CLASSIFICATION

CLS 1 & 2. LR Models

1

4 Lectures on Classification

CLS 1 & 2. Logistic Regression (LR) Models

CLS 3. Stochastic Gradient Descent & Evaluation Metrics

CLS 4. Neural Classifiers for Extremely Large Classification

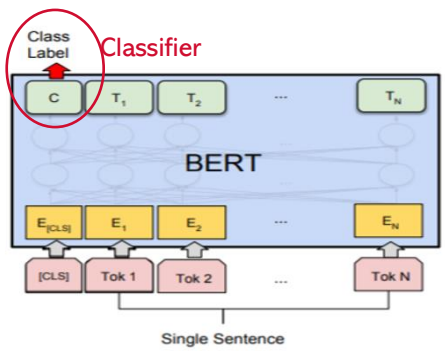
2

Outline on LR Models

- Introduction
- Decision boundaries
- Binary LR
- Optimization algorithms
- Convexity
- Regularization
- Softmax LR

3

BERT Fine Tuning for Classification



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

4

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 1 out of $M > 2$ category labels.
- **Wikipedia page subject topics** (multi-label classification)
  - Given input text, predict the relevant labels;
  - Choosing 1 or more out of $M > 2$ category labels.

5

Mathematical Definition

- Find the mapping $f: X \rightarrow Y$ for $X \in \mathbb{R}^d$ and $Y \in \{0,1\}^M$.
- Practically, predict vector $f(x) \in \mathbb{R}^M$ and apply a threshold to the elements of $f(\mathrm{x})$ for yes/no decisions

  Option 1. Assigning *yes* to the k top-ranking label and *no* to the rest where the k is a prespecified hyper-parameter;

  Option 2. Assigning *yes* to label j is $f_j(\mathrm{x}) \geq 0.5$ for $j = 1, \ldots, M$;

  Option 3. …

  (see Y Yang, SIGIR 2001)

6

Terminology

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

7

Scoring Functions

- **Linear Function** (e.g., linear regression or Naïve Bayes models)

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

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

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

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

$$f_{\boldsymbol{w}}(\boldsymbol{x}) \equiv \widehat{P_{\boldsymbol{w}}}(Y = 1|\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{w}^T \boldsymbol{x}}}$$

8

Scoring Functions (cont'd)

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

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

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

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

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

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

Example of a linear classifier

- A linear decision boundary is defined as a set of data points

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

which is

  - a line in 2D
  - a plane in 3D
  - a hyperplane in $\mathbb{R}^d$

- If the decision boundary by a classifier is linear, we call it a linear classifier.

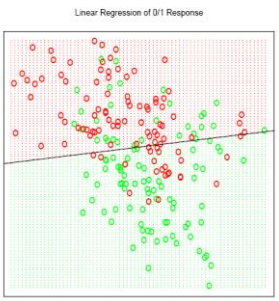


Figure 2.1: *A classification example in two dimensions. The classes are coded as a binary variable— GREEN = 0, RED = 1—and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$.

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

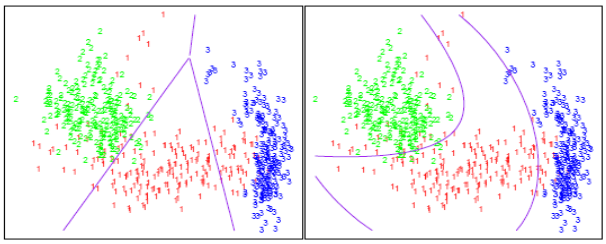


Figure 4.1: *The left plot shows some data from three classes, with linear decision boundaries found by linear discriminant analysis. The right plot shows quadratic decision boundaries. These were obtained by finding linear boundaries in the five-dimensional space* $X_1, X_2, X_{12}, X_1^2, X_2^2$. *Linear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

(Hastie et al., ESL)

11

Decision Boundaries of kNN (non-linear)

