

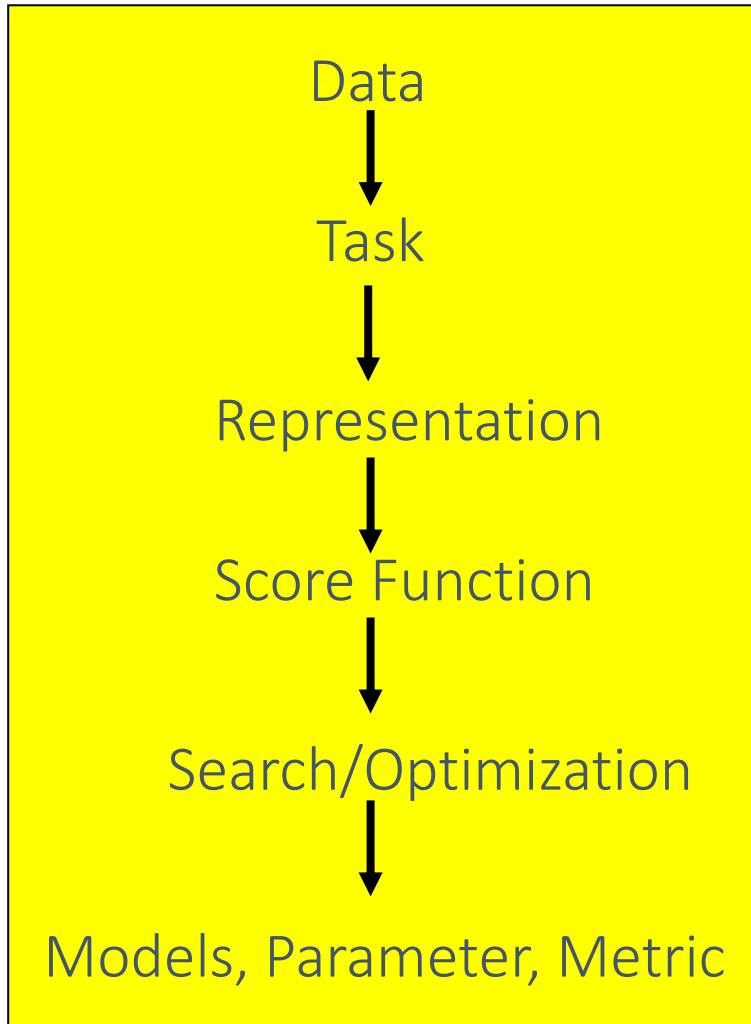
UVA CS 4774: Machine Learning

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

~~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, X_1, \dots, X_p

Reinforcement Learning

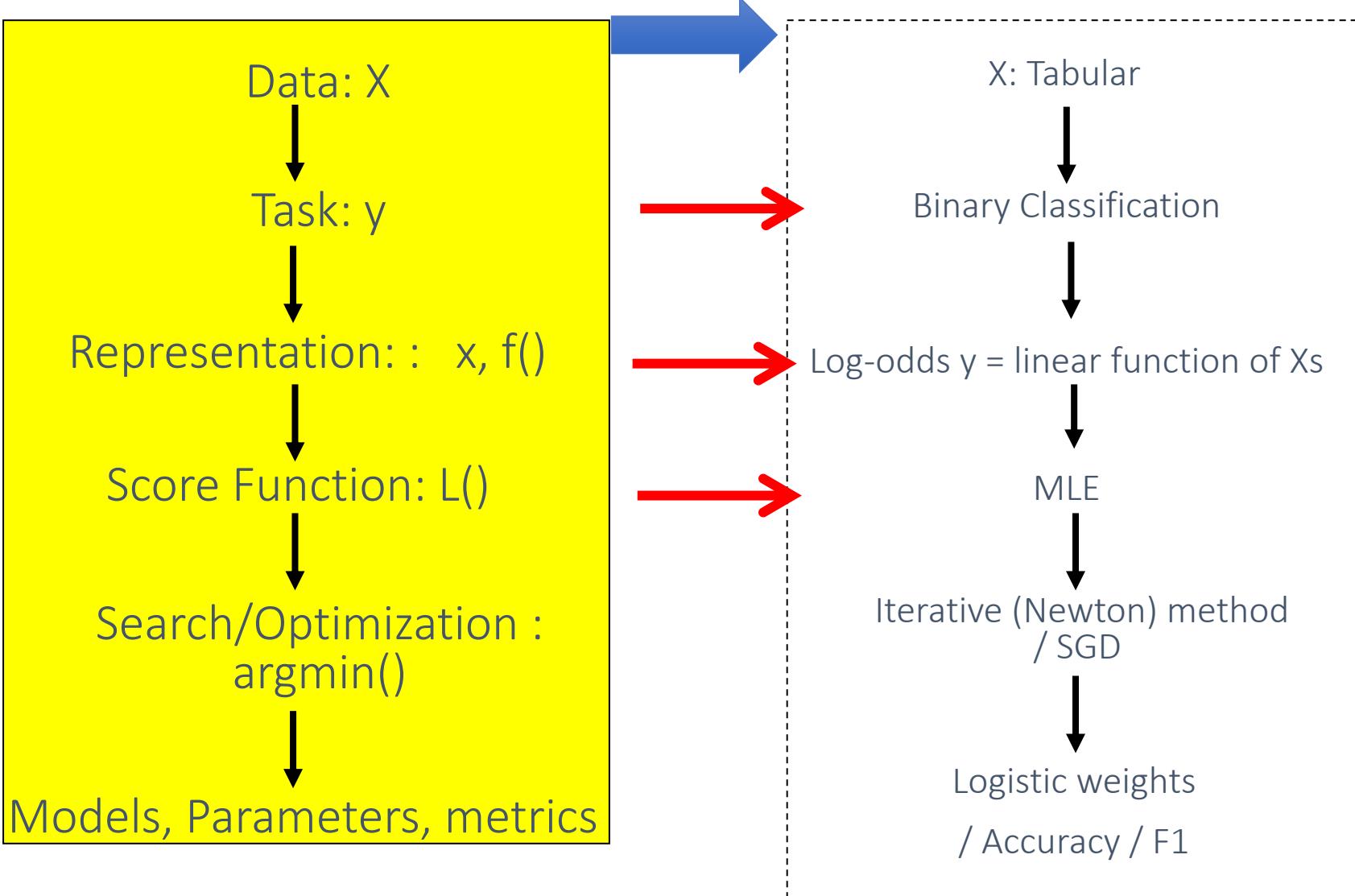
Learn to Interact with environment

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.

$$\overline{\{c_1, \dots, c_L\}}$$

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 $p(C=C_i | x) = p(C_i | x_1, x_2, \dots, x_p)$ directly from reference data?

Bayes classifiers

- Treat each feature attribute and the class label as random variables.
- **Testing:** Given a sample vector \mathbf{x} with attributes as (x_1, x_2, \dots, x_p) :
 - Goal is to predict its class
 - HOW: → we want to **find the class that maximizes** $p(c | x_1, x_2, \dots, x_p)$.

Bayes Classifiers – Predict via MAP Rule

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

based on:

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} P(c_j | x_1, x_2, \dots, x_p)$$

MAP Rule

MAP = Maximum Aposteriori Probability

Bayes Classifiers – Predict via MAP Rule

- Establishing a probabilistic model for classification
 - ➔ MAP classification rule
 - ➔ To Assign x to c^* if

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

for $c \neq c^*$, $c = c_1, \dots, c_L$

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

Recap: Statistical Decision Theory (Extra)

- Random input vector: X
- Random output variable: Y
- Joint distribution: $\Pr(X, Y) \Rightarrow D = \boxed{(\bar{x}_1, \bar{y}_1), \dots, (\bar{x}_n, \bar{y}_n)}$
- Loss function $L(Y, f(X))$
- Expected prediction error (EPE):

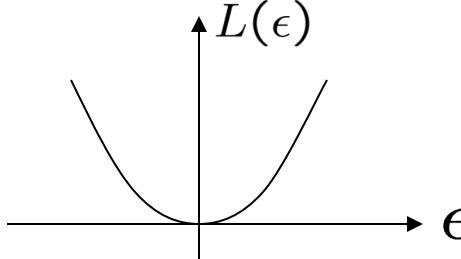
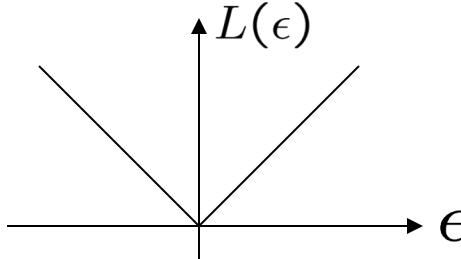
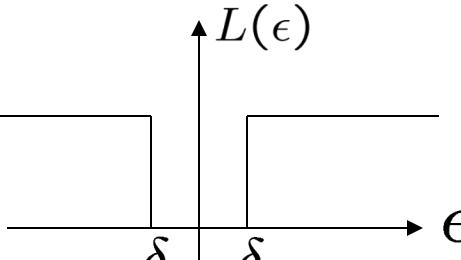
$$\text{EPE}(f) = E(L(Y, f(X))) = \int L(y, f(x)) \Pr(dx, dy)$$

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

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  $L(\epsilon) = \epsilon^2$	$\hat{f}(x) = E[Y X = x]$
L_1  $L(\epsilon) = \epsilon $	$\hat{f}(x) = \text{median}(Y X = x)$
$0-1$  $L(\epsilon) = \begin{cases} 0 & \text{if } -\delta \leq \epsilon \leq \delta \\ 1 & \text{otherwise} \end{cases}$	$\hat{f}(x) = \arg \max_Y P(Y X = x)$ (Bayes classifier / MAP)

$$f : [X] \longrightarrow [C]$$

Establishing a probabilistic model
for classification

Output as Discrete
Class Label
 C_1, C_2, \dots, C_L

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} P(c_j | x_1, x_2, \dots, x_p)$$

|| Generative

$$\operatorname{argmax}_{c \in C} P(c | X) = \operatorname{argmax}_{c \in C} P(X, c) = \operatorname{argmax}_{c \in C} P(X | c)P(c)$$

$$\frac{P(x, c)}{P(x)}$$

Later!

|| Discriminative

$$\operatorname{argmax}_{c \in C} P(c / \mathbf{X}) \quad C = \{c_1, \dots, c_L\}$$

