Logistic Regression

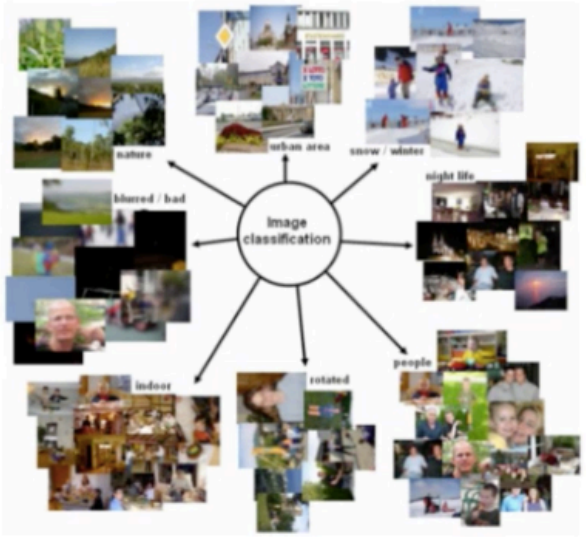
Notes adapted from Logistic Regression in Python Course

Classification Problem

- Supervised vs. Unsupervised Classification
- Supervised classification
  - Regression: predict the value of a function
  - Classification: put data into classes
    - Logistic regression

