Probabilistic Discriminative Models: Fixed Basis Functions

Karan Nathwani

Topics in Linear Classification using Probabilistic Discriminative Models

- Distinction between Generative vs Discriminative

1. Fixed basis functions
2. Logistic Regression (two-class)
3. Iterative Reweighted Least Squares (IRLS)
4. Multiclass Logistic Regression
5. Probit Regression
6. Canonical Link Functions

Probabilistic Generative Models

- **Two-class classification:** Posterior of class C_1: $p(C_1|\boldsymbol{x})$ can be written as a logistic sigmoid operating on linear function of $\boldsymbol{x}$, i.e., $\sigma(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0)$, for wide choice of forms for $p(\boldsymbol{x}|C_k)$
  - For Gaussians with same covariance matrix $\boldsymbol{w} = \Sigma^{-1}(\mu_1 - \mu_2)$ $w_0 = -\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 + \ln\frac{p(C_1)}{p(C_2)}$
  - For Discrete binary features x_i
    - which are linear functions of features

$$a_k(\boldsymbol{x}) = \ln(p(\boldsymbol{x} \mid C_k)p(C_k))$$
$$= \sum_{i=1}^{D}\left\{x_i \ln \mu_{ki} + (1-x_i)\ln(1-\mu_{ki}\right\} + \ln p(C_k)$$

- **Multiclass case:**
  - Posterior probability of class C_k i.e.,
    - $p(C_k|\boldsymbol{x})$ is given by a softmax transformation of a linear function of $\boldsymbol{x}$

$$p(C_k \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_k)p(C_k)}{\sum_j p(\boldsymbol{x} \mid C_j)p(C_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

Normalized Exponential Or Softmax

- **MLE used to get parameters of $p(\boldsymbol{x}|C_k)$ and $p(C_k)$**
  - This indirect approach to find parameters of a generalized linear model is called generative
    - Since we can use such a model to generate synthetic data from marginal $p(\boldsymbol{x})$

Probabilistic Discriminative Models

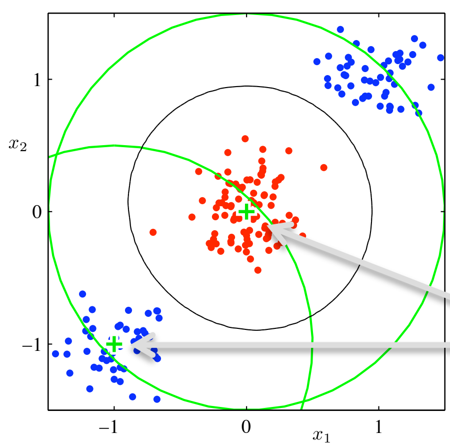
- An alternative approach
  - Use the functional form of the generalized linear model explicitly
  - Determine its parameters using maximum likelihood
  - There is an efficient algorithm to find such solutions known as Iterative reweighted least squares (IRLS)
- Advantages
  - Fewer adaptive parameters
  - Improved predictive performance when $p(\mathbf{x}|C_k)$ assumptions are poor approximations

Fixed Basis Functions

Although we use linear classification models

Linear-separability in feature space does not imply linear-separability in input space

Original Input Space (x_1,x_2)

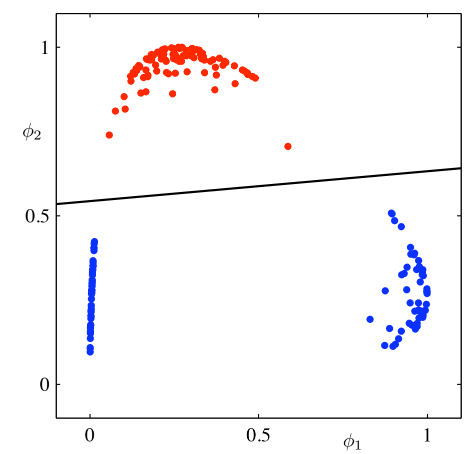


not linearly separable

Nonlinear transformation of inputs using vector of basis functions $\boldsymbol{\varphi}(\boldsymbol{x})$

Basis functions are Gaussian with centers Shown by crosses and Green contours

Feature Space (ϕ_1,ϕ_2)



linearly separable

Basis functions with increased dimensionality is often used

Limitation of Fixed Basis Functions

- Nonlinear transformations cannot remove overlap between classes
  - They can even increase the overlap!
  - Still fixed nonlinear basis functions play an important role
- One solution: basis functions that adapt to data
  - SVMs use basis functions centered on the data points and select a fixed subset of them