Three major groups of classifiers

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

1. Discriminative

directly estimate a decision rule/boundary

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

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

2. Generative:

build a generative statistical model

e.g., Bayesian networks, Naïve Bayes classifier

3. Instance based classifiers

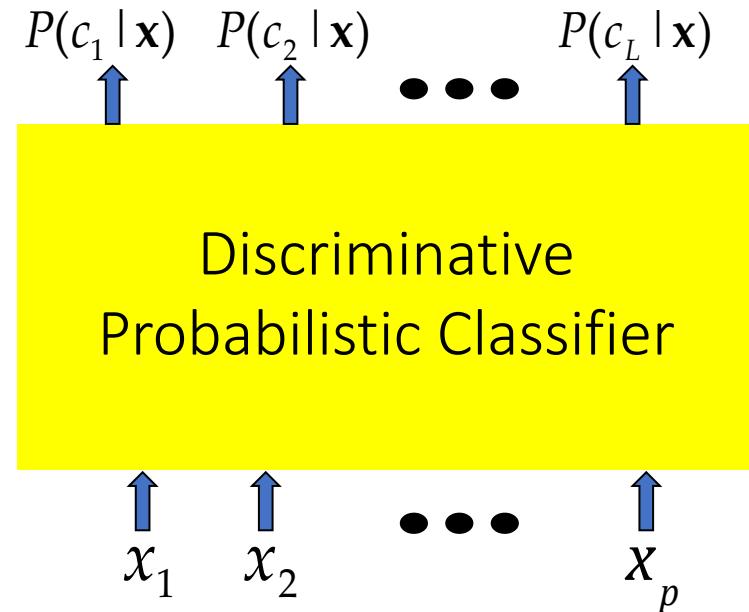
- Use observation directly (no models)
- ~~e.g. K nearest neighbors~~

Our Whole S2:

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

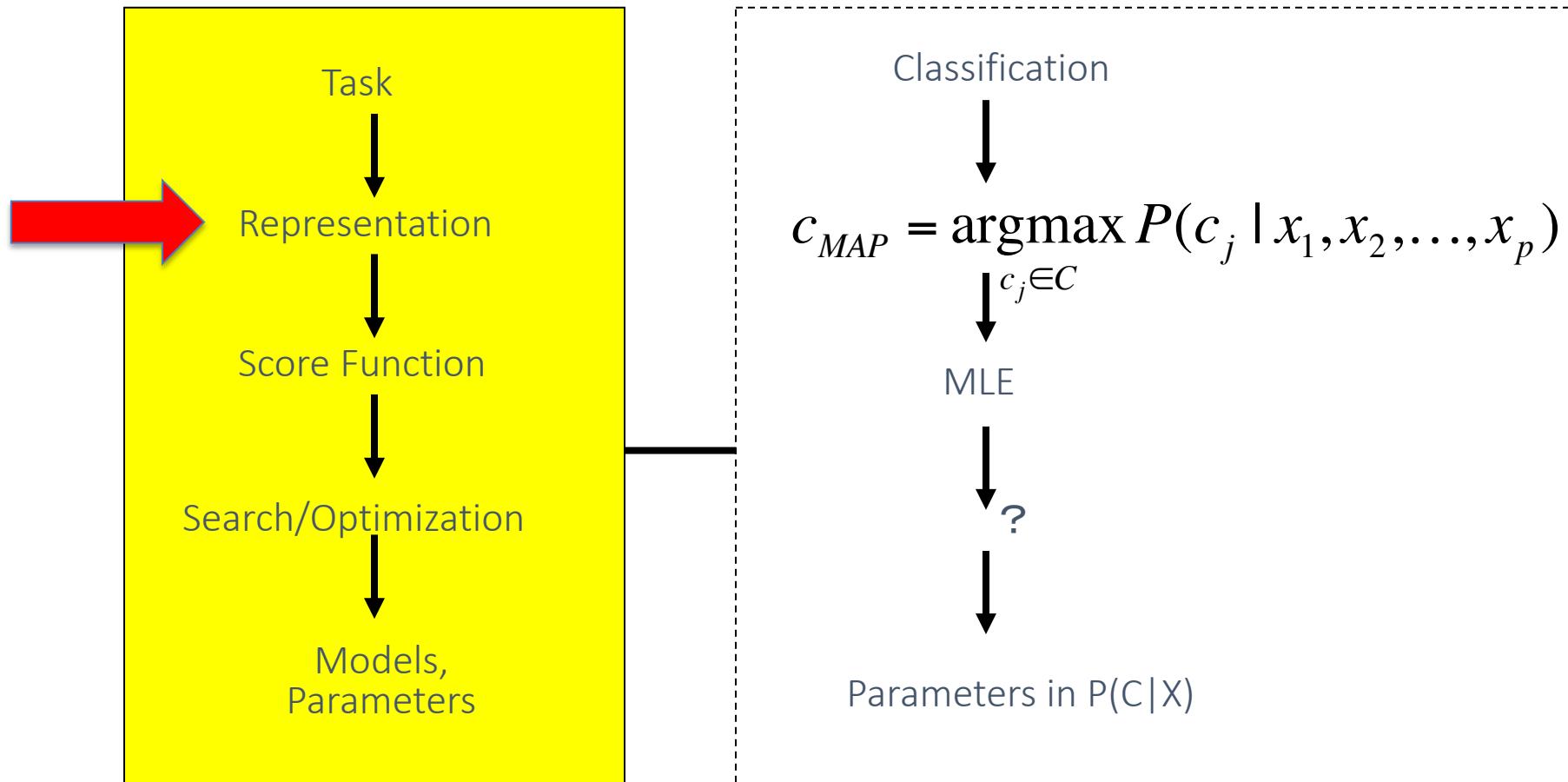
- Discriminative Classifiers

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



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

Bayes Classifier

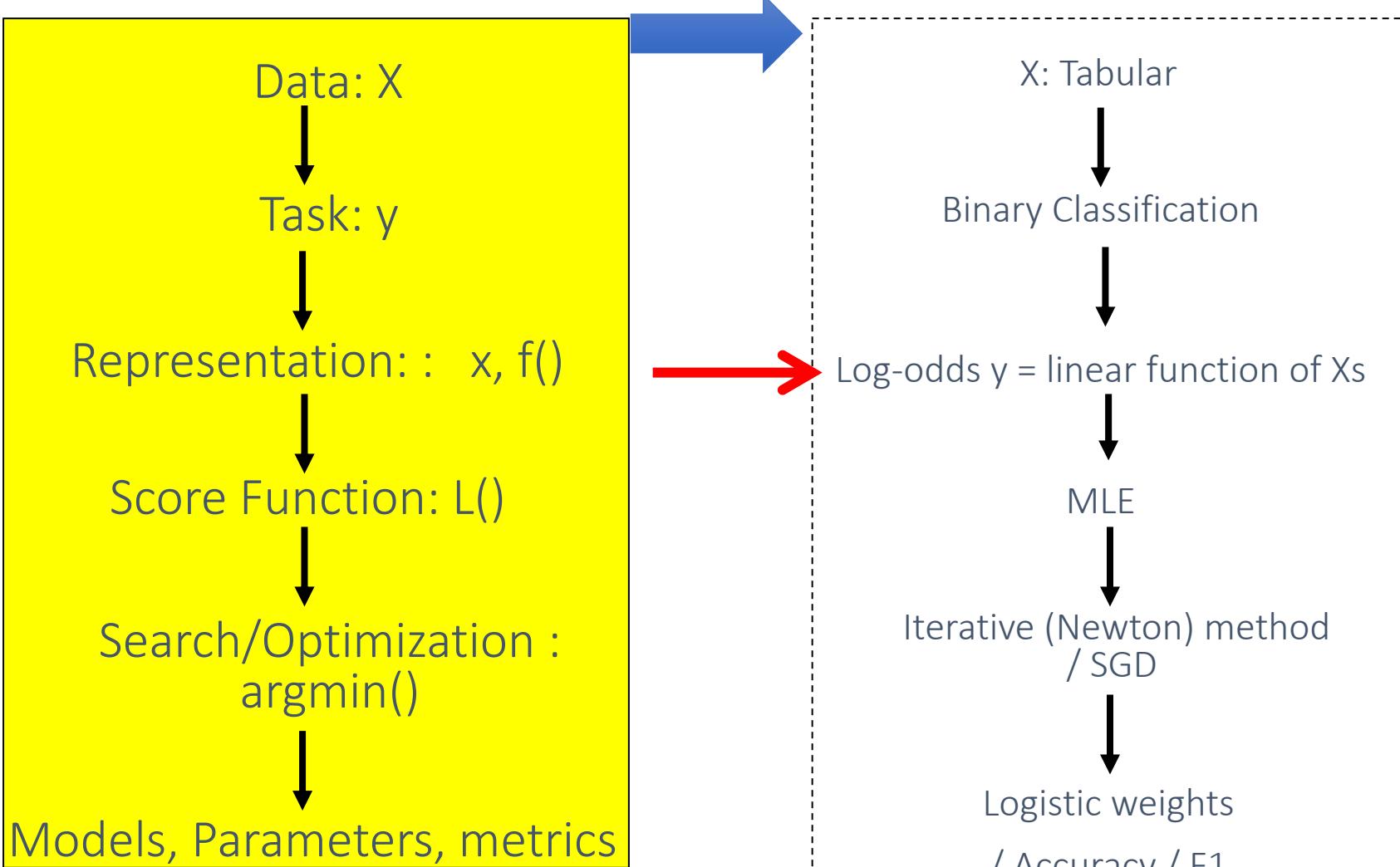


Roadmap

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

Today: Logistic Regression **Classifier**

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



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

Multivariate linear regression to Logistic Regression

$$\underline{y} = \underline{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p}$$

Logistic regression for
binary classification

Log-odds

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

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

Logistic Regression $p(y|x)$

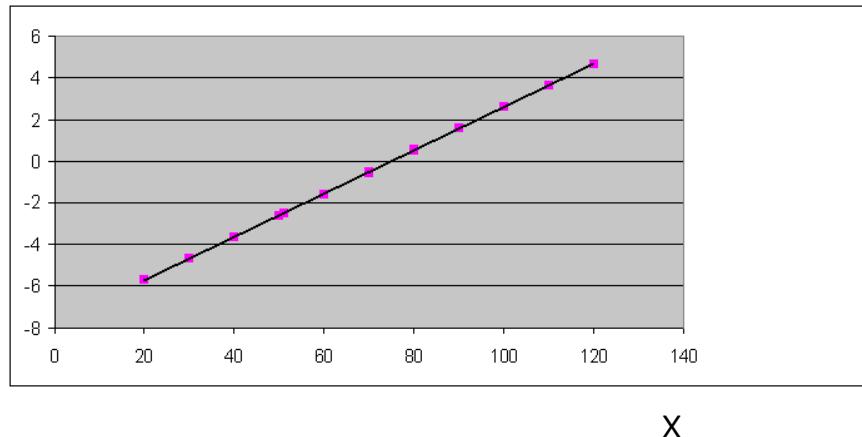
$$\ln \left[\frac{P(y|x)}{1 - P(y|x)} \right] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$



