UVA CS 4774: Machine Learning

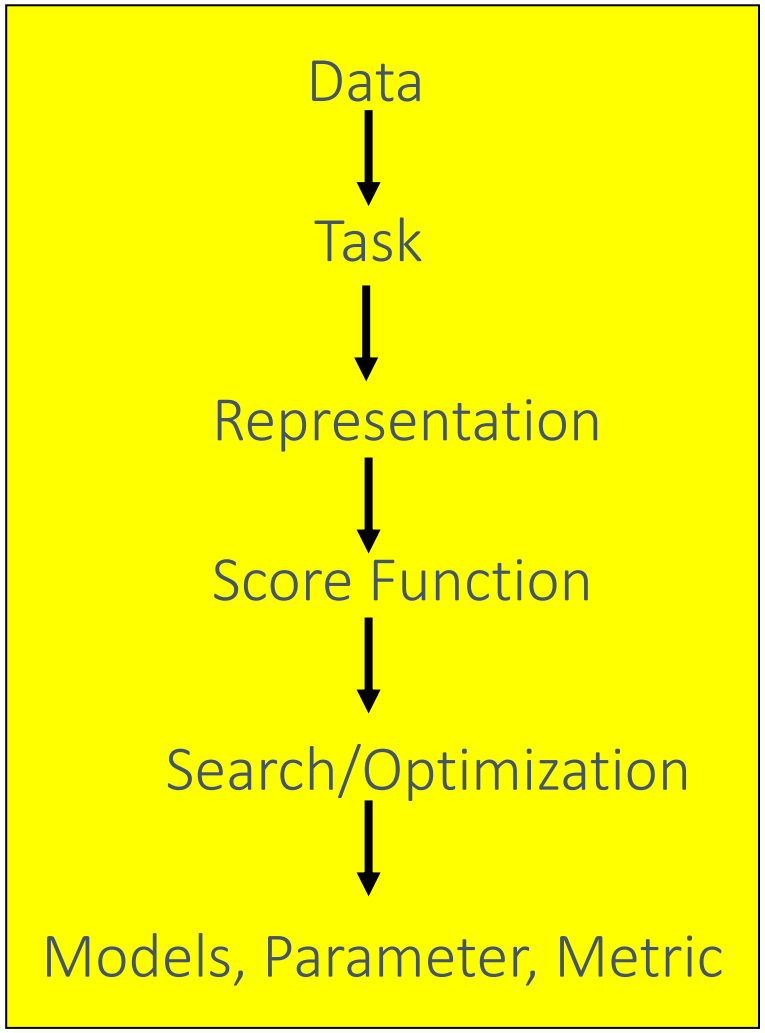
Lecture 11: Logistic Regression

Dr. Yanjun Qi

University of Virginia

Department of Computer Science

Machine Learning in a Nutshell



ML grew out of work in AI

Optimize a performance criterion using example data or past experience,

Aiming to generalize to unseen data

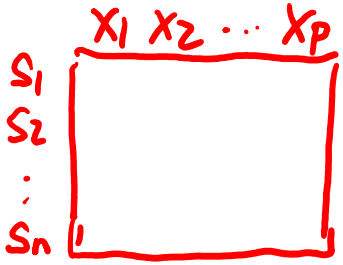
Course Content Plan ➔ Regarding Tasks

IID

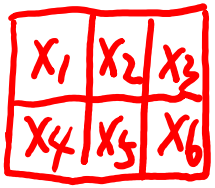
- ~~Regression (supervised)~~ — Y is a continuous
- ~~Learning theory~~ — About f()
- Classification (supervised) — Y is a discrete
- Unsupervised models — NO Y
- Graphical models — About interactions among Y,X1,. Xp
- Reinforcement Learning — Learn to Interact with environment

Course Content Plan ➔ Regarding Data

- ❑ Tabular / Matrix



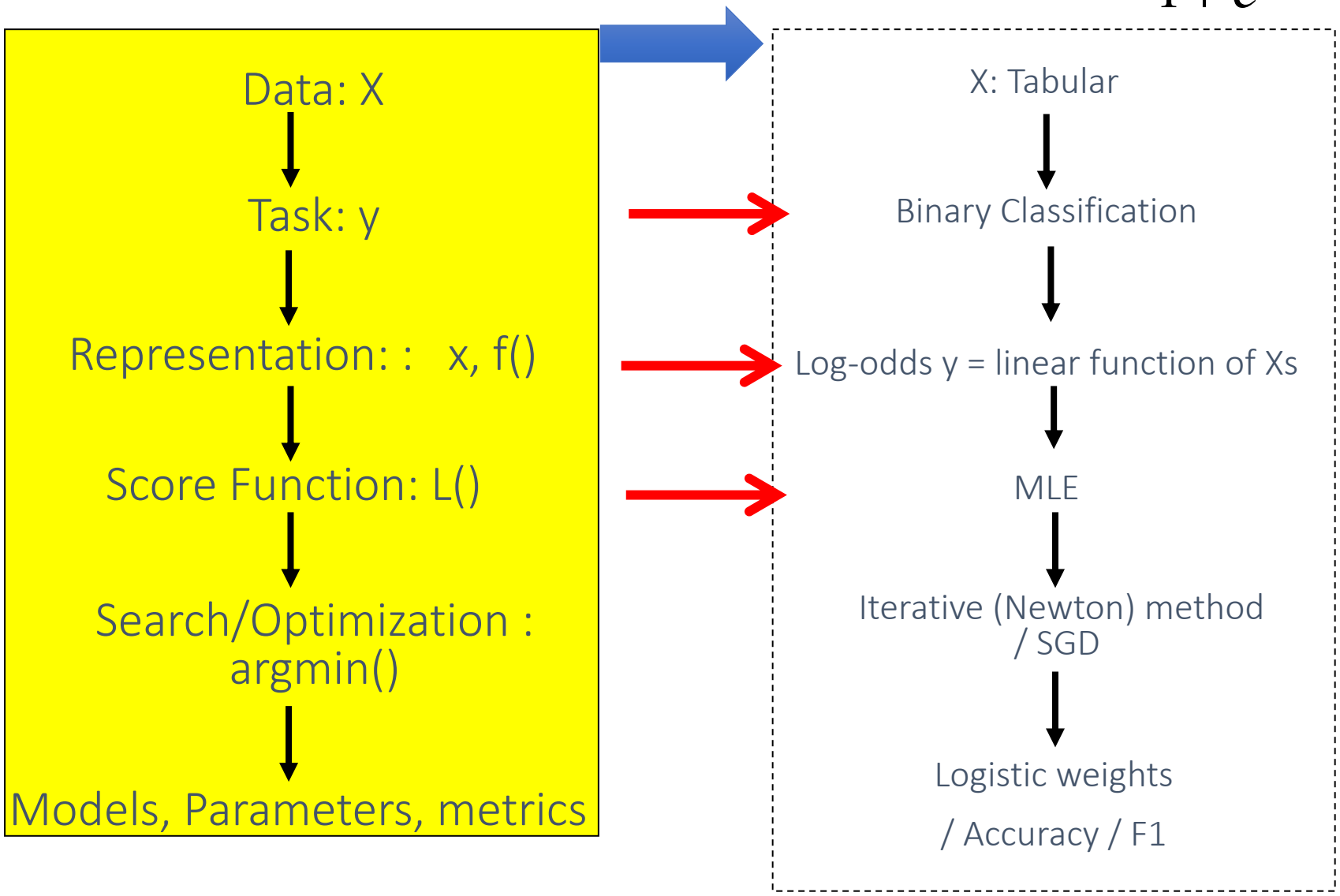
- ❑ 2D Grid Structured: Imaging



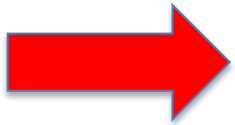
- ❑ 1D Sequential Structured: Text
- ❑ Graph Structured (Relational)
- ❑ Set Structured / 3D /

Today: Logistic Regression **Classifier**

$$P(y=1|x)=\frac{e^{\beta_0+\beta^T x}}{1+e^{\beta_0+\beta^T x}}$$



Roadmap



- ❑ Bayes Classifier: Classification MAP Rule
- ❑ Logistic Regression Representation
- ❑ Training LG by MLE

Bayes classifiers

- Treat each feature attribute and the class label as random variables.

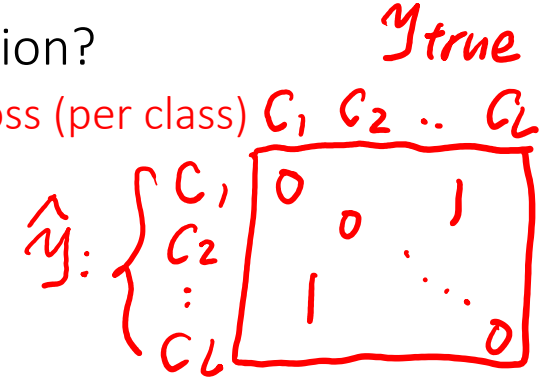
$C : \{c_1, \cdots c_L\}$

Bayes classifiers

- Treat each feature attribute and the class label as random variables.

- Training: What is an appropriate loss for classification?
  - E.g. Compare training class to output class: Zero-One loss (per class)

$$L(\hat{y}, y) = I(\hat{y} \neq y),$$

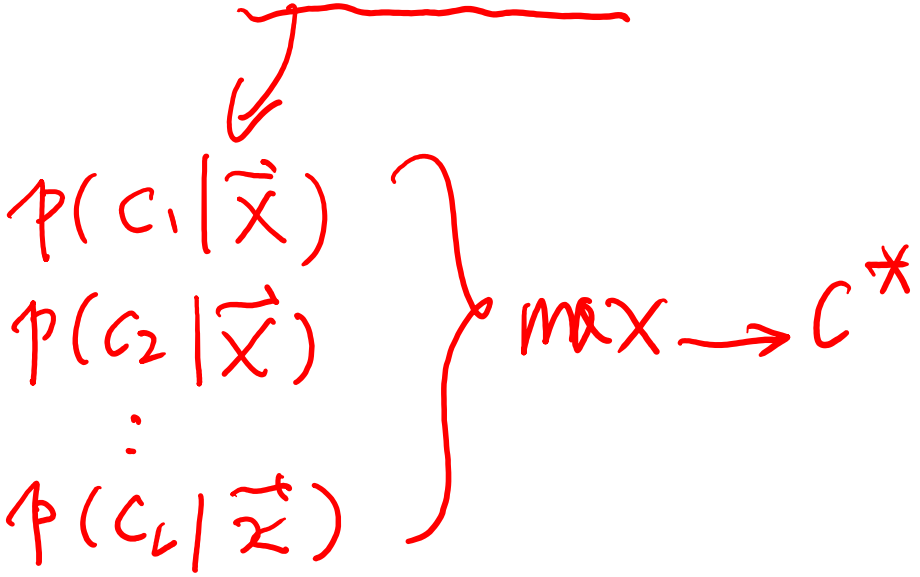


  - Can we model/estimate $p(C=C_i \mid x) = p(C_i \mid x_1, x_2, \ldots, x_p)$ directly from reference data?

Bayes classifiers

- Treat each feature attribute and the class label as random variables.

- Testing: Given a sample vector **x** with attributes as (x_1, x_2, … , x_p):
  - Goal is to predict its class
  - HOW: ➔we want to find the class that maximizes p(c | x_1, x_2, … , x_p).



