5. Logistic Regression for Classification

Dongwoo Kim

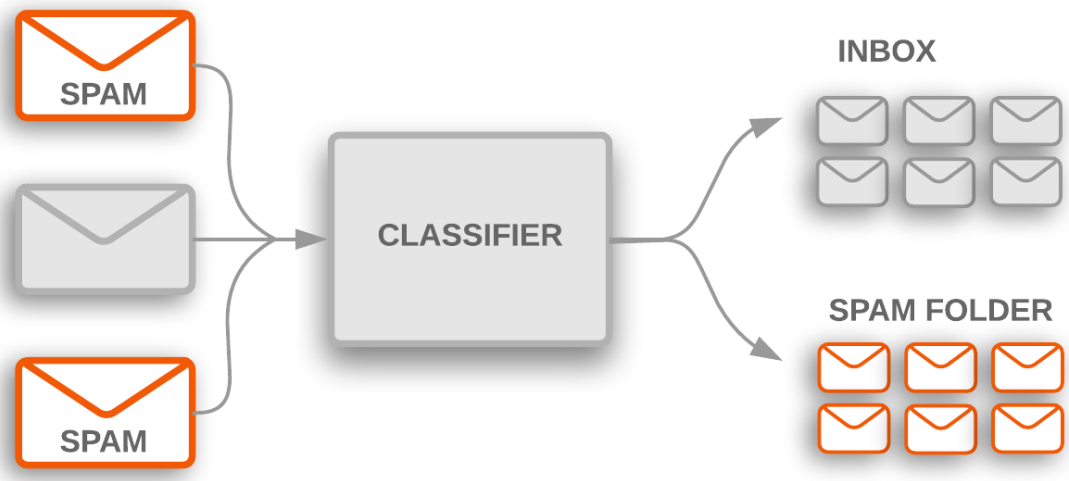
dongwookim.ac.kr

CSED515 - 2023 Spring

Classification

Goal: to assign an input feature vector $\boldsymbol{x}$ to one of K discrete class $\mathcal{C}_k$ where $k = 1, ..., K$.

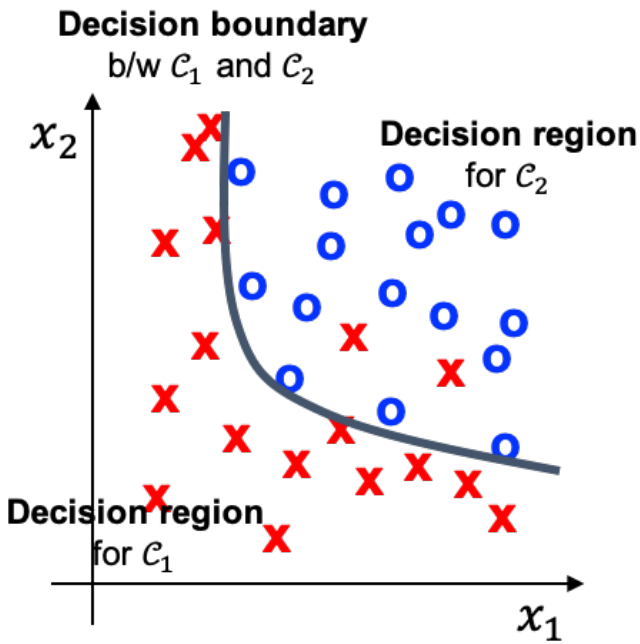
- e.g., handwriting recognition, speech recognition, biological classification, drug discovery, ..., spam filtering



https://medium.com/@naveeen.kumar.k/naive-bayes-spam-detection-7d087cc96d9d

Decision Boundary

- Suppose that there are two classes $\mathcal{C}_1, \mathcal{C}_2$
- A classifier assigns a new data point x to $\mathcal{C}_1$ only if it is in the decision region $\mathcal{R}_1$ for $\mathcal{C}_1$, i.e., classification function $h(x) = k$ if $x \in \mathcal{R}_k$.
- Decision boundaries (or surface) are boundaries between decision regions



developing classifier = drawing decision boundaries

Linear Regression for Classification?

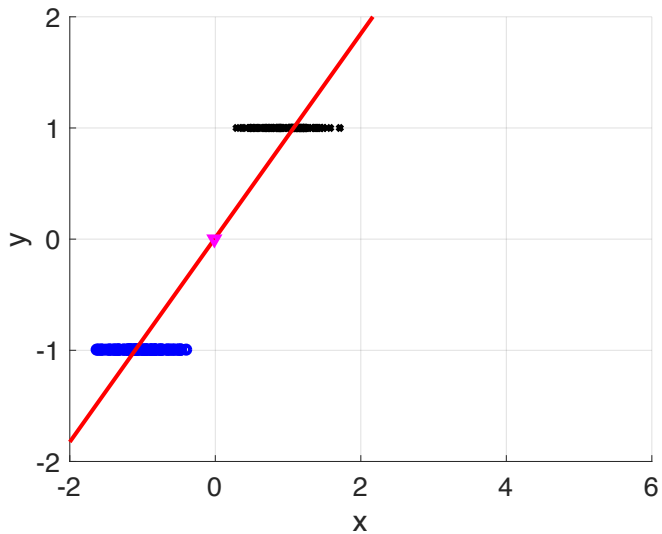
- Consider dataset $\mathcal{D} = (x_n, y_n)_{n \in [N]}$ for classification where $x_n \in \mathbb{R}$ and $y_n = \{-1, +1\}$.
- Assuming linear model $y = w_1 x + w_2$, then a classifier h can be defined as:

$$h(x) = \text{sign}(w_1 x + w_2) = \begin{cases} +1 & \text{if } w_1 x + w_2 \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