Logistic Regression

Karan Nathwani

Topics in Logistic Regression

- Logistic Sigmoid and Logit Functions
- Parameters in discriminative approach
- Determining logistic regression parameters
  - Error function
  - Gradient of error function
  - Simple sequential algorithm
  - An example
- Generative vs Discriminative Training
  - Naiive Bayes vs Logistic Regression

Logistic Sigmoid and Logit Functions

- In two-class case, *posterior* of class C_1 can be written as as a logistic sigmoid of feature vector $\phi=[\phi_1,..\phi_M]^T$

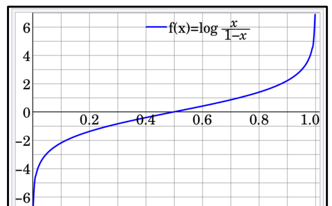
  $$p(C_1|\phi) = y(\phi) = \sigma\,(\boldsymbol{w}^T\phi)$$

  with $p(C_2|\phi) = 1 - p(C_1|\phi)$

  Here σ (.) is the logistic sigmoid function

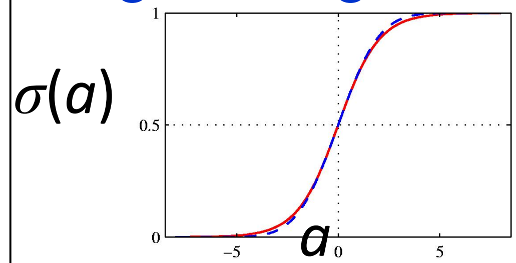
  – Known as logistic regression in statistics
    - Although a model for classification rather than for regression

- Logit function:
  – It is the log of the odds ratio
    - It links the probability to the predictor variables



Plot of logit(p) in the domain of 0 to 1, where the base of logarithm is e

Logistic Sigmoid



Properties:

A. Symmetry

$$\sigma\,(-a) = 1 - \sigma\,(a)$$

B. Inverse

$$a = \ln(\sigma\,/1-\sigma)$$

known as *logit*.
Also known as *log odds* since it is the ratio

$$\ln[p(C_1/\phi)/p(C_2/\phi)]$$

C. Derivative

$$d\sigma/da = \sigma\,(1-\sigma)$$

Fewer Parameters in Linear Discriminative Model

- Discriminative approach (Logistic Regression)
  - For M -dim feature space ϕ:
  - M adjustable parameters
- Generative based on Gaussians (Bayes/NB)
  - $2M$ parameters for mean
  - $M(M+1)/2$ parameters for shared covariance matrix
  - Two class priors
  - Total of $M(M+5)/2 + 1$ parameters
    - Grows quadratically with M
  - If features assumed independent (naïve Bayes) still needs $M+3$ parameters

Determining Logistic Regression parameters

- Maximum Likelihood Approach for Two classes
- For a data set $(\boldsymbol{\phi}_n, t_n)$ where $t_n \,\varepsilon\, \{0,1\}$ and

$\boldsymbol{\phi}_n = \boldsymbol{\phi}(\boldsymbol{x}_n)$, $n = 1,..,N$

- Likelihood function can be written as

$$p(\mathrm{t} \mid \boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n} \left\{1 - y_n\right\}^{1-t_n}$$

where $\mathrm{t} = (t_1,..,t_N)^{\mathrm{T}}$ and $y_n = p(C_1|\boldsymbol{\phi}_n)$

y_n is the probability that $t_n = 1$

Error Fn for Logistic Regression

- Likelihood function is

$$\boxed{p(\mathrm{t} \mid \boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n} \left\{1 - y_n\right\}^{1-t_n}}$$

- By taking negative logarithm we get the
  Cross-entropy Error Function

$$\boxed{E(\boldsymbol{w}) = -\ln p(\mathrm{t} \mid \boldsymbol{w}) = -\sum_{n=1}^{N} \left\{ t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right\}}$$

where $y_n = \sigma(a_n)$ and $a_n = \boldsymbol{w}^T \boldsymbol{\phi}_n$

- We need to minimize $E(\boldsymbol{w})$
  At its minimum, derivative of $E(\boldsymbol{w})$ is zero
  So we need to solve for w in the equation