Bayes Classifiers – Predict via MAP Rule

Task: Classify a new instance X: $X = \langle X_1, X_2, \ldots, X_p \rangle$

based on:

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid x_1, x_2, \ldots, x_p)$$

MAP Rule

MAP = Maximum Aposteriori Probability

Please read extra slides for how to get MAP-rule from minimize EPE-0/1

Bayes Classifiers – Predict via MAP Rule

- Establishing a probabilistic model for classification

➔ MAP classification rule

➔ To Assign x to c* if

$$P(C = c^* \mid \mathbf{X} = \mathbf{x}) > P(C = c \mid \mathbf{X} = \mathbf{x})$$

$$\text{for } c \neq c^*,\ c = c_1, \cdots, c_L$$

$$\sum_{j=1}^{L} P(C = c_j \mid x) = 1$$

Adapt from Prof. Ke Chen NB slides

Recap: Statistical Decision Theory (Extra)

- Random input vector: X
- Random output variable: Y
- Joint distribution: Pr(X,Y) $\Rightarrow D = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\}$
- Loss function L(Y, f(X))

- Expected prediction error (EPE):

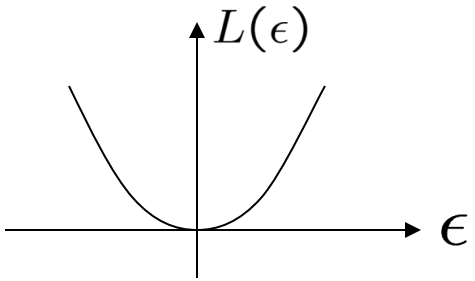
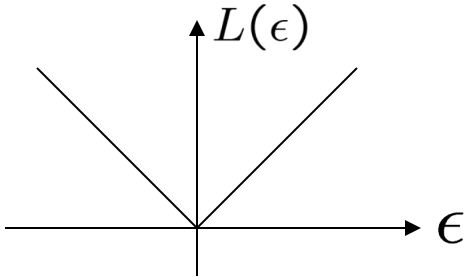
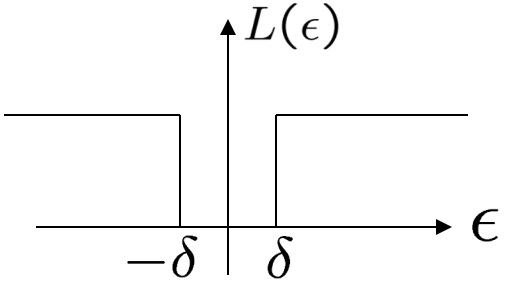
$$\mathrm{EPE}(f) = \mathrm{E}(L(Y, f(X))) = \int L(y, f(x)) \Pr(dx, dy)$$

$$\text{e.g.} = \int (y - f(x))^2 \Pr(dx, dy)$$

e.g. Squared error loss (also called L2 loss)

Consider population distribution

SUMMARY: WHEN Expected prediction error (EPE) USES DIFFERENT LOSS (Extra)

Loss Function	Estimator $\hat{f}(x)$
L_2 	$\hat{f}(x) = E[Y\|X = x]$ regression / KNN
L_1 	$\hat{f}(x) = \text{median}(Y\|X = x)$
0-1 	$\hat{f}(x) = \arg\max_Y P(Y\|X = x)$ (Bayes classifier / MAP)

Please read extra slides for how to get MAP-rule from minimize EPE-0/1

$$f : X \longrightarrow C$$

Output as Discrete Class Label
$C_1, C_2, \ldots, C_L$

Establishing a probabilistic model for classification

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid x_1, x_2, \ldots, x_p)$$

$P(C|\vec{x})$

Generative

$$\underset{c \in C}{\operatorname{argmax}} P(c \mid X) = \underset{c \in C}{\operatorname{argmax}} P(X, c) = \underset{c \in C}{\operatorname{argmax}} P(X \mid c) P(c)$$

$\frac{P(X,C)}{P(X)}$

Later!

Discriminative

$$\underset{c \in C}{\arg\max} P(c / \mathbf{X}) \quad C = \{c_1, \cdots, c_L\}$$

Three major groups of classifiers

- We can divide the large variety of classification approaches into roughly three major types

1. Discriminative

$P(C|\vec{x})$

directly estimate a decision rule/boundary

e.g., support vector machine, decision tree, ~~logistic regression,~~

e.g. neural networks (NN), deep NN

2. Generative:

$P(C|\vec{x}) =$ Bayes Rule

build a generative statistical model

e.g., Bayesian networks, Naïve Bayes classifier

3. Instance based classifiers

- Use observation directly (no models)

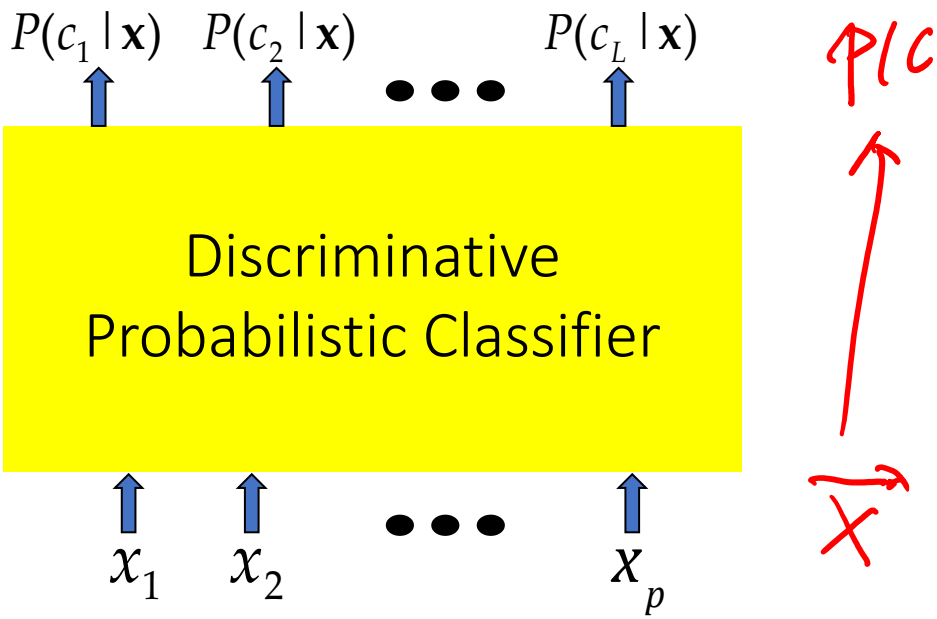
~~- e.g. K nearest neighbors~~

Our Whole S2:

$X \rightarrow C$: $\underbrace{P(C|X)}_{f(x)}$

- Discriminative Classifiers

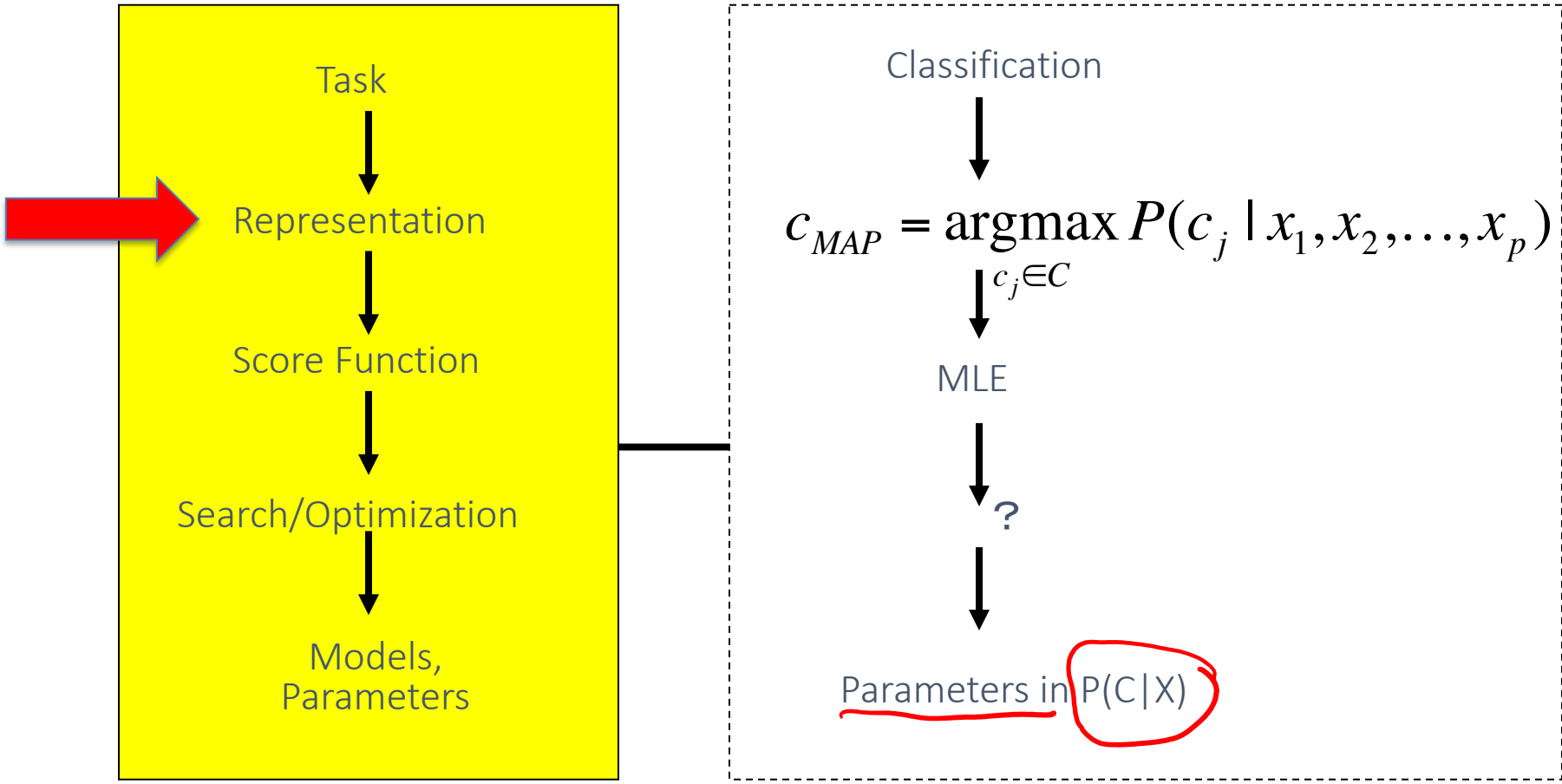
$$\arg\max_{c \in C} P(c / \mathbf{X}), \quad C = \{c_1, \cdots, c_L\}$$



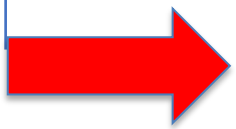
$$\mathbf{x} = (x_1, x_2, \cdots, x_p)$$

Adapt from Prof. Ke Chen NB slides

Bayes Classifier



Roadmap



- ❑ Bayes Classifier: Classification MAP Rule
- ❑ Logistic Regression Representation
- ❑ Training LG by MLE

Today: Logistic Regression **Classifier**

$$P(y=1|x) = \frac{e^{\beta_0+\beta^T x}}{1+e^{\beta_0+\beta^T x}}$$

Data: X

↓

Task: y

↓

Representation: : x, f()

↓

Score Function: L()

↓

Search/Optimization : argmin()

↓

Models, Parameters, metrics

→

X: Tabular

↓

Binary Classification

↓

Log-odds y = linear function of Xs

↓

MLE

↓

Iterative (Newton) method / SGD

↓

Logistic weights

/ Accuracy / F1

$P(y=0|x) = 1 - P(y=1|x)$

Multivariate linear regression to Logistic Regression

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

Logistic regression for binary classification

Binary Variable

y : Yes, No

1, 0

H, T

Log-odds

$$\ln\left[\frac{P(y=1|x)}{1-P(y=1|x)}\right] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

$P(y=0|x) = 1 - P(y=1|x)$

Logistic Regression p(y|x)

$$\ln\left[\frac{P(y|x)}{1-P(y|x)}\right] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

log [P(Head|x) / P(Tail|x)]

Logit

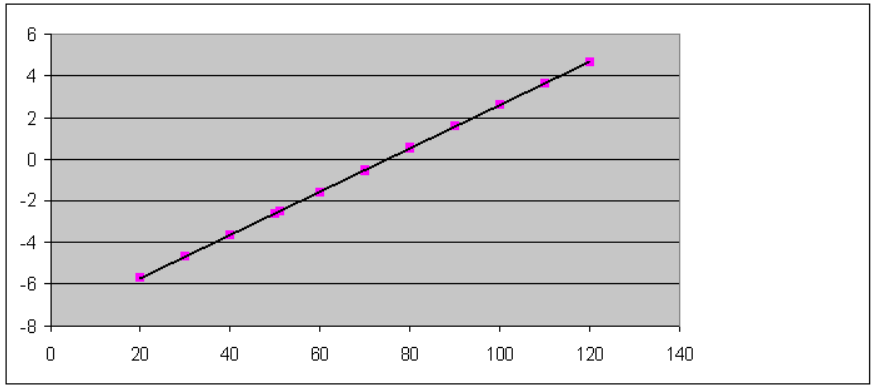
p(c|x)

$$= P(y|x) = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_p x_p}}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_p x_p}} = \frac{1}{1+e^{-(\beta_0+\beta^T X)}}$$

p(Head)

Binary Logistic Regression: Geometric (X: 1D)

ln[p/(1-p)]

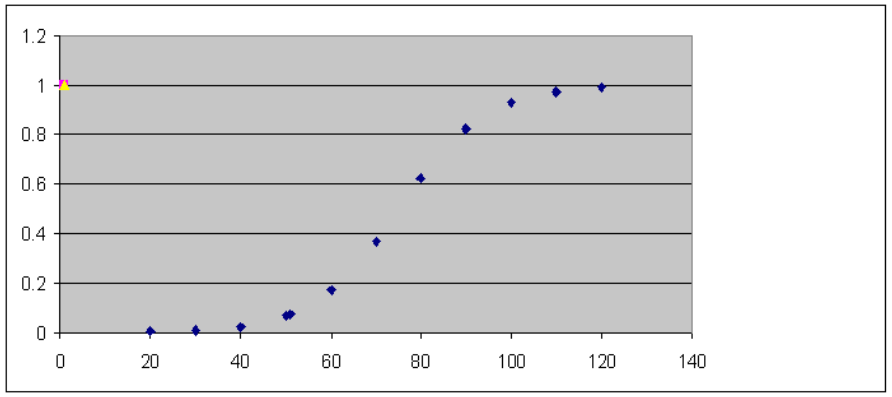


X

p(Head)