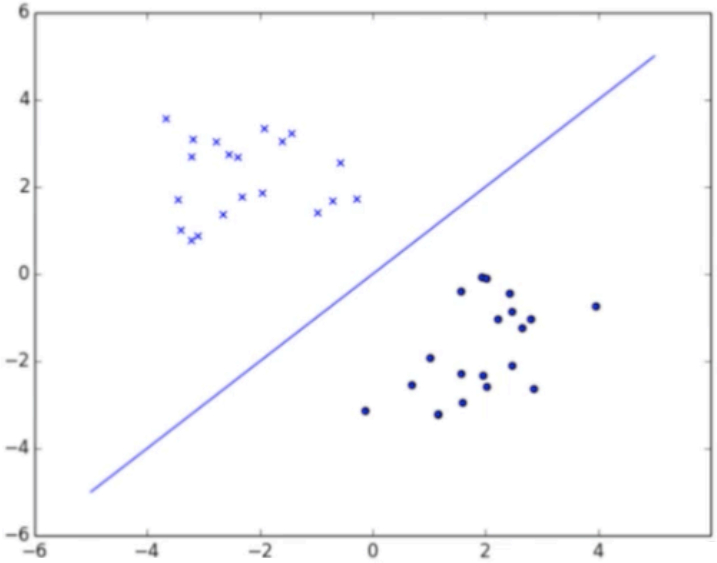
MNIST

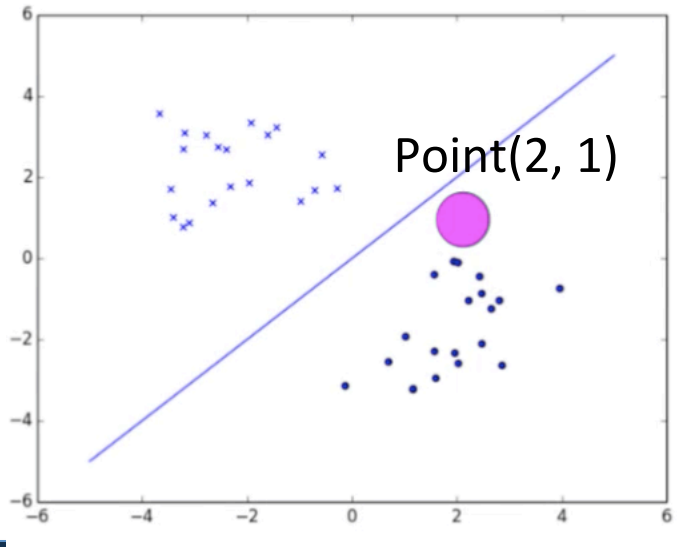
Image classification

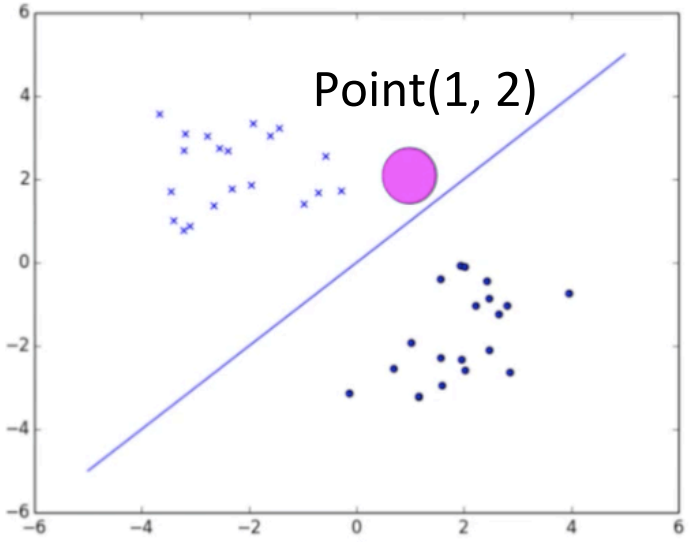


Linear Regression

y=mx+b, or
ax+by+c=0
a=1, b=-1, c=0
x-y=0



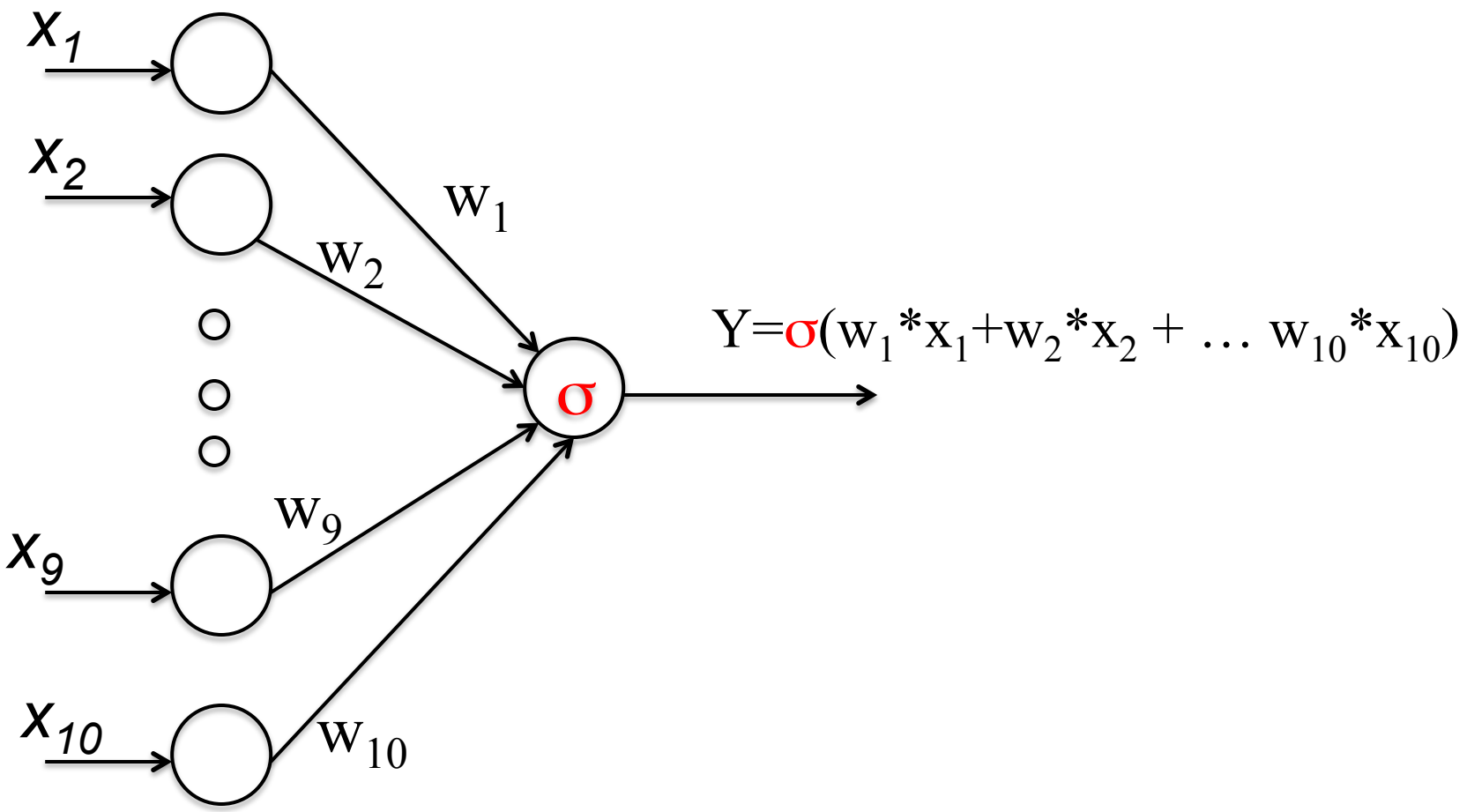




Regression

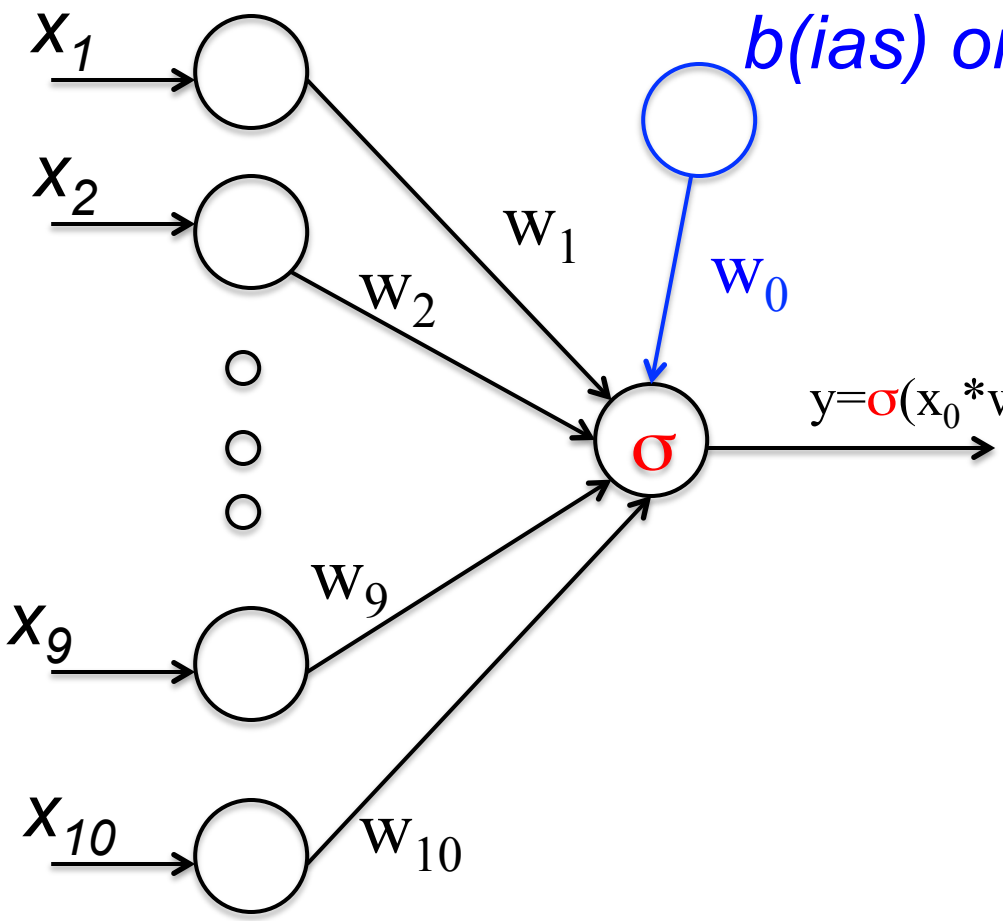
- $h(x) = w_0 + w_1x_1 + w_2x_2$, h() is a linear combination of the components of x
  - In vector form : $h(x) = w^Tx$
- The class separating function:
  - In 2-dimensions: a line
  - In 3-dimensions: a plane
  - In >3 dimension: hyperplane

Logistic Regression



10 features

Logistic Regression