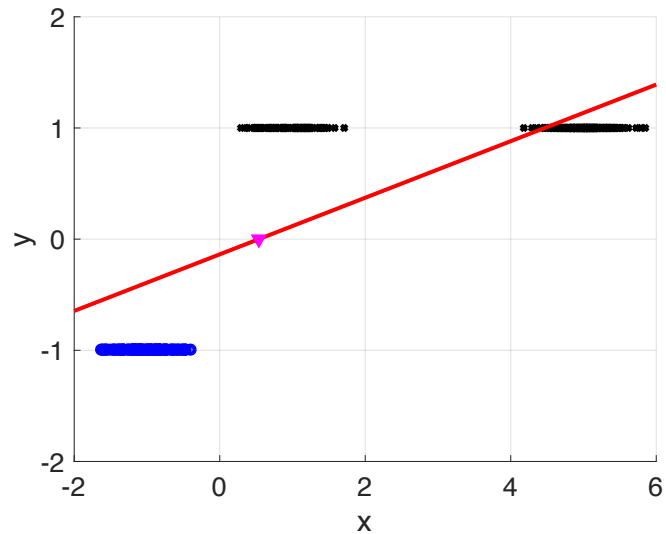
- Learning the linear regression aims at finding $\boldsymbol{w} = (w_1, w_2)$ such that the square error is small

$$\arg\min_{\boldsymbol{w}} \quad \frac{1}{2} \sum_{n=1}^{N} \big(y_n - (w_1 x_n + w_2)\big)^2$$

Linear Regression for Classification?



perfect classification



decision boundary shifted

- Note that x_i having $w_1 x_i + w_2$ farther from zero is easier sample
- Employing square error penalizes samples that are very easy to classify
- This problem is from the large mismatch between linear model $y = w_2 x + w_1$ and data $(x_n, y_n)_{n=1,\ldots,N}$

Outline

1. A systemic formulation: logistic regression
   - The optimality of Bayes decision rule
   - Logistic regression and probabilistic model
2. A systemic solver: gradient-based optimization
   - Gradient decent/ascent
   - Newton method

Discriminant Function

An alternative description of classifier:

$$\text{assign } x \text{ to } \mathcal{C}_k \text{ if } f_k(x) > f_j(x) \text{ for all } j \neq k$$

- $\{f_k\}_{k=1,\ldots,K}$: discriminant function/score
  - e.g., $f_k(x) = p(\mathcal{C}_k \mid x)$
- i.e., classifier $h(x) = \arg\max_k f_k(x)$
- If $f_k(\cdot)$'s are a set of discriminant functions, then for any monotonically increasing function $g : \mathbb{R} \to \mathbb{R}$ $g_k(\cdot) = g(f_k(\cdot))$'s are also a set of discriminant functions
  - e.g.1, $a\, f_k(x) + b$ for any $a > 0$ and b
  - e.g.2, $\log f_k(\cdot)$
  - ...