= P (Y=1|x)

S-shape



X

p(Head) p(Tail)

P(y=1|x) 1-p(y=1x)

View I: logit of p(y=1|x) is linear function of x

$$\ln\left(\frac{p}{1-p}\right) = p(y=1|x)$$

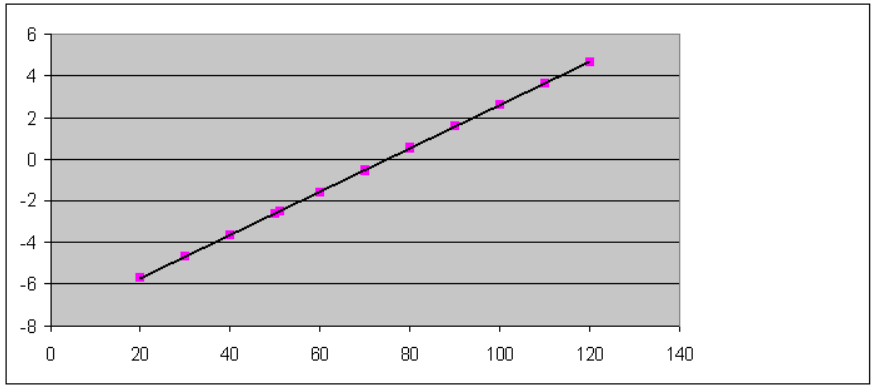
$$\ln\left[\frac{P(y|x)}{1-P(y|x)}\right] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

Logit function

Logit of P(y|x)

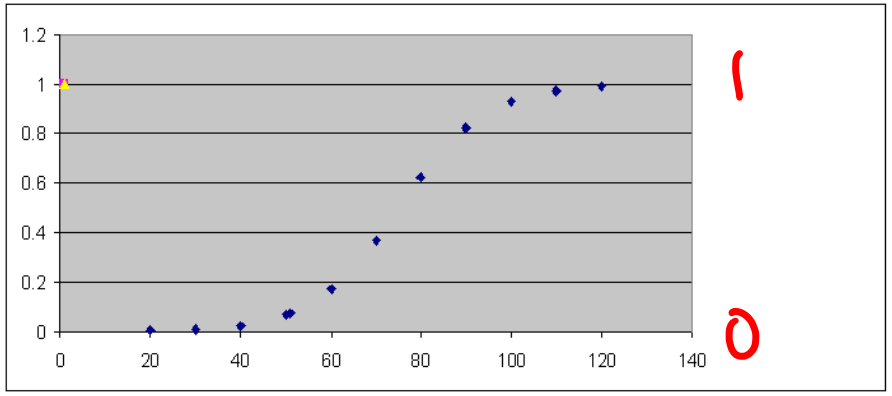
View II: Model Y as Bernoulli:

ln[p/(1-p)]



X

P (Y=1|x)



1

0

S shape

X

Bernoulli Distribution

p_{Head}

$\Downarrow$

$Y \in \{0, 1\}$ (T, H)

$p_{Head} = P(y=1|x) = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}}$

P(y=1|x) 1-p(y=1x)

Review: Bernoulli Distribution e.g. Coin Flips

- You flip a coin
  - Z: {Who is Up: Head or Tail} is a discrete Random Variable
  - Head with probability p
  - Binary random variable
  - Bernoulli trial with success probability p

Bernoulli(p)

{H, T}

p(Head)

$\{z_1, z_2, \ldots, z_n\}$

{H H T .. H} → MLE → p(Head)

Logistic Regression—Bernoulli Distribution with p(Head) as a function of x

$\Rightarrow$ y is model with Bernoulli (p)

Logistic regression models are appropriate when the target variable is coded as 0/1.

$\Rightarrow p$ is a func of x

We only observe "0" and "1" for the target variable—but we think of the target variable conceptually as a probability that "class=1" will occur.

$$p(\text{Head}) = p(y=1|x) = \frac{e^{\beta^T x}}{e^{\beta^T x}+1}$$

This means we use Bernoulli distribution to model the target variable with its Bernoulli parameter p=p(y=1|x) predefined.

The main interest ➔ predicting the probability that an event occurs (i.e., the probability that p(y=1|x)).

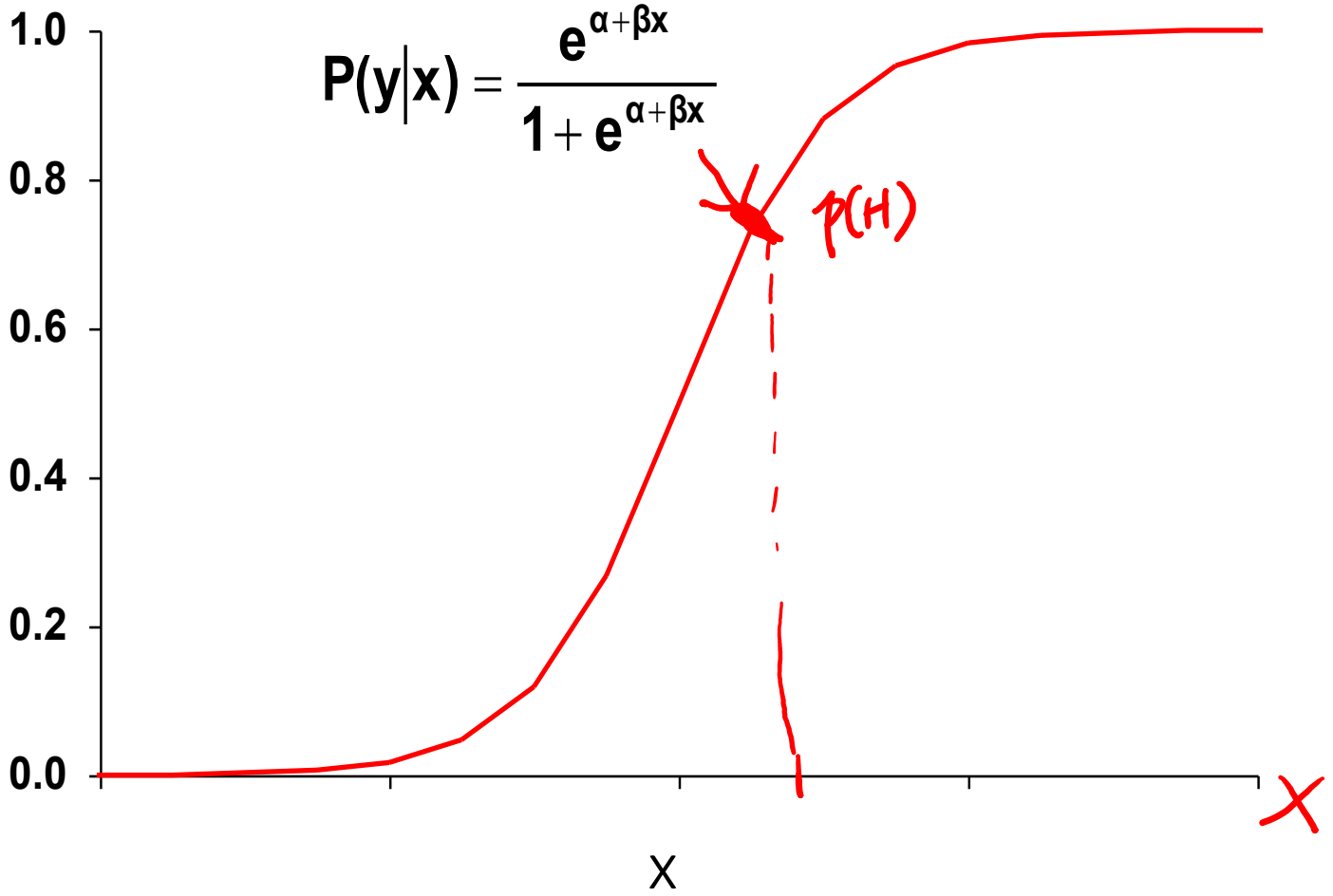
View II: Model Y as Bernoulli:

e.g.
Probability of disease

1-p(y=1x)

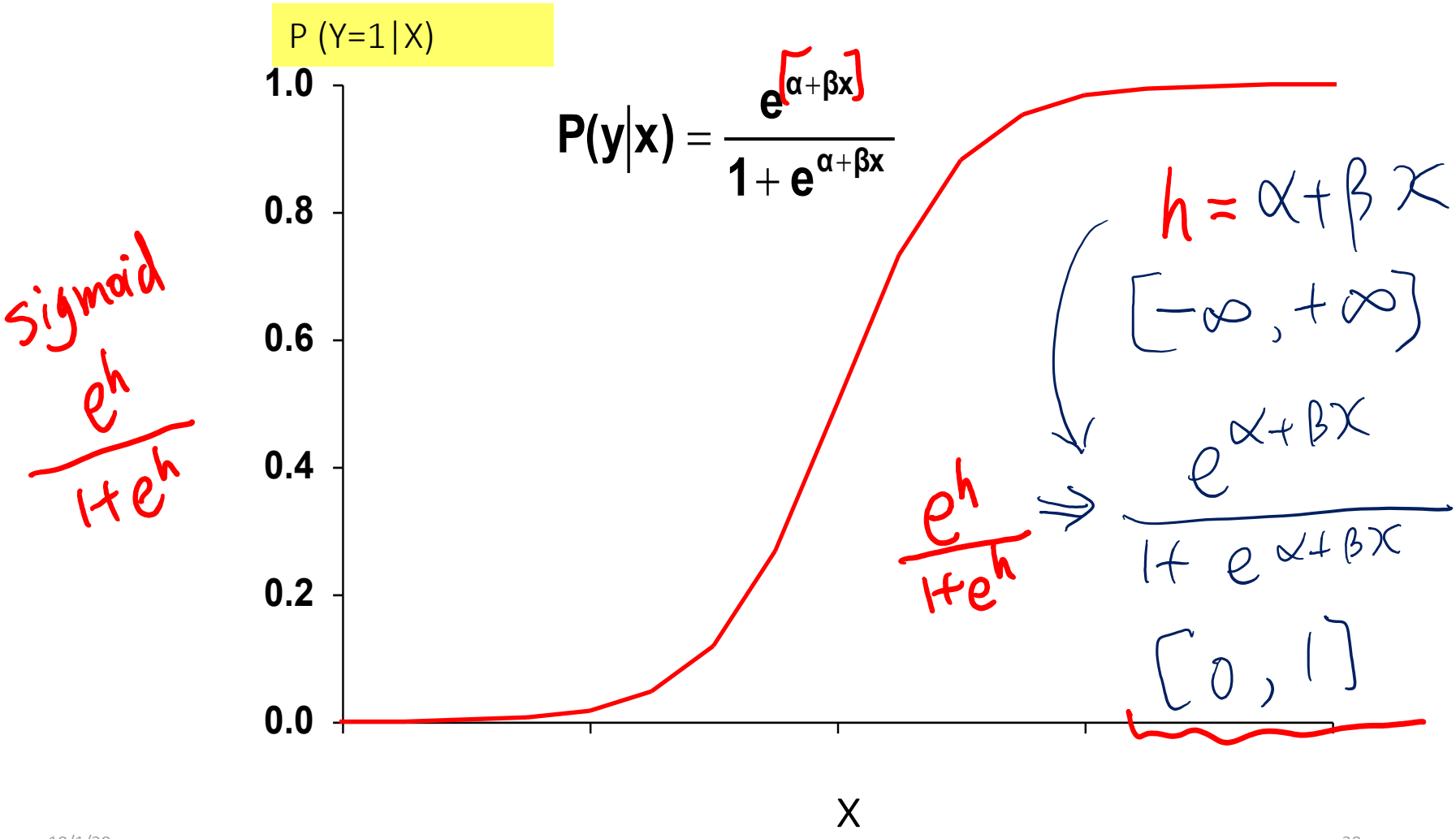
P(y=1|x)

P (Y=1|X)



View III: "S" shape function compress output to [0,1]

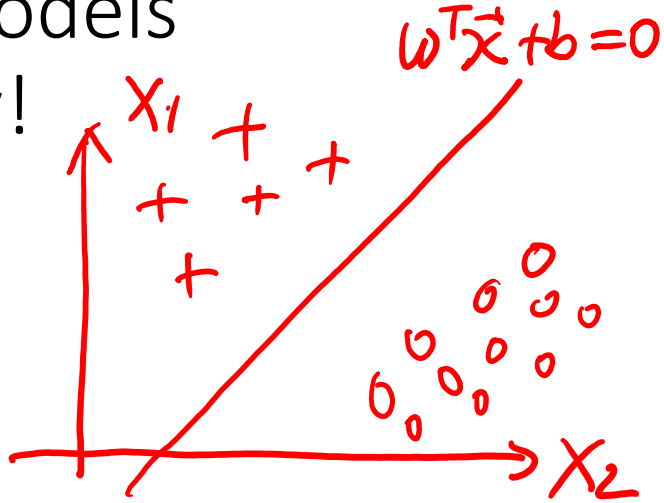
LogistiR = LR + Sigmoid



View IV: Logistic Regression models a linear classification boundary!

