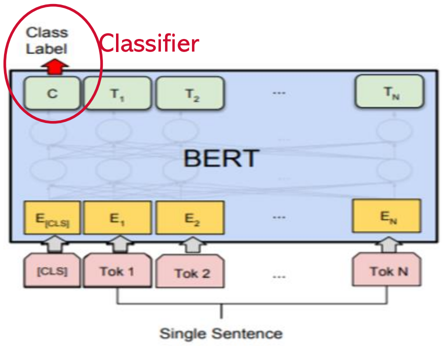
CLASSIFICATION

Class 1. Logistic Regression Models

Outline

- Introduction
- Binary LR
- Softmax LR

BERT Fine Tuning for Classification



(b) Single Sentence Classification Tasks: SST-2, CoLA

Application Examples

- **Email spam detection** (binary classification)
  - Given message $\boldsymbol{x} \in \mathbb{R}^d$, predict $y \in \{yes, no\}$.
- **Hand-written digit recognition** (multi-class classification)
  - Given image $\boldsymbol{x} \in \mathbb{R}^d$, predict $y \in \{0,1, \ldots, 9\}$;
  - Choosing one and only one output label.
- **News article topic tagging** (multi-label classification)
  - Given article $\boldsymbol{x} \in \mathbb{R}^d$, predict $\boldsymbol{y} \subseteq \{0,1\}^M$ where M is the number of category labels and $\boldsymbol{y}$ specifies the relevant subset given the input (e.g., China, Taiwan, economy, politics).

Terminology

X	*Y*
Input Variables	Output Variables
Independent Variables	Dependent Variables
Predictors	Responses
Features	Categories or labels
Factors	Outcomes

Classification via Label Scoring

- We usually learn scoring function $f_\theta(\boldsymbol{x}) \in \mathbb{R}^M$ via model training, where θ is the set of learnable parameters, and each element $f_\theta^{(j)}(\boldsymbol{x})$ is the predicted score with respect to label $j \in \{1, \dots, M\}$.
- We can obtain a *yes/no* decision on each category by assigning *yes* if $f_\theta^{(j)}(\boldsymbol{x}) \geq 0.5$ and *no* otherwise.
- Or we can use sort the labels based on their scores and assign *yes* to the top one and *no* to the rest as

$$\hat{y}\ (\boldsymbol{x}) = \mathrm{argmax_j}\{f_\theta^{(j)}(\boldsymbol{x})\}$$

Scoring Functions

- **Linear Function**

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = w_0 + w_1\, x_1 + \cdots + \, w_d\, x_d = \boldsymbol{w}^T \boldsymbol{x}$$

where $\boldsymbol{x} = (1,\ x_1, \cdots, x_d)$ is a data point;

$\boldsymbol{w} = (w_0, w_1, \ldots, w_d)$ are the model parameters.

- **Sigmoid Logistic Regression** (in binary LR)

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \frac{1}{1+\mathrm{e}^{-\boldsymbol{w}^T\boldsymbol{x}}} \equiv \hat{P}(\,Y = 1|\boldsymbol{x})$$

Scoring Functions (cont'd)

- **SoftMax LR:** for $j \in \{1,2, \ldots, M\}$ and $W = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M)$

$$f_j(\boldsymbol{x}; W) \equiv \hat{P}(\,Y = j\,|\boldsymbol{x}; W) = \frac{\exp\left(\boldsymbol{W}_j^T\boldsymbol{x}\right)}{\sum_{m=1}^{M} \exp(\boldsymbol{W}_m^T\boldsymbol{x})}$$

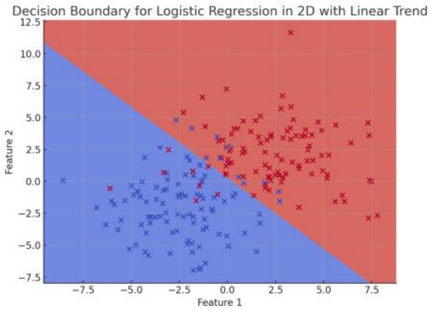
- **k-Nearest Neighbors (kNN)** (Non-parametric)

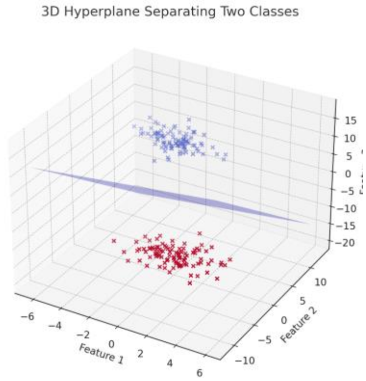
$$f_j(\boldsymbol{x}|D) = \frac{\sum_{x_i \in kNN(x)} \delta(y_i, j)}{k}\ , \quad \delta(y_i, j) = \begin{cases} 1 & \text{if } y_i = j \\ 0 & \text{otherwise} \end{cases}$$

$D = \{(x_i, y_i)\}_{\mathrm{i=1,\ldots N}}$ is a labeled training set;

$kNN(\boldsymbol{x})$ is the set of k-nearest-neighbors of $\boldsymbol{x}$ in D.

Decision Boundaries of Binary LR
(Generated by GPT4)

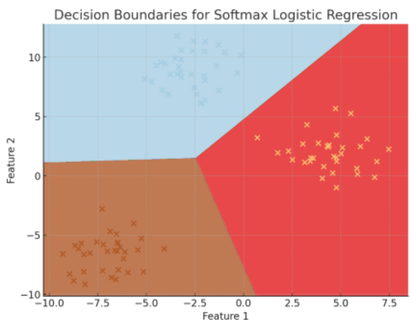


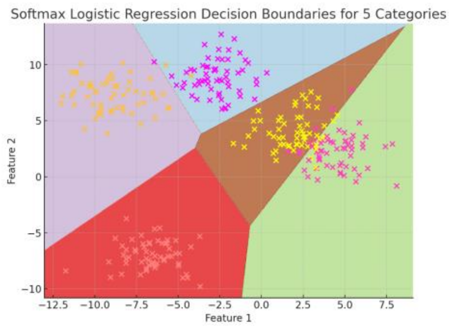


Binary LR has a linear decision boundary (e.g., in 2D and 3D).