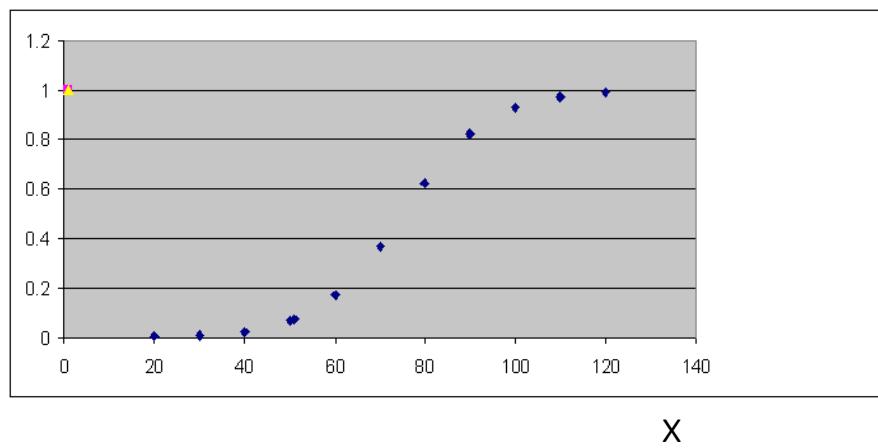
$$P(y|x) = \frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p}}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p}} = \frac{1}{1 + e^{-(\beta_0 + \beta^T X)}}$$

Binary Logistic Regression: Geometric

$\ln[p/(1-p)]$



$P(Y=1|x)$



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

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

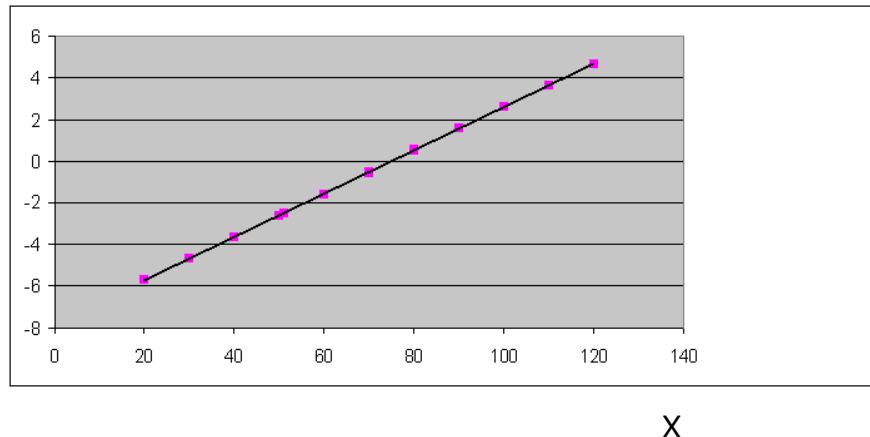
$$\ln\left(\frac{P}{1-P}\right) \xrightarrow{\text{Logit function}} \ln\left[\frac{P(y|x)}{1-P(y|x)}\right] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

Logit of $P(y|x)$

View II: Model Y as Bernoulli:

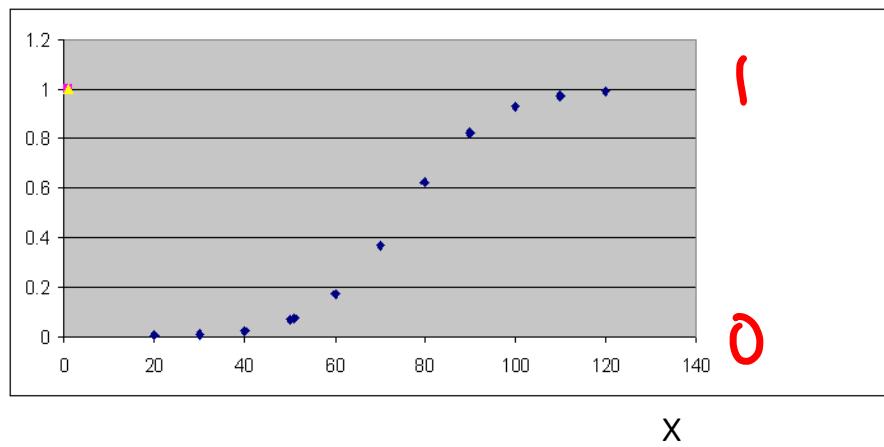
Bernoulli Distribution

$\ln[p/(1-p)]$



$P(Y=1|x)$

S shape



P_{Head}
↓

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

$$P_{\text{Head}} = P(Y=1|x) = \frac{e^{\alpha + \beta x}}{1 + e^{\alpha + \beta x}}$$



$$P(Y=1|x) \quad 1 - P(Y=1|x)$$

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 }

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

⇒ y is model with Bernoulli (p)

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

⇒ 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.

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

$P(Y=1|X)$

1.0

0.8

0.6

0.4

0.2

0.0

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

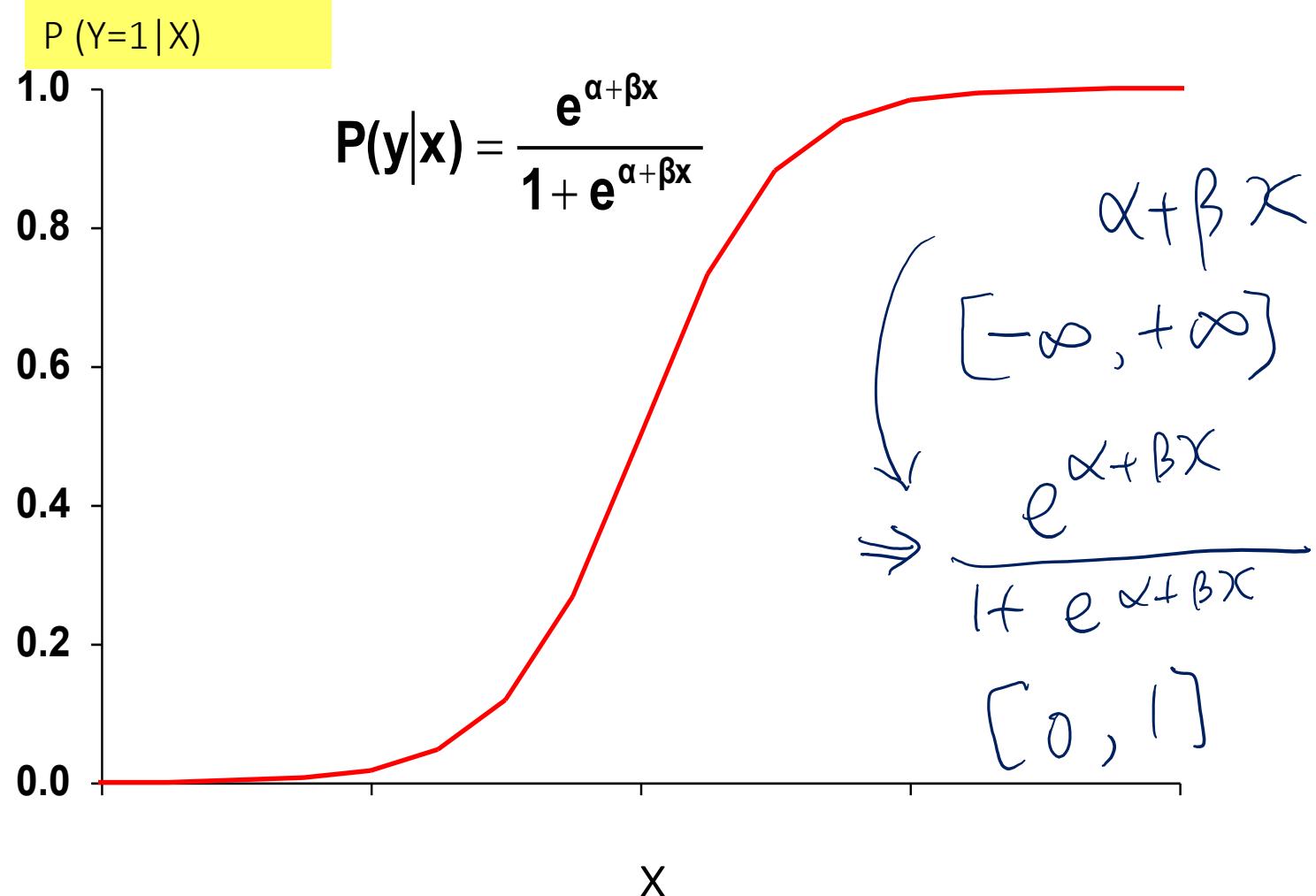
X

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



$P(y=1|x)$

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



View IV: Logistic Regression models a linear classification boundary!

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

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

Decision Boundary → equals to zero

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

View IV: Logistic Regression models a linear classification boundary!