Discriminant Function: Bayes Decision Rule

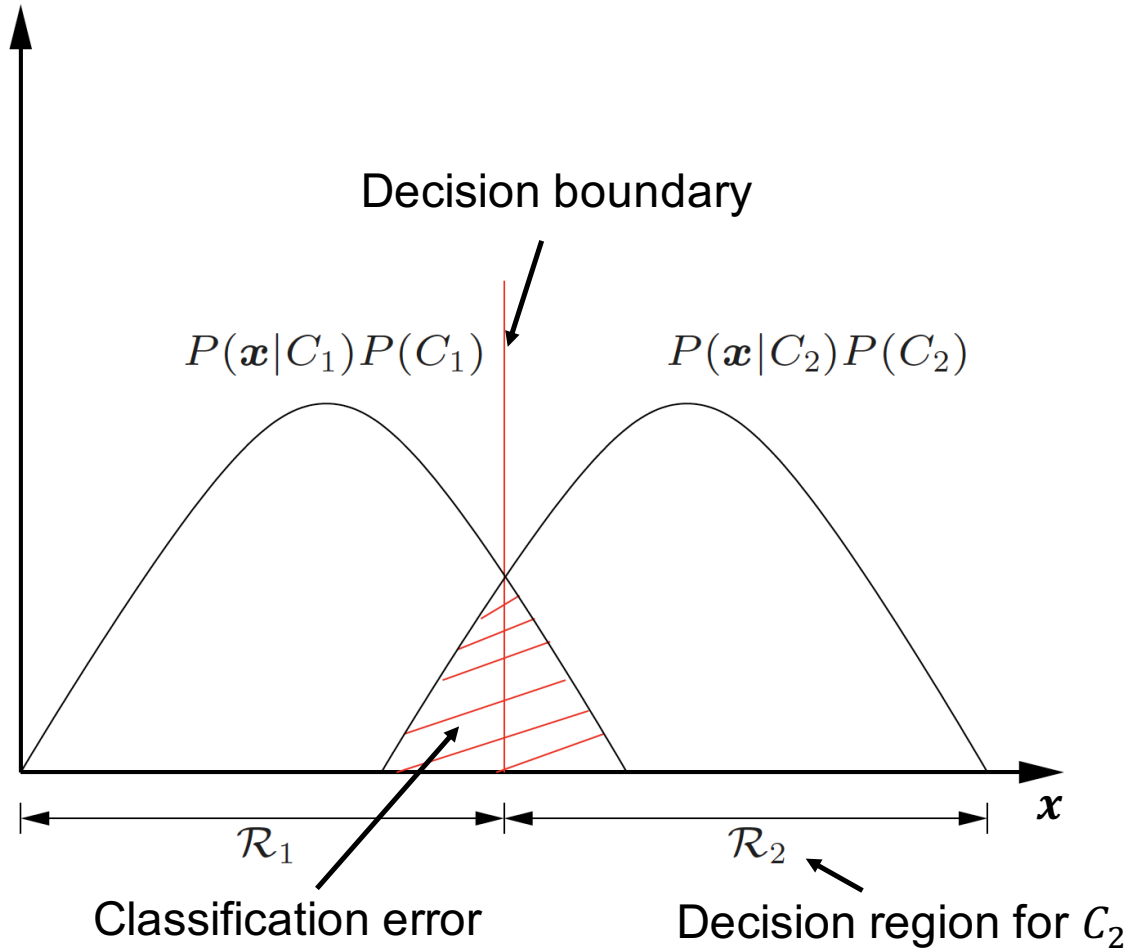
Bayes decision rule:

$$f_k(x) = p(\mathcal{C}_k \mid x) \text{ or equivalently, } f_k(x) = p(x \mid \mathcal{C}_k)p(\mathcal{C}_k)$$

- Classify $x \in \mathcal{C}_k$ if $p(\mathcal{C}_k \mid x) > p(\mathcal{C}_j \mid x)$ for all $j \neq k$.
- We can use discriminant function $f_k(x) = p(x \mid \mathcal{C}_k)p(\mathcal{C}_k)$ without normalization factor thanks to Bayes rule:

$$p(\mathcal{C}_k \mid x) = \frac{\overbrace{p(x \mid \mathcal{C}_k)}^{\text{Class-conditional density}} \overbrace{p(\mathcal{C}_k)}^{\text{Prior}}}{\underbrace{\sum_{j=1}^{K} p(x \mid \mathcal{C}_j)p(\mathcal{C}_j)}_{\text{Normalization factor}}}$$

Graphical Illustration of Bayes Decision Rule



Optimality of Bayes Decision Rule

The accuracy or $(1 - p(\text{error}))$ is given by:

$$\begin{aligned} p(\text{correct}) &= \sum_{k=1}^{K} p(x \in \mathcal{R}_k, \mathcal{C}_k) \\ &= \sum_{k=1}^{K} p(x \in \mathcal{R}_k \mid \mathcal{C}_k) p(\mathcal{C}_k) \\ &= \sum_{k=1}^{K} \int_{\mathcal{R}_k} p(x \mid \mathcal{C}_k) p(\mathcal{C}_k) dx \ . \end{aligned}$$

By dividing the domain of x into infinitesimal regions ε's, in order to maximize the accuracy (or to minimize the error rate), a region ε should be included in $\mathcal{R}_k$ such that $p(x \mid \mathcal{C}_k)p(\mathcal{C}_k) > p(x \mid \mathcal{C}_j)p(\mathcal{C}_j)$ for all $j \neq k$. (A formal argument can be found at link)

Outline

1. A systemic formulation: logistic regression
   - The optimality of Bayes decision rule
   - Logistic regression and probabilistic model
2. A systemic solver: gradient-based optimization
   - Gradient decent/ascent
   - Newton method

Logistic Regression: Binary Classifier

Inspired by Bayes decision rule, a probabilistic formulation for classification ($y_n \in \{-1, 1\}$) [1]:

$$p(y_n = 1 \mid \boldsymbol{x}_n, \boldsymbol{w}) = \frac{1}{1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x}_n)}$$

$$p(y_n = -1 \mid \boldsymbol{x}_n, \boldsymbol{w}) = 1 - p(y_n = 1 \mid x_n) = \frac{1}{1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$$