$y=\sigma(x_0*w_0+w_1*x_1+w_2*x_2+\ldots w_{10}*x_{10})$

$$y=\sum_{i=0}^{N} w_i * x_i$$

$y=\sigma(w^T x)$

10 features

Activation Functions

- Tanh()

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

$$f'(x) = 1 - f(x)^2$$

- Sigmoid/Logistic

$$f(x) = \frac{1}{1+e^{(-x)}}$$

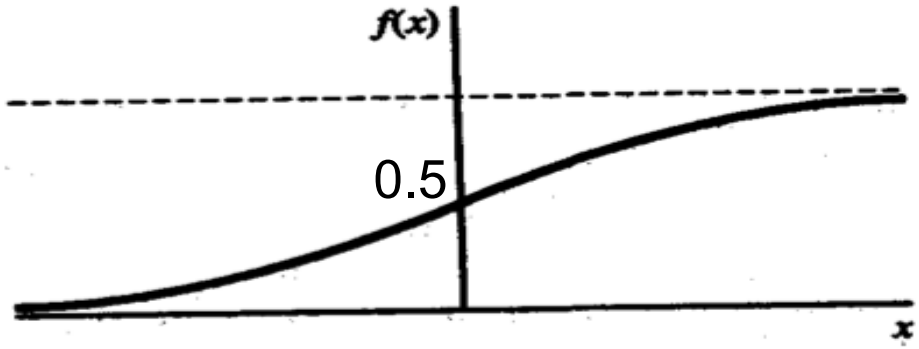
$$f'(x) = f(x)[1 - f(x)]$$

- Bipolar Sigmoid

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

$$f'(x) = \frac{1}{2}[1 + f(x)][1 - f(x)]$$

Sigmoid Function for Classification



$$y=\sigma(w^T x)$$

if $\sigma(w^T x)>0.5$, predict class 1
else if $\sigma(w^T x)<0.5$, predict class 0