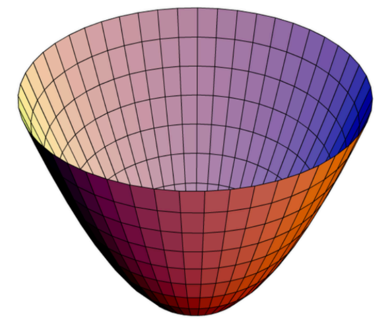
$$\boxed{\nabla E(\boldsymbol{w}) = 0}$$

Gradient of Error Function

Error function

$$E(\boldsymbol{w}) = -\ln p(t \mid \boldsymbol{w}) = -\sum_{n=1}^{N} \left\{ t_n \ln y_n + (1-t_n)\ln(1-y_n) \right\}$$

where $y_n = \sigma(\boldsymbol{w}^{\mathrm{T}} \phi_n)$



Using Derivative of logistic sigmoid $d\sigma/da = \sigma(1-\sigma)$

Gradient of the error function

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N} \underbrace{(y_n - t_n)\phi_n}_{\text{Error x Feature Vector}}$$

Contribution to gradient by data point n is error between target t_n and prediction $y_n = \sigma\,(\mathrm{w}^{\mathrm{T}}\varphi_n)$ times basis φ_n

Proof of gradient expression

$Let\ z = z_1 + z_2$

where $z_1 = t \ln \sigma(w\phi)$ and $z_2 = (1-t)\ln[1-\sigma(w\phi)]$

$$\frac{dz_1}{dw} = \frac{t\sigma(w\phi)[1-\sigma(w\phi)]\phi}{\sigma(w\phi)} \qquad \frac{d\sigma}{da} = \sigma(1-\sigma)$$

and

Using $\frac{d}{dx}(\ln ax) = \frac{a}{x}$

$$\frac{dz2}{dw} = \frac{(1-t)\sigma(w\phi)[1-\sigma(w\phi)](-\phi)}{[1-\sigma(w\phi)]}$$

Therefore $\frac{dz}{dw} = (\sigma(w\phi) - t)\phi$

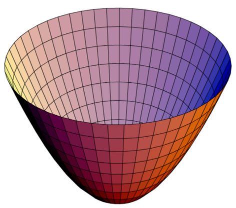
Simple Sequential Algorithm

- Given Gradient of error function

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N} \left(y_n - t_n\right)\phi_n$$

where $y_n = \sigma(\boldsymbol{w}^T \boldsymbol{\phi}_n)$

- Solve using an iterative approach



$$\boldsymbol{w}^{\tau+1} = \boldsymbol{w}^{\tau} - \eta \nabla E_n$$

- where

$$\nabla E_n = \underbrace{(y_n - t_n)\phi_n}_{\text{Error x Feature Vector}}$$

Takes precisely same form as Gradient of Sum-of-squares error for linear regression

Samples are presented one at a time in which each each of the weight vectors is updated

Python Code for Logistic Regression

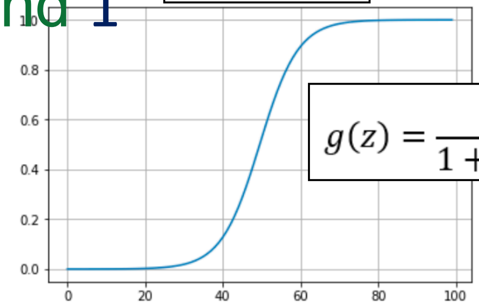
$$w^T.x + b = z$$

Sigmoid function to produce value between 0 and 1

```
def sigmoid(z):
    return (1 / (1 + np.exp(-z)))
```

Prediction

$$s(z) = p$$



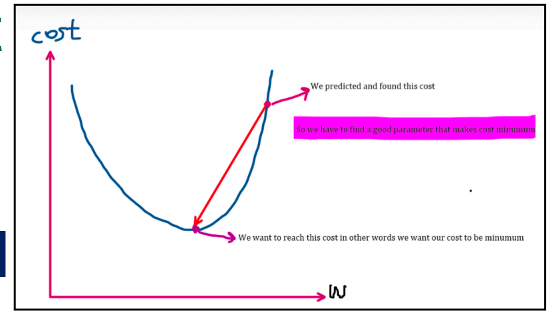
$$g(z) = \frac{1}{1+e^{-z}}$$

Loss and Cost function

Loss function is the loss for a training example

Cost is the loss for whole training set

z



$$L(p,y) = -(ylogp + (1-y)\,log(1-p))$$

Updating weights and biases

p is our prediction and y is correct val

$$\boxed{\begin{aligned} b &:= b - \alpha db \\ w &:= w - \alpha dw \end{aligned}}$$