$y \in \{0,1\}$

$P(y=1|x) = P(y=0|x)$ Boundary no decision



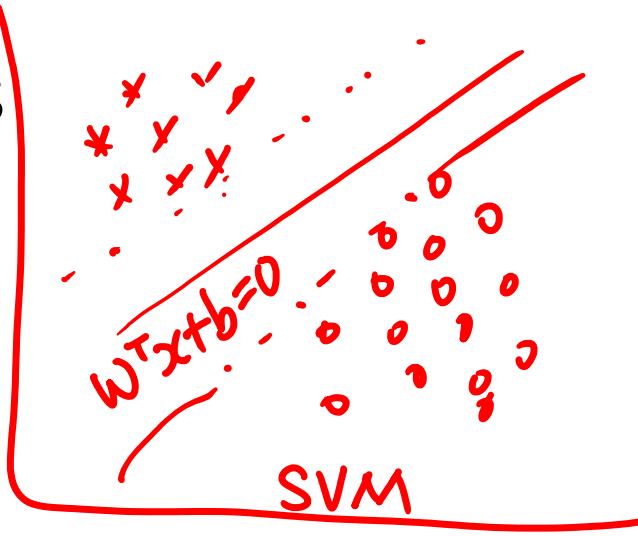
$$\ln\left[\frac{P(y|x)}{1-P(y|x)}\right] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

Decision Boundary ➔ equals to zero

$$\ln\left[\frac{P(y=1|x)}{P(y=0|x)}\right] = \ln\left[\frac{P(y=1|x)}{1-P(y=1|x)}\right] = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

$= w^T\vec{x} + b = 0$

View IV: Logistic Regression models a linear classification boundary!



$$y = \{H, T\} \quad \{1, 0\}$$

$$\underset{y \in \{0,1\}}{\operatorname{argmax}} \; p(y \mid \underline{x})$$

$$p(y=0 \mid x) = p(y=1 \mid x) \Rightarrow \text{Decision Boundary}$$

$$\frac{p(y=1|x)}{p(y=0|x)} = 1$$

$$\log\left(\frac{p(y=1|x)}{p(y=0|x)}\right) = \vec{\beta}^T \vec{x} = \log(1) = 0$$


Thank You

UVA CS 4774: Machine Learning

Lecture 11: Logistic Regression

Module II

Dr. Yanjun Qi

University of Virginia

Department of Computer Science

Roadmap

- ❑ Bayes Classifier: Classification MAP Rule
- ❑ Logistic Regression Representation
- ❑ Training LG by MLE

$$p(c|\vec{x}) \downarrow p(y=1|\vec{x}) = \frac{e^{\beta^T \vec{x}}}{e^{\beta^T x}+1}$$

Today: Logistic Regression **Classifier**

$$P(y=1|x) = \frac{e^{\beta_0+\beta^T x}}{1+e^{\beta_0+\beta^T x}}$$

Data: X

↓

Task: y

↓

Representation: : x, f()

↓

Score Function: L()

↓

Search/Optimization : argmin()

↓

Models, Parameters, metrics

→

X: Tabular

↓

Binary Classification

↓

Log-odds y = linear function of Xs

↓

→ MLE

↓

Iterative (Newton) method / SGD

↓

Logistic weights

/ Accuracy / F1

$P(y=0|x) = 1 - P(y=1|x)$

Maximum Likelihood Estimation

A general Statement

Consider a sample set T=(Z_1...Z_n) which is drawn from a probability distribution P(Z|\theta) where \theta are parameters.

If the Zs are independent with probability density function $P(Z_i|\theta)$, the joint probability of the whole set is

$$\theta^* = \arg\max\{P(Z_1 \dots Z_n|\theta)\} = \prod_{i=1}^{n} P(Z_i|\theta)$$

this may be maximised with respect to \theta to give the maximum likelihood estimates.

The idea is to

- ✓ assume a particular model with unknown parameters, θ
- ✓ we can then define the probability of observing a given event conditional on a particular set of parameters. $P(Z_i|\theta)$
- ✓ We have observed a set of outcomes in the real world. $Z_1, Z_2, \ldots, Z_n$
- ✓ It is then possible to choose a set of parameters which are most likely to have produced the observed results.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, P(Z_1 \ldots Z_n|\theta) = \prod_{i=1}^{n} P(Z_i|\theta)$$

Likelihood

$0 < \quad < 1$

This is maximum likelihood. In most cases it is both consistent and efficient.

$n \sim 30{,}000$

$$log(L(\theta)) = \sum_{i=1}^{n} log(P(Z_i|\theta))$$

Log-Likelihood

It is often convenient to work with the Log of the likelihood function.

Review: Defining Likelihood for basic Bernoulli

Given: $\{z_1, z_2, \ldots, z_n\}$

$\Downarrow$

$\{H, H, T, \ldots H\}_n$

$\Downarrow$ reformulate

$\{1, 1, 0, \ldots, 1\}_n$

Constant

$\theta = \{p\}$

$= \{p(\text{Head})\}$

$p(z_i \mid \underline{\theta}) = p^{z_i}(1-p)^{1-z_i}$ (Here $z_i \in \{0, 1\}$)

$p(z_i) = \begin{cases} p, & \text{if } z_i = H/1 \\ 1-p, & \text{if } z_i = T/0 \end{cases} \Rightarrow \underset{p_{\text{Head}}}{\operatorname{argmax}} \prod_{i=1}^{n} p^{z_i}(1-p)^{1-z_i}$

Review: Defining Likelihood for basic Bernoulli

$$\log(L(p)$$

$$= \log\left[\prod_{i=1}^{n} p^{z_i}(1-p)^{1-z_i}\right]$$

$$= \sum_{i=1}^{n}(z_i \log p + (1-z_i)\log(1-p))$$

$$= \log p \sum_{i=1}^{n} z_i + \log(1-p)\sum_{i=1}^{n}(1-z_i)$$

$$= x\log p + (n-x)\log(1-p)$$

Observed data ➔ A total of x heads-up from n trials

Extra

Review: Defining Likelihood for basic Bernoulli

$$\arg\min_p \{-l(p)\} = \arg\min_p \{-x\log(p)-(n-x)\log(1-p)\}$$

i.e. negative log-likelihood

$$\frac{dl(p)}{dp} = 0 - \frac{x}{p} - \frac{-(n-x)}{1-p} \simeq 0$$

$$0 = -\frac{x}{p} + \frac{n-x}{1-p}$$

$$0 = \frac{-x(1-p)+p(n-x)}{p(1-p)}$$

$$0 = -x + px + pn - px$$

$$0 = -x + pn$$

Minimize the negative log-likelihood

➔ MLE parameter estimation

$$\hat{p} = \frac{x}{n}$$

i.e. Relative frequency of a binary event

The proportion of heads!

Extra

From Basic Bernoulli to Logistic Regression

Observing binary samples z_i

PMF:

$$\Pr(z_i|p) = p^{z_i}(1-p)^{1-z_i}$$

LIKELIHOOD:

$$L(p) = \prod_{i=1}^{n} p^{z_i}(1-p)^{1-z_i}$$

function of p=Pr(head)

$\{H, H, T, \ldots H\}_n$

Logistic Regression $z_i = y_i|x_i$

$\{y_1|x_1, y_2|x_2, \ldots, y_n|x_n\}$

$p(z_i|\beta)$

$p(y_i=1|\beta) = \frac{e^{\beta^T \vec{x}}}{e^{\vec{\beta}\vec{x}}+1}$

Now we just rewrite

$\hat{y}_i = p(y_i=1|x_i)$

$p(z_i|\beta) = \hat{y}_i^{y_i}(1-\hat{y}_i)^{1-y_i}$

Extra

LIKELIHOOD:

Basic Bernoulli

$$L(p) = \prod_{i=1}^{n} p^{z_i} (1-p)^{1-z_i}$$

function of p=Pr(head)

$$\log(L(p)) = \log\left[\prod_{i=1}^{n} p^{z_i} (1-p)^{1-z_i}\right]$$
$$= \sum_{i=1}^{n} (z_i \log p + (1-z_i)\log(1-p))$$

Logistic / Bernoulli

$$L(\beta) = \prod_{i=1}^{n} p(y_i=1|x_i)^{y_i} (1-p(y_i=1|x_i))^{1-y_i}$$
$$= \prod_{i=1}^{n} \hat{y}_i^{\,y_i} (1-\hat{y}_i)^{1-y_i}$$

Log likelihood [Cross Entropy]

$$ll(\beta) = \sum_{i=1}^{n} \left(y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i) \right)$$

Extra

ll $l(\beta) = \sum_{i=1}^{N} \{\log \Pr(Y = y_i \mid X = x_i)\}$

When training set includes (x_i, y_i), i=1,…,N

$$ll(\beta) = \sum_{i=1}^{N} \log P(y_i \mid x_i)$$

$$\text{Here } P(y_i \mid x_i) = \begin{cases} p(y=1 \mid x_i), & \text{if } y_i = 1 \\ p(y=0 \mid x_i), & \text{if } y_i = 0 \end{cases}$$

$$= \left(p(y=1 \mid x_i)\right)^{y_i} \left(1 - p(y=1 \mid x_i)\right)^{1-y_i}$$

Extra

MLE for Logistic Regression Training

Training set: (x_i, y_i), i=1,…,N

$$l(\beta) = \sum_{i=1}^{N} \{\log \Pr(Y = y_i \mid X = x_i)\}$$

$$= \sum_{i=1}^{N} \{y_i \log(\Pr(Y = 1 \mid X = x_i)) + (1 - y_i) \log(\Pr(Y = 0 \mid X = x_i))\}$$

$$= \sum_{i=1}^{N} (y_i \log \frac{\exp(\beta^T x_i)}{1 + \exp(\beta^T x_i)}) + (1 - y_i) \log \frac{1}{1 + \exp(\beta^T x_i)})$$

$$= \sum_{i=1}^{N} (y_i \beta^T x_i - \log(1 + \exp(\beta^T x_i)))$$

i.e. negative cross entropy

Today: Logistic Regression **Classifier**

$$P(y=1|x) = \frac{e^{\beta_0+\beta^T x}}{1+e^{\beta_0+\beta^T x}}$$

Data: X

↓

Task: y

↓

Representation: : x, f()

↓

Score Function: L()

↓

Search/Optimization : argmin()

↓

Models, Parameters, metrics

X: Tabular

↓

Binary Classification

↓

Log-odds y = linear function of Xs

↓

[MLE] / neg loglike

↓

Iterative (Newton) method / SGD

↓

Logistic weights / Accuracy / F1

$P(y=0|x) = 1 - P(y=1|x)$

Extra

Summary: MLE for Logistic Regression Training

Let's fit the logistic regression model for K=2, i.e., number of classes is 2

Training set: (x_i, y_i), i=1,…,N

(conditional)
Log-likelihood:

How?

