assuming the true label as y = 1 (fruit), i.e., our classifier predicted correctly.

Predict $\hat{y} = 0$ (computer):

(1) If $y = 0$:

$$-\log \mathcal{L}(\theta) = -\{0 \cdot \log(\sigma(x_i;\theta)) + 1 \cdot \log(1-\sigma(x_i;\theta))\}$$

$$= -\log(1-\sigma(x_i;\theta))$$

$$= -\log(1-0.17)$$

$$= 0.19 \quad \text{(lower loss)}$$

(2) If $y = 1$:

$$-\log \mathcal{L}(\theta) = -\{1 \cdot \log(\sigma(x_i;\theta)) + 0 \cdot \log(1-\sigma(x_i;\theta))\}$$

$$= -\log(\sigma(x_i;\theta))$$

$$= -\log(0.17)$$

$$= 1.77$$

3. 4. For the model created in question ~~4~~ 2, compute a single gradient descent update for parameter θ_1 given the training instances given above. Recall that for each feature j, we compute its weight update as

$$\theta_j \leftarrow \theta_i - \eta \sum_i (\sigma(x_i;\theta) - y_i)x_{ij}$$

$\frac{\partial}{\partial\theta} - L(\theta)$

Summing over all training instances i. We will compute the update for θ_j assuming the current parameters as specified above, and a learning rate $\eta = 0.1$.

$\underset{\sim}{\theta}$: = [0.2 , 0.3 , -2.2 , 3.3, -0.2]

① Compute $\sigma(x_i;\theta)$ for all i (training instances)

x_{1A} to x_{1E}

$$\sigma(x_A;\theta) = \sigma(0.2 + (0.3 \times 1 + (-2.2) \times 0 + 3.3 \times 1 + (-0.2) \times 5)) = 0.94$$

$$\sigma(x_B;\theta) = \sigma(0.2 + (0.3 \times 1 + (-2.2) \times 0 + 3.3 \times 1 + (-0.2) \times 2)) = 0.97$$

$$\sigma(x_C;\theta) = \sigma(0.2 + (0.3 \times 2 + (-2.2) \times 0 + 3.3 \times 0 + (-0.2) \times 1)) = 0.65$$

$$\sigma(x_D;\theta) = \sigma(0.2 + (0.3 \times 2 + (-2.2) \times 2 + 3.3 \times 0 + (-0.2) \times 0)) = 0.03$$

$$\sigma(x_E;\theta) = \sigma(0.2 + (0.3 \times 1 + (-2.2) \times 2 + 3.3 \times 1 + (-0.2) \times 7)) = 0.12$$

② Update params (e.g. θ_1)

$$\theta_1 = \theta_1 - \eta \sum_{i\in\{A,B,C,D,E\}} (\sigma(x_i;\theta) - y_i)\, x_{1i}$$

$$\theta_1 = 0.3 - 0.1 \sum_{i\in\{A,B,C,D,E\}} (\sigma(x_i;\theta) - y_i)\, x_{1i}$$

η (learning rate)

$$\theta_1 = 0.3 - 0.1\,[((\sigma(x_A;\theta) - y_A).x_{1A}) + ((\sigma(x_B;\theta) - y_B).x_{1B}) + ((\sigma(x_C;\theta) - y_C).x_{1C})$$

$$+((\sigma(x_D;\theta) - y_D).x_{1D}) + ((\sigma(x_E;\theta) - y_E).x_{1E})] \quad \Sigma$$

$$= 0.3 - 0.1\,[((0.94 - 1) \times 1) + ((0.97 - 1) \times 1) + ((0.65 - 1) \times 2) + ((0.03 - 0) \times 2)$$

$$+((0.12 - 0) \times 1)]$$

$$= 0.3 - 0.1\big((-0.06) + (-0.03) + (-0.70) + 0.06 + 0.12\big) = 0.3 - 0.1\,(-0.61)$$

$$= 0.3 + 0.061 = 3.061 \quad \text{(new } \theta_1\text{)}$$

⇒ Do same thing for other θ's.

Note: update all params at once.

4.

~~4~~. [OPTIONAL] What is the relation between "odds" and "probability"? (optional)

prob : $p = p(\text{success})$

odds : ratio of $p(\text{success})$ to $p(\text{failure})$: $\frac{p}{1-p}$

E.g. 8 balls : 5 red $\Rightarrow$ $p(\text{red}) = \frac{5}{8}$

odds of drawing red ball :

$$\text{odds} = \frac{\frac{5}{8}}{1-\frac{5}{8}} = \frac{\frac{5}{8}}{\frac{3}{8}} = \frac{5}{3} = 1.7$$

(optional)

5.

~~5~~. Why is a perceptron (which uses a **sigmoid** activation function) equivalent to *logistic regression*?

- Has a weight associated with each input (attribute)
- Output is acquired by applying an activation function
  - If use sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$ to linear comb of inputs $\theta_0 + \theta_1 x_1 + \cdots \Rightarrow \sigma(\underset{\sim}{\theta}^T \underset{\sim}{x})$: logistic regression

6.