9

Decision Boundaries of Softmax LR
(Generated by GPT4)





The decision boundaries are piecewise linear but globally non-linear.

10

Linear Decision Boundary Formally

- In 2D, a line can be written as $ax + by + \mathrm{c} = 0$.
- In 3D, a plane can be written as $ax + by + \mathrm{cz} + \mathrm{d} = 0$.
- In d-dimensional vector space, a hyperplane (h) can be defined as

$$h = \{\boldsymbol{x}: \boldsymbol{w}^T\boldsymbol{x} = 0\}$$

where $\boldsymbol{x} = (1, \mathrm{x}_1, \mathrm{x}_2, \ldots, x_d)$ and $\boldsymbol{w} = (\mathrm{w}_0, \mathrm{w}_1, \ldots, w_d)$.

- If the decision boundary of a classifier can be written as $\boldsymbol{w}^T\boldsymbol{x} = 0$, we call it a linear classifier.
- We cannot tell by looking at the scoring function $f_\theta(\boldsymbol{x})$ along; instead, we must take the decision rule (thresholding strategy) into account.

Is binary LR a linear classifier?

- Scoring function given $\boldsymbol{x}$ is sigmoid (non-linear)

$$\sigma_{\boldsymbol{w}}(\boldsymbol{x}) = (1 + e^{-\boldsymbol{w}^T\boldsymbol{x}})^{-1} \qquad (1)$$

- A popular threshold is set as

$$\sigma_{\boldsymbol{w}}(\boldsymbol{x}) = 0.5 \qquad (2)$$

- The decision boundary is

$$h = \left\{x: \ (1 + e^{-\boldsymbol{w}^T\boldsymbol{x}})^{-1} = 0.5\right\} \qquad (3)$$

$$\Rightarrow \ 1 + e^{-\boldsymbol{w}^T\boldsymbol{x}} = 2 \ \Rightarrow \ \mathrm{e}^{-\mathrm{w}^\mathrm{T}\mathrm{x}} = 1 \ \Rightarrow \ \mathrm{w}^\mathrm{T}\mathrm{x} = 0$$

$$\Rightarrow \ h = \{x: w^T x = 0\} \qquad (4)$$

How about softmax LR?

- Scoring function is defined for $k = 1, 2, \dots, K$ as

$$\Pr(y = k|x) = \frac{\exp(\mathbf{w}_k{}^T x)}{\sum_{k'=1}^{K} \exp\left(w_{k'}^T x_i\right)} \equiv \hat{p}_k(\boldsymbol{x}) \qquad (5)$$

- Decision boundary between labels j and k

$$h_{jk} = \left\{\boldsymbol{x}: \hat{p}_j(\boldsymbol{x}) = \hat{p}_k(\boldsymbol{x})\right\} \qquad (6)$$

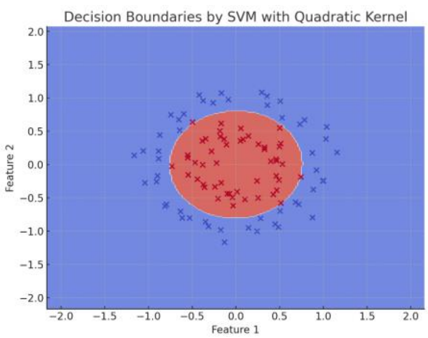
$$\Rightarrow \frac{\exp(\mathbf{w}_j{}^T x)}{\sum_{k'=1}^{K} \exp\left(w_{k'}^T x_i\right)} = \frac{\exp(\mathbf{w}_k{}^T x)}{\sum_{k'=1}^{K} \exp\left(w_{k'}^T x_i\right)} \Rightarrow \mathbf{w}_j{}^T x = \mathbf{w}_k{}^T x$$

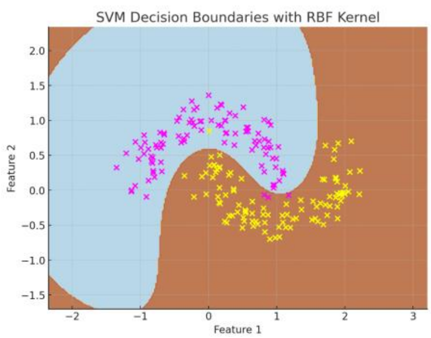
$$\Rightarrow \mathbf{w}_j{}^T x = \mathbf{w}_k{}^T x, \quad \mathbf{w}_j{}^T x = \mathbf{w}_k{}^T x, \quad (\mathbf{w}_j{}^T - \mathbf{w}_k{}^T) x = 0$$

$$\Rightarrow \quad h_{jk} = \{x: \underbrace{(\mathbf{w}_j - \mathbf{w}_k)}_{w}{}^T x = 0\} \qquad (7)$$

13

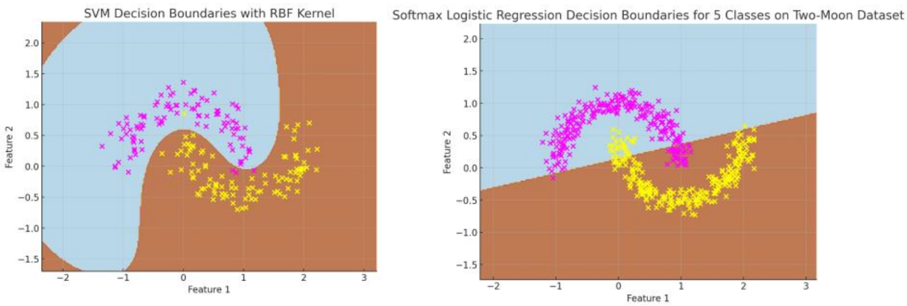
Other Examples: Support Vector Machines





14

SVM vs. Softmax (for 5 classes)



