CLASSIFICATION

CLS 1 & 2. LR Models

4 Lectures on Classification

CLS 1 & 2. Logistic Regression (LR) Models

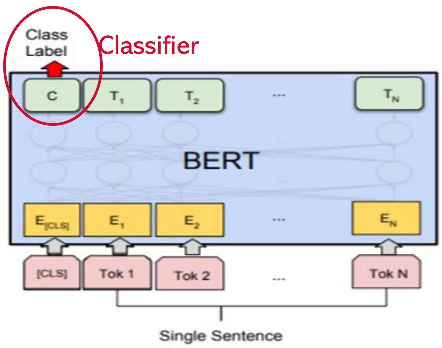
CLS 3. Stochastic Gradient Descent & Evaluation Metrics

CLS 4. Neural Classifiers for Extremely Large Classification

Outline on LR Models

- Introduction
- Decision boundaries
- Binary LR
- Optimization algorithms
- Convexity
- Regularization
- 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 $y \in \{0,1, \dots, 9\}$;
  - 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.

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 $j = 1, \dots, M$;

  Option 3. …

  (see Y Yang, SIGIR 2001)

Terminology

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

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)

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

Scoring Functions (cont'd)

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

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

- **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.

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.

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

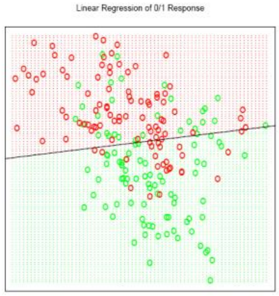


Figure 2.1: *A classification example in two dimensions. The classes are coded as a binary variable—* GREEN $= 0$, RED $= 1$ *—and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$.

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

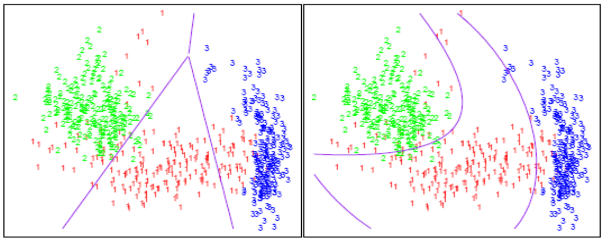


Figure 4.1: *The left plot shows some data from three classes, with linear decision boundaries found by linear discriminant analysis. The right plot shows quadratic decision boundaries. These were obtained by finding linear boundaries in the five-dimensional space* $X_1, X_2, X_{12}, X_1^2, X_2^2$. *Linear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

(Hastie et al., ESL)

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

Decision Boundaries of kNN (non-linear)

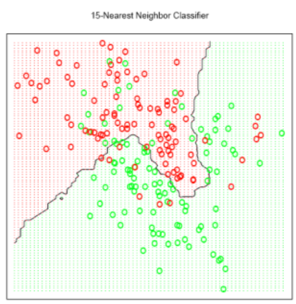


Figure 2.2: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (GREEN $= 0$, RED $= 1$) *and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

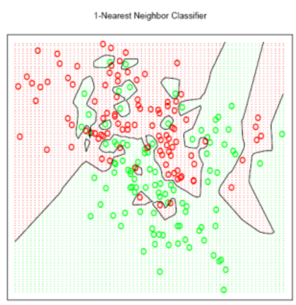


Figure 2.3: *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (GREEN $= 0$, RED $= 1$), *and then predicted by 1-nearest-neighbor classification.*

How to tell if a classifier is linear or not?

- We cannot tell by just looking at f_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)

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

- A popular threshold for a binary decision is set as

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

- Denoting the decision boundary as h we have

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

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

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

Is softmax LR a linear classifier?

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

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

- Decision boundary between labels j and k

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

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

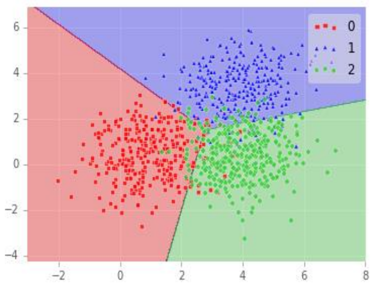
$$\Rightarrow (\mathbf{w}_j{}^T - \mathbf{w}_k{}^T)x = 0$$

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

Is softmax LR a linear classifier?

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

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



Outline

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

LR for Binary Classification

- Label probabilities estimated using a sigmoid function

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

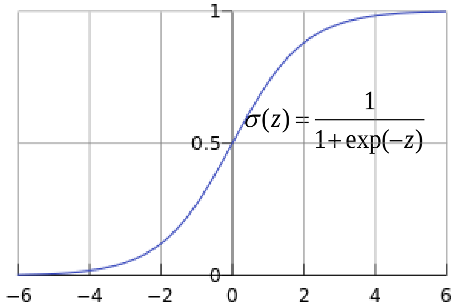
$$P_{\boldsymbol{w}}(y=0|\boldsymbol{x}) = 1 - \sigma_{\boldsymbol{w},\boldsymbol{x}} = \frac{\exp(-\boldsymbol{w}^T\boldsymbol{x})}{1+\exp(-\boldsymbol{w}^T\boldsymbol{x})}$$

- Compact formula

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

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

Sigmoid Function



z (the logit)

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

Logit = logarithm of the odds

$$p = \frac{1}{1 + \exp(-z)}$$

☜ 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

$$z = \log \frac{p}{1-p}$$

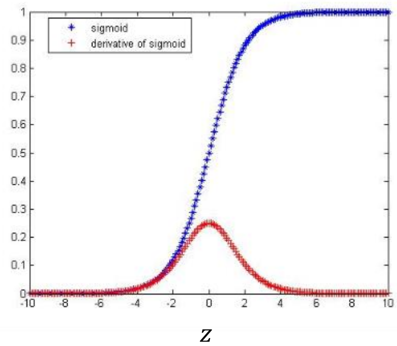
logit

odds of p

☜ Take the log on both sides

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

Training a Binary Classifier

- Labeled Training Data

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

Parameter Optimization

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

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

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

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

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

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

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

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

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

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:

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

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

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