Finding db and dw

Derivative wrt p → Derivative wrt z.

$$\frac{d}{dx}\log(mx) = \frac{(mx)'}{mx} = \frac{m}{mx} = \frac{1}{x}$$

$$\frac{\partial L(p,y)}{\partial p} = \frac{\partial}{\partial p}\Big(-\big(ylogp + (1-y)log(1-p)\big)\Big)$$

$$\frac{\partial L(p,y)}{\partial p} = \left(\frac{1-y}{1-p} - \frac{y}{p}\right)$$

$$\frac{\partial L(p,y)}{\partial z} = \frac{\partial L(p,y)}{\partial p}\frac{\partial p}{\partial z} \text{ ...chain rule}$$

$$\frac{\partial p}{\partial z} = p(1-p) \text{ because of the sigmoid function}$$

$$\frac{\partial L(p,y)}{\partial z} = \left(\frac{1-y}{1-p} - \frac{y}{p}\right)(p(1-p))$$

$$\boxed{\frac{\partial L(p,y)}{\partial z} = p - y}$$

$$\frac{\partial L(p,y)}{\partial b} = \frac{\partial L(p,y)}{\partial z}\frac{\partial z}{\partial b} = dz\frac{\partial}{\partial b}(w^T.x + b)$$

$$\frac{\partial L(p,y)}{\partial b} = db = \frac{1}{\underbrace{m}_{training\ example}}\sum_{k=1}^{m} dz^k$$

db

$$\frac{\partial L(p,y)}{\partial w} = \frac{\partial L(p,y)}{\partial z}\frac{\partial z}{\partial w} = dz\frac{\partial}{\partial w}(w^T.x + b)$$

$$\frac{\partial L(p,y)}{\partial w} = dw = \frac{1}{\underbrace{m}_{training\ example}} X dz^T$$

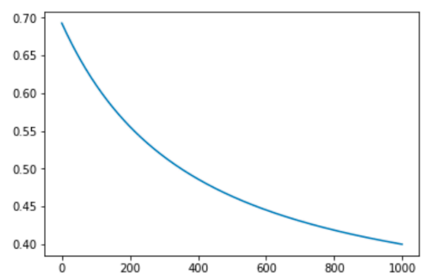
dw

https://towardsdatascience.com/logistic-regression-from-very-scratch-ea914961f320

Logistic Regression Code in Python

use sci-kit learn to create a data set.

```
import sklearn.datasets
import matplotlib.pyplot as plt
import numpy as np
X, Y = sklearn.datasets.make_moons(n_samples=500, noise=.2)
X, Y = X.T, Y.reshape(1, Y.shape[0])
epochs = 1000
learningrate = 0.01
def sigmoid(z):
        return 1 / (1 + np.exp(-z))
losstrack = []
m = X.shape[1]
w = np.random.randn(X.shape[0], 1)*0.01
b = 0
for epoch in range(epochs):
        z = np.dot(w.T, X) + b
        p = sigmoid(z)
        cost = -np.sum(np.multiply(np.log(p), Y) + np.multiply((1 - Y), np.log(1 - p)))/m
        losstrack.append(np.squeeze(cost))
        dz = p-Y
dw = (1 / m) * np.dot(X, dz.T)
db = (1 / m) * np.sum(dz)
w = w - learningrate * dw
b = b - learningrate * db
plt.plot(losstrack)
```

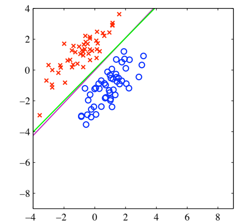


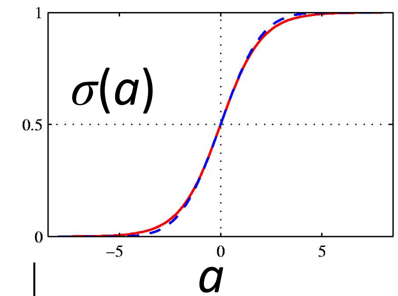
for epoch in range(epochs):

$$z = w^T . x + b$$
$$p = s(z)$$
$$dz = p - y$$
$$dw = \frac{1}{m} X dz^T$$
$$db = \frac{1}{m} \sum_{k=1}^{m} dz^k$$
$$b := b - \alpha db$$
$$w := w - \alpha dw$$