LDA (linear) vs. QDA (non-linear)

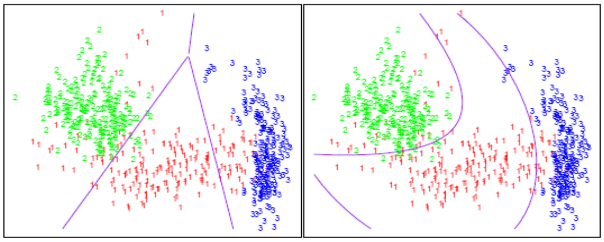


Figure 4.1: *The left plot shows some data from three classes, with linear decision boundaries found by linear discriminant analysis. The right plot shows quadratic decision boundaries. These were obtained by finding linear boundaries in the five-dimensional space* $X_1, X_2, X_{12}, X_1^2, X_2^2$. *Linear inequalities in this space are quadratic inequalities in the original space.*

(Hastie et al., ESL)

Elements of Statistical Learning ©Hastie, Tibshirani & Friedman 2001 Chapter 2

Elements of Statistical Learning ©Hastie, Tibshirani & Friedman 2001 Chapte

Decision Boundaries of kNN (non-linear)

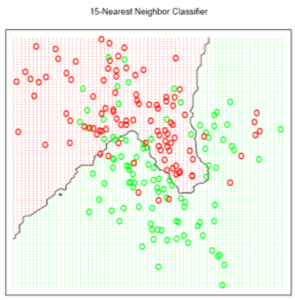


Figure 2.2: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (GREEN $= 0$, RED $= 1$) and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

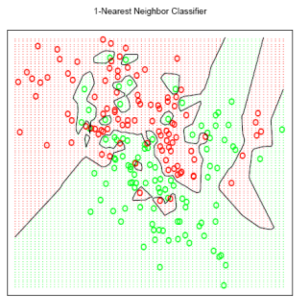


Figure 2.3: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (GREEN $= 0$, RED $= 1$), and then predicted by 1-nearest-neighbor classification.*

17

Outline

- ✓Introduction
- Binary LR
  - Optimization algorithms
  - Convexity
  - Regularization
- Softmax LR

18

LR for Binary Classification

- Using sigmoid function to estimate label probabilities

$$P_{\boldsymbol{w}}(y=1|\boldsymbol{x}) = \sigma_{\boldsymbol{w},\boldsymbol{x}} = \frac{1}{1+\exp(-w^T\boldsymbol{x})}$$

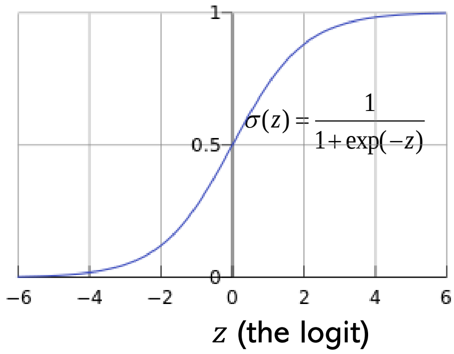
$$P_{\boldsymbol{w}}(y=0|\boldsymbol{x}) = 1-\sigma_{\boldsymbol{w},\boldsymbol{x}} = \frac{\exp(-\boldsymbol{w}^T\boldsymbol{x})}{1+\exp(-\boldsymbol{w}^T\boldsymbol{x})}$$

- Merged formula

$$P_{\boldsymbol{w}}(y|\boldsymbol{x}) = (\sigma_{\boldsymbol{z}})^{\mathrm{y}}(1-\sigma_{\boldsymbol{z}})^{(1-\mathrm{y})} \text{ with } z = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}$$

$$\log\, P_{\boldsymbol{w}}(y|\boldsymbol{x}) = \mathrm{y}\log\sigma_{\boldsymbol{z}} + (1-\mathrm{y})\log((1-\sigma_{\boldsymbol{z}})$$

Sigmoid Function



$$z \in (-\infty, \infty), \quad \sigma(z) \in (0,1), \quad \sigma(0) = 0.5, \quad \sigma(z) = (1-\sigma(-z))$$

Logit = logarithm of the odds

$p = \dfrac{1}{1 + \exp(-z)}$ — Start with the sigmoid

$p(1 + \exp(-z)) = 1$ — Multiply the denominator on both sides

$\exp(-z) = \frac{1-p}{p}$ — Arrange p to the RHS

$\exp(z) = \frac{p}{1-p}$ — Flip over

$z = \log\frac{p}{1-p}$ — Take the log on both sides

logit

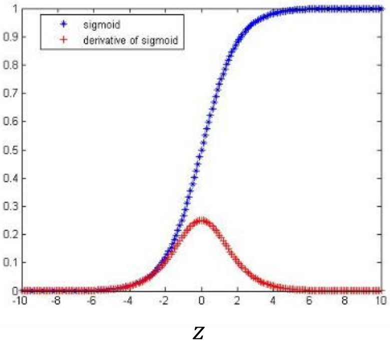
odds of p

02/06/2024 @Yiming Yang, S24 Lecture on LR Models 21

21

Derivative of Sigmoid

$$\frac{d\sigma(z)}{dz} = \frac{d}{dz}\left(\frac{1}{1+\exp(-z)}\right)$$
$$= (-1)(-1)\frac{\exp(-z)}{(1+\exp(-z))^2}$$
$$= \frac{1}{(1+\exp(-z))}\frac{\exp(-z)}{(1+\exp(-z))}$$
$$= \sigma(z)(1-\sigma(z))$$



02/06/2024 @Yiming Yang, S24 Lecture on LR Models 22

22

Training Binary LR

- Labeled Training Data

$$D = \{(x_i, y_i)\}_{i=1}^{n} \text{ with } x_i \in R^d \text{ and } y_i \in \{0,1\}$$

- Model Training

$$\hat{\boldsymbol{w}} = argmin_{\boldsymbol{w}}\{Loss(D; \boldsymbol{w}) + C\|\boldsymbol{w}\|^2\}$$

  - $Loss(D; \boldsymbol{w})$ is the training-set loss, measuring how well the model fits the data;
  - $\|\boldsymbol{w}\|^2$ is the regularization term (on model parameters only), controlling the model complexity to avoid overfitting.