For Bernoulli distribution

$$p(y \mid x)^{y}(1-p)^{1-y}$$

$$l(\beta)=\sum_{i=1}^{N}\{\log \Pr(Y=y_i \mid X=x_i)\}$$

$$=\sum_{i=1}^{N} y_i \log(\Pr(Y=1 \mid X=x_i))+(1-y_i)\log(\Pr(Y=0 \mid X=x_i))$$

$$=\sum_{i=1}^{N}(y_i \log\frac{\exp(\beta^T x_i)}{1+\exp(\beta^T x_i)})+(1-y_i)\log\frac{1}{1+\exp(\beta^T x_i)})$$

$$=\sum_{i=1}^{N}(y_i \beta^T x_i-\log(1+\exp(\beta^T x_i)))$$

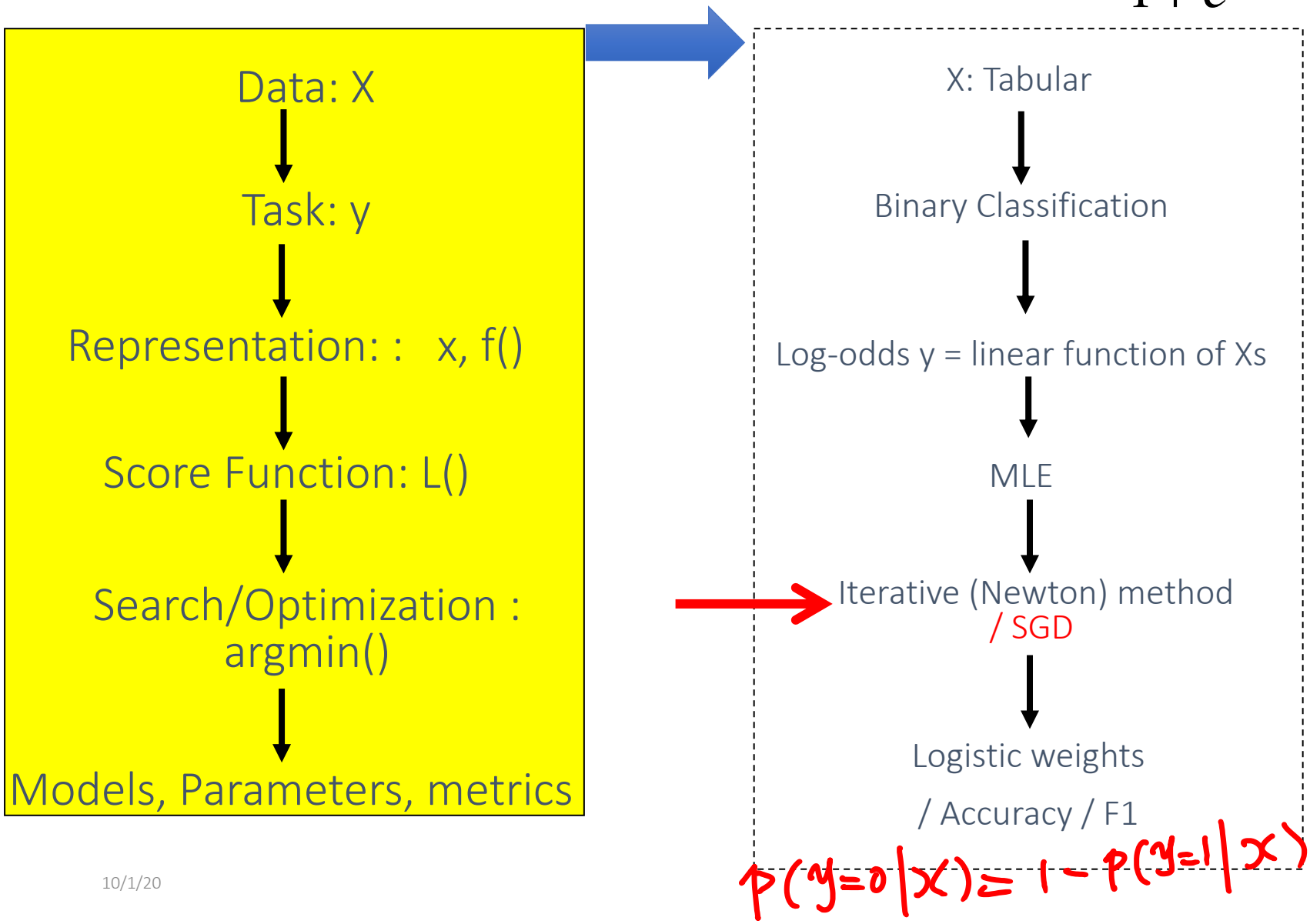
x_i are (p+1)-dimensional input vector with leading entry 1
\beta is a (p+1)-dimensional vector

We want to maximize the log-likelihood in order to estimate \beta

See Extra Slides How to used Newton-Raphson optimization

Today: Logistic Regression **Classifier**

$$P(y=1|x) = \frac{e^{\beta_0+\beta^T x}}{1+e^{\beta_0+\beta^T x}}$$



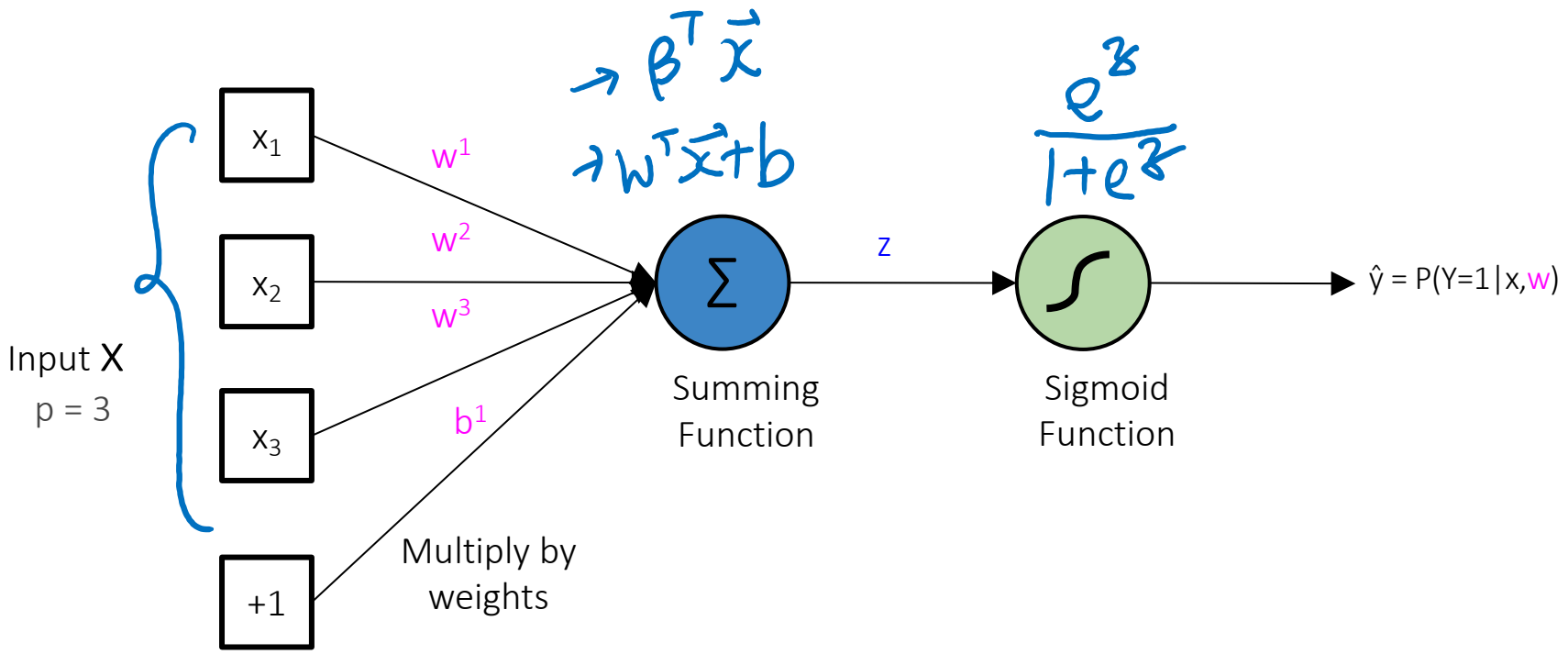
ReWrite Logistic Regression as two stages:

VIEW II

First:
Summing $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$

Second:
Sigmoid
Squashing $\hat{y} = P(y=1|x) = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_p x_p}}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_p x_p}} = \frac{e^z}{1+e^z}$

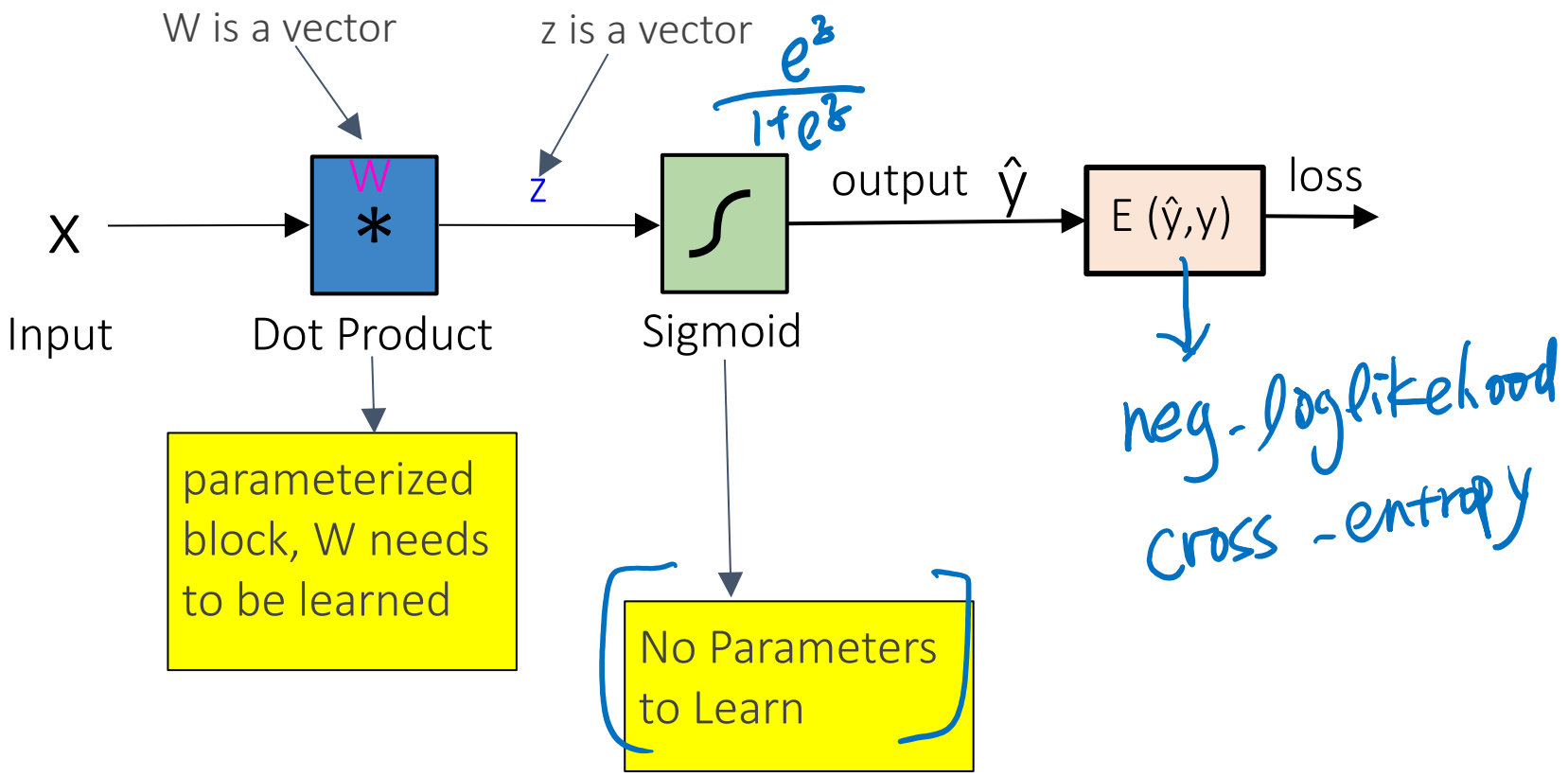
One "Neuron": Block View of Logistic Regression



$$z = w^T \cdot x + b$$

$$y = \text{sigmoid}(z) = \frac{e^z}{1 + e^z}$$

Next: "Block View" of Logistic Regression (plus loss)



Next: Stochastic GD ➔

- For LR: linear regression, We have the following gradient descent rule:

$$\theta_j^{t+1} = \theta_j^{t} - \alpha \frac{\partial}{\partial \theta_j} J(\theta)\Big|_t$$

J(θ) ↓ ⇓ θ

- ➔ For neural network, we have the delta rule

$$\Delta \mathbf{w} = -\eta \frac{\partial E}{\partial W^t}$$

$$W^{t+1} = W^t - \eta \frac{\partial E}{\partial W^t} = W^t + \Delta w$$

E(w) ↓ ⇓ w

Takeaway: Logistic Regression **Classifier**

- View I: logit(y) as linear of Xs
- View II: model Y as Bernoulli with p(y=1|x) as p(Head)
- View III: S" shape function compress to [0,1]
- View IV: models a linear classification boundary!
- View V: Two stages: summation + sigmoid

Today: Logistic Regression **Classifier**

P(Head)

$$P(y=1|x)=\frac{e^{\beta_0+\beta^T x}}{1+e^{\beta_0+\beta^T x}}$$

Data: X

↓

Task: y

↓

Representation: : x, f()

↓

Score Function: L()

↓

Search/Optimization : argmin()

↓

Models, Parameters, metrics

X: Tabular

↓

Binary Classification

↓

Log-odds y = linear function of Xs

↓

MLE

↓

Iterative (Newton) method / SGD

↓

Logistic weights

/ Accuracy / F1


Thank You

References

- Prof. Tan, Steinbach, Kumar's "Introduction to Data Mining" slide
- Prof. Andrew Moore's slides
- Prof. Eric Xing's slides
- Prof. Ke Chen NB slides
- Hastie, Trevor, et al. *The elements of statistical learning*. Vol. 2. No. 1. New York: Springer, 2009.