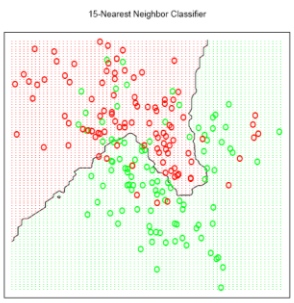


Figure 2.2: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (GREEN $= 0$, RED $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

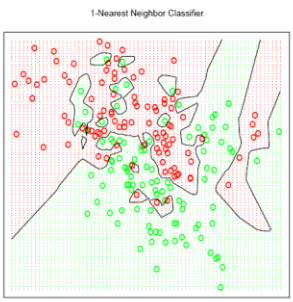


Figure 2.3: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (GREEN $= 0$, RED $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

12

How to tell if a classifier is linear or not?

- We cannot tell by just looking at $f_{\boldsymbol{w}}$ (as being linear or non-linear).
- Instead, we must check if the decision boundary can be written as

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

- Let's take a look at a binary LR and Softmax classifiers.

Is binary LR a linear classifier?

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

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

- A popular threshold for a binary decision is set as

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

- Denoting the decision boundary as h we have

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

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

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

Is softmax LR a linear classifier?

- Scoring function for $k = 1, 2, \ldots, K$

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

- Decision boundary between labels j and k

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

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

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

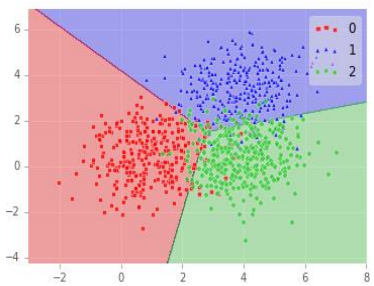
$$\Rightarrow h_{jk} = \{x: \underbrace{(\mathbf{w}_j - \mathbf{w}_k)}_{w}{}^T x = 0\} \qquad (7)$$

15

Is softmax LR a linear classifier?

- Answer
  - Locally (pairwise) linear but globally nonlinear
- Thresholding strategy

$$\hat{y}(\mathbf{x}) = \text{argmax}_k\{\hat{p}_k(\boldsymbol{x})\}$$



16

Outline

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

17

LR for Binary Classification

- Label probabilities estimated using a sigmoid function

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

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

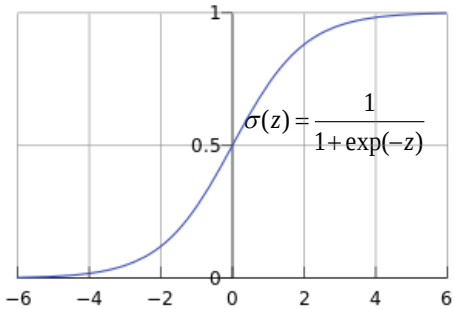
- Compact formula

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

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

18

Sigmoid Function



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

Logit = logarithm of the odds

$p = \frac{1}{1 + \exp(-z)}$ ☜ 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

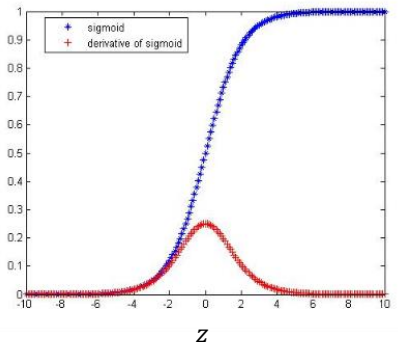
$\exp(z) = \frac{p}{1-p}$ ☜ Flip over

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

logit odds of p

Derivative of Sigmoid

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



A good online lecture on LR by Andrew Ng

- Andrew Ng on LR decision boundary

Outline

- ✓Introduction
- ✓Decision boundaries
- ✓Binary Logistic Regression (LR)
- Optimization algorithms
- Convexity
- Regularization
- Softmax LR

23

Training a Binary Classifier

- Labeled Training Data

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

- 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 labeled data;
  - $\|\boldsymbol{w}\|^2$ is the regularization term, controlling the model complexity to avoid overfitting.