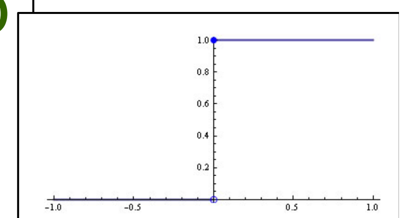
Prediction: From the code above, you find p. It will be between 0 a

ML solution can over-fit

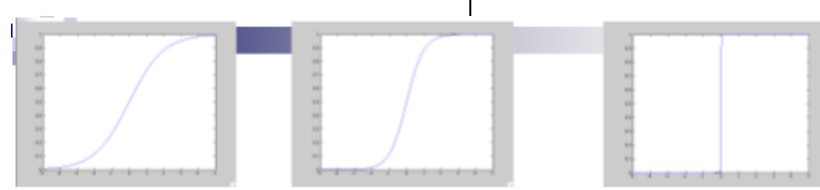




- Severe over-fitting for linearly separable data
  - Because ML solution occurs at $\sigma = 0.5$
    - With $\sigma>0.5$ and $\sigma< 0.5$ for the two classes
    - Solution equivalent to $a=\mathbf{w}^T\phi = 0$
  - Logistic sigmoid becomes infinitely steep
    - A Heavyside step function $||\mathbf{w}||$ goes to ∞



  - Solution
    - Penalizing wts
    - Recall in linear regression



$$\frac{1}{1+e^{-x}} \qquad \frac{1}{1+e^{-2x}} \qquad \frac{1}{1+e^{-100x}}$$

$$\boxed{\nabla E_n = -\sum_{n=1}^{N}\left\{ t_n - \mathrm{w}^T\phi(\mathrm{x}_n) \right\} \phi(\mathrm{x}_n)^T}$$ without reg

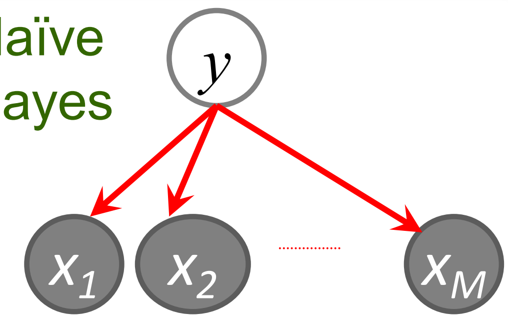
$$\boxed{\nabla E_n = \left[-\sum_{n=1}^{N}\left\{ t_n - \mathrm{w}^T\phi(\mathrm{x}_n) \right\} \phi(\mathrm{x}_n)^T \right] + \lambda \mathrm{w}}$$ with reg

Generative vs Discriminative Training

Variables $\boldsymbol{x} = \{x_1,..x_M\}$ and classifier target y

1. Generative: estimate parameters of variables independently

Naïve Bayes



For classification:
Determine joint:

$$p(y,\boldsymbol{x}) = p(y)\prod_{i=1}^{M} p(x_i \mid y)$$

From joint get required conditional $p(y/\boldsymbol{x})$

Simple estimation

independently estimate M sets of parameters

But independence is usually false

We can estimate $M(M+1)/2$ covariance matrix

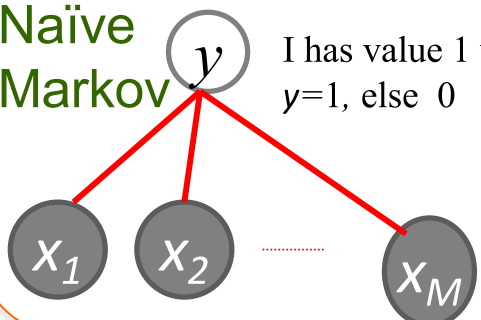
2. Discriminative: estimate joint parameters w_i

Potential Functions (log-linear)

$\phi_i(x_i,y)=\exp\{w_i x_i \, \mathrm{I}\{y=1\}\}$,

$\phi_0(y)=\exp\{w_0 \, \mathrm{I}\{y=1\}\}$

Naïve Markov

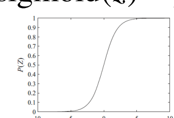


I has value 1 when y=1, else 0

For classification:

Unnormalized $\tilde{P}(y=1 \mid \boldsymbol{x}) = \exp\left\{w_0 + \sum_{i=1}^{M} w_i x_i\right\}$ $\tilde{P}(y=0 \mid \boldsymbol{x}) = \exp\{0\} = 1$

Normalized $\boxed{P(y=1 \mid \boldsymbol{x}) = \mathrm{sigmoid}\left\{w_0 + \sum_{i=1}^{M} w_i x_i\right\}}$ where $\mathrm{sigmoid}(z) = \dfrac{e^z}{1+e^z}$



Logistic Regression

Jointly optimize M parameters

More complex estimation but correlations accounted for

Can use much richer features:

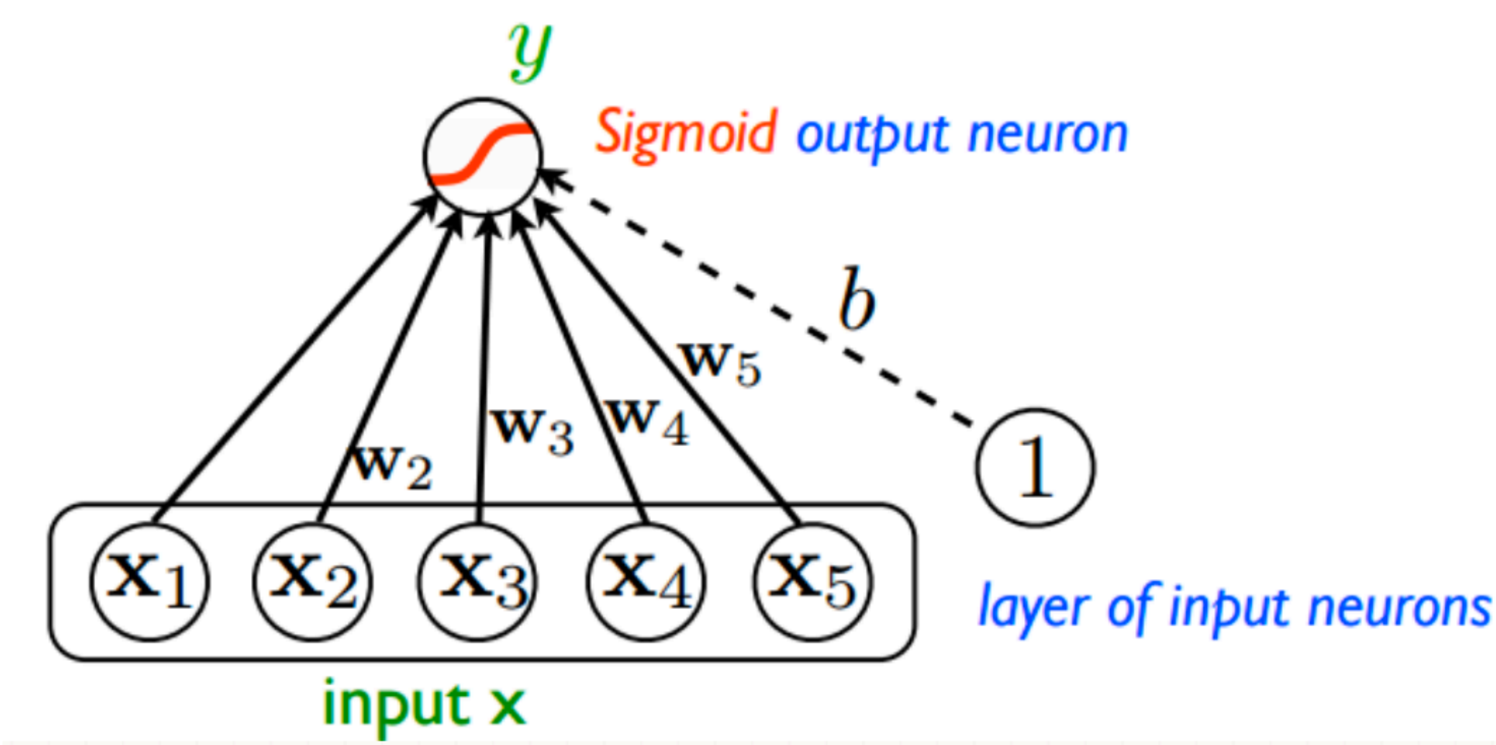
Edges, image patches sharing same pixels

