Logistic Regression

Mingjun Zhong
Department of Computing Science
University of Aberdeen

Classification example: cat and dog classification

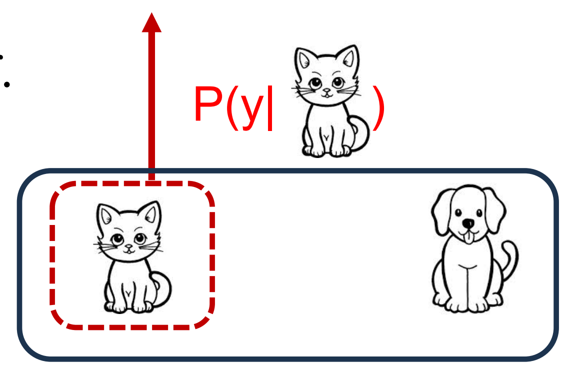
- Images are 28*28 pixels
- Represent input image as a vector $x \in R^{784}$ (i.e., 28x28=784 pixels)
- The target is $y \in \{0, 1\}$. Cat: y=0, and Dog: $y = 1$
- Learn a classifier f (x) such that,

$$f\colon \text{x} \rightarrow \{0, 1\}$$

- As the target is a discrete value, we assign a probability to each x, for example:

$$P(y = 0|x) = 0.9, P(y = 1|x) = 0.1$$

- And $\sum_{i=0}^{1} P(y = i|x) = 1.0$
- We want to find such map f.



$$P(y = 0|x) = 0.9$$

When the input image is x (in this example, a cat image), the probability of y=0 is 0.9.

Classification

- **Learn**: f: **X**->Y
  - **X** – features (images in image classification)
  - Y – target classes
- Suppose you know P(Y|**X**) exactly, how should you classify?
  - Bayes classifier: $y^* = f(x) = \underset{y}{\operatorname{argmax}} P(Y = y|X = x)$
  - The notation `argmax' means finding the `argument' y which `maximises' the function P(Y=y|X=x)
- For example:

$$P(y = 0|x) = \mathbf{0.9}, P(y = 1|x) = 0.1$$

then $y^* = 0$ (which means the classifier recognized x as a cat)

- We can see that the key is to know how to compute the probability $P(Y|X)$
- Two approaches to estimate $P(Y|X)$: 1) Generative Classifier, and 2) Discriminative Classifier

Generative classifier

- Also called ***Naïve Bayes***:
  - Target: to compute the probability $P(Y|X)$
  - Using ***Bayes rule***:

$$P(Y|X) = \frac{p(X|Y)P(Y)}{P(X)}$$

  - Assume some functional form for **P(X|Y), P(Y)**
  - Estimate parameters of P(X|Y), P(Y) directly from training data
  - Then calculate P(Y|X=x) using Bayes rule
  - This is 'generative' model
    - Indirect computation of P(Y|X) through Bayes rule
    - It can generate a sample of the data using:

$$P(X) = \sum_y P(y)P(X \mid y)$$

Discriminative classifier

- Discriminative classifier, e.g., Logistic Regression:
  - Assume some functional form for **P(Y|X)**
  - Estimate parameters of P(Y|X) directly from training data
  - This is the 'discriminative' model
    - Directly learn P(Y|X)
    - But cannot sample data, because P(X) is not available
    - Neural networks (deep learning) are discriminative classifiers
    - Logistic regression is the simplest neural networks

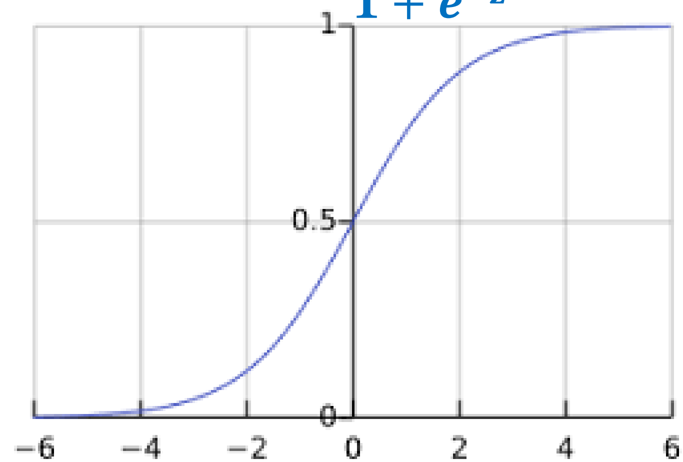
Logistic Regression

- We do binary classification – two classes
- Let X be the data instance, and Y be the class label: Learn P(Y|X) directly
  - Sigmoid function: $\boldsymbol{S(z) = \frac{1}{1+e^{-z}}}$
  - Logistic regression:
    - Set $Z = WX$
    - Here $WX = w_1x_1 + w_2x_2 + \cdots + w_nx_n$ is a linear function
    - In the image example, $n = 784$. Each pixel has a weight w_i
    - Then