$$y = \begin{cases} H, T \\ 1, 0 \end{cases}$$

$$\underset{y \in \{0, 1\}}{\operatorname{argmax}} \quad p(y|x)$$



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

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

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

Thank You



UVA CS 4774: Machine Learning

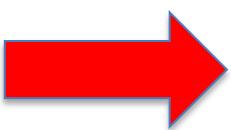
Lecture 11: Logistic Regression

Module II

Dr. Yanjun Qi

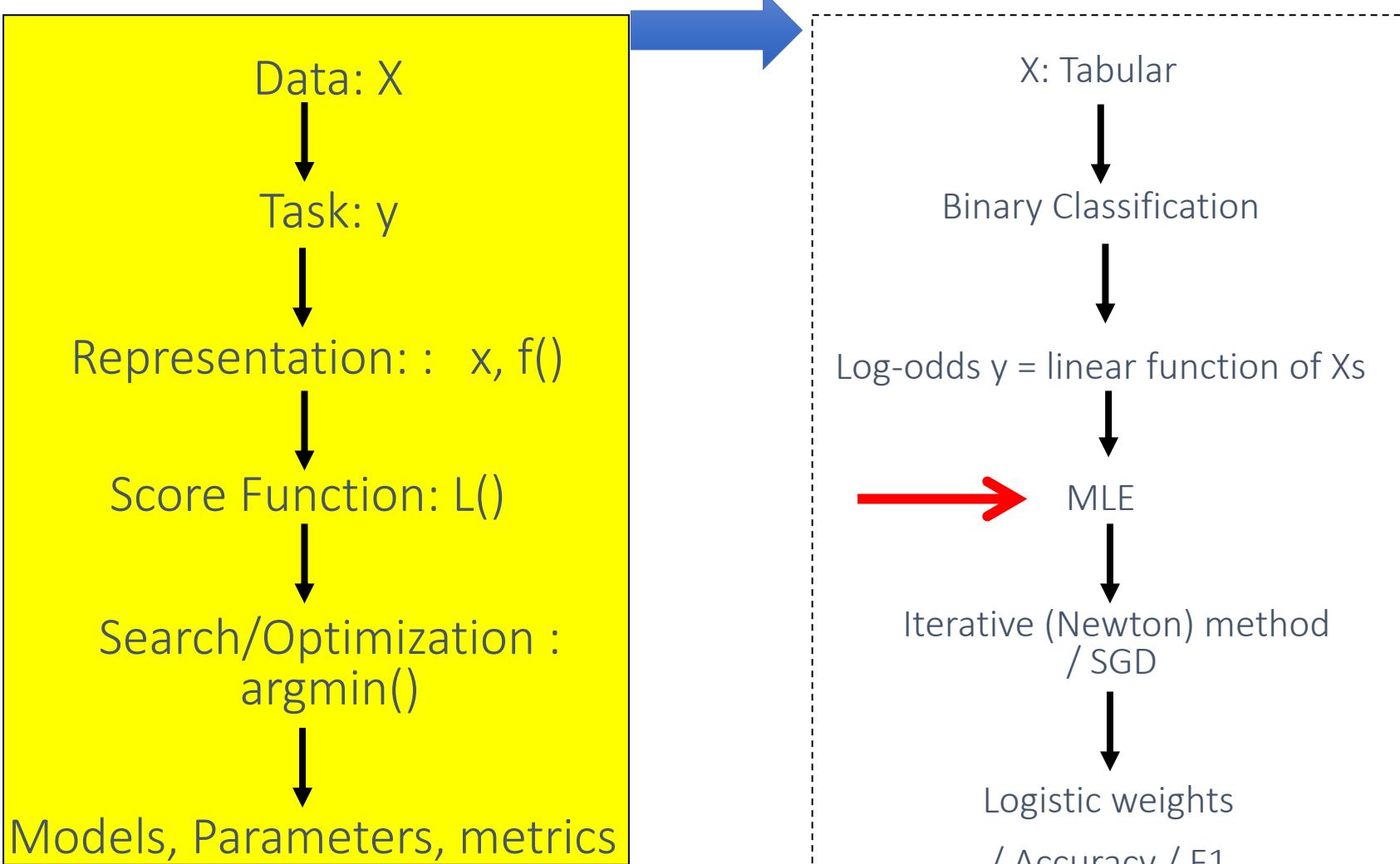
University of Virginia
Department of Computer Science

Roadmap

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

Today: Logistic Regression **Classifier**

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



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

Maximum Likelihood Estimation

A general Statement

$$\begin{aligned} Z &: \{H, T\} \rightarrow \underbrace{\{Z_1, Z_2, \dots, Z_n\}}_{P} \\ P(Z) &: \{P, 1-P\} \end{aligned}$$

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

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

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

this may be maximised with respect to θ 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, \dots, Z_n
- ✓ It is then possible to choose a set of parameters which are most likely to have produced the observed results.

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


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

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


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

Review: Defining Likelihood for basic Bernoulli

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

\downarrow
 $\{H, H, T, \dots, H\}_n$

\downarrow reformulate

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

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

$$p(z_i) = \begin{cases} p, & \text{if } z_i = 1 \\ 1-p, & \text{if } z_i = 0 \end{cases} \xrightarrow{\text{argmax}} \prod_{i=1}^n p^{z_i} (1-p)^{1-z_i}$$

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

Review: Defining Likelihood for **basic** Bernoulli

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

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

Review: Defining Likelihood for **basic** Bernoulli

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

i.e. negative log-likelihood

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

$$0 = -x + pn$$

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

Minimize the negative log-likelihood

→ MLE parameter estimation

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

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

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

i.e. Relative frequency of a binary event

The proportion of heads!

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 $p = \Pr(\text{head})$

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

$$\text{Logistic Regression } z_i = y_i|x_i$$

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

$$P(z_i|\beta)$$

Now we just rewrite

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

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

LIKELIHOOD:

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

↑
function of $p = \Pr(\text{head})$

Basile Bernoulli

Logistic / Bernoulli

$$L(\beta)$$

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

$$= \prod_{i=1}^n \hat{y}_i^{y_i} (1-\hat{y}_i)^{1-y_i}$$

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

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

Log likelihood

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

$$+ (1-y_i) \log(1-\hat{y}_i))$$

i.e. negative
cross entropy

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

When training set includes $(x_i, y_i), i=1, \dots, N$

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

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

$$= (p(y=1|x_i))^{y_i} (1 - p(y=1|x_i))^{1-y_i}$$

MLE for Logistic Regression Training

Training set: $(x_i, y_i), i=1, \dots, N$

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

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

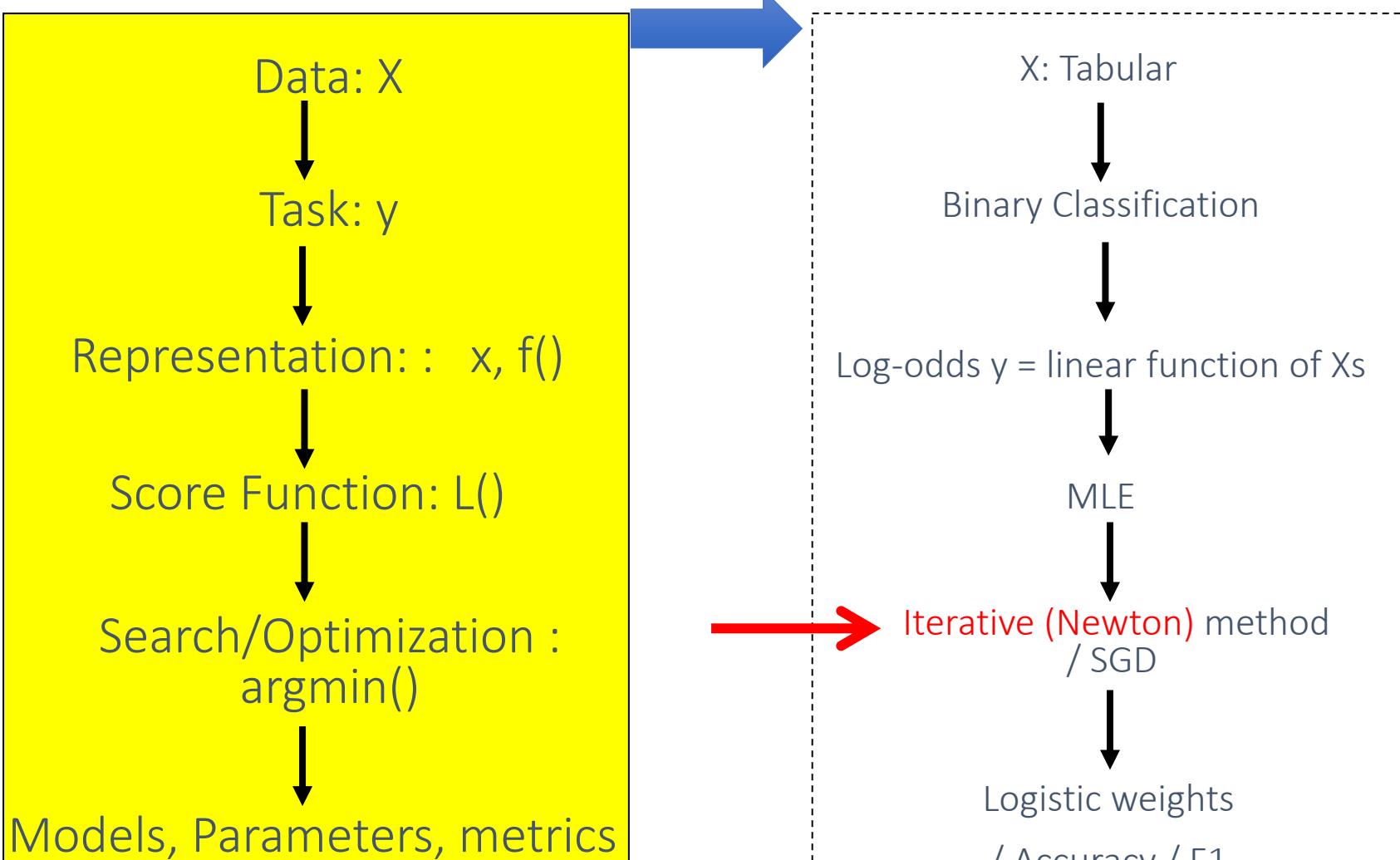
$$= \sum_{i=1}^N (y_i \log \frac{\exp(\beta^T x_i)}{1 + \exp(\beta^T x_i)}) + (1 - y_i) \log \frac{1}{1 + \exp(\beta^T x_i)}$$

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

i.e. negative
cross entropy

Today: Logistic Regression **Classifier**

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



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

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: $(x_i, y_i), i=1, \dots, N$

(conditional)
Log-likelihood:



For Bernoulli distribution

$$p(y|x)^y(1-p)^{1-y}$$

$$\begin{aligned} l(\beta) &= \sum_{i=1}^N \{\log \Pr(Y = y_i | X = x_i)\} \\ &= \sum_{i=1}^N y_i \log(\Pr(Y = 1 | X = x_i)) + (1 - y_i) \log(\Pr(Y = 0 | X = x_i)) \\ &= \sum_{i=1}^N \left(y_i \log \frac{\exp(\beta^T x_i)}{1 + \exp(\beta^T x_i)} \right) + (1 - y_i) \log \frac{1}{1 + \exp(\beta^T x_i)} \\ &= \sum_{i=1}^N (y_i \beta^T x_i - \log(1 + \exp(\beta^T x_i))) \end{aligned}$$