24

Parameter Optimization

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

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

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

the 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

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

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

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

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

Gradient ascent on a training set

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

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

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:

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

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:

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

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

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

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:

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

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

31

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)

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

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

32

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

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

Outline

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

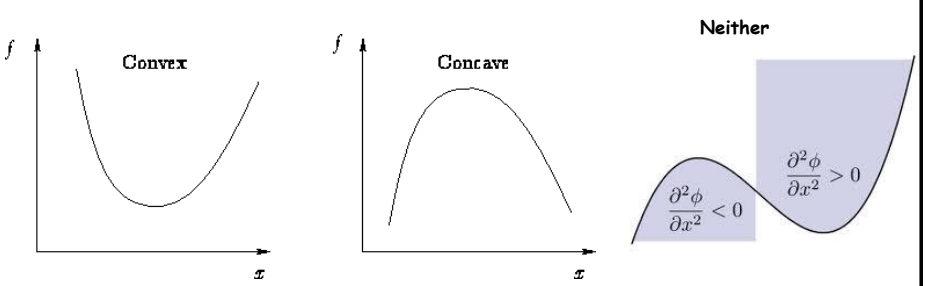
Globally optimal solution guaranteed?

- 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.

35

Convexity

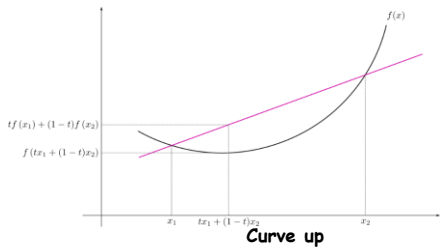
Examples of 1-dimensional functions



36

Convex Function

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



Curve up

- f is called **convex** if:

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

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

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

37

Convex or Not? (one-dimensional case)

- http://mathworld.wolfram.com/ConvexFun

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 $x \in [a, b]$.

- Examples
  - $f(x) = 2x + 3$: $f' = 2,\ f'' = 0$ → Convex but not strictly
  - $f(x) = x^2 + 2x + 1$: $f' = 2x,\ f'' = 2$ → Strictly convex
  - $f(x) = e^x$ → ?

38

Convexity (in general)

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

- **Proposition.** Suppose f is twice differentiable. Then
  1) f is convex if and only if $\nabla^2 f(\boldsymbol{x}) \succcurlyeq 0$ for all $\boldsymbol{x} \in dom\ f$.
  2) If $\nabla^2 f(\boldsymbol{x}) \succ 0$ for all $\boldsymbol{x} \in dom\ f$, then f is strictly convex.
  3) f is m-strongly convex if and only if $\nabla^2 f(\boldsymbol{x}) \succcurlyeq mI\ (m > 0)$ for all $\boldsymbol{x} \in dom\ f$, where $A \succcurlyeq B$ means that $A - B$ is positive semi-definite.
- Functions that are *convex but not strictly convex*
  - $f(\boldsymbol{x})=\boldsymbol{w}^T\boldsymbol{x}$-$b$ for any $\boldsymbol{x} \in \mathbb{R}^d, b \in \mathbb{R}$
  - $f(\boldsymbol{x}) = |\boldsymbol{x}|_1$ for any $\boldsymbol{x} \in \mathbb{R}^d$

39

Positive (Semi-)Definite Matrices

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

40

Hessian of the L1 norm of a vector

$$f(\boldsymbol{x}) = |\boldsymbol{x}|_1 = \sum_{i=1}^{d} |x_i|$$

$$\frac{\partial f}{\partial x_i} = \begin{cases} 1 & x_i \geq 0 \\ -1 & x_i < 0 \end{cases}$$

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = 0$$

$$\nabla\nabla f = \mathbf{0}_{d \times d}$$

41

Convexity (cont'd)

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

- Functions that are *strictly but not strongly convex*
  - $f(x) = e^x$ for any $x \in \mathbb{R}$ (its bounded below but has no local minimum.)
  - $f(x) = x^4$ for any $x \in \mathbb{R}$

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

- Functions that is *strongly convex*
  - $f(\boldsymbol{x}) = \| \boldsymbol{x} \|_2^2$

$f(x) = x^T x \quad \nabla f = 2x \quad \nabla^2 f = 2I$

42

$f(x)$ with multi-dimensional input

Check the convexity of function f by examining its hessian ($H \succcurlyeq 0$?)