$$P(Y = 1|\mathrm{X}) = \frac{1}{1 + e^{-\mathrm{wx}}}$$

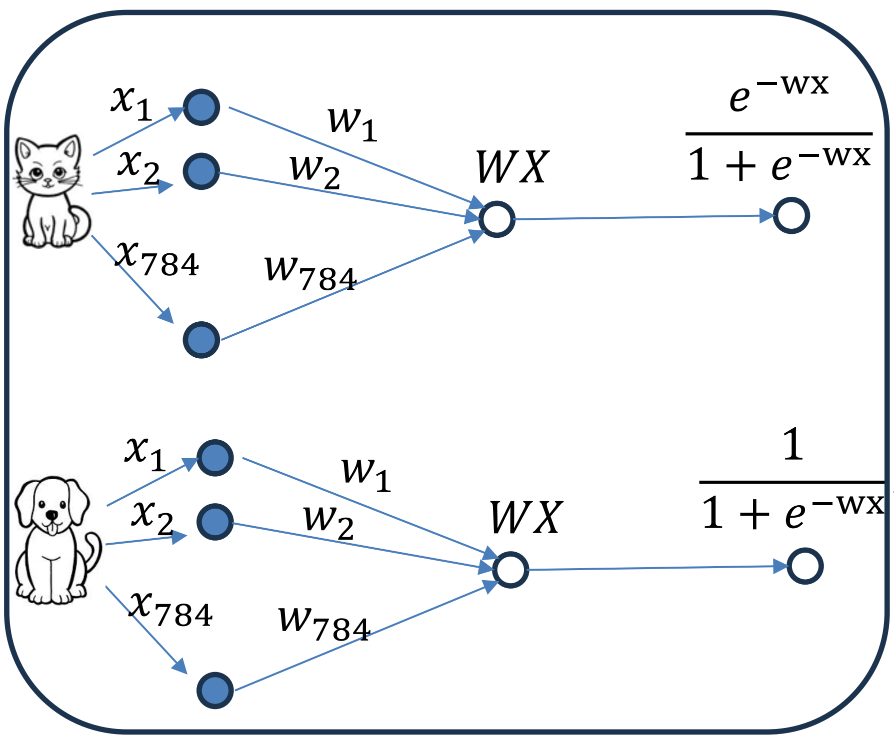
$$P(Y = 0|\mathrm{X}) = 1 - \frac{1}{1+e^{-\mathrm{wx}}} = \frac{e^{-\mathrm{wx}}}{1+e^{-\mathrm{wx}}}$$

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



Logistic Regression

- We sum up the sigmoid functions of all the examples for the cat and the dog
- Before summing up, we apply the logarithmic function to the sigmoid function, only for easier computation
- The loss function is then
- $loss(w) = \sum_{cat} \log(\frac{e^{-\mathrm{wx}_{cat}}}{1+e^{-\mathrm{wx}_{cat}}}) + \sum_{dog} \log(\frac{1}{1+e^{-\mathrm{wx}_{dog}}})$



Logistic Regression

- The loss function $loss(w) = \sum_{cat} \log(\frac{e^{-\mathrm{w}\mathrm{x}_{cat}}}{1+e^{-\mathrm{w}\mathrm{x}_{cat}}}) + \sum_{dog} \log(\frac{1}{1+e^{-\mathrm{w}\mathrm{x}_{dog}}})$
- We need to find the best w to maximize the loss function (why?)
- We can use gradient descent algorithm to find the best w.
- TensorFlow and PyTorch can be used to find w.
- After the optimal parameters are found, denote it as w^*.
- For predicting the class of a new image X_{new}, we compute $P(Y = 1|\mathrm{X}_{new}) = \frac{1}{1+e^{-\mathrm{w}^*\mathrm{x}_{new}}}$ and $P(Y = 0|\mathrm{X})$= $\frac{e^{-\mathrm{w}^*\mathrm{x}_{new}}}{1+e^{-\mathrm{w}^*\mathrm{x}_{new}}}$
- Then decide the class label of the new image

Regularization in Logistic Regression

- Overfitting the training data is a problem that can arise in Logistic Regression, especially when data has very high dimensions and is sparse.

- One approach to reducing overfitting is regularization, in which we create a modified "penalized loss function," which penalizes large values of **w**.

$$w^* = \underset{\mathrm{w}}{\operatorname{argmax}}\,(loss(w) - \frac{\lambda}{2}||\mathrm{w}||^2)$$

- Again, the best w can be found using gradient descent algorithm

What you should know LR

- In general, NB and LR make different assumptions
  - NB: Features independent given class -> assumption on P(X|Y)
  - LR: Functional form of P(Y|X), no assumption on P(X|Y)
- LR is a linear classifier
  - decision rule is a hyperplane
- LR optimized by conditional likelihood
  - no closed-form solution
  - concave -> global optimum with gradient ascent

Summary of Logistic Regression

- Learns the Conditional Probability Distribution P(y|x)
- Local Search using gradient descent
  - Begins with initial weight vector.
  - Modifies it iteratively to maximize the loss function.
  - The loss function is the conditional log likelihood of the data – so the algorithm seeks the probability distribution P(y|x) that is most likely given the data.