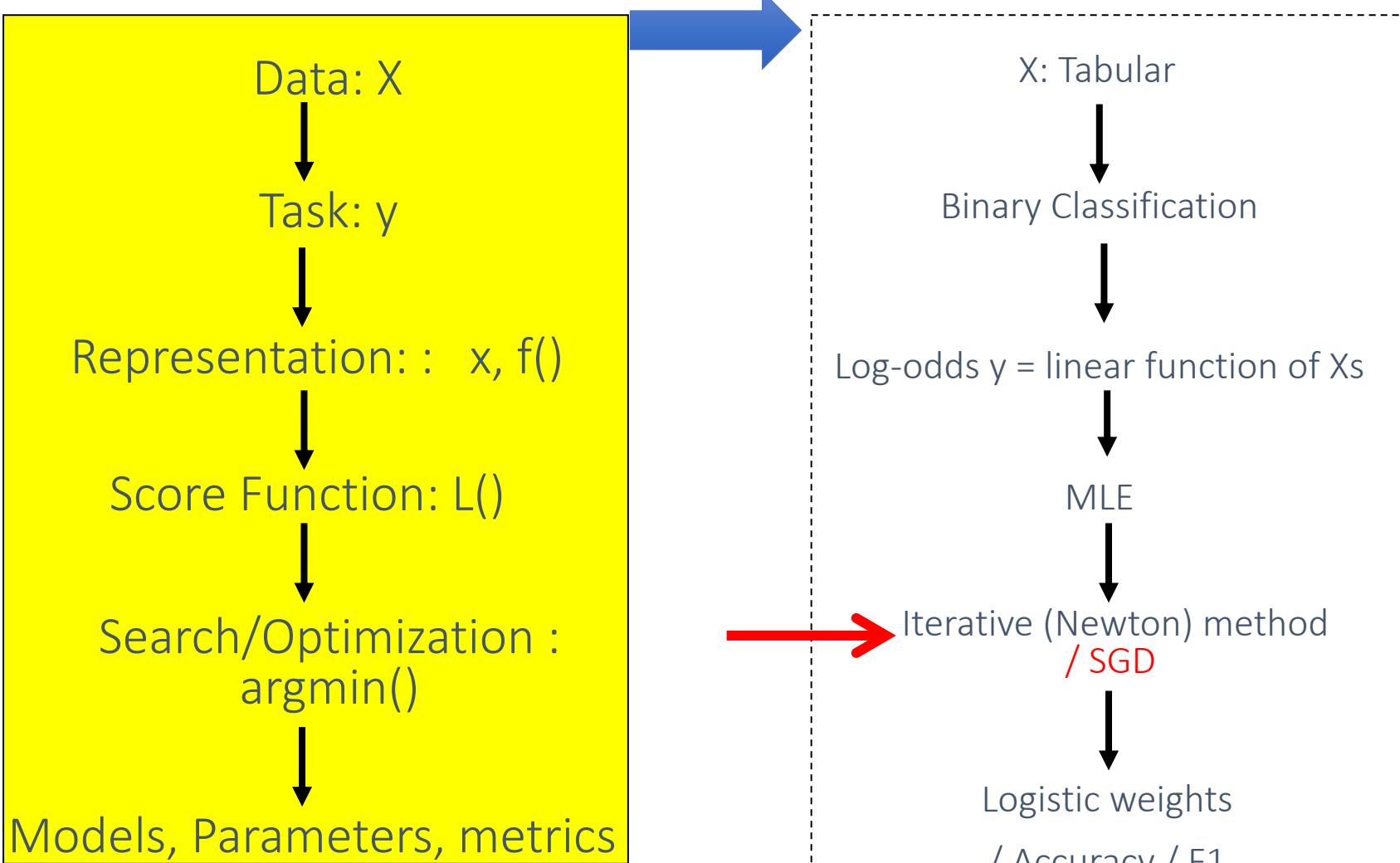
x_i are $(p+1)$ -dimensional input vector with leading entry 1
 β is a $(p+1)$ -dimensional vector

We want to **maximize** the log-likelihood in order to estimate β

See Extra Slides How to used Newton-Raphson optimization

Today: Logistic Regression **Classifier**

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



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

ReWrite Logistic Regression as two stages:

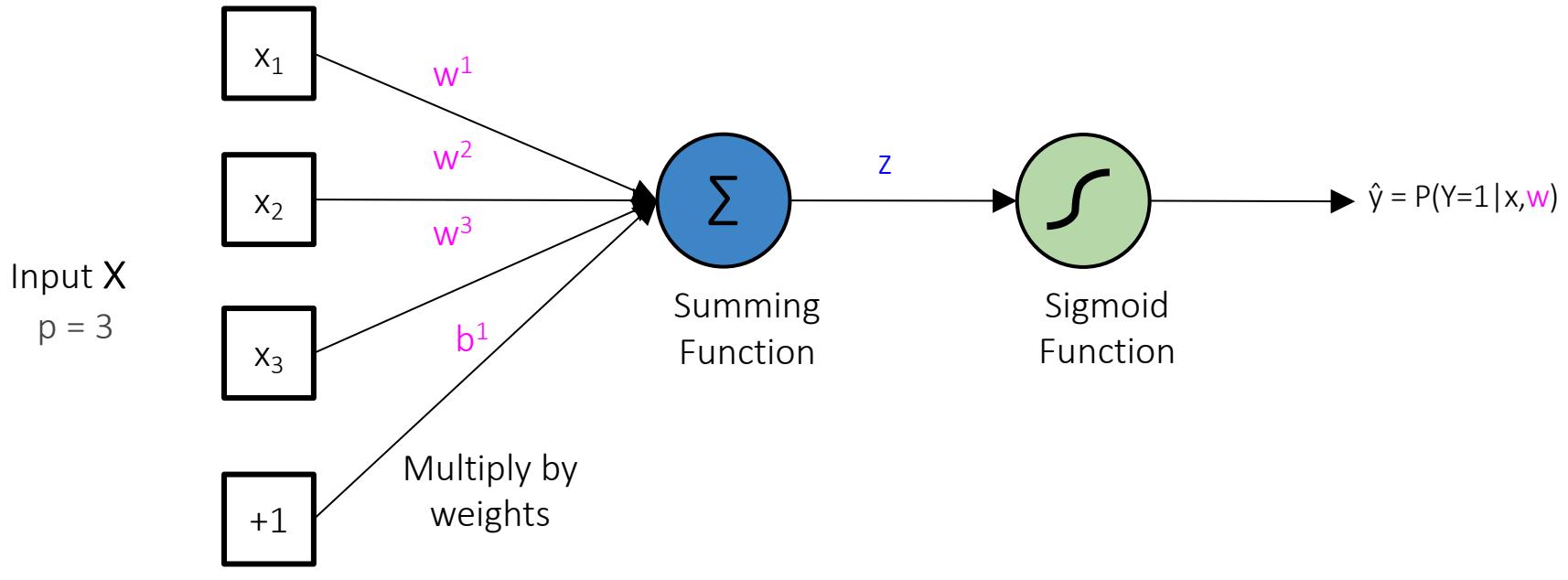
First:

Summing $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$

Second:

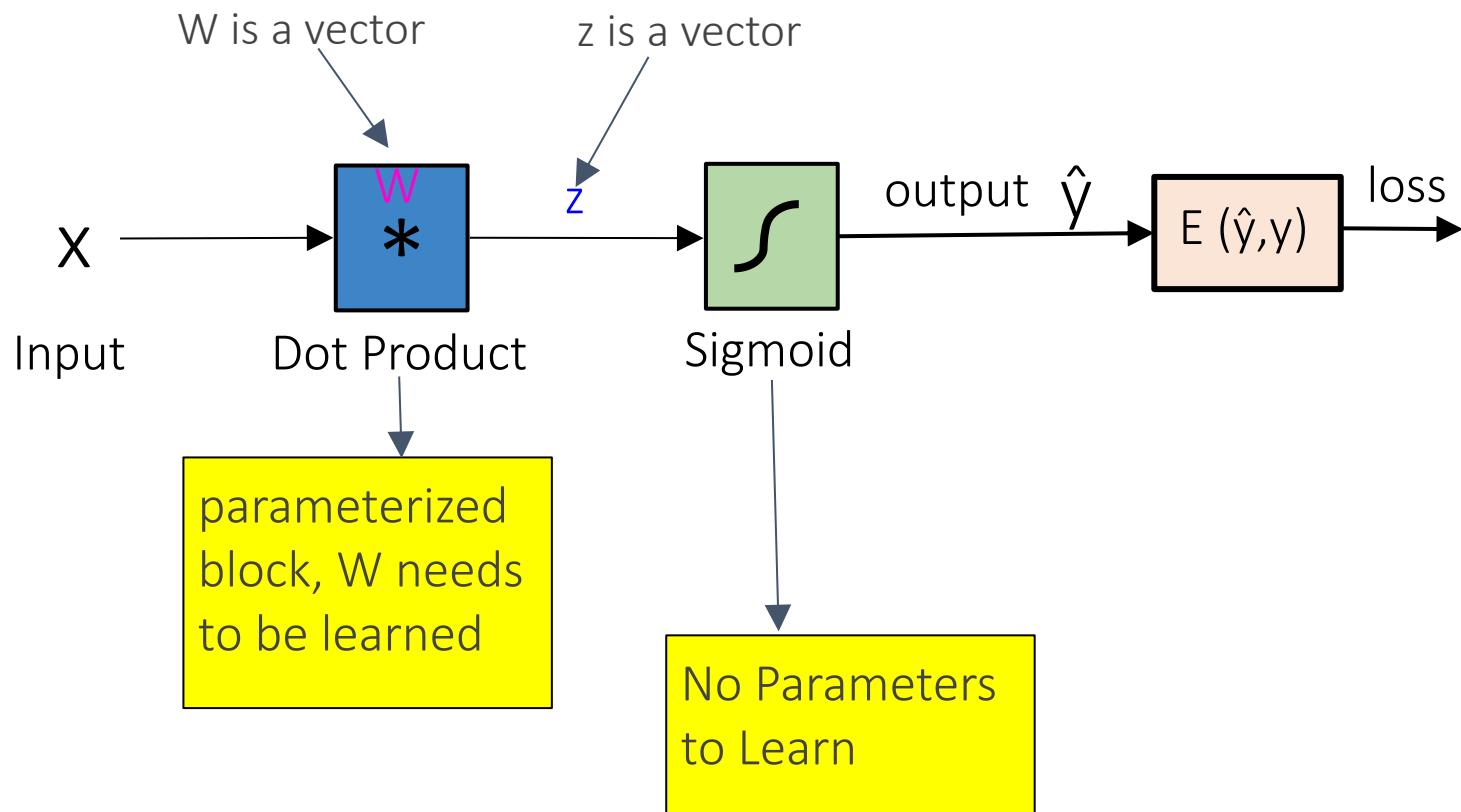
Sigmoid $\hat{y} = P(y=1|x) = \frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p}}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p}} = \frac{e^z}{1 + e^z}$

One “Neuron”: Block View of Logistic Regression



$$Z = \mathbf{W}^T \cdot \mathbf{X} + b$$
$$y = \text{sigmoid}(Z) =$$
$$\frac{e^z}{1 + e^z}$$