Today: Extra

- ✓ Bayes Classifier and MAP Rule?
  - Bayes Classifier
  - Expected Prediction Error
  - 0-1 Loss function for Bayes Classifier
- ✓ Logistic regression
  - Parameter Estimation for LR

$$p(y|x) = \frac{e^{\beta x}}{1+e^{\beta x}} \Rightarrow \beta$$

Bayes classifiers

- Treat each feature attribute and the class label as random variables.

- Given a sample x with attributes ($x_1, x_2, \ldots, x_p$):
  - Goal is to predict its class C.
  - Specifically, we want to find the value of C_i that maximizes $p(C_i \mid x_1, x_2, \ldots, x_p)$.

- Can we estimate $p(C_i \mid x) = p(C_i \mid x_1, x_2, \ldots, x_p)$ directly from data?

Bayes classifiers

- Treat each feature attribute and the class label as random variables.

- Given a sample x with attributes ($x_1, x_2, \ldots, x_p$):
  - Goal is to predict its class C.
  - Specifically, we want to find the value of C_i that maximizes $p(C_i \mid x_1, x_2, \ldots, x_p)$.

- Can we estimate $p(C_i \mid x) = p(C_i \mid x_1, x_2, \ldots, x_p)$ directly from data?

→ $\{C_1, C_2, \ldots, C_L\}$

Bayes classifiers
➔ MAP classification rule

- Establishing a probabilistic model for classification

➔ MAP classification rule

  - MAP: Maximum A Posterior
  - Assign x to c* if

$$P(C = c^* \mid \mathbf{X} = \mathbf{x}) > P(C = c \mid \mathbf{X} = \mathbf{x})$$

$$\text{for } c \neq c^*, \; c = c_1, \cdots, c_L$$

Bayes classifiers
➔ MAP classification rule

- Establishing a probabilistic model for classification

➔ MAP classification rule

- MAP: Maximum A Posterior
- Assign x to c* if

$$P(C = c^* \mid \mathbf{X} = \mathbf{x}) > P(C = c \mid \mathbf{X} = \mathbf{x}) \quad c \neq c^*,\ c = c_1, \cdots, c_L$$

$$\left\{ \begin{array}{l} P(C = c_1 \mid x) \\ P(C = c_2 \mid x) \\ P(C = c_3 \mid x) \end{array} \right\} \Rightarrow \max \Rightarrow c_i$$

Adapt from Prof. Ke Chen NB slides

Bayes Classifiers – MAP Rule

Task: Classify a new instance X based on a tuple of attribute values into one of the classes $X = \langle X_1, X_2, \ldots, X_p \rangle$

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid x_1, x_2, \ldots, x_p)$$

WHY ?

MAP = Maximum Aposteriori Probability

Recap: Statistical Decision Theory (Extra)

- Random input vector: X
- Random output variable: Y
- Joint distribution: Pr(X,Y) $\Rightarrow D = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\}$
- Loss function L(Y, f(X))

- Expected prediction error (EPE):

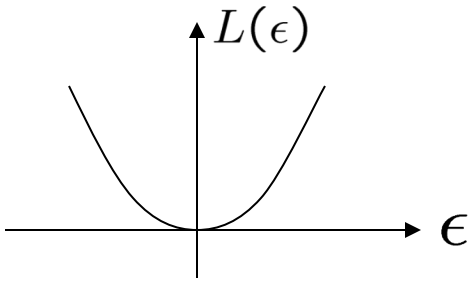
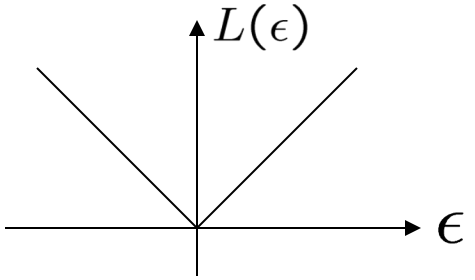
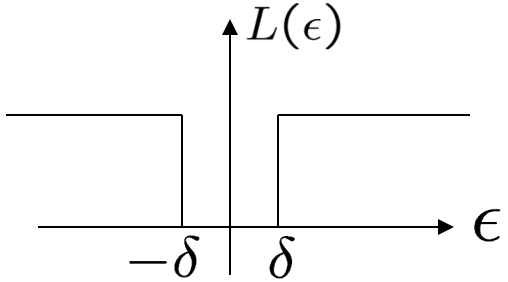
$$\mathrm{EPE}(f) = \mathrm{E}(L(Y, f(X))) = \int L(y, f(x)) \Pr(dx, dy)$$

$$\text{e.g.} = \int (y - f(x))^2 \Pr(dx, dy)$$

e.g. Squared error loss (also called L2 loss)

Consider population distribution

SUMMARY: WHEN Expected prediction error (EPE) USES DIFFERENT LOSS (Extra)

Loss Function	Estimator $\hat{f}(x)$
L_2 	$\hat{f}(x) = E[Y\|X = x]$
L_1 	$\hat{f}(x) = \text{median}(Y\|X = x)$
0-1 	$\hat{f}(x) = \arg\max_Y P(Y\|X = x)$ (Bayes classifier / MAP)

Please read extra slides for how to get MAP-rule from minimize EPE-0/1

0-1 LOSS for Classification

if $k = \ell$, $L(k, \ell) = 0$

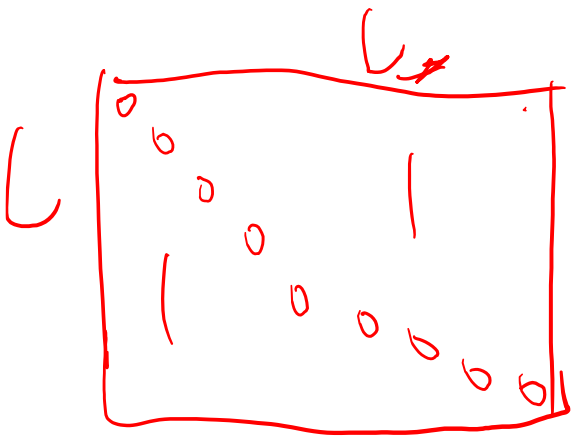
- Procedure for categorical output variable C

if $k \neq \ell$, $L(k, \ell) = 1$

- Frequently, 0-1 loss function used: L(k, ℓ)

- L(k, ℓ) is the price paid for misclassifying an element from class C_k as belonging to class C_ℓ

  ➔ L*L matrix



$C_1, C_2, \cdots, C_L$.

Expected prediction error (EPE)

- Expected prediction error (EPE), with expectation taken w.r.t. the joint distribution Pr(C,X)
  - Pr(C,X)=Pr(C |X)Pr(X)

e.g. 0-1 loss

$E_x(X)$

$E_x(g(X))$

$$\mathrm{EPE}(f) = E_{X,C}(L(C, f(X))$$

$$= \mathrm{E}_X \sum_{k=1}^{L} L[C_k, f(X)]\Pr(C_k \mid X)$$

Consider sample population distribution

$$EPE(f) = E_{X,C}\left(L(C, f(X))\right)$$

$$= E_X E_{C|X}\left[L(C, f(X)) \mid X\right]$$

Discrete RV's Expectation

$$= E_X \sum_{k=1}^{L} L[C_k, f(X)]\, Pr(C_k|X)$$

$$E_C(C) = \sum_{i=1}^{L} C_i P(C_i)$$

$\underset{f}{\operatorname{argmin}}\ EPE(f(X))$

$\Rightarrow$ Pointwise minimization when $X = x$

$$\Rightarrow \hat{f}(X=x) = \underset{f(x)\in C}{\operatorname{argmin}} \sum_{k=1}^{L} L[C_k, f(x)]\, Pr(C_k|X=x)$$

$$\Rightarrow \hat{f}(x) = \underset{C_k \in \{C_1, C_2, C_3, \ldots, C_L\}}{\operatorname{argmax}} Pr(C_k|X=x)$$

$\{P(C_1|x), P(C_2|x), \ldots, P(C_L|x)\}$

Expected prediction error (EPE)

$$\mathrm{EPE}(f) = E_{X,C}(L(C, f(X)) = \mathrm{E}_X \sum_{k=1}^{K} L[C_k, f(X)]\Pr(C_k \mid X)$$

Consider sample population distribution

- Pointwise minimization suffices
- ➔ simply

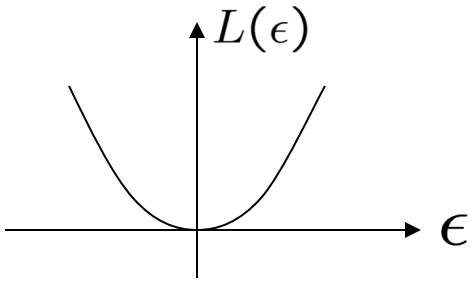
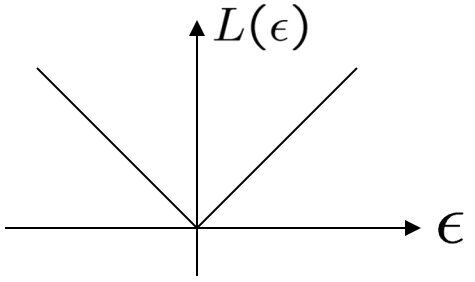
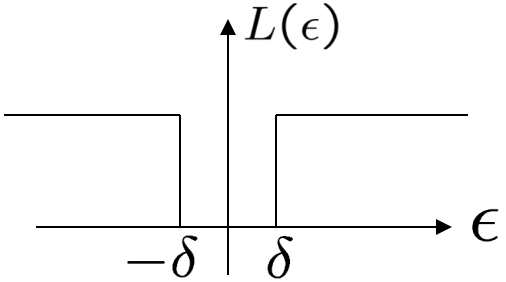
$$\hat{f}(X) = \operatorname{argmin}_{g \in C} \sum_{k=1}^{K} L(C_k, g)\Pr(C_k \mid X = x)$$

$$\hat{f}(X) = C_k \text{ if}$$

$$\Pr(C_k \mid X = x) = \max_{g \in C} \Pr(g \mid X = x)$$

Bayes Classifier

SUMMARY: WHEN Expected prediction error (EPE) USES DIFFERENT LOSS

Loss Function	Estimator $\hat{f}(x)$
L_2 	$EPE = E_{X,Y}\left(Y - f(X)\right)^2$ $\hat{f}(x) = E[Y \mid X = x]$
L_1 	$\hat{f}(x) = \text{median}(Y \mid X = x)$
0-1 	$\hat{f}(x) = \arg\max_Y P(Y \mid X = x)$ (Bayes classifier / MAP)

Today: Extra

- ✓ Why Bayes Classification – MAP Rule?
  - Expected Prediction Error
  - 0-1 Loss function for Bayes Classifier
- ✓ Logistic regression
  - Parameter Estimation for LR

$$p(y|x) = \frac{e^{\beta x}}{1+e^{\beta x}}$$

$\Downarrow$

β

Logistic Regression Assumptions

- Linearity in the logit – the regression equation should have a linear relationship with the logit form of the target variable

- There is no assumption about the feature variables / target predictors being linearly related to each other.