Parameter Optimization

$$\hat{\boldsymbol{w}} = \text{argmax}_{\boldsymbol{w}}\{\ logP(D|\boldsymbol{w})\ \} \qquad \text{(the log-likelihood)}$$

$$= \text{arg}min_{\boldsymbol{w}}\{\ -logP(D|\boldsymbol{w})\} \qquad \text{(the negative log-likelihood)}$$

$$= \underset{\boldsymbol{w}}{\text{argmin}} \underbrace{-\ \textstyle\sum_{i=1}^{n} \{y_i \ln \sigma_i + (1-y_i)\ln(1-\sigma_i)\}}_{l(\boldsymbol{w})}$$

cross-entropy loss

Popular Algorithms

- Gradient Descent
  - Use the first-order derivative of $l(\boldsymbol{\beta})$
  - Need to pre-specify the "learning rate" (step size)
  - Fast to compute in each step but may take many steps
- Newton-Raphson
  - Use the first-order and second-order derivatives of $l(\boldsymbol{\beta})$
  - Automatically find the optimal step size for each iteration
  - Converge faster but may be too costly in each step

Gradient on a single training pair

$$l_i(\mathbf{w}) = y_i \ln \underbrace{\sigma(z_i)}_{\sigma_i} + (1-y_i)\ln(1-\sigma(z_i)) \qquad z_i = \mathbf{w}^{\mathbf{T}}\mathbf{x}_i = w_0 + \sum_{j=1}^{m} w_j x_{ij}$$

$$\frac{\partial}{\partial w_j} l_i(\mathbf{w}) = \boxed{\frac{dl_i}{d\sigma_i}}\boxed{\frac{d\sigma_i}{dz_i}}\boxed{\frac{\partial z_i}{\partial w_j}}$$

$$= \boxed{\left(y_i \frac{1}{\sigma_i} - (1-y_i)\frac{1}{1-\sigma_i}\right)}\boxed{\sigma_i(1-\sigma_i)}\boxed{x_{ij}} = (y_i - \sigma_i)\, x_{ij}$$

$$\nabla l_i(\mathbf{w}) \equiv \left(\frac{\partial}{\partial w_0} l_i(\mathbf{w}),\ \frac{\partial}{\partial w_1} l_i(\mathbf{w}),\ \cdots,\ \frac{\partial}{\partial w_m} l_i(\mathbf{w})\right)^T$$

Gradient ascent on a training set

The single-instance version: $D = \left\{ (\mathbf{x}^{(i)}, y^{(i)}) \right\}$

Loop until convergence {

for $i = 1$ to $|D|$ {

$\mathbf{w} := \mathbf{w} + \eta \nabla l_i(\mathbf{w})$ ($\eta > 0$ is prespecified or adapted via backtracking line search)

}

}

The batch version:

Loop until convergence {

$$\mathbf{w} := \mathbf{w} + \eta \sum_{i=1}^{|D|} \nabla l_i(\mathbf{w})$$

}

Guaranteed: $l\left(\mathbf{w}^{(0)}\right) \le l\left(\mathbf{w}^{(1)}\right) \le l\left(\mathbf{w}^{(2)}\right) \cdots$

Newton-Raphson Method
(in the case of one-dimensional w)

- Given current w, we want move it with the optimal step size (ε) in the right direction.
- Taylor series:

$$l(w+\varepsilon) = l(w) + \frac{l'(w)}{1!}\varepsilon + \frac{l''(w)}{2!}\varepsilon^2 + \cdots$$

- At the mode (with respect to ε)

$$0 = \frac{d}{d\varepsilon} l(w+\varepsilon) \approx l'(w) + l''(w)\varepsilon \quad \Rightarrow \quad \varepsilon = -\frac{l'(w)}{l''(w)}$$

- Update rule: $w := w - \dfrac{l'(w)}{l''(w)}$

Newton-Raphson Method
(in the case of multi-dimensional **w**)

- Taylor series:

$$l(\mathbf{w}+\boldsymbol{\varepsilon}) = l(\mathbf{w}) + \nabla l(\mathbf{w})\boldsymbol{\varepsilon} + \boldsymbol{\varepsilon}^T \frac{\nabla\nabla l(\mathbf{w})}{2!}\boldsymbol{\varepsilon} + \cdots$$

- Update rule:

$$\mathbf{w} := \mathbf{w} - \underbrace{\left(\nabla\nabla l(\mathbf{w})\right)}_{\mathrm{H}(\mathbf{w})}^{-1} \underbrace{\nabla l(\mathbf{w})}_{\text{the gradient}}$$

$$\nabla l(\mathbf{w}) = \left(\frac{\partial}{\partial w_0} l(\mathbf{w}), \frac{\partial}{\partial w_1} l(\mathbf{w}), \cdots, \frac{\partial}{\partial w_m} l(\mathbf{w})\right)^T$$

$$\nabla\nabla l(\mathbf{w}) \equiv \mathbf{H}(\mathbf{w}) = \left(H_{jj'}\right), \quad H_{jj'} = \frac{\partial^2}{\partial w_j \partial w_{j'}} l(\mathbf{w})$$

Newton-Raphson Method (cont'd)

First order derivative (as shown in slide #11):

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \sum_{i=1}^{n} (y_i - \sigma(\underbrace{\mathbf{w}^T\mathbf{x}_i}_{z_i}))x_{ij}$$