or equivalently,

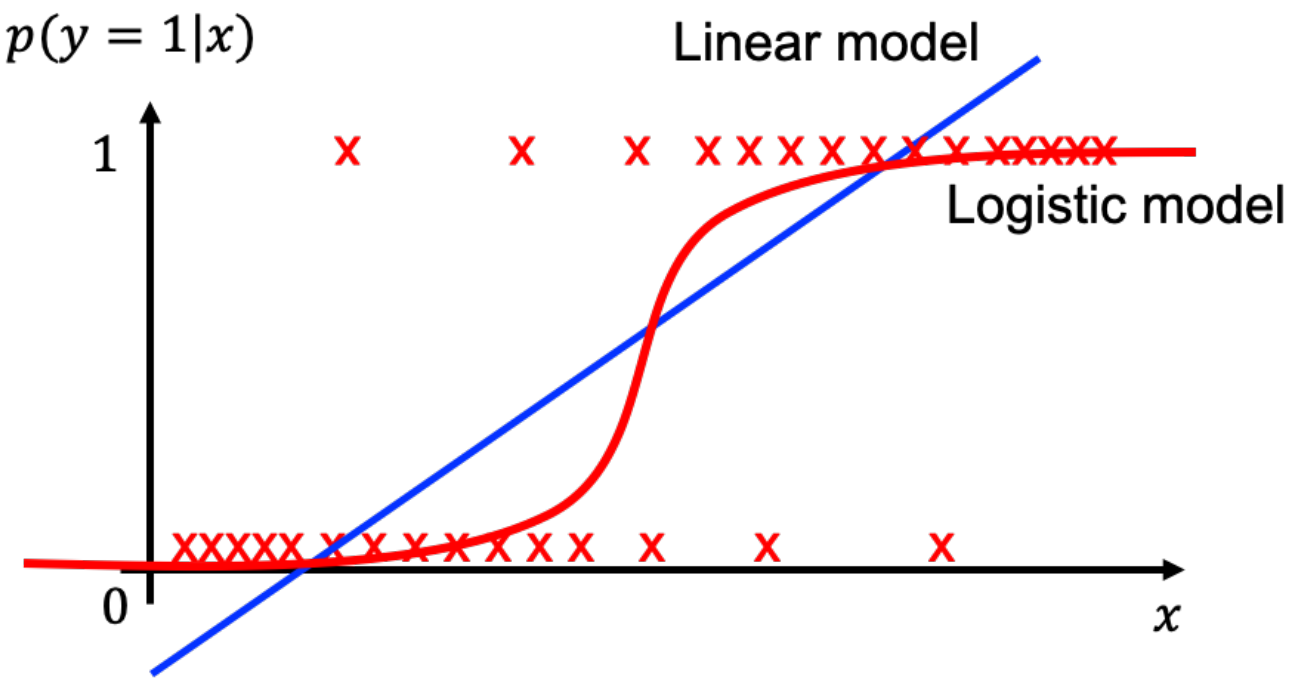
$$p(y_n \mid \boldsymbol{x}_n, \boldsymbol{w}) = \frac{1}{1 + \exp(-y_n \boldsymbol{w}^\top \boldsymbol{x}_n)}$$

[1] $\boldsymbol{x}_n$ can be replaced with $\phi(\boldsymbol{x}_n)$.

Linear vs. Logistic

- Linear regression ($y = w^\top x + b$) is easy estimation (closed form solution) but not suitable: (i) the target function (probability) is bounded $[0, 1]$, (ii) linear regression is not robust to outliers, ...
- Logistic regression (logistic or sigmoid function $\sigma(\cdot)$)

$$p(y = 1 \mid x) = \sigma(\xi(x)) = \frac{1}{1 + \exp(-\xi(x))} \in [0, 1]$$

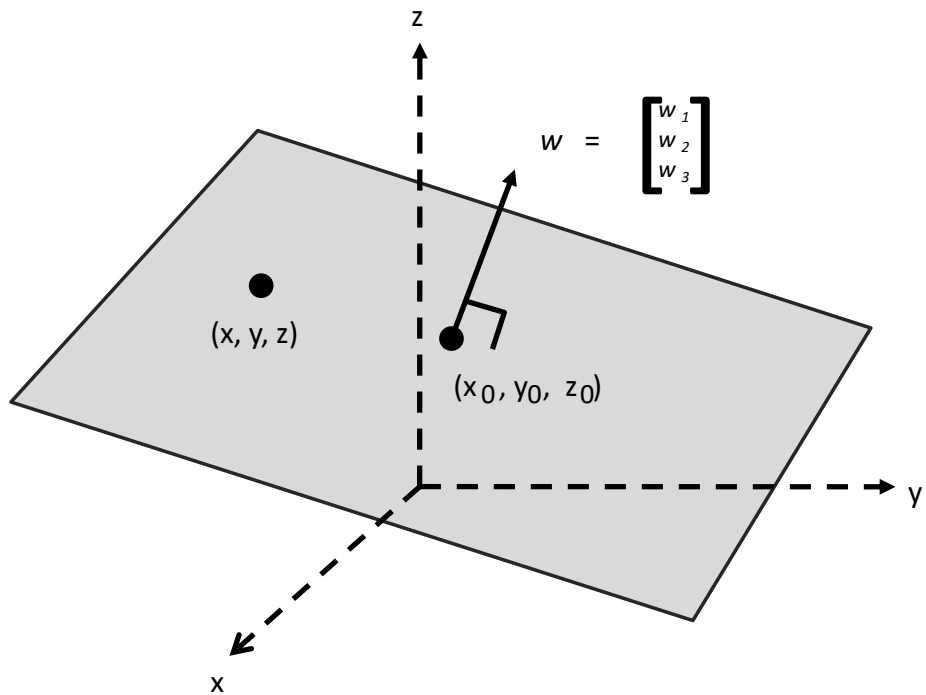


Logistic Decision Boundary

Note that we classify $\boldsymbol{x}_n$ based on the sign of $-\boldsymbol{w}^\top \boldsymbol{x}_n$.