Next: “Block View” of Logistic Regression



Next: Stochastic GD →

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

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

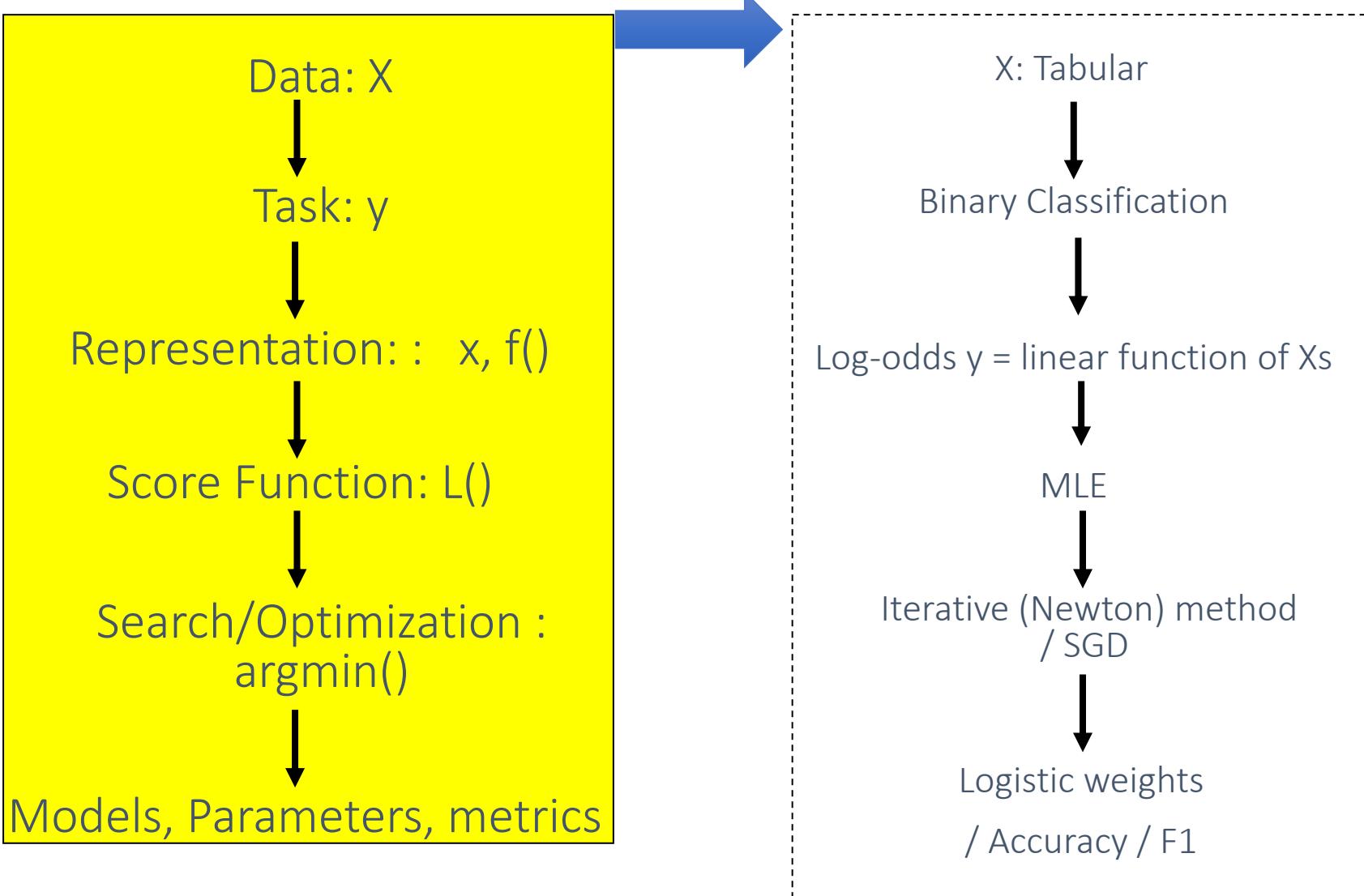
- → For neural network, we have the delta rule

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

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

Today: Logistic Regression **Classifier**

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



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

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

↓
 $\hat{\beta}$

- Parameter Estimation for LR

Bayes classifiers

- Treat each feature attribute and the class label as random variables.
- Given a sample x with attributes (x_1, x_2, \dots, x_p) :
 - Goal is to predict its class C .
 - Specifically, we want to **find the value of C_i that maximizes $p(C_i | x_1, x_2, \dots, x_p)$** .
- Can we estimate $p(C_i | x) = p(C_i | x_1, x_2, \dots, x_p)$ 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, \dots, x_p) :
 - Goal is to predict its class C .
 - Specifically, we want to **find the value of C_i that maximizes $p(C_i | x_1, x_2, \dots, x_p)$** .
- Can we estimate $p(C_i | x) = p(C_i | x_1, x_2, \dots, x_p)$ directly from data?

$$\rightarrow \{C_1, C_2, \dots, C_L\}$$

Bayes classifiers

→ MAP classification rule

- Establishing a probabilistic model for classification
- MAP classification rule
- MAP: Maximum A Posterior
 - Assign x to c^* if

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

for $c \neq c^*$, $c = c_1, \dots, c_L$

Bayes classifiers

→ MAP classification rule

- Establishing a probabilistic model for classification
- MAP classification rule
 - MAP: Maximum A Posterior
 - Assign x to c^* if

$$P(C = c^* | \mathbf{X} = \mathbf{x}) > P(C = c | \mathbf{X} = \mathbf{x}) \quad c \neq c^*, c = c_1, \dots, c_L$$
$$\left\{ \begin{array}{l} P(C = c_1 | \mathbf{x}) \\ P(C = c_2 | \mathbf{x}) \\ P(C = c_3 | \mathbf{x}) \end{array} \right\} \max \Rightarrow c_i$$

Bayes Classifiers – MAP Rule

Task: Classify a new instance X based on a tuple of attribute values

into one of the classes $X = \langle X_1, X_2, \dots, X_p \rangle$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} P(c_j | x_1, x_2, \dots, x_p)$$

WHY ?

MAP = Maximum Aposteriori Probability

Recap: Statistical Decision Theory (Extra)

- Random input vector: X
- Random output variable: Y
- Joint distribution: $\Pr(X, Y) \Rightarrow D = \boxed{(\bar{x}_1, \bar{y}_1), \dots, (\bar{x}_n, \bar{y}_n)}$
- Loss function $L(Y, f(X))$
- Expected prediction error (EPE):

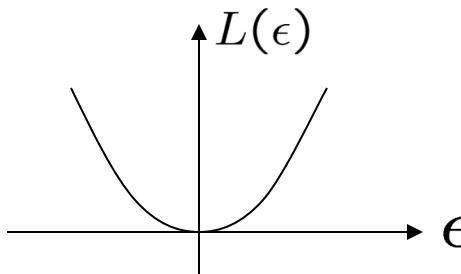
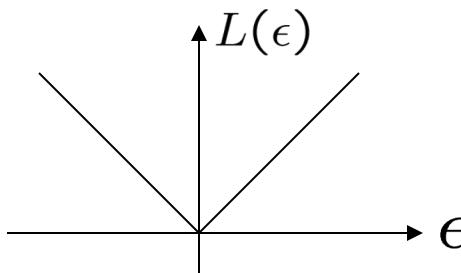
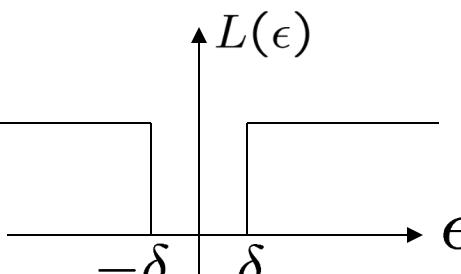
$$\text{EPE}(f) = E(L(Y, f(X))) = \int L(y, f(x)) \Pr(dx, dy)$$

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

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  $L(\epsilon) = \epsilon^2$	$\hat{f}(x) = E[Y X = x]$
L_1  $L(\epsilon) = \epsilon $	$\hat{f}(x) = \text{median}(Y X = x)$
$0-1$  $L(\epsilon) = \begin{cases} 0 & \text{if } -\delta \leq \epsilon \leq \delta \\ 1 & \text{otherwise} \end{cases}$	$\hat{f}(x) = \arg \max_Y P(Y X = x)$ (Bayes classifier / MAP)

0-1 LOSS for Classification

if $k = l$, $L(k, l) = 0$

- Procedure for categorical output variable C

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

- 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

$$L \begin{bmatrix} 0 & 1 & & & \\ 1 & 0 & 1 & & \\ & 1 & 0 & 1 & \\ & & 1 & 0 & 1 \\ & & & 1 & 0 \end{bmatrix}$$

C_1, C_2, \dots, 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

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

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

$$\begin{aligned} & E_x(X) \\ & E_x(g(x)) \end{aligned}$$

Consider sample population distribution

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

$$= E_{\mathcal{X}} E_{C|\mathcal{X}} [L(C, f(\mathcal{X})) | \mathcal{X}]$$

Discrete RV's Expectation

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

$$\arg \min_f EPE(f(\mathcal{X}))$$

\Rightarrow pointwise minimization when $\mathcal{X}=x$

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

0	0	1
1	0	0
0	0	0



$$\Rightarrow \hat{f}(x) = \arg \max_{C_k \in C} \Pr(C_k | \mathcal{X}=x)$$



$$\begin{cases} p(C_1 | x) \\ p(C_2 | x) \\ \vdots \\ p(C_L | x) \end{cases}$$

Expected prediction error (EPE)

$$\text{EPE}(f) = E_{X,C}(L(C, f(X))) = E_X \sum_{k=1}^K L(C_k, f(X)) \Pr(C_k | X)$$

Consider sample population distribution

- Pointwise minimization suffices

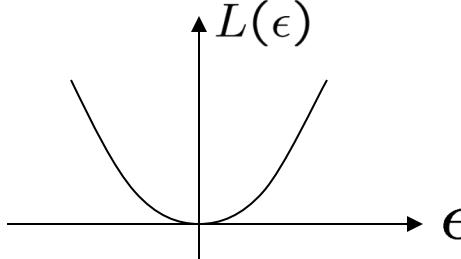
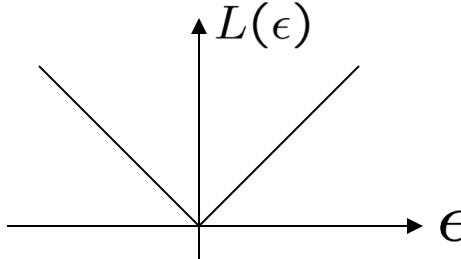
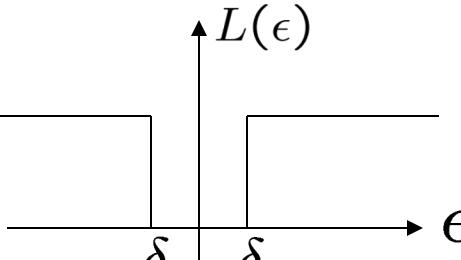
- → simply