Second order derivative:

$$\frac{\partial^2}{\partial w_j \partial w_{j'}} l(\mathbf{w}) = \frac{\partial}{\partial w_{j'}}\left(\frac{\partial}{\partial w_j} l(\mathbf{w})\right) = \sum_{i=1}^{n} \frac{\partial}{\partial w_{j'}}(y_i - \sigma(\underbrace{\mathbf{w}^T\mathbf{x}_i}_{z_i}))x_{ij}$$

$$= -\sum_{i=1}^{n} \frac{d\sigma}{dz_i}\frac{\partial z_i}{\partial w_{j'}} x_{ij} = -\sum_{i=1}^{n} \sigma_i(1-\sigma_i) x_{ij'} x_{ij}$$

Newton-Raphson Method (cont'd)

The gradient (compact notion) :

$$\nabla l(\mathbf{w}) = \sum_{i=1}^{n} (y_i - \sigma_i)\ \mathbf{x}_i = \mathbf{X}^{\mathbf{T}}(\mathbf{y} - \boldsymbol{\sigma})$$

$\nabla l(\mathbf{w})$ is the weighted sum of the training documents;
$\mathbf{X}^T$ is (m+1) × n, whose columns ($\mathbf{x}_i$'s) are the training documents;
$\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$ is the vector of true labels of n training doc's;
$\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \cdots, \sigma_n)^T$ is the vector of predicted probilities $\sigma_i = \sigma(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i)$.

The Hessian (compact notion)

$$\mathbf{H}(\mathbf{w}) = \ -\sum_{i=1}^{n} \sigma_i (1 - \sigma_i)\,\mathbf{x}_i\ (\mathbf{x}_i)^T = -\mathbf{X}^{\mathbf{T}}\Lambda\mathbf{X}$$

$$\Lambda = diag\left(\sigma_1(1-\sigma_1),\ \ \sigma_2(1-\sigma_2), \cdots, \sigma_n(1-\sigma_n)\right)$$

31

Newton-Raphson Method (cont'd)

- Update rule in 1-dimentional LR

$$w := w^{\text{old}} - \frac{l'(w^{\text{old}})}{l''(w^{\text{old}})}$$

- Update rule in high-dimensional LR

$$\mathbf{w} := \mathbf{w}^{\text{old}} - H(\mathbf{w}^{\text{old}})^{-1} \nabla l(\mathbf{w}^{\text{old}})$$
$$:= \mathbf{w}^{\text{old}} + \left(\mathbf{X}^{\mathbf{T}}\boldsymbol{\Lambda}\mathbf{X}\right)^{-\mathbf{1}} \mathbf{X}^{\mathbf{T}}(\mathbf{y} - \boldsymbol{\sigma})$$

32

Outline

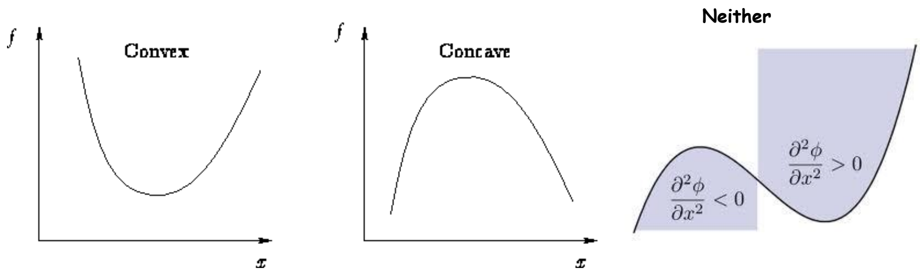
- ✓Introduction
- Binary LR
  - ✓Optimization algorithms
  - Convexity
  - Regularization
- Softmax LR

Globally optimal solution guaranteed?

- Check the convexity of the objective function
  - If it is convex, then there is a single global minimum.
  - If it is concave, then there is a single global maximum.
  - If it is neither, then the global optimal is not guaranteed.

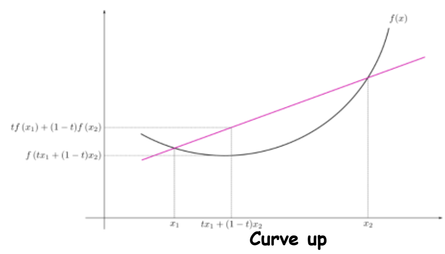
Convexity

Examples of 1-dimensional functions



Convex Function

https://en.wikipedia.org/wiki/Convex_function



- f is called **convex** if:

$$\forall x_1, x_2 \in X, \forall t \in [0,1]: \qquad f(tx_1 + (1-t)x_2) \le tf(x_1) + (1-t)f(x_2).$$

- f is called **strictly convex** if:

$$\forall x_1 \neq x_2 \in X, \forall t \in (0,1): \qquad f(tx_1 + (1-t)x_2) < tf(x_1) + (1-t)f(x_2).$$

Convexity of One-dimensional Function

- If $f(x)$ has a second derivative in $[a, b]$, then a necessary and sufficient condition for it to be convex on that interval is that the second derivative $f''(x) \geq 0$ for all $x \in [a, b]$.
- We call it strictly convex If $f''(x) > 0$ for all $x \in [a, b]$.
- We call it strongly convex If $f''(x) > m$ with a positive m for all $x \in [a, b]$.

02/06/2024 @Yiming Yang, S24 Lecture on LR Models 37

37

Examples (One-dimensional Case)

(Garret Thomas, https://gwthomas.github.io/docs/math4ml.pdf)

- Consider real $x \in \{-\infty, \infty\}$

1) $f(x) = 2x + 3$: $f' = 2$, $f'' = 0$ ← Convex but not strictly nor strongly
2) $f(x) = x^2 + 2x + 1$: $f' = 2x$, $f'' = 2$ ← Strongly (and strictly)
3) $f(x) = e^x$: $f' = f'' = e^x$ ← Convex? Strictly? Strongly?
4) $f(x) = x^4$ ← ???