$$p(y_n = 1 \mid \boldsymbol{x}_n, \boldsymbol{w}) = \frac{1}{1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x}_n)}$$

The decision boundary[2] of $\boldsymbol{w}^\top \boldsymbol{x}$:



[2] We assume that the last element of $\boldsymbol{x}$ is 1

Property of Logistic Function

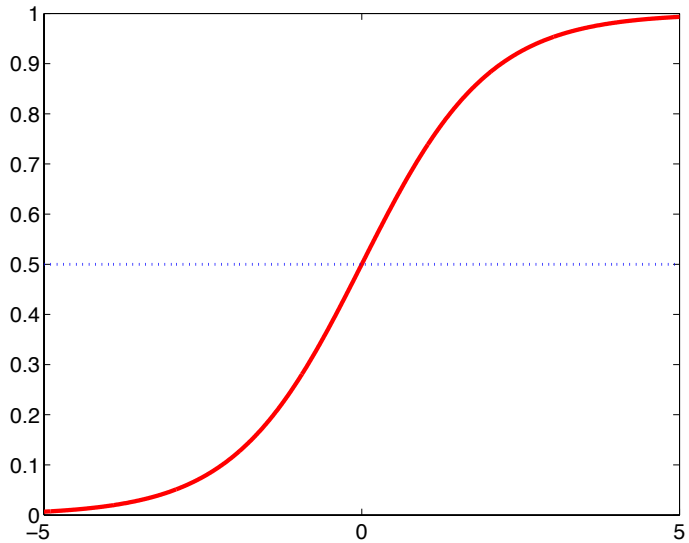


Figure: $\sigma(t) = \frac{1}{1+e^{-t}}$

- $\sigma(t) \to 0$ as $t \to -\infty$
- $\sigma(t) \to 1$ as $t \to \infty$
- $\sigma(-t) = 1 - \sigma(t)$
- $\frac{d}{dt}\sigma(t) = \sigma(t)\sigma(-t) = \sigma(t)(1 - \sigma(t))$

Optimization for Logistic Regression

Maximum likelihood estimate (to learn Bayes decision rule)

- Assume each data points are i.i.d., i.e.,

$$\log p(\mathcal{D} \mid \boldsymbol{w}) = \sum_{n=1}^{N} \log p(y_n \mid x_n, \boldsymbol{w})$$

- Recalling $p(y_n \mid x_n, \boldsymbol{w}) = \frac{1}{1+\exp(-y_n \boldsymbol{w}^\top \boldsymbol{x}_n)}$ in logistic regression, MLE is

$$\arg\min_{\boldsymbol{w}} \quad \sum_{n=1}^{N} \log\left(1 + \exp(-y_n \boldsymbol{w}^\top \boldsymbol{x}_n)\right)$$

Outline

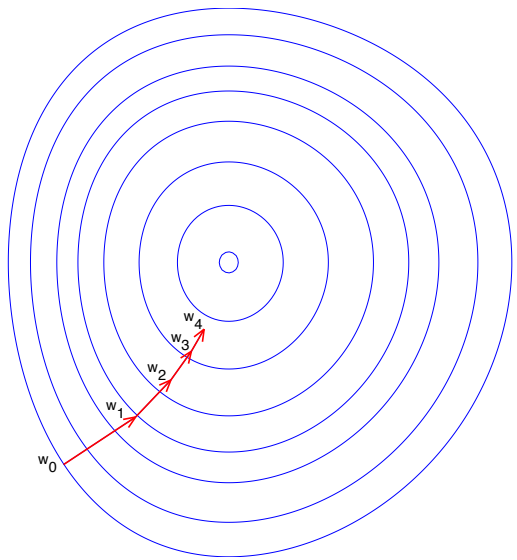
1. A systemic formulation: logistic regression
   - The optimality of Bayes decision rule
   - Logistic regression and probabilistic model
2. A systemic solver: gradient-based optimization
   - Gradient decent/ascent
   - Newton method

How to Optimize?

Logistic regression aims at finding

$$\arg\min_{\boldsymbol{w}} \quad \sum_{n=1}^{N} \log\left(1 + \exp(-y_n \boldsymbol{w}^\top \boldsymbol{x}_n)\right)$$

- Unfortunately, no analytic solution[3] for $\boldsymbol{w}$ in general
- Gradient descent



[3] https://en.wikipedia.org/wiki/Closed-form_expression

How to Optimize?