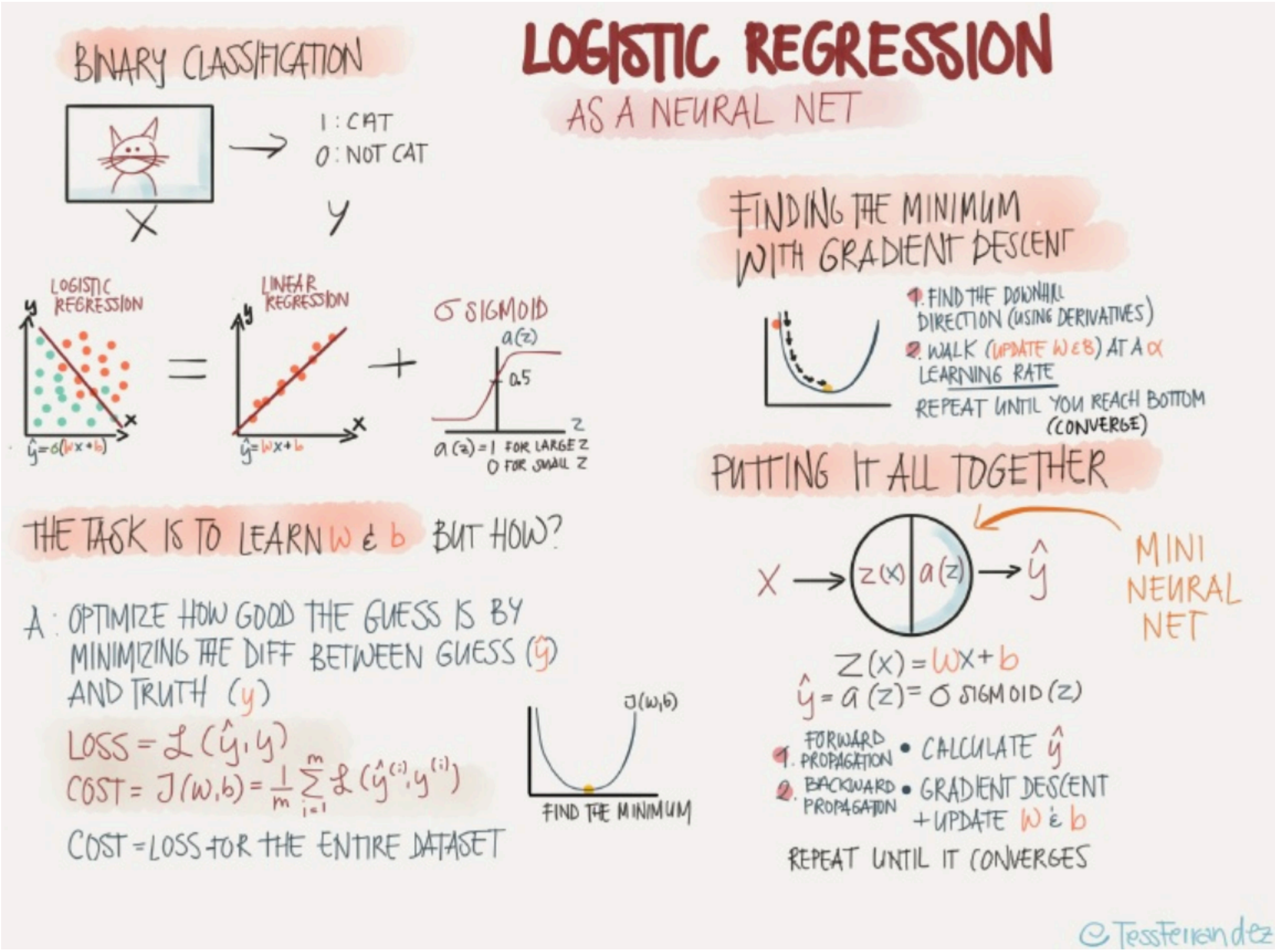
multiclass

$$p(y_i \mid \phi) = y_i(\phi) = \frac{\exp(a_i)}{\sum_j \exp(a_j)}$$

where $a_j = \boldsymbol{w}_j^{T}\phi$

Logistic Regression is a special architecture of a neural network





https://storage.ning.com/topology/rest/1.0/file/get/2408482975?profile=original