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

43

The Hessian of LR

- We have shown $\mathbf{H}^{(LR)} = -\mathbf{X}^\mathbf{T}\mathbf{\Lambda}\mathbf{X}$
- We check its convexity as

$$\mathbf{H}^{(LR)} = -\mathbf{X}^\mathbf{T}\mathbf{\Lambda}\mathbf{X} = -\underbrace{\mathbf{X}^\mathbf{T}\mathbf{\Lambda}^{1/2}}_{\mathbf{X}'^\mathbf{T}}\underbrace{\mathbf{\Lambda}^{1/2}\mathbf{X}}_{\mathbf{X}'}$$

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

$$\forall \boldsymbol{u} \in R^{m+1},\ \mathbf{u}^\mathbf{T}\mathbf{H}^{(LR)}\mathbf{u} = -\underbrace{\mathbf{u}^\mathbf{T}\mathbf{X}'^\mathbf{T}}_{\mathbf{v}^\mathbf{T}}\underbrace{\mathbf{X}'\mathbf{u}}_{\mathbf{v}} = -\mathbf{v}^\mathbf{T}\mathbf{v} = -||\mathbf{v}||^2 \leq 0$$

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

44

Outline

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

45

Regularized Logistic Regression (RLR)

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

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

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

- Now we add a regularization term as:

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

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

46

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

$$\begin{aligned}\hat{\mathbf{w}}^{RLR} &= \text{argmax}_{\mathbf{w}} P(\mathbf{w}|D) = \text{argmax}_{\mathbf{w}} P(D|\mathbf{w})P(\mathbf{w}) \\ &= \text{argmax}_{\mathbf{w}} \{\underbrace{\log P(D|\mathbf{w})}_{l(\mathbf{w})} + \log P(\mathbf{w})\} \\ &= \text{argmax}_{\mathbf{w}} \{l(\mathbf{w}) - \lambda \mathbf{w}^{\mathbf{T}}\mathbf{w} + \text{some constant}\} \\ \text{where } \lambda &= \frac{1}{2\sigma^2}\end{aligned}$$

- MAP solution for RLR assumes a "non-informative" Gaussian prior of **w**.

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

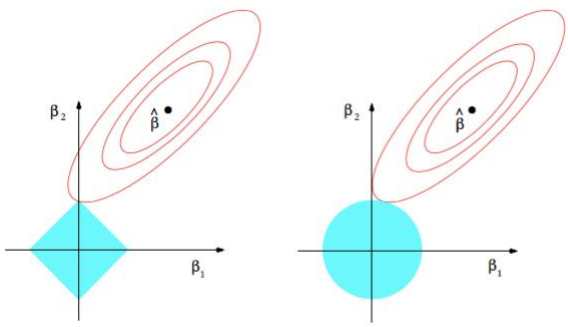


FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are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

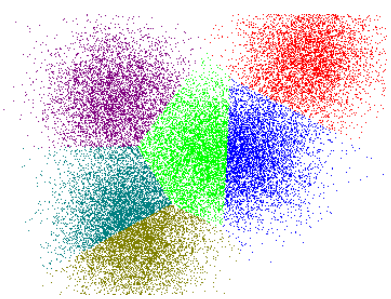
Outline

- ✓Introduction
- ✓Decision boundaries
- ✓Binary Logistic Regression (LR)
- ✓Optimization algorithms
- ✓Convexity
- ✓Regularization
- Softmax LR