$f' = 4x^3, \quad f'' = 12x^2, \quad f''(0) = 0$

02/06/2024 @Yiming Yang, S24 Lecture on LR Models 38

38

Convexity of High-dimensional Function

We can its Hessian matrix $H \in \mathbb{R}^{d \times d}$.

- If $H \succcurlyeq 0$ (positive semi-definite), meaning that $\forall u,\ u^T H u \geq\ 0$, then f is convex.
- If $H \preccurlyeq 0$ (negative semidefinite), meaning that $\forall u,\ u^T H u \leq\ 0$), then f is concave.
- If none of the above is true, we can only reach a local optimal.

Convexity of High-dimensional Function

- **Proposition.** Suppose $f(\boldsymbol{x})$ is twice differentiable in its domain, and denote its Hessian as $H \overset{\text{def}}{=} \nabla^2 f(\boldsymbol{x}) \in \mathbb{R}^{d \times d}$

  1) f is convex if and only i $H \succcurlyeq 0$ for every $\boldsymbol{x} \in dom\ f$.

  2) f is strictly convex if and only if $H \succ 0$ for every $\boldsymbol{x} \in dom\ f$.

  3) f is m-strongly convex if and only if $H \succcurlyeq mI\ (m > 0)$ for every $\boldsymbol{x} \in dom\ f$, where $A \succcurlyeq B$ means that $A - B$ is positive semi–definite.

Positive (Semi-)Definite Matrices

- **Definitions**
  - A symmetric real matrix is positive-definite if the real number $\boldsymbol{z}^T M\boldsymbol{z}$ is positive for every nonzero vector $\boldsymbol{z} \in \mathbb{R}^n$.
  - A symmetric real matrix is positive semi-definite if the real number $\boldsymbol{z}^T M\boldsymbol{z}$ is non-negative for every nonzero vector $\boldsymbol{z} \in \mathbb{R}^n$.
- **Examples**
  - Identity matrix I $(\mathrm{n} \times n)$ is positive-definite as $\boldsymbol{z}^T I\boldsymbol{z} > \boldsymbol{0}$ for every nonzero vector $\boldsymbol{z}$.
  - Zero matrix 0 $(\mathrm{n} \times n)$ is positive semi-definite as $\boldsymbol{z}^T 0\boldsymbol{z} = 0$ for every nonzero vector $\boldsymbol{z}$..
  - What about real matrix $M = A^T A$ where $A \in \mathbb{R}^{n\times m}$ is rectangular?

Let's check if $M = A^T A$ is positive(semi-)definite

- For every nonzero $\boldsymbol{z} \in \mathbb{R}^d$, we have

$$\boldsymbol{z}^T M\boldsymbol{z} = \boldsymbol{z}^T A^{\mathbf{T}} A\boldsymbol{z} = \|Az\|^2 \geq 0$$

- If M has a full rank (but we did not assume that), then $M \succcurlyeq mI$ and m is the smallest eigenvalue (positive) of the matrix (we omit the explanation for now).

Examples (High Dimensional Cases)
(Garret Thomas, https://gwthomas.github.io/docs/math4ml.pdf)

- $f(\boldsymbol{x}) = \boldsymbol{a}^T\ \boldsymbol{x}\text{-}b$ for $\boldsymbol{x} \in \mathbb{R}^d, \boldsymbol{a} \in \mathbb{R}^d, b \in \mathbb{R}$

  $\nabla_x f = \boldsymbol{a}, \quad \nabla_x^2 = 0_{\boldsymbol{d}\times\boldsymbol{d}}$ ← Convex but not strictly nor strongly

- $f(\boldsymbol{x}) = |\boldsymbol{x}|_{\mathbf{1}} = \sum_{i=1}^{d}|x_i|$ for $\boldsymbol{x} \in \mathbb{R}^d$

  $\nabla_x f = (sign(x_1), sign(x_2), \dots, sign(x_d)),, \quad \nabla_x^2 = ?$

- $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_2^2 = \sum_{i=1}^{d} x_i^2$ for $\boldsymbol{x} \in \mathbb{R}^d$

  $\nabla_x f = (2x_1, 2x_2, \dots, 2x_d), \quad \nabla_x^2 = 2I$ ← Strongly convex

- $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\boldsymbol{T}} A^{\boldsymbol{T}} A\boldsymbol{x}$ for $\boldsymbol{x} \in \mathbb{R}^d$ and $A \in \mathbb{R}^{d\times k}$

  $\nabla_x f = A^T A\boldsymbol{x}, \quad \nabla\nabla_x f = A^T A$ ← Convexity?

The Hessian of LR

- We have shown $H^{(LR)} = -X^T \Lambda X$
- It can be re-written as

$$\mathbf{H}^{(LR)} = -\mathbf{X}^{\mathbf{T}}\boldsymbol{\Lambda}\mathbf{X} = -\underbrace{\mathbf{X}^{\mathbf{T}}\boldsymbol{\Lambda}^{1/2}}_{A^T}\underbrace{\boldsymbol{\Lambda}^{1/2}\mathbf{X}}_{A}$$

where $\boldsymbol{\Lambda} = diag(\sigma_i(1-\sigma_i))_{i=1\cdots n}$

$$\forall \boldsymbol{u} \in R^{m+1},\ \mathbf{u}^{\mathbf{T}}\mathbf{H}^{(LR)}\mathbf{u} = -\mathbf{u}^{\mathbf{T}} A^T A\boldsymbol{u} = -\|Au\|_2^2 \le\ 0$$

H is negative semi-definite. LR has a concave objective function.