$P(y=1|x)$ $1-p(y=1x)$

func of x with parameter $\vec{\beta}$ to learn from training data

Newton's method for optimization

- The most basic second-order optimization algorithm
- Updating parameter with

GD: $\theta_{k+1} = \theta_k - \alpha g_k$

Newton: $$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mathbf{H}_K^{-1}\mathbf{g}_k$$

$p \times p$ $p \times 1$

$p \times 1$

Review: Hessian Matrix / n==2 case

Singlevariate → multivariate

$$f(x,y)$$

- 1st derivative to gradient,

$$g = \nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

- 2nd derivative to Hessian

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}$$

Review: Hessian Matrix

Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is a function that takes a vector in $\mathbb{R}^n$ and returns a real number. Then the ***Hessian*** matrix with respect to x, written $\nabla_x^2 f(x)$ or simply as H is the $n \times n$ matrix of partial derivatives,

$$\nabla_x^2 f(x) \in \mathbb{R}^{n \times n} = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}.$$

Newton's method for optimization

- Making a quadratic/second-order Taylor series approximation

$$\hat{f}_{quad}(\boldsymbol{\theta}) = f(\boldsymbol{\theta}_k) + \mathbf{g}_k^T(\boldsymbol{\theta} - \boldsymbol{\theta}_k) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_k)^T\mathbf{H}_k(\boldsymbol{\theta} - \boldsymbol{\theta}_k)$$

Finding the minimum solution of the above right quadratic approximation (quadratic function minimization is easy !)

$$\hat{f}(\theta) = f(\theta_k) + g_k^T(\theta - \theta_k) + \frac{1}{2}(\theta - \theta_k)^T H_k (\theta - \theta_k)$$

$$\Downarrow$$

$$\frac{1}{2}\left(\theta^T H_k \theta - 2\theta^T H_k \theta_k + \theta_k^T H_k \theta_k\right)$$

$$\frac{\partial \hat{f}(\theta)}{\partial \theta} = 0 + g_k + \frac{2}{2} H_k \theta - \frac{2}{2} H_k \theta_k := 0$$

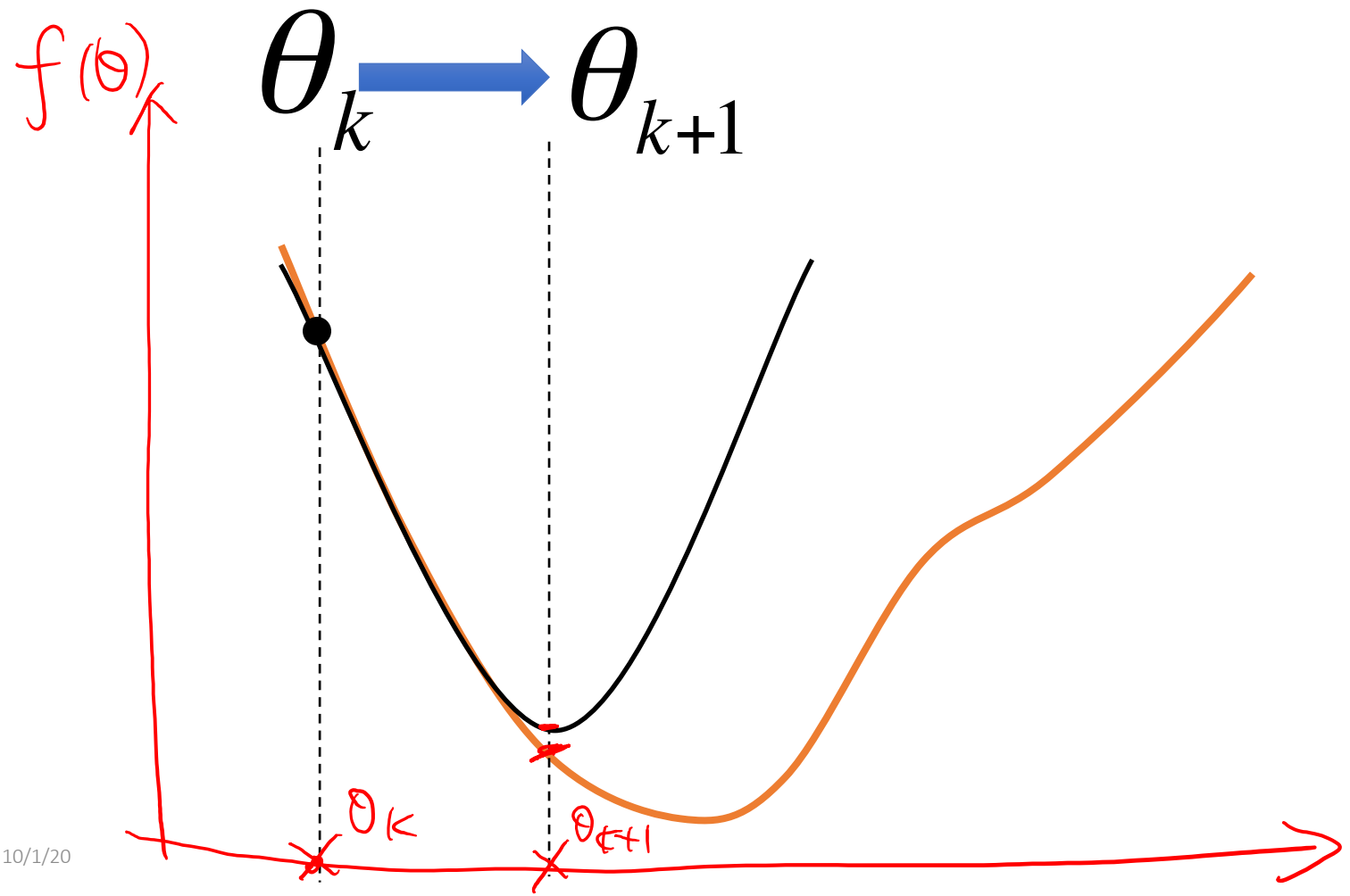
see p24 handout

$$g_k + H_k(\theta - \theta_k) = 0$$

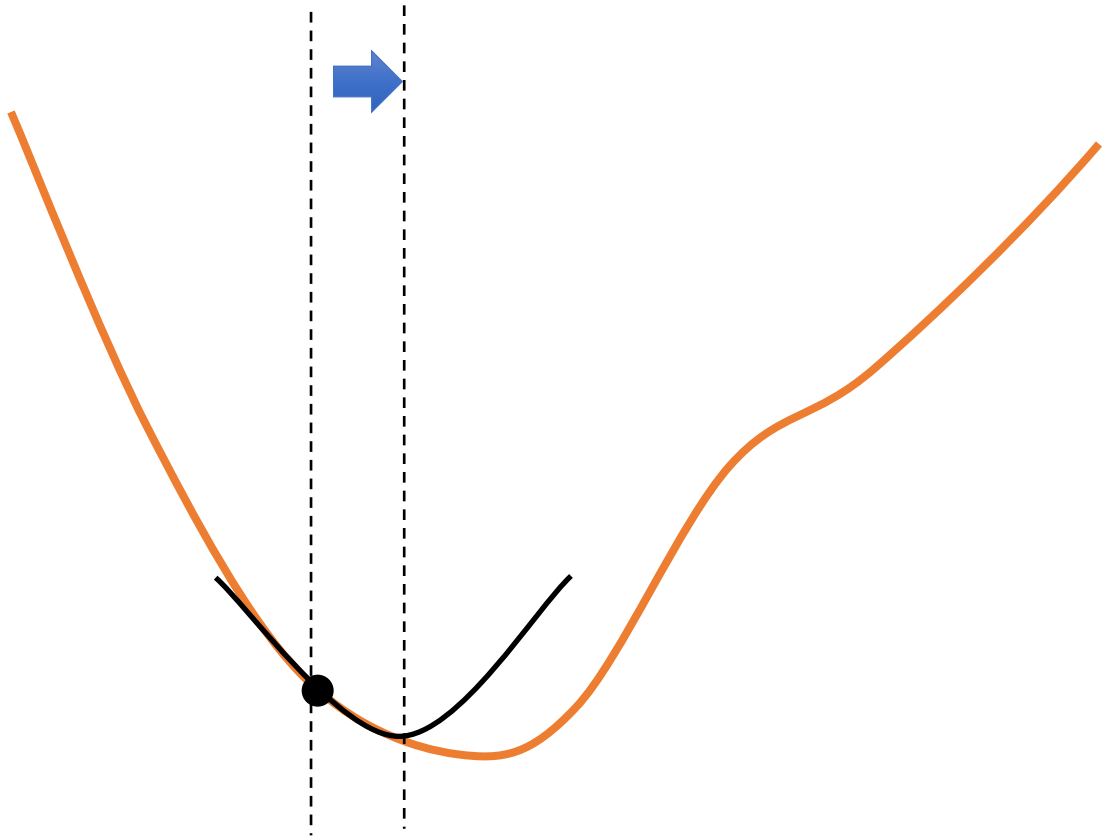
$$\Rightarrow \quad \theta = \theta_k - H_k^{-1} g_k$$

where $H_k \in \mathbb{R}^{p \times p}$, $g_k \in \mathbb{R}^p$

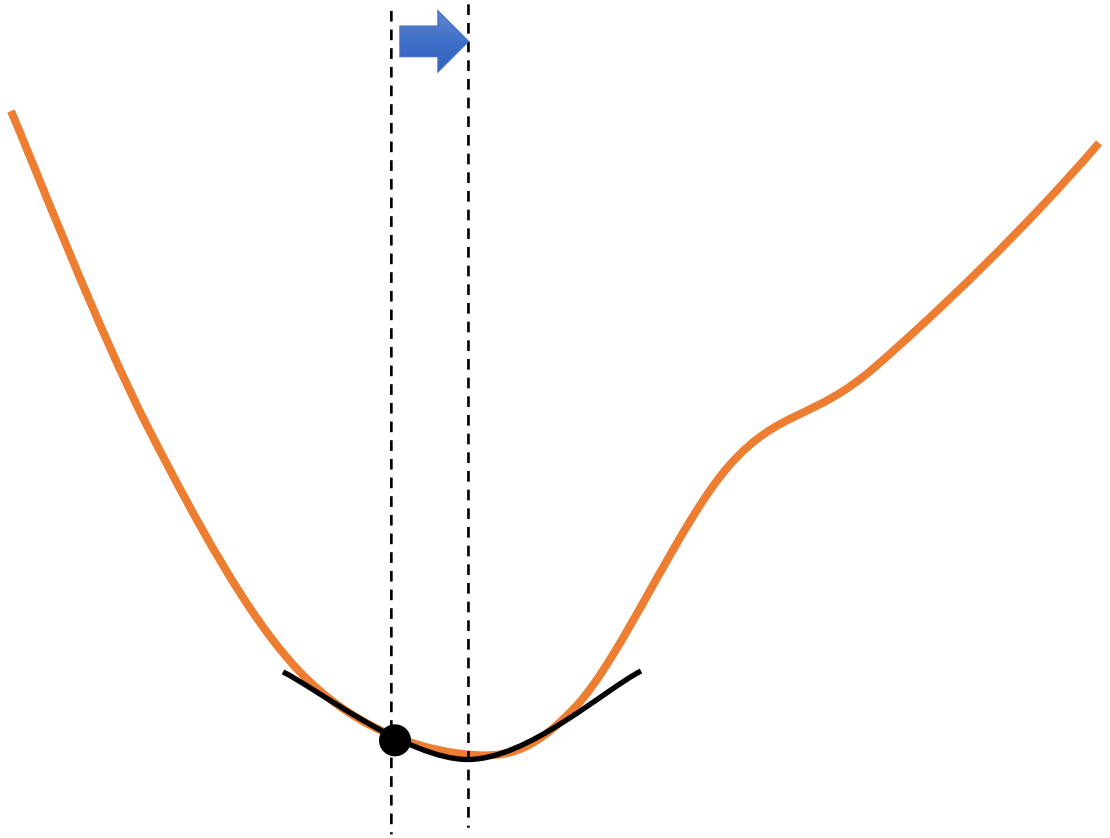
Newton's Method / second-order Taylor series approximation



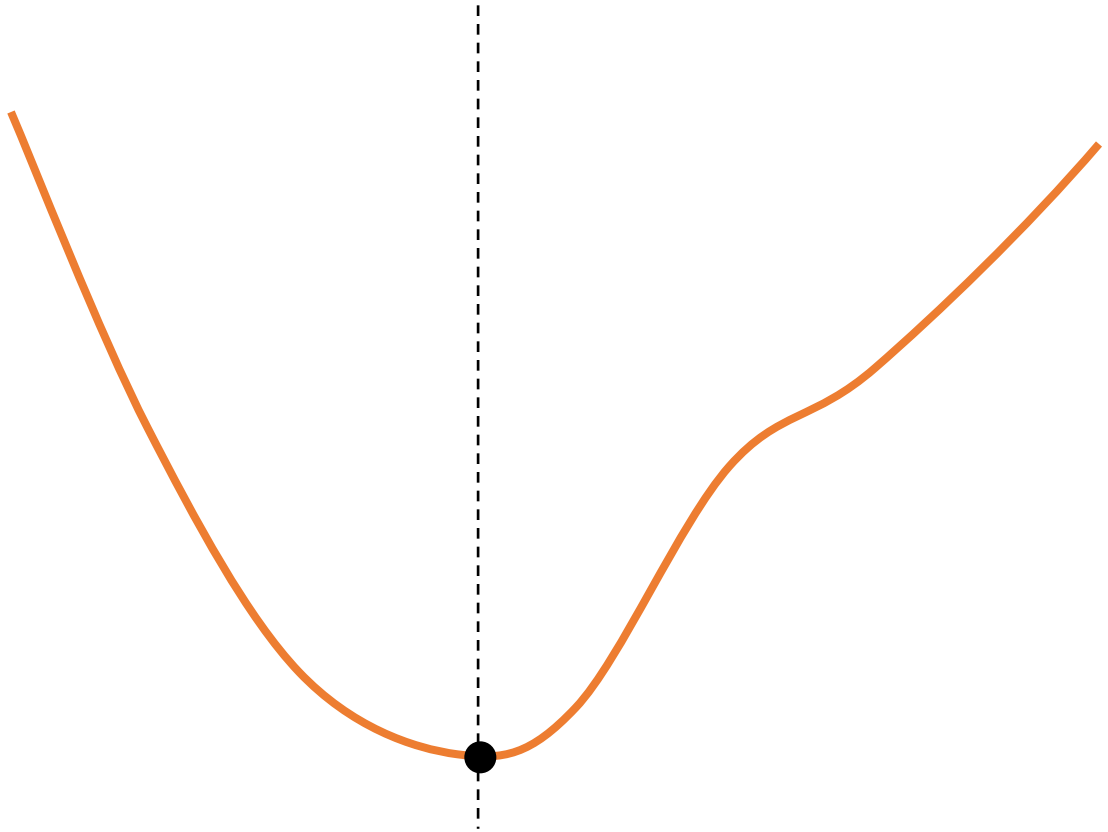
Newton's Method / second-order Taylor series approximation