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

Bayes Classifier

$$\begin{aligned}\hat{f}(X) &= C_k \text{ if} \\ \Pr(C_k | X = x) &= \max_{g \in C} \Pr(g | X = x)\end{aligned}$$

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

Loss Function	Estimator $\hat{f}(x)$
L_2  $L(\epsilon)$ ϵ	$EPE = E_{X,Y} (Y - \hat{f}(x))^2$ $\hat{f}(x) = E[Y X = x]$
L_1  $L(\epsilon)$ ϵ	$\hat{f}(x) = \text{median}(Y X = x)$
$0-1$  $L(\epsilon)$ $-\delta$ δ ϵ	$\hat{f}(x) = \arg \max_Y P(Y X = x)$ (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.



$$\underbrace{P(y=1 | x)}_{\text{func of } x} \quad 1 - p(y=1|x)$$

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

Newton's method for optimization

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

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

$$\text{Newton: } \theta_{k+1} = \theta_k - \mathbf{H}_K^{-1} \mathbf{g}_k$$

$$\underbrace{\begin{matrix} p \times p & p \times 1 \\ \hline p \times 1 \end{matrix}}$$

Review: Hessian Matrix / n==2 case

Singlevariate

→ multivariate

$$f(x, y)$$

- 1st derivative to gradient,
- 2nd derivative to Hessian

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

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

Review: Hessian Matrix

Suppose that $f : \mathbb{R}^n \rightarrow \mathbb{R}$ 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,

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

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 !)

$$\hat{f}(\theta) = f(\theta_K) + g_K^\top (\theta - \theta_K) +$$

$$\frac{1}{2} (\theta - \theta_K)^\top H_K (\theta - \theta_K)$$
$$\frac{1}{2} (\theta^\top H_K \theta - 2\theta^\top H_K \theta_K + \theta_K^\top H_K \theta_K)$$

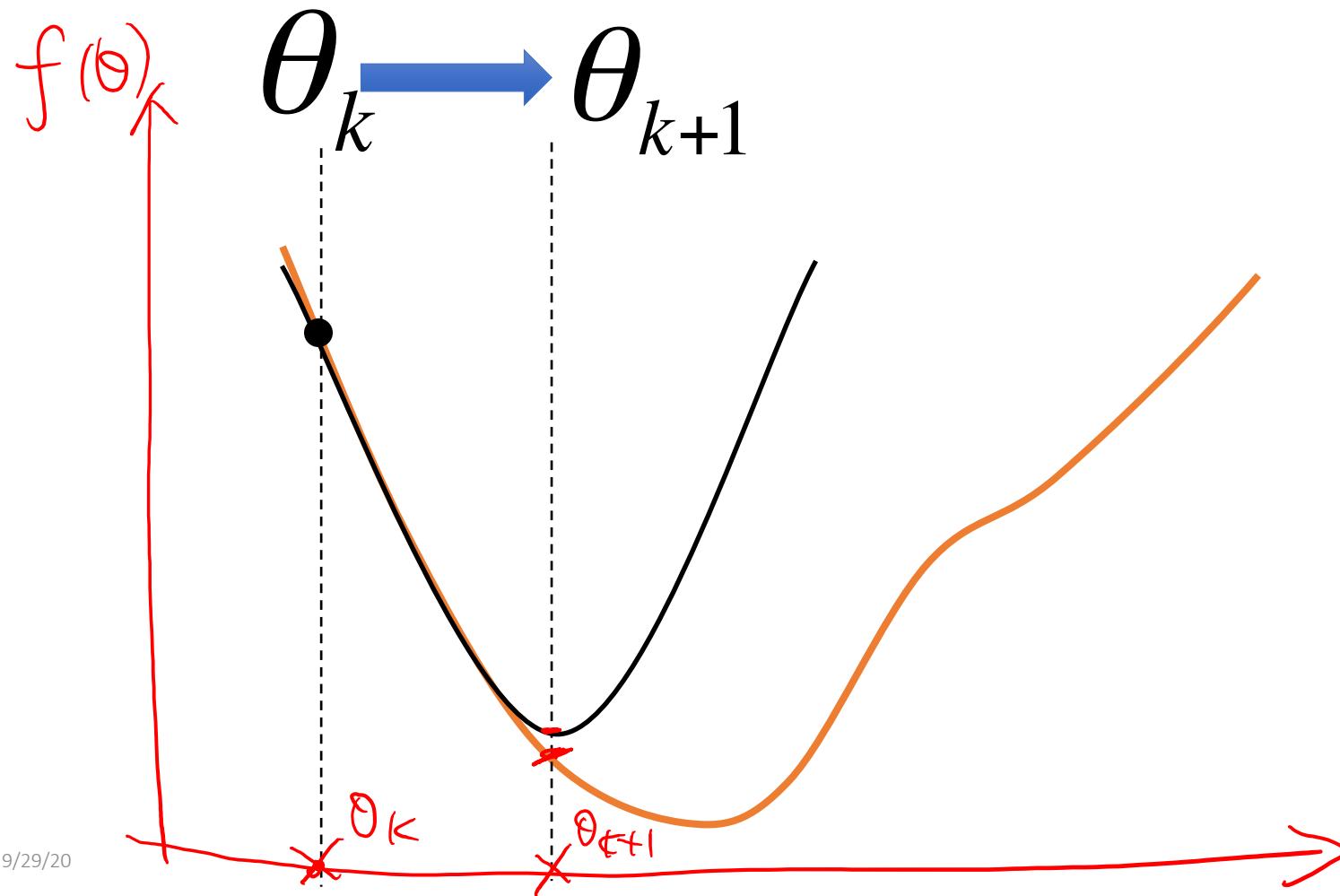
$$\frac{\partial \hat{f}(\theta)}{\partial \theta} = 0 + g_K + \underbrace{\frac{2}{2} H_K \theta - \frac{2}{2} H_K \theta_K}_{\text{See P24 handout}} := 0$$

$$g_K + H_K (\theta - \theta_K) = 0$$

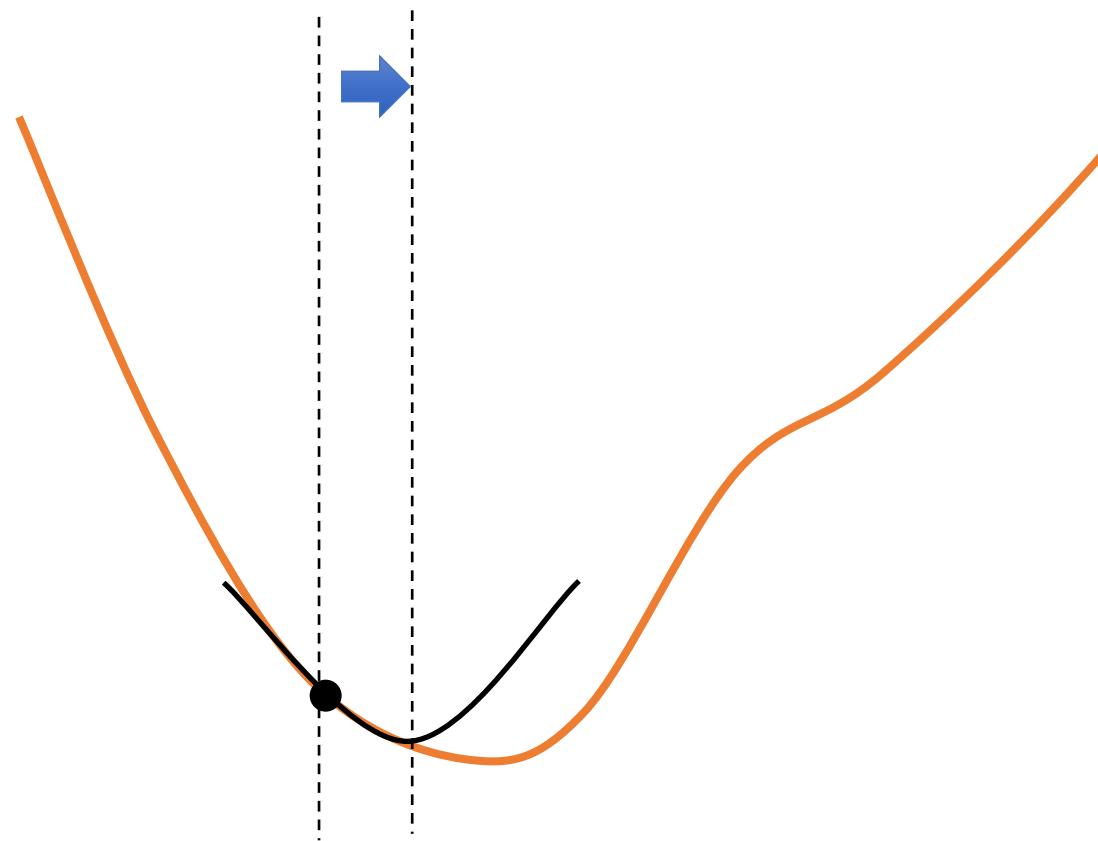
$$\Rightarrow \theta = \theta_K - H_K^{-1} g_K$$

where $\begin{cases} H_K \in \mathbb{R}^{P \times P} \\ g_K \in \mathbb{R}^P \end{cases}$