Outline

✓ Introduction

- Binary LR
  - ✓ Optimization algorithms
  - ✓ Convexity
  - Regularization
- Softmax LR

Regularized Logistic Regression (RLR)

- So far, we have focus on the MLE objective as:

$$\hat{\mathbf{w}}^{LR} = \mathrm{argmax}_{\mathbf{w},w_0}\{\, l_D(\mathbf{w})\}$$

$$l_D(\mathbf{w}) = \sum_{i=1}^{n} \{\, y_i \ln \sigma\,(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i) + (1 - y_i)\ln(\,1 - \sigma(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i))\,\}$$

- Now we add a regularization term as:

$$\hat{\mathbf{w}}^{RLR} = \mathrm{argmax}_{\mathbf{w}}\left\{\, l_D(\mathbf{w}) - \frac{1}{2}C||\mathbf{w}||^2\right\}$$

- Equivalent to adding a Bayesian prior for w (next slide)

Maximum A Posterior (MAP) Solution

- Bayesian Prior (Assumption)

$$\mathbf{w} \sim N(0, \sigma^2 I) \qquad P(\mathbf{w}) = \frac{1}{Z_0} \exp\left(-\frac{\mathbf{w}^{\mathbf{T}}\mathbf{w}}{2\sigma^2}\right)$$

where Z_0 is some constant (normalization factor).

- Posterior Probability $P(\mathbf{w}|D) = \frac{P(D|\mathbf{w})P(\mathbf{w})}{P(D)} \propto P(D|\mathbf{w})P(\mathbf{w})$

- Objective

$$\begin{aligned}\hat{\mathbf{w}}^{RLR} &= \text{argmax}_{\mathbf{w}} P(\mathbf{w}|D) = \text{argmax}_{\mathbf{w}} P(D|\mathbf{w})P(\mathbf{w}) \\ &= \text{argmax}_{\mathbf{w}} \left\{\underbrace{\log P\,(D|\mathbf{w})}_{l(\mathbf{w})} + \log P\,(\mathbf{w})\right\} \\ &= \text{argmax}_{\mathbf{w}} \{l(\mathbf{w}) - \lambda \mathbf{w}^{\mathbf{T}}\mathbf{w} + \text{some constant}\}\end{aligned}$$

where $\lambda = \frac{1}{2\sigma^2}$

- MAP solution for RLR assumes a "non-informative" Gaussian prior of $\mathbf{w}$.

L1 vs L2 regularization
(figure from Elements of Stat. Learn., Hastie et al.)

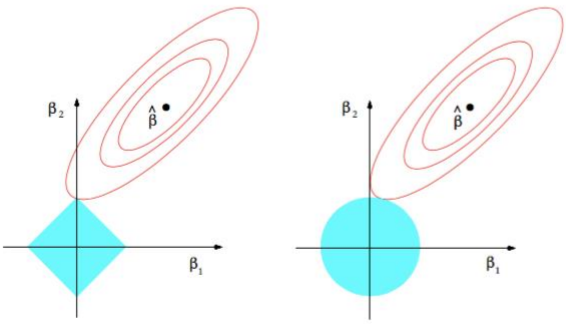


FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions $|\beta_1| + |\beta_2| \le t$ and $\beta_1^2 + \beta_2^2 \le t^2$, respectively, while the red ellipses are the contours of the least squares error function.*

Outline

- ✓Introduction
- ✓Binary LR
  - ✓Optimization algorithms
  - ✓Convexity
  - ✓Regularization
- Softmax LR

Softmax LR

- Let $\boldsymbol{x} \in R^d$ be the input vector, $k = 1, 2, \dots, K$ be the index of category labels, and $Y \in \{1,\ 2,\ \cdots,\ \text{K}\}$ be the output variable.
- Assume a categorical (multinomial) distribution of labels given $\boldsymbol{x}$, with $p_k(\boldsymbol{x}) \overset{\text{def}}{=} \text{P}(Y = k\ |\boldsymbol{x})$ and $\sum_{k=1}^{K} p_k(\boldsymbol{x}) = 1$,

$$Y|\boldsymbol{x} \sim Cat\big(p_1(\boldsymbol{x}), p_2(\boldsymbol{x}), \dots, p_k(\boldsymbol{x})\big)$$

- The system predicts the probability of each category as

$$\hat{p}_k(\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_k^T \boldsymbol{x})}{\sum_{k'=1}^{K} \exp\left(\boldsymbol{w}_{k'}^T \boldsymbol{x}\right)},\ \ k = 1, 2, \dots, K$$

  where $\boldsymbol{w}_k^T \in R^d$ is a category-specific vector of model parameters .

- Decision Rule: $\hat{Y}|\boldsymbol{x} = \text{argmax}_k\{\ \hat{p}_k(\boldsymbol{x})\ \}$

Log-likelihood of Softmax LR

$P(D|W) = \prod_{i=1}^{N} \prod_{k-1}^{K} p_k(x_i)^{y_{ik}}$ $\quad\quad y_{ik} \in \{0,1\}$

$\log P(D|W) = \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} log p_k(x_i)$

$= \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} log \frac{\exp(w_k^T x_i)}{\sum_{k'=1}^{K} \exp(w_{k'}^T x_i)}$

$= \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} w_k^T x_i - \sum_{i=1}^{N} \log \sum_{k'=1}^{K} \exp(w_{k'}^T x_i)$

where $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}, W = \{\boldsymbol{w}_k\}_{k=1}^{K}$ and $y_{ik} = I(y(x_i = k)$

Loss Function of Regularized LR

$J(W; D) = \frac{\lambda}{2} \sum_{k=1}^{K} ||\boldsymbol{w}_k||^2 - \log P(D|W)$ $\quad\quad (\lambda > 0)$

$= \frac{\lambda}{2} \sum_{k=1}^{K} ||\boldsymbol{w}_k||^2 - \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \boldsymbol{w}_k^T \boldsymbol{x}_i + \sum_{i=1}^{N} \log \sum_{k'=1}^{K} \exp(\boldsymbol{w}_{k'}^T \boldsymbol{x}_i)$

- Is Softmax convex? (Yes, but tricks are needed for proving)
- How to minimize the loss function?
- Where is the computational bottleneck, and how to alleviate it?

