Gradient Descent Learning (II): Classification

Shan He

School for Computational Science
University of Birmingham

Module 06-27818 and 27819:
Introduction to Neural Computation (Level 4/M)
Neural Computation (Level 3)

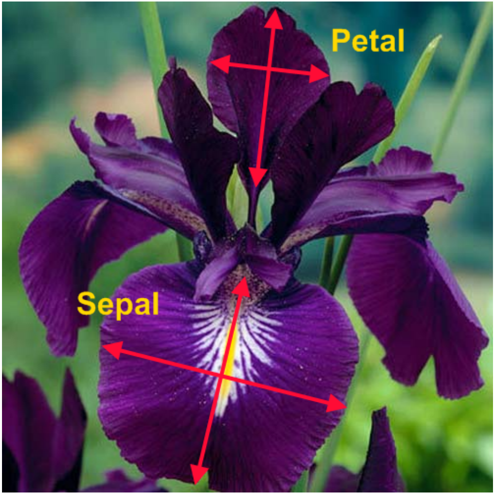
Outline of Topics

Classification

Logistic function

Solving Classification problems using ANNs

An motivating example: Iris classification



Classification: The Iris dataset

- The data set consists of 50 samples (training data) from each of three species of Iris:
  - Iris-setosa
  - Iris-virginica
  - Iris-versicolor
- Measures of four features of each samples in cm:
  - Petal length
  - Petal width
  - Sepal length
  - Sepal width
- Task: to train a model to learn from the training data to classify unseen Iris flowers based on the four features
- Download the dataset and Detailed explanation of the dataset

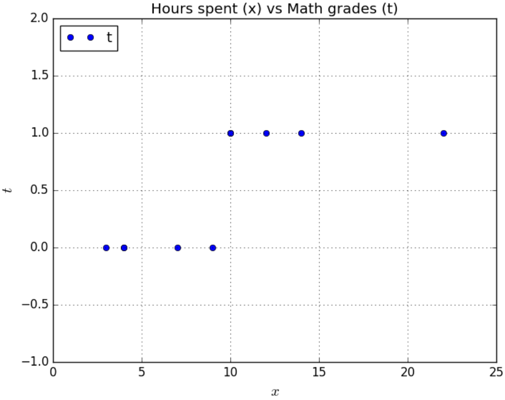
Classification: a less interesting simple problem

- The Time spent and Math score/grade problem:

Student ID	1	2	3	4	5	6	7	8	9	10
Hours spent	4	9	10	14	4	7	12	22	3	10
Math Score	39	58	65	73	41	50	60	79	40	64
Grades	F	F	P	P	F	F	P	P	F	P

- Task: to train a classification model based on the hours spent and grades to predict unseen students' grades based on the hours spent

Logistic function



Classification Problems

- **Classification**: determining the most likely class that an input pattern belongs to.
- **Formally**: modelling the posterior probabilities of class membership conditioned on the input variables.
- Artificial neural networks: one output unit for each class, and for each input pattern we have
  - 1 for the output unit corresponding to that class
  - 0 for all the other output units
- The simplest case: binary classification $\rightarrow$ one output unit
  - 1: passed
  - 0: failed

Logistic regression

- **Logistic regression**: also known as logit regression, is a regression model where the prediction (dependent variable) is categorical, e.g., binary.
- **Goal**: to predict the probability that a given example belongs to the 1 class versus the probability that it belongs to the 0 class.

Logistic regression

- **Logistic regression**: to learn a function of the form:

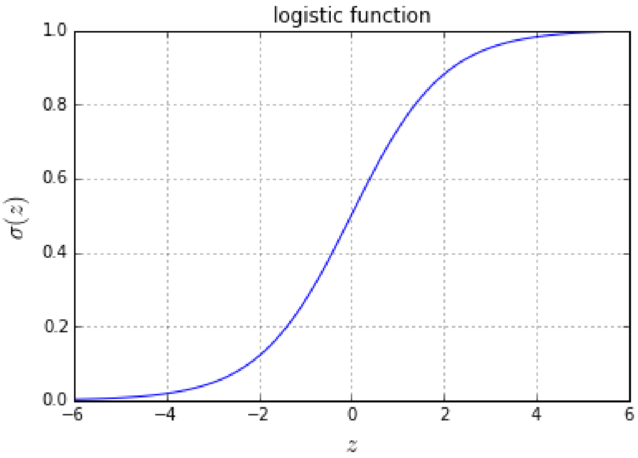
$$P(y=1|\mathbf{x}) = \frac{1}{1+exp(-\mathbf{w}^{\mathrm{T}}\mathbf{x})} \equiv \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