Logistic regression aims at finding

$$\arg\min_{\boldsymbol{w}} \underbrace{\sum_{n=1}^{N} \log\left(1+\exp(-y_n \boldsymbol{w}^\top \boldsymbol{x}_n)\right)}_{=:\mathcal{L}(\boldsymbol{w}) \text{ or } \mathcal{L}}$$

- Gradient

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) = \sum_{n=1}^{N} \frac{-y_n \exp(-y_n \boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(-y_n \boldsymbol{w}^\top \boldsymbol{x}_n)} \boldsymbol{x}_n$$

- Gradient descent:

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \alpha \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}_t}$$

Alternative form: $y_n \in \{0, 1\}$

If $y_n \in \{0, 1\}$, then the likelihood becomes

$$\prod_{n=1}^{N} p(y_n | \boldsymbol{x}_n, \boldsymbol{w}) = \prod_{n=1}^{N} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)^{y_n} \left(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)\right)^{(1-y_n)}$$

Then the optimization goal is

$$\arg\max_{\boldsymbol{w}} \sum_{n=1}^{N} y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))$$

Logistic Regression: Gradient Ascent ($y_n \in \{0, 1\}$)

The gradient ascent learning has the form

$$\boldsymbol{w}^{\text{new}} = \boldsymbol{w}^{\text{old}} + \alpha \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} \right).$$

Calculate the gradient

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \sum_{n=1}^{N} [y_n - \sigma(\boldsymbol{w}^\top \mathbf{x}_n)] \mathbf{x}_n$$

Hence, the gradient ascent update rule for $\boldsymbol{w}$ is

$$\boldsymbol{w}^{\text{new}} = \boldsymbol{w}^{\text{old}} + \alpha \sum_{n=1}^{N} \left[y_n - \sigma \left(\left(\boldsymbol{w}^{\text{old}} \right)^\top \mathbf{x}_n \right) \right] \mathbf{x}_n.$$

Detailed Calculation of Gradient

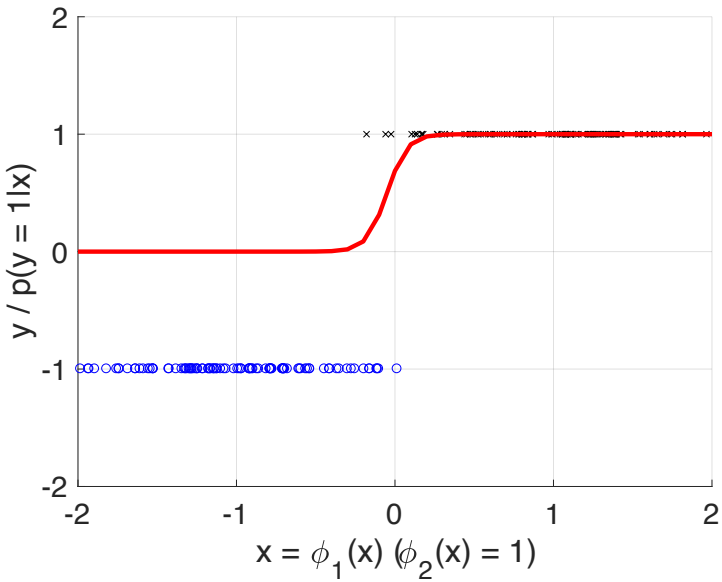
Let $\hat{y}_n = \sigma(\boldsymbol{w}^\top \mathbf{x}_n)$. Recall the log-likelihood

$$\mathcal{L} = \sum_{n=1}^{N} \Big[y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n) \Big].$$

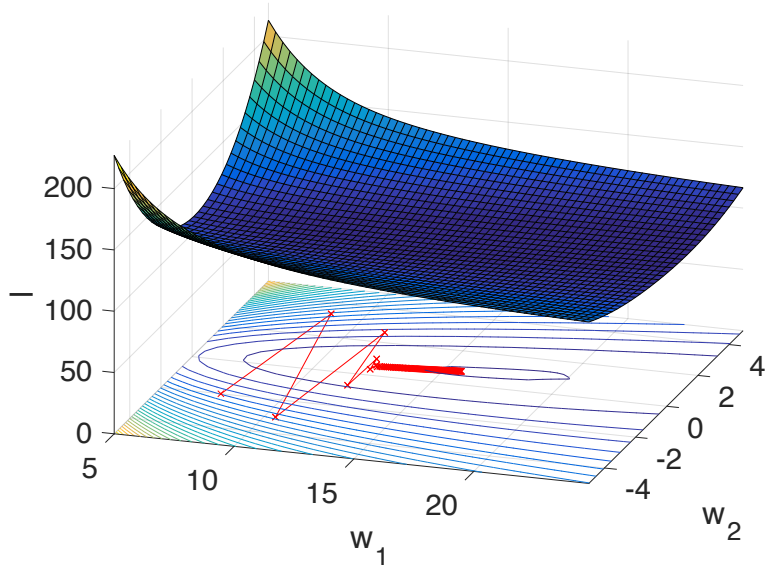
Calculate

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} &= \sum_{n=1}^{N} \left[y_n \frac{\hat{y}_n'}{\hat{y}_n} \mathbf{x}_n + (1 - y_n) \frac{-\hat{y}_n'}{1 - \hat{y}_n} \mathbf{x}_n \right] \\
&= \sum_{n=1}^{N} \left[y_n \frac{\hat{y}_n(1 - \hat{y}_n)}{\hat{y}_n} - (1 - y_n) \frac{\hat{y}_n(1 - \hat{y}_n)}{1 - \hat{y}_n} \right] \mathbf{x}_n \\
&= \sum_{n=1}^{N} \left[y_n(1 - \hat{y}_n) - (1 - y_n)\hat{y}_n \right] \mathbf{x}_n \quad = \sum_{n=1}^{N} (y_n - \hat{y}_n) \mathbf{x}_n
\end{aligned}$$