Demo logistic1.py

Real Data Application

- Features:
  - Is_mobile (0/1)
  - N_Products_viewed (int >0)
  - Visit_duration (real>=0)
  - Is_returning_visitor (0/1)
  - Time_of_day (1, 2, 3, 4 = 4 24 hour split into 4 categories)
  - Class: User_action (bounce/add_to_cart/ begin_checkout)

Demo ecommerce_data.csv

Data Preprocessing

- Data pre-processing
- One-hot encoding:

12am-6am ,	6am-12pm,	12pm-6pm,	6pm - 12am
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

- Use Z-score for numeric features: n_products_viewed and visitor_duration

Demo process.py

Prediction on Ecommerce Data

- Random weights are used. How good is the prediction accuracy?

- Two things to come:
  - How to evaluate the quality of the logistic regression model learned?
  - How to improve the prediction accuracy?

Demo logistic_predict.py

Cross Entropy Error Cost Function

- Logistic Regression Error
  - 0 if correct, >0 if not correct, more wrong ==> bigger cost
- Cross-Entropy Error cost function

$$J = -(t\log(y) + (1-t)\log(1-y))$$

t is the target, y is the predicted value

t=1, y=1 --> 0

t=0, y=0 --> 0

t=1, y=0.9 --> 0.11

t=1, y=0.5 --> 0.69

t=1, y=0.1 --> 2.3

Demo logistic2.py

Minimizing the Cost Function

Multiple Training Examples

$$J = -\sum_{n=1}^{N} t_n log(y_n) + (1 - t_n) log(1 - y_n)$$

- To minimizing the cost function over the entire data set

$$\frac{\partial J}{\partial w_i} = 0$$

  - Generally, there is no closed form solution for this minimization problem, except for special cases

Minimizing the Cost Function

Gradient Descent

Idea: take small steps in direction of derivative

Step size == learning rate (1 for this example)

Ex:
w = -2
w = -2 - 1*(-1) = -1
Now we're closer to the optimal point! (w=0)

Note: slope is 0 at the bottom, so no more changes will occur

Minimizing the Cost Function

Gradient Descent for Logistic Regression

$$J = -\sum_{n=1}^{N} t_n log(y_n) + (1 - t_n) log(1 - y_n)$$

Split into 3 derivatives:

$$\frac{\partial J}{\partial w_i} = \sum_{n=1}^{N} \frac{\partial J}{\partial y_n} \frac{\partial y_n}{\partial a_n} \frac{\partial a_n}{\partial w_i}$$

$$a_n = w^T x_n$$

Minimizing the Cost Function

Derivatives

$$J = -\sum_{n=1}^{N} t_n log(y_n) + (1 - t_n) log(1 - y_n)$$

$$\frac{\partial J}{\partial y_n} = - \quad t_n \frac{1}{y_n} + (1 - t_n) \frac{1}{1 - y_n} (-1)$$

Minimizing the Cost Function

Derivatives

$$y_n = \sigma(a_n) = \frac{1}{1+e^{-a_n}}$$

$$\frac{\partial y_n}{\partial a_n} = \frac{-1}{(1+e^{-a_n})^2}(e^{-a_n})(-1)$$

$$\frac{\partial y_n}{\partial a_n} = \frac{e^{-a_n}}{(1+e^{-a_n})^2} = \frac{1}{1+e^{-a_n}}\frac{e^{-a_n}}{1+e^{-a_n}} = y_n(1-y_n)$$

Minimizing the Cost Function

Derivatives

$$a_n = w^T x_n$$

$$a_n = w_0 x_{n0} + w_1 x_{n1} + w_2 x_{n2} + \ldots$$

$$\frac{\partial a_n}{\partial w_i} = x_{ni}$$

Minimizing the Cost Function

Putting them all together

$$\frac{\partial J}{\partial w_i} = -\sum_{n=1}^{N} \frac{t_n}{y_n} y_n(1-y_n)x_{ni} - \frac{1-t_n}{1-y_n} y_n(1-y_n)x_{ni}$$

$$\frac{\partial J}{\partial w_i} = -\sum_{n=1}^{N} t_n(1-y_n)x_{ni} - (1-t_n)y_n x_{ni}$$

$$\frac{\partial J}{\partial w_i} = -\sum_{n=1}^{N} [t_n - t_n y_n - y_n + t_n y_n]x_{ni}$$

$$\frac{\partial J}{\partial w_i} = \sum_{n=1}^{N} (y_n - t_n)x_{ni}$$

Weight Updates

Repeat for M epoch:

on each epoch:

$$w_i += \nabla w,$$

$$\nabla w = \lambda * (y - t) * x_i$$

λ: Learning Rate → step size

t : target class value

y: predicted class value

Demo logistic3.py
Demo logistic_train.py