Newton's Method / second-order Taylor series approximation



Newton's Method / second-order Taylor series approximation



Newton's Method

- At each step:

$$\theta_{k+1} = \theta_k - \frac{f'(\theta_k)}{f''(\theta_k)}$$

$$\theta_{k+1} = \theta_k - H^{-1}(\theta_k)\nabla f(\theta_k)$$

- Requires 1st and 2nd derivatives
- Quadratic convergence
- ➔ However, finding the inverse of the Hessian matrix is often expensive

Newton vs. GD for optimization

- Newton: a quadratic/second-order Taylor series approximation

$$\hat{f}_{quad}(\boldsymbol{\theta}) = f(\boldsymbol{\theta}_k) + \mathbf{g}_k^T(\boldsymbol{\theta} - \boldsymbol{\theta}_k) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_k)^T \mathbf{H}_k(\boldsymbol{\theta} - \boldsymbol{\theta}_k)$$

$\theta_{k+1} = \theta_k - \frac{1}{H(\theta_k)} g(\theta_k)$

Finding the minimum solution of the above right quadratic approximation (quadratic function minimization is easy !)

$$\hat{f}_{quad}(\boldsymbol{\theta}) = f(\boldsymbol{\theta}_k) + \mathbf{g}_k^T(\boldsymbol{\theta} - \boldsymbol{\theta}_k) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_k)^T \frac{1}{\alpha}(\boldsymbol{\theta} - \boldsymbol{\theta}_k)$$

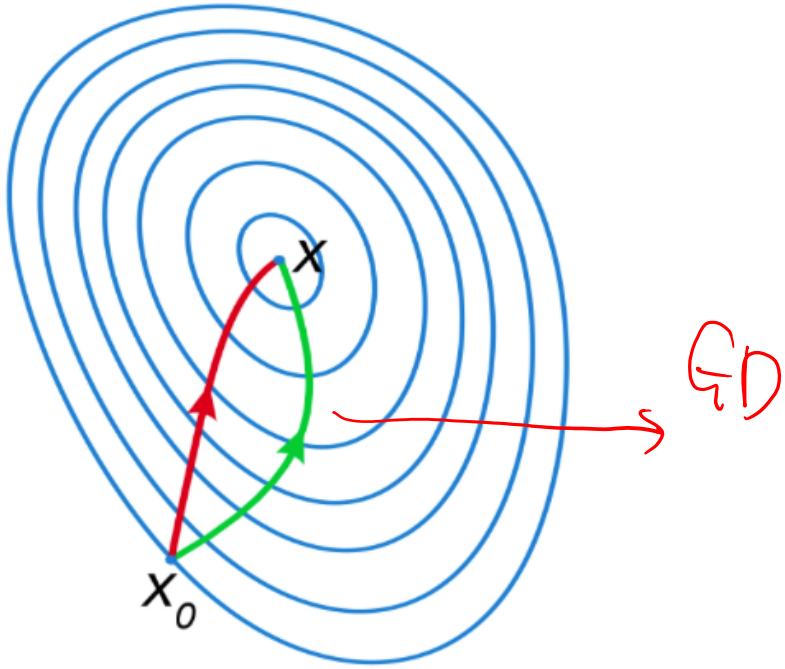
$\theta_{k+1} = \theta_k - \alpha\, g(\theta_k)$

Comparison

- Newton's method vs. Gradient descent

A comparison of gradient descent (green) and Newton's method (red) for minimizing a function (with small step sizes).

Newton's method uses curvature information to get a more direct route …



MLE for Logistic Regression Training

Let's fit the logistic regression model for K=2, i.e., number of classes is 2

Training set: (x_i, y_i), i=1,…,N

For Bernoulli distribution

$$p(y \mid x)^y (1-p)^{1-y}$$

(conditional) Log-likelihood:

How?

$$l(\beta) = \sum_{i=1}^{N} \{\log \Pr(Y = y_i \mid X = x_i)\}$$

$$= \sum_{i=1}^{N} y_i \log(\Pr(Y=1 \mid X = x_i)) + (1-y_i)\log(\Pr(Y=0 \mid X = x_i))$$

$$= \sum_{i=1}^{N} (y_i \log \frac{\exp(\beta^T x_i)}{1+\exp(\beta^T x_i)}) + (1-y_i)\log \frac{1}{1+\exp(\beta^T x_i)})$$

$$= \sum_{i=1}^{N} (y_i \beta^T x_i - \log(1+\exp(\beta^T x_i)))$$

x_i are (p+1)-dimensional input vector with leading entry 1
\beta is a (p+1)-dimensional vector

We want to maximize the log-likelihood in order to estimate \beta

$$l(\beta) = \sum_{i=1}^{N} \{\log \Pr(Y = y_i \mid X = x_i)\}$$

y_i

$\log \{\Pr(Y = y_i \mid X = x_i) = p(y_i \mid x_i)\} \Rightarrow$ $y_i = 1$, $y_i = 0$ ($p(y_i=1|x)$, $1-p(y_i=1|x)$)

$= \log \{p(y_i=1|x)^{y_i} (1 - p(y_i=1|x_i))^{1-y_i}\}$

$= y_i \log p(y_i=1|x) + (1-y_i) \log (1-p(y_i=1|x))$

Newton-Raphson for LR (optional)

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{N} (y_i - \frac{\exp(\beta^T x)}{1+\exp(\beta^T x)}) x_i = 0$$

Vector

(p+1) Non-linear equations to solve for (p+1) unknowns

β

Solve by Newton-Raphson method:

$$\beta^{new} \leftarrow \beta^{old} - [(\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T})]^{-1} \frac{\partial l(\beta)}{\partial \beta},$$

$$\text{where, } (\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T}) = -\sum_{i=1}^{N} x_i x_i^T (\frac{\exp(\beta^T x_i)}{1+\exp(\beta^T x_i)})(\frac{1}{1+\exp(\beta^T x_i)})$$

minimizes a quadratic approximation
to the function we are really interested in.

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mathbf{H}_K^{-1} \mathbf{g}_k$$

$p(x_i ; \beta)$

$1 - p(x_i ; \beta)$

Newton-Raphson for LR…

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{N} (y_i - \frac{\exp(\beta^T x)}{1+\exp(\beta^T x)}) x_i = X^T (y - p)$$

$$p(y=1|x) = \frac{e^{\beta^T x}}{1+e^{\beta^T x}}$$

$$(\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T}) = -X^T W X$$

So, NR rule becomes:

$$\beta^{new} \leftarrow \beta^{old} + (X^T W X)^{-1} X^T (y - p),$$

$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix}_{N-by-(p+1)}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}_{N-by-1}, \quad p = \begin{bmatrix} \exp(\beta^T x_1)/(1+\exp(\beta^T x_1)) \\ \exp(\beta^T x_2)/(1+\exp(\beta^T x_2)) \\ \vdots \\ \exp(\beta^T x_N)/(1+\exp(\beta^T x_N)) \end{bmatrix}_{N-by-1},$$

$X : N \times (p+1)$ matrix of x_i

$y : N \times 1$ matrix of y_i

$p : N \times 1$ matrix of $p(x_i; \beta^{old})$

$W : N \times N$ diagonal matrix of $p(x_i; \beta^{old})(1 - p(x_i; \beta^{old}))$

$$(\frac{\exp(\beta^T x_i)}{(1+\exp(\beta^T x_i))})(1 - \frac{1}{(1+\exp(\beta^T x_i))})$$

Newton-Raphson for LR…

- Newton-Raphson
  - $\beta^{new} = \beta^{old} + (X^T W X)^{-1} X^T (y - p)$

    $= (X^T W X)^{-1} X^T W (X\beta^{old} + W^{-1}(y - p))$

    $= (X^T W X)^{-1} X^T W z$

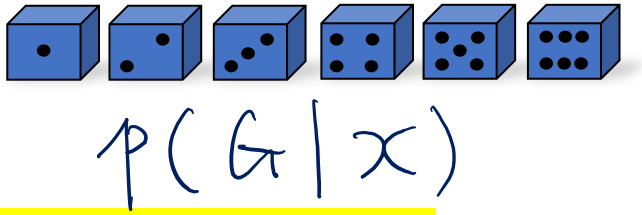
    Re expressing Newton step as weighted least square step
  - Adjusted response

    $z = X\beta^{old} + W^{-1}(y - p)$
  - Iteratively reweighted least squares (IRLS)

$$\beta^{new} \leftarrow \arg\min_{\beta} (z - X\beta^T)^T W (z - X\beta^T)$$

$$\leftarrow \arg\min_{\beta} (y - p)^T W^{-1} (y - p)$$

Binary ➔ Multinoulli Logistic Regression Model



Directly models the posterior probabilities as the output of regression

$$\Pr(G=k \mid X=x)=\frac{\exp(\beta_{k0}+\beta_k^T x)}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\beta_l^T x)},\ k=1,\ldots,K-1$$

$$\Pr(G=K \mid X=x)=\frac{1}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\beta_l^T x)}$$

x is p-dimensional input vector

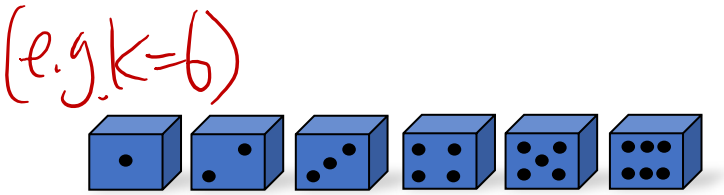
β_k^T is a p-dimensional vector for each class k

Total number of parameters is (K-1)(p+1)

$\beta_{k0}, \vec{\beta_k}$, $k=1,2,\ldots,K-1$

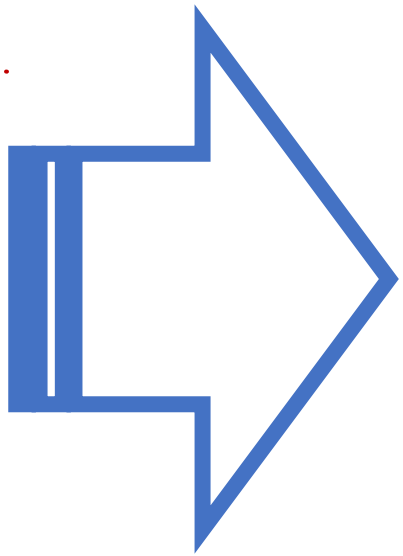
Note that the class boundaries are linear

Binary ➔ Multinoulli Logistic Regression Model



$$p(y=1|x) \Rightarrow \frac{e^{\beta x}}{1+e^{\beta x}}$$

$$p(y=0|x) \Rightarrow \frac{1}{1+e^{\beta x}}$$



$$\frac{e^{\beta_k^T x}}{1+e^{\beta_1^T x}+e^{\beta_2^T x}\cdots}$$

e.g.

$$\ln \frac{P(G=k|x)}{P(G=K|x)} = 0 \Rightarrow \text{linear}$$

$$\beta_{k0}+\beta_k^T x$$

Note that the class boundaries are linear

References

- Prof. Tan, Steinbach, Kumar's "Introduction to Data Mining" slide
- Prof. Andrew Moore's slides
- Prof. Eric Xing's slides
- Prof. Ke Chen NB slides
- Hastie, Trevor, et al. *The elements of statistical learning*. Vol. 2. No. 1. New York: Springer, 2009.