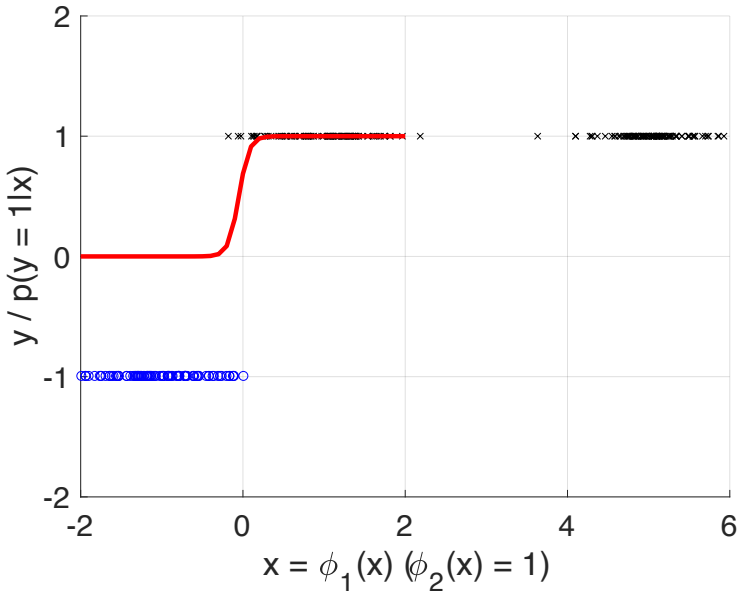
Data


2
1
0
-1
-2
-2
-1
0
1
2
y / p(y = 1|x)
x = ϕ_1(x) (ϕ_2(x) = 1)

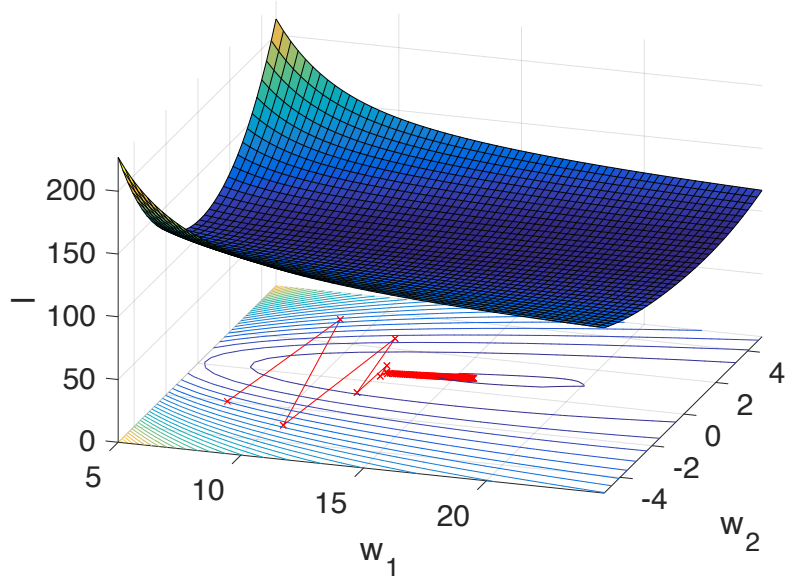
Loss


200
150
100
50
0
l
5
10
15
20
w1
4
2
0
-2
-4
w2

Data


2
1
0
-1
-2
-2
0
2
4
6
y / p(y = 1|x)
x = ϕ_1(x) (ϕ_2(x) = 1)