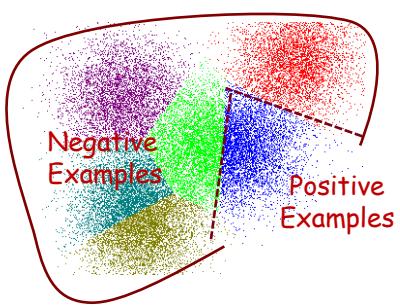
49

SoftMax LR vs. Binary LR for Multi-class Classification

Training Data for Softmax LR



Training data for Binary OVA (one-vs-all)) Models



50

Softmax LR

- Let $Y \in \{1, 2, \cdots, K\}$ be the target variable in categorical distribution

$$Y|\boldsymbol{x} \sim Cat\big(p_1(\boldsymbol{x}), p_2(\boldsymbol{x}), \ldots, p_k(\boldsymbol{x})\big) \text{ where } p_1 + p_2 + \cdots + p_k = 1$$

$$\hat{p}_k(\boldsymbol{x}) \equiv \hat{P}(Y = k|\boldsymbol{x}) = \frac{\exp\big(w_k^T \boldsymbol{x}\big)}{\sum_{k'=1}^{K} \exp\big(w_{k'}^T \boldsymbol{x}\big)} \quad k = 1, 2, \ldots, K$$

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

51

Log-likelihood of Softmax LR

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

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

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

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

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

52

Loss Function of Regularized LR

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

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

- 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?

53

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.

54

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.

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

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

57

Concrete Example

$$\boldsymbol{w} = (w_1, w_2, w_3)^T$$

$$f(\boldsymbol{w}) = w^T w = w_1^2 + w_2^2 + w_3^2$$

$$\nabla_{\boldsymbol{w}} f \equiv \left[\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \frac{\partial f}{\partial w_3}\right] = [2w_1, 2w_2, 2w_3] = 2\boldsymbol{w}$$

A trick: just consider w as a 1-dimensional variable

$$f(w) = \mathrm{w}^2, \quad \nabla_w f = 2\mathrm{w}$$

58

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)?

59

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

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

- 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

60

The Modified Objective Function

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

$$F(\boldsymbol{W},\boldsymbol{\alpha}) = \frac{\lambda}{2}\sum_{k1}^{K}\|w_k\|^2 - \sum_{i=1}^{N}\sum_{k=1}^{K}\left(y_{ik}w_k^T x_i - \alpha_i \exp\left(w_k^T, x_i\right) - log\alpha_i - 1\right)$$

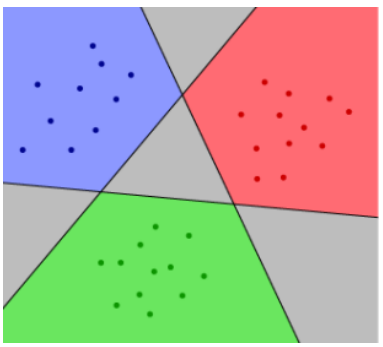
Convergence-related Properties (proof in Gopal's ICML 2013 paper)

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.

61

Decision boundaries by K (OVA) classifiers if each classifier makes its independent decisions

https://shapeofdata.wordpress.com/

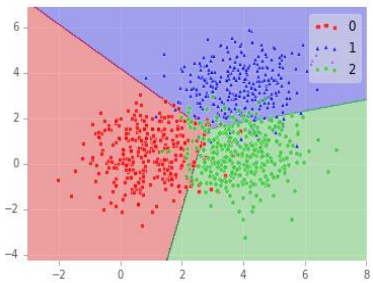


The number of labels per instance could be 0, 1, 2, ... if we set the threshold to be 0.5 for each OVA model.

62

Softmax Decision Boundary

$$\hat{P}(y = k|x) = \frac{\exp(\mathbf{w}_k{}^T x)}{\sum_{k'=1}^{K} \exp(w_{k'}^T x_i)}$$



- Mutually exclusive and exhaustive labels (MEE), i.e., no empty region in the middle.

https://docs.microsoft.com/en-us/cognitive-toolkit/tutorial/tutorial#softmax

63

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

64