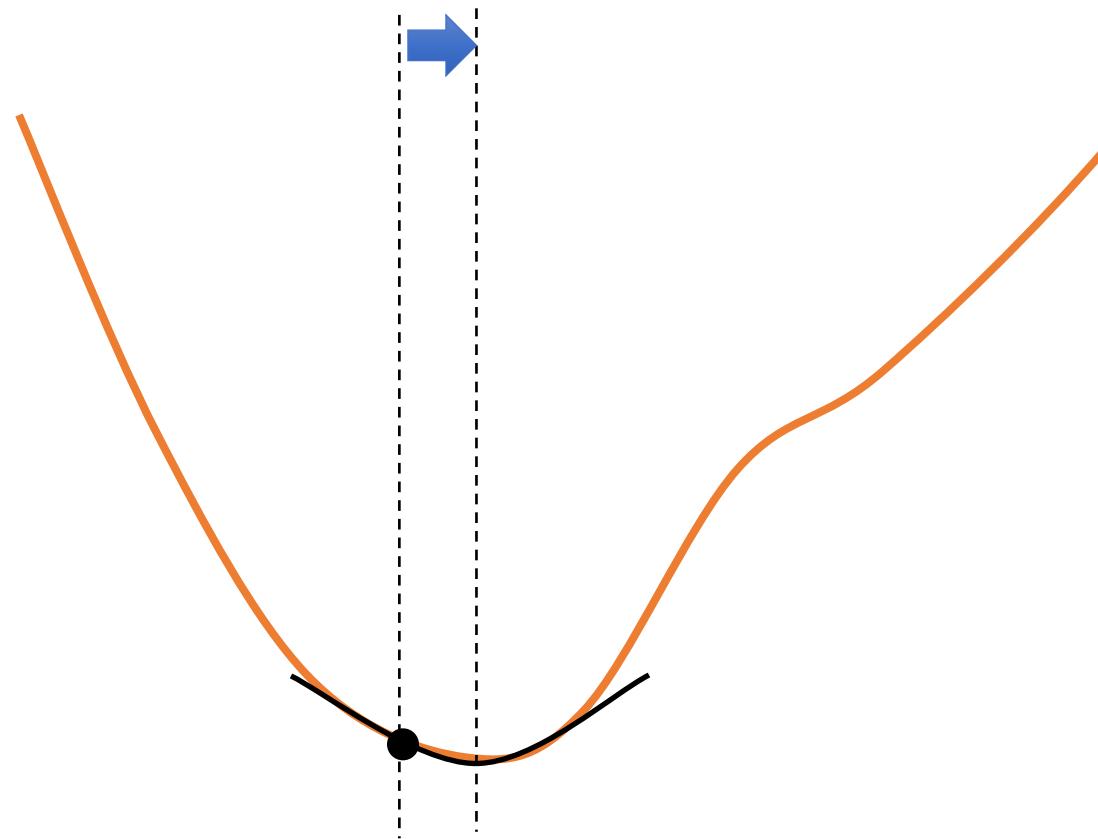
Newton's Method / second-order Taylor series approximation



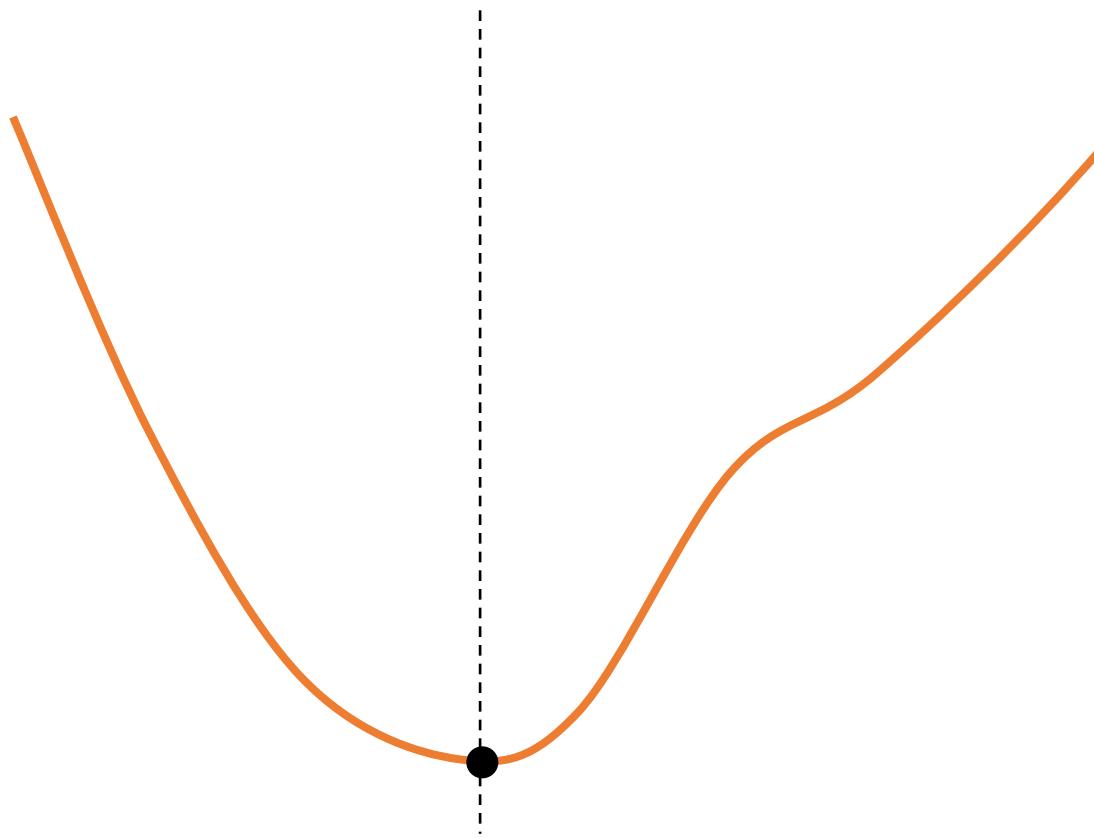
Newton's Method / second-order Taylor series approximation



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

$$\Theta_{k+1} = \Theta_k - \frac{1}{H(\Theta_k)} g(\Theta_k)$$

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

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

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

↓

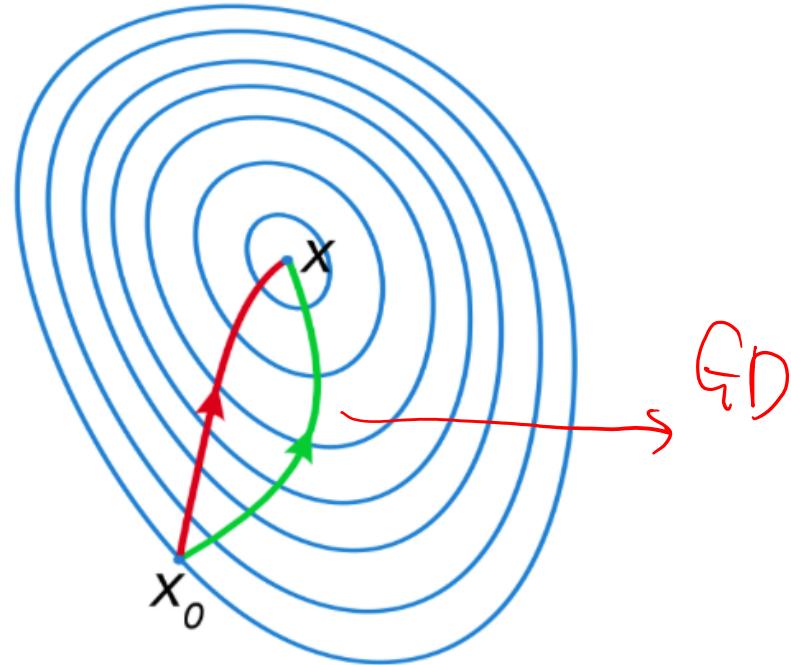
$$\Theta_{k+1} = \Theta_k - \alpha g(\Theta_k)$$

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: $(x_i, y_i), i=1, \dots, N$

For Bernoulli distribution

$$p(y|x)^y(1-p)^{1-y}$$

(conditional)

Log-likelihood:

How?

$$\begin{aligned} l(\beta) &= \sum_{i=1}^N \{\log \Pr(Y = y_i | X = x_i)\} \\ &= \sum_{i=1}^N y_i \log(\Pr(Y = 1 | X = x_i)) + (1 - y_i) \log(\Pr(Y = 0 | X = x_i)) \\ &= \sum_{i=1}^N \left(y_i \log \frac{\exp(\beta^T x_i)}{1 + \exp(\beta^T x_i)} \right) + (1 - y_i) \log \frac{1}{1 + \exp(\beta^T x_i)} \\ &= \sum_{i=1}^N (y_i \beta^T x_i - \log(1 + \exp(\beta^T x_i))) \end{aligned}$$

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

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

y_i

$$P(y_i=1|x)$$

$$\log \Pr(Y = y_i | X = x_i) = P(y_i|x_i)$$

$$= \log \left\{ P(y_i=1|x)^{y_i} (1 - P(y_i=1|x))^{\bar{y}_i} \right\}$$

$$\begin{cases} y_i = 1 \\ y_i = 0 \end{cases}$$

$$1 - P(y_i=1|x)$$

$$= y_i \log P(y_i=1|x) + (\bar{y}_i) \log (1 - P(y_i=1|x))$$

Newton-Raphson for LR (optional)

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

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

Vector

β

Solve by Newton-Raphson method:

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

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.

$p(x_i ; \beta)$

$1 - p(x_i ; \beta)$

$$\theta_{k+1} = \theta_k - H_K^{-1} g_k$$

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 $p(x_i; \beta^{old})(1 - p(x_i; \beta^{old}))$

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

Newton-Raphson for LR...

↑

- Newton-Raphson

$$\begin{aligned}\beta^{new} &= \beta^{old} + (X^T W X)^{-1} X^T (y - p) \\ &= (X^T W X)^{-1} X^T W (X \beta^{old} + W^{-1} (y - p)) \\ &= (X^T W X)^{-1} X^T W z\end{aligned}$$

Re expressing
Newton step as
weighted least
square step

- Adjusted response

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

$$(X^T X)^{-1} X^T Y$$

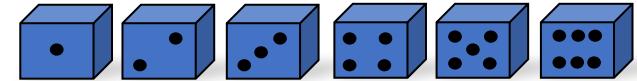
\underbrace{W}_{WZ}

- 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



$$P(G|x)$$

Directly models the posterior probabilities as the output of regression

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

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

x is p-dimensional input vector

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

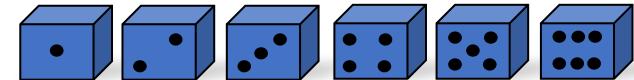
Total number of parameters is $(K-1)(p+1)$

β_{k0}, β_k , $k=1, 2, \dots, K-1$

Note that the class boundaries are linear

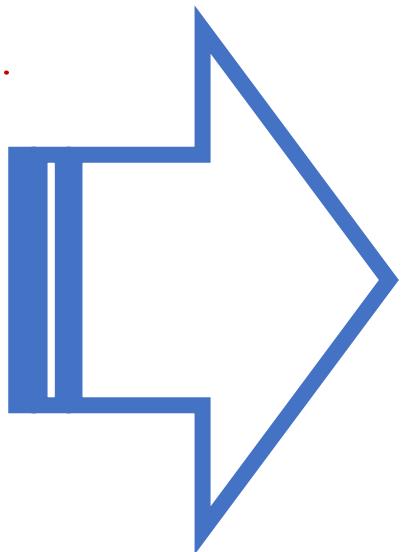
Binary → Multinoulli Logistic Regression Model

(e.g. k=6)



$$p(y=1|x) \rightarrow \frac{e^{\beta_1 x}}{1 + e^{\beta_1 x}}$$

$$p(y=0|x) \rightarrow \frac{1}{1 + e^{\beta_1 x}}$$



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

e.g.

$$\ln \frac{P(G=k|x)}{P(G=1|x)} = 0 \Rightarrow \text{linear}$$

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

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.