Loss


200
150
100
50
0
L
5
10
15
20
w_1
4
2
0
-2
-4
w_2

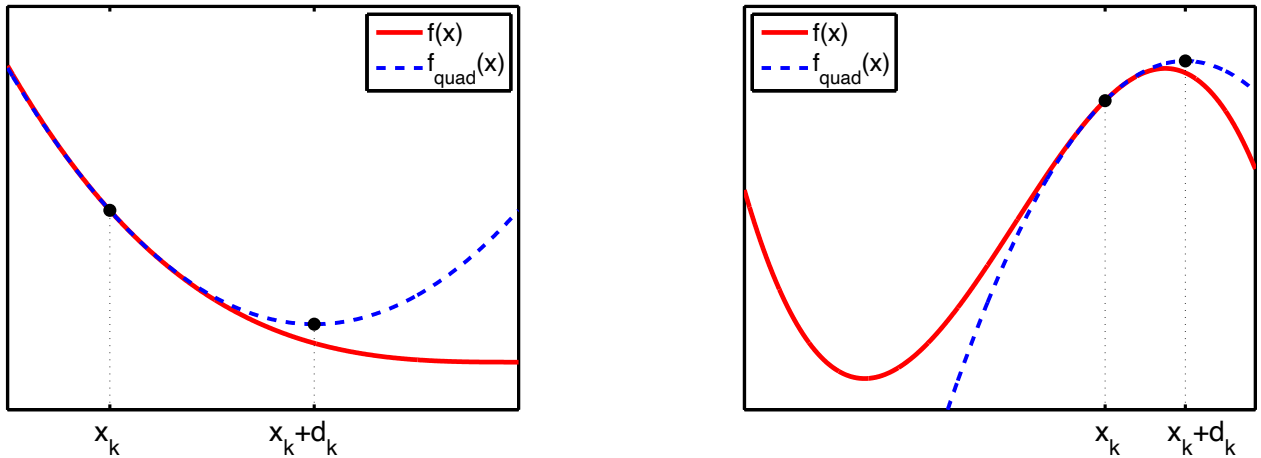
Outline

1. A systemic formulation: logistic regression
   - The optimality of Bayes decision rule
   - Logistic regression and probabilistic model
2. A systemic solver: gradient-based optimization
   - Gradient decent/ascent
   - Newton method

Newton's Method: Second-order Optimization

- *Iterative re-weighted least squares (IRLS)* is a popular algorithm, derived from Newton's method.
- Mathematical Preliminaries
  - Gradient
  - Gradient descent/ascent
  - *Hessian matrix*
  - *Newton's method*

Intuition behind Newton's Method



(a) convex (b) non-convex

[Figure source: Murphy's book]

- $f(x)$ is our objective function (which is expensive to evaluate)
- $f_{\text{quad}}(x)$ is a function approximating $f(x)$ at x_k(relatively easy to approximate)
- We can easily obtain the minimum/maximum from $f_{\text{quad}}(x)$.

Hessian Matrix

- If $f(\mathbf{x})$ is twice differentiable, the Hessian matrix $\mathbf{H}$ is defined as the symmetric matrix with the (i, j) element $\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}$

$$
\begin{aligned}
\mathbf{H} &= \nabla^2 f(\mathbf{x}) \\
&= \left[\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right] \\
&= \begin{bmatrix}
\frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_D} \\
\vdots & & & \vdots \\
\frac{\partial^2 f(\mathbf{x})}{\partial x_D \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_D \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_D^2}
\end{bmatrix} \\
&= \frac{\partial}{\partial \mathbf{x}} \left[\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right]^\top = \frac{\partial}{\partial \mathbf{x}} \left[\nabla f(\mathbf{x})\right]^\top
\end{aligned}
$$

Newton's Method

The basic idea of Newton's method is to optimize the *quadratic approximation* of the objective function $\mathcal{J}(\boldsymbol{w})$ around the current point $\boldsymbol{w}^{(k)}$.
The second-order Taylor series expansion of $\mathcal{J}(\boldsymbol{w})$ at the current point $\boldsymbol{w}^{(k)}$ gives

$$\begin{aligned}\mathcal{J}_2(\boldsymbol{w}) =& \mathcal{J}(\boldsymbol{w}^{(k)}) + \left[\nabla \mathcal{J}(\boldsymbol{w}^{(k)})\right]^\top \left(\boldsymbol{w} - \boldsymbol{w}^{(k)}\right) \\ & \frac{1}{2}\left(\boldsymbol{w} - \boldsymbol{w}^{(k)}\right)^\top \nabla^2 \mathcal{J}(\boldsymbol{w}^{(k)}) \left(\boldsymbol{w} - \boldsymbol{w}^{(k)}\right),\end{aligned}$$

where $\nabla^2 \mathcal{J}(\boldsymbol{w}^{(k)})$ is the Hessian of $\mathcal{J}(\boldsymbol{w})$ evaluated at $\boldsymbol{w} = \boldsymbol{w}^{(k)}$.

Newton's Method

The basic idea of Newton's method is to optimize the *quadratic approximation* of the objective function $\mathcal{J}(\boldsymbol{w})$ around the current point $\boldsymbol{w}^{(k)}$.
The second-order Taylor series expansion of $\mathcal{J}(\boldsymbol{w})$ at the current point $\boldsymbol{w}^{(k)}$ gives

$$\begin{aligned}\mathcal{J}_2(\boldsymbol{w}) =& \mathcal{J}(\boldsymbol{w}^{(k)}) + \left[\nabla \mathcal{J}(\boldsymbol{w}^{(k)})\right]^\top \left(\boldsymbol{w} - \boldsymbol{w}^{(k)}\right) \\ & \frac{1}{2}\left(\boldsymbol{w} - \boldsymbol{w}^{(k)}\right)^\top \nabla^2 \mathcal{J}(\boldsymbol{w}^{(k)}) \left(\boldsymbol{w} - \boldsymbol{w}^{(k)}\right),\end{aligned}$$

where $\nabla^2 \mathcal{J}(\boldsymbol{w}^{(k)})$ is the Hessian of $\mathcal{J}(\boldsymbol{w})$ evaluated at $\boldsymbol{w} = \boldsymbol{w}^{(k)}$.
Differentiate $\mathcal{J}_2(\boldsymbol{w})$ w.r.t. $\boldsymbol{w}$ and set