2. Consider the following training set:

(x_1, x_2)	y
(0,0)	0
(0,1)	1
(1,1)	1

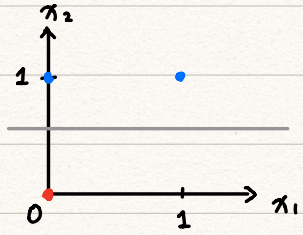
With the bias value of 1, the initial weight function of $\theta = \{\theta_0, \theta_1, \theta_2\} = \{0.1, 0.1, -0.3\}$ and learning rate of $\eta = 0.2$.

Consider the activation function of the perceptron as the step function $f = \begin{cases} 1 & if\ \Sigma > 0 \\ 0 & otherwise \end{cases}$

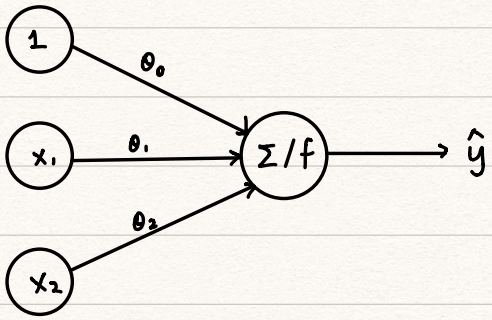
a) Can the perceptron learn a perfect solution for this data set?

Linearly separable?

Y ⇒ Can learn perfect solution.



b) Draw the perceptron graph and calculate the accuracy of the perceptron on the training data before training?



Acc : $\theta = \{0.2, -0.4, 0.1\}$

$f = \begin{cases} 1 & \text{if } \Sigma > 0 \\ 0 & \text{otherwise} \end{cases}$

(x_1, x_2)	$\Sigma = \theta_0 + \theta_1 x_1 + \theta_2 x_2$	$\hat{y} = f(\Sigma)$	y
(0,0)	0.2	1	0
(0,1)	0.2 + 0.1 = 0.3	1	1
(1,1)	0.2 − 0.4 + 0.1 = −0.1	0	1

Acc : $\frac{1}{3}$

c) Using the perceptron *learning rule* and the learning rate of $\eta = 0.2$. Train the perceptron **for one epoch**. What are the weights after the training?

Perceptron weight training rule: (for each instance) (online algorithm, contrast to batch algorithm e.g. NB, LR with GD)

$$\theta_j^{(t)} \leftarrow \theta_j^{(t-1)} + \eta (y^i - \hat{y}^{i,(t)}) x_j^i$$

Can be more efficient for large dataset

for $\eta = 0.2$

$\theta = \{0.2, -0.4, 0.1\}$

(x_1, x_2)	$\Sigma = \theta_0 + \theta_1 x_1 + \theta_2 x_2$	$\hat{y} = f(\Sigma)$	y	
(0, 0)	0.2	1	0	✗
	Update θ: $\theta_0^{(1)} = \theta_0^{(0)} + \eta(y^1 - \hat{y}^1) x_0^1$ $= 0.2 + 0.2(0-1)\cdot 1 = 0$ $\theta_1^{(1)} = \theta_1^{(0)} + \eta(y^1 - \hat{y}^1) x_1^1 = -0.4$ $\theta_2^{(1)} = \theta_2^{(0)} + \eta(y^1 - \hat{y}^1) x_2^1 = 0.1$			
(0, 1)	$0 - 0.4\times 0 + 0.1 = 0.1$	1	1	✓
	Correct ⇒ No update			
(1, 1)	$0 - 0.4\times 1 + 0.1 = -0.3$	0	1	✗
	Update θ: (instance 3) $\theta_0^{(1)} = \theta_0^{(0)} + \eta(y^3 - \hat{y}^3) x_0^3$ $= 0 + 0.2(1-0)\cdot 1 = 0.2$ $\theta_1^{(1)} = \theta_1^{(0)} + \eta(y^3 - \hat{y}^3) x_1^3$ $= -0.4 + 0.2(1-0)\cdot 1 = -0.2$ $\theta_2^{(1)} = \theta_2^{(0)} + \eta(y^3 - \hat{y}^3) x^3$ $= 0.1 + 0.2(1-0)\cdot 1 = 0.3$			

$\theta^{(1)} = \{0.2, -0.2, 0.3\}$

d) What is the accuracy of the perceptron on the training data after training for one epoch? Did the accuracy improve?

Acc : $\theta = \{0.2, -0.2, 0.3\}$

$f = \begin{cases} 1 & \text{if } \Sigma > 0 \\ 0 & \text{otherwise} \end{cases}$

(x_1, x_2)	$\Sigma = \theta_0 + \theta_1 x_1 + \theta_2 x_2$	$\hat{y} = f(\Sigma)$	y
$(0,0)$	0.2	1	0
$(0,1)$	$0.2 + 0.3 = 0.5$	1	1
$(1,1)$	$0.2 - 0.2 + 0.3 = 0.3$	1	1

Acc : $\frac{2}{3}$