Convexity of Softmax LR

$J(W; D) = \frac{\lambda}{2}\sum_{k=1}^{K}\|\boldsymbol{w}_k\|^2 - \log P(D|W)$

$= \frac{\lambda}{2}\sum_{k=1}^{K}\|\boldsymbol{w}_k\|^2 - \sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\boldsymbol{w}_k^T\boldsymbol{x}_i + \sum_{i=1}^{N}\log\sum_{k'=1}^{K}\exp(\boldsymbol{w}_{k'}^T\boldsymbol{x}_i)$

- The first term (regularization) is convex because the non-negatively weighted sum of convex function is convex.
- The 2nd term is a linear function (convex and concave) of model parameters.
- The 3rd term is about the convexity of the log-sum-exp (LSE) function.

Convexity of log-sum-exponential

- https://math.stackexcwhy-is-log-of-sum-of-exponentials-fx-log-left-sum-i-1n-e-x-i-righthange.com/questions/2418554/
  - Proving based on the convexity definition
- https://math.stackexchange.com/questions/2416837/the-second-derivative-of-log-left-sum-limits-i-1nex-i-right-seems-neg
  - By showing the Hessian of Softmax LR to be positive semi-definite

Optimization of Softmax LR

$$\min_W J(W; D) =$$

$$= \min_w \frac{\lambda}{2} \sum_{k=1}^{K} \|\boldsymbol{w}_k\|^2 - \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \boldsymbol{w}_k^T \boldsymbol{x}_i + \sum_{i=1}^{N} \log \sum_{k'=1}^{K} \exp(\boldsymbol{w}_{k'}^T \boldsymbol{x}_i)$$

- It cannot be solved analytically.
- Instead, it should be solved with GD or SGD iteratively.
- Exercise by yourself in HW2: derive $\nabla_{w_k} J(W; D)$

Computing the gradients

$$J(W; D) = \underbrace{\frac{\lambda}{2} \sum_{k=1}^{K} \|\boldsymbol{w}_k\|^2}_{f(w_1, w_2, \ldots, w_K)} - \underbrace{\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \boldsymbol{w}_k^T \boldsymbol{x}_i}_{g(w_1, w_2, \ldots, w_K)} + \underbrace{\sum_{i=1}^{N} \log \sum_{k'=1}^{K} \exp(\boldsymbol{w}_{k'}^T \boldsymbol{x}_i)}_{\varphi(w_1, w_2, \ldots, w_K)}$$

$$\frac{\partial f}{\partial \boldsymbol{w}_k} = \frac{\lambda}{2} \frac{\partial}{\partial w_k} \|\boldsymbol{w}_k\|^2 = \lambda \boldsymbol{w}_k$$

input is a vector; output is a scalar
→ the gradient is a vector.

$$\frac{\partial g}{\partial \boldsymbol{w}_k} = \sum_{i=1}^{N} y_{ik} \frac{\partial}{\partial \boldsymbol{w}_k} \boldsymbol{w}_k^T \boldsymbol{x}_i = \sum_{i=1}^{N} y_{ik} \boldsymbol{x}_i$$

$$\frac{\partial \varphi}{\partial \boldsymbol{w}_k} = \sum_{i=1}^{N} \frac{\partial}{\partial \boldsymbol{w}_k} LSE = \sum_{i=1}^{N} \frac{\partial (\log v)}{\partial v} \ldots \frac{\partial z}{\partial \boldsymbol{w}_k} \quad \left(v = \sum_{k'=1}^{K} \exp(\boldsymbol{w}_{k'}^T \boldsymbol{x}_i)\right)$$

Parallel Computing Bottleneck

$$\min_{w} \frac{\lambda}{2}\sum_{k=1}^{K}\|w_k\|^2 - \sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} w_k^T x_i + \sum_{i=1}^{N}\log\sum_{k'=1}^{K}\exp(w_{k'}^T x_i)$$

- Easy to decouple the updates of w_k's updates for first two terms
- But how can we decouple for 3rd term (on different processors)?

57

Distributed training of regularized LR [Gopal & Yang, ICML 2013]

- Objective

$$L(W) = \frac{\lambda}{2}\sum_{k1}^{K}\|w_k\|^2 - \sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} w_k^T x_i + \sum_{i=1}^{N}\log\sum_{k'=1}^{K}\exp(w_{k'}^T x_i)$$

- Log-concavity bound (the 1st order concavity property) of log function

$$\log(v) \le \alpha v - \log(a) - 1 \quad \forall v, \quad a > 0$$

- Log-partition term for each instance i in LR is bounded as:

substitute

$$\log\left(\sum_{k=1}^{K}\exp(w_k^T x_i)\right) \le \alpha_i \sum_{k=1}^{K}\exp(w_k^T x_i) - \log(\alpha_i) - 1 \quad \alpha_i > 0$$

variational parameter

58

The Modified Objective Function

$$\min_{\alpha > \mathbf{0}, \mathbf{w}} \quad F(\mathbf{W}, \alpha)$$

$$F(\boldsymbol{W}, \boldsymbol{\alpha}) = \frac{\lambda}{2} \sum_{k1}^{K} \|w_k\|^2 - \sum_{i=1}^{N} \sum_{k=1}^{K} \left(y_{ik} w_k^T x_i - \alpha_i \exp\left(w_{k'}^T x_i\right) - log\alpha_i - 1 \right)$$

Convergence-related Properties (proof in Gopal's ICML 2013 paper)

1) It does not preserve the convexity of the original objective function (because we have α_i*'s* as the additional variables).
2) However, it has exactly one stationary point that is the same stationary point of the original convex function of softmax.
3) A block co-ordinate descent procedure guarantees to converge to the stationary point which is the optimal solution of softmax.

Concluding Remarks about LR

- Explicit probabilistic reasoning
- Easy to extend with regularization terms (e.g., L1 or L2 norm of the parameter vector)
- Commonly used as building blocks in neural nets
- Expressive enough for complicated decision boundaries?

A good online lecture on LR by Andrew Ng

- Andrew Ng on LR decision boundary