  and

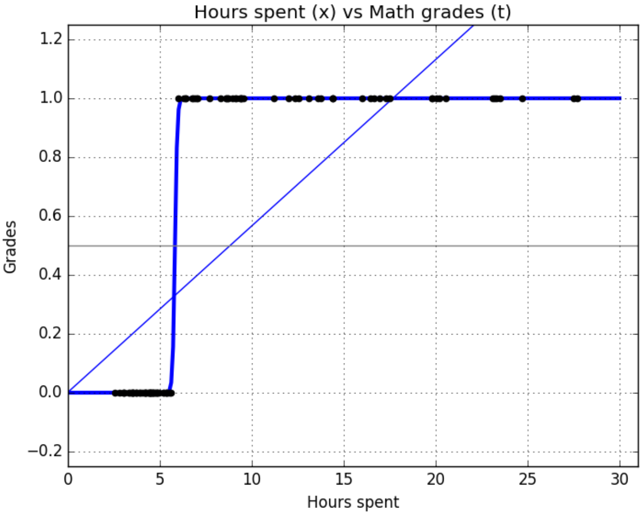
$$P(y=0|\mathbf{x}) = 1 - P(y=1|\mathbf{x})$$

- The function $\sigma(x) = \frac{1}{1+exp(-x)}$ is often called the "logistic" or "sigmoid" function
- We usually use Maximum Likelihood estimation to obtained **w**

Logistic function



Logistic function

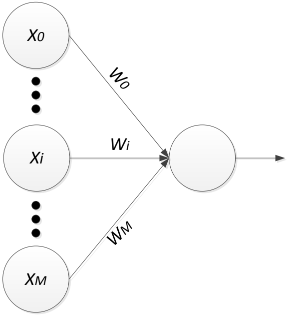


Solving Classification problems using ANNs?

- How to solve this simple linear regression using neural networks.
- We need three things to define a neural networks:
  - **Network topology:** to define how neurons are connected by weights
  - **Activation function:** to convert a neuron's weighted input to its output activation
  - **Learning process:** to update the weights
    - **Essence of supervised learning process**: adjusting the network weights w_{ij} to minimise a **cost function**

Solving classification problems using a percetron?

- Network topology: Single Layer Networks
- Assuming binary classification: one output neuron



- Activation function
  - Question: can we use the simple linear activation function $f(x) = x$?

Solving classification problems using a percetron?

- Activation function
  - Question: can we use the simple linear activation function $f(x) = x$?
    - Answer: No. Linear function outputs a continuous value, not a categorical output i.e., Iris-setosa or Iris-virginica, or Passed/Failed.
  - We need different activation functions output categorical values:
    - Logistic function or Sigmoid function:

$$y = f(\mathbf{w}^{\mathrm{T}}\mathbf{x}) = \frac{1}{1 + exp(-\mathbf{w}^{\mathrm{T}}\mathbf{x})} \equiv \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

Learning process

- **Learning process:** to update the weights
  - **Essence of supervised learning process**: adjusting the network weights w_{ij} to minimise a **cost function**

Gradient Descent Learning

- **Aim**: to develop a learning algorithm that minimises a cost function (such as Sum Squared Error) by making appropriate **iterative adjustments** to the weights w_{ij}.
- **Idea**: to apply a series of small updates to the weights $w_{ij}^{t+1} = w_{ij}^{t} + \Delta w_{ij}$ until the cost $E(w_{ij})$ is "small enough".
- **Question**: how to obtain Δw_{ij}
- **Answer**: $\Delta w_{ij} = -\eta \frac{\partial E(w_{ij})}{\partial w_{ij}}$
- **Explanation**: we repeatedly adjust the weights by small steps against the gradient, we will move through weight space, descending along the gradients towards a minimum of the cost function.

Cross Entropy Cost Function for Two Classes

- Reminder: the Sum Squared Error cost function is for regression problems $\rightarrow$ we need a new cost function for classification problems
- If the output $\mathbf{y}$ of a network represents the probability of a particular class, and $\mathbf{t}$ is the **binary** target output, the probability of observing the whole training data set is:

$$P(\mathbf{t}|\mathbf{x}) = \prod_{p=1}^{P} y_p^{t_p} \cdot (1 - y_p)^{1-t_p}$$

- The ANN model aims to maximise this probability
- It is more convenient to work with a negative log-likelihood function since it is monotonically decreasing:

$$-\log \mathcal{L}(\mathbf{w}|t, y) = -\sum_{p} [t^p \log(y^p) + (1 - t^p) \log(1 - y^p)]$$

Cross Entropy Cost Function for Two Classes

- We define the Cross Entropy Cost Function

$$E_{ce} = -\log \mathcal{L}(\mathbf{w}|t, y) = -\sum_p [t^p \log(y^p) + (1 - t^p) \log(1 - y^p)]$$

$$= -\sum_p [t^p \log(\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})) + (1 - t^p) \log(1 - \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}))$$

- The cost function looks very complicated but we shall see some tricks to derive its derivative

Gradient Descent Learning for Cross Entropy Cost Function

- Cross Entropy Cost Function for the p-th training sample $(\mathbf{x}^p, t^p)$:

$$\Delta \mathbf{w}^p = -\eta \frac{\partial E_{ce}^p(\mathbf{w})}{\partial \mathbf{w}}$$

- Reminder: Using sigmoid activation: $y^p = \sigma(\mathbf{w}^{\mathrm{T}} \mathbf{x}^p)$.
- Using the chain rule:

$$\Delta \mathbf{w}^p = -\eta \frac{\partial E_{ce}^p(\mathbf{w})}{\partial \mathbf{w}} = -\eta \frac{\partial E_{ce}^p(\mathbf{w})}{\partial y^p} \frac{\partial y^p}{\partial \mathbf{w}} = -\eta \frac{\partial E_{ce}^p(\mathbf{w})}{\partial y^p} \frac{\partial \sigma(\mathbf{w}^{\mathrm{T}} \mathbf{x}^p))}{\partial \mathbf{w}}$$

$$= -\eta \frac{\partial E_{ce}^p(\mathbf{w})}{\partial y^p} \frac{\partial \sigma(\mathbf{w}^{\mathrm{T}} \mathbf{x}^p))}{\partial \mathbf{w}^{\mathrm{T}} \mathbf{x}^p} \frac{\partial \mathbf{w}^{\mathrm{T}} \mathbf{x}^p}{\partial \mathbf{w}} = -\eta \frac{\partial E_{ce}^p(\mathbf{w})}{\partial y^p} \sigma' \mathbf{x}^p$$

The Derivative of a Sigmoid: σ'

- Let $\sigma(x) = \frac{1}{1+exp(-x)} = g(h(x))$ with $g(h) = h^{-1}$ and $h(x) = 1 + exp(-x)$, so

$$\frac{\partial g(h)}{\partial h} = -\frac{1}{h^2}, \quad \frac{\partial h(x)}{\partial x} = -exp(-x)$$

- Using the chain rule:

$$\sigma' = \frac{\partial \sigma(x)}{\partial x} = \frac{\partial g(h)}{\partial h}\frac{\partial h(x)}{\partial x} = -\frac{1}{h^2} \cdot -exp(-x) = \frac{exp(-x)}{(1+exp(-x))^2}$$

$$= \frac{1+exp(-x)-1}{(1+exp(-x))^2} = \frac{1}{(1+exp(-x))} - \frac{1}{(1+exp(-x))^2}$$

$$= \sigma(x)(1-\sigma(x))$$

Question: Why use Sigmoid function?

- Question: Why not use step function?

Cross Entropy Cost Function for Two Classes

- We then derive

$$\frac{\partial E_{ce}^{p}}{\partial y^{p}} = -\frac{\partial(t^{p}\log(y^{p}) + (1-t^{p})\log(1-y^{p}))}{\partial y^{p}}$$
$$= -\left[\frac{\partial(t^{p}\log(y^{p}))}{\partial y^{p}} + \frac{\partial((1-t^{p})\log(1-y^{p}))}{\partial y^{p}}\right]$$

- Since:

$$\frac{\partial(t^{p}\log(y^{p}))}{\partial y^{p}} = \frac{\partial(t^{p}\log(y^{p}))}{\partial\log(y^{p})}\frac{\partial\log(y^{p})}{\partial y^{p}} = \frac{t^{p}}{y^{p}}$$

and

$$\frac{\partial((1-t^{p})\log(1-y^{p}))}{\partial y^{p}} = -\frac{1-t^{p}}{1-y^{p}}$$

- Finally:

$$\frac{\partial E_{ce}^{p}}{\partial y^{p}} = -\frac{t^{p}}{y^{p}} + \frac{1-t^{p}}{1-y^{p}} = \frac{y^{p}-t^{p}}{y^{p}(1-y^{p})}$$

Cross Entropy Cost Function for Two Classes

- Since $y^p = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}^p)$, and we know:

$$\sigma'(\mathbf{w}^{\mathrm{T}}\mathbf{x}^p) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}^p)(1 - \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}^p)) = y^p(1 - y^p)$$

- Putting them together:

$$\Delta\mathbf{w}^p = -\eta\frac{\partial E_{ce}^p(\mathbf{w})}{\partial y^p}\sigma'\mathbf{x}^p = -\eta\frac{y^p - t^p}{y^p(1 - y^p)}y^p(1 - y^p)\mathbf{x}^p$$

$$= \eta(t^p - y^p)\mathbf{x}^p$$

- In the batch processing, we just add up all the gradients for each sample:

$$\Delta\mathbf{w} = \eta\sum_{p=1}^{P}(t^p - y^p)\mathbf{x}^p$$

Conclusion

- We learned how to solve classification problems using a perceptron with sigmoid output activations and a Cross Entropy cost function
  - Bishop: Sections 3.1, 6.1, 6.7, 6.8, 6.9
  - Haykin-1999: Sections 3.5, 3.7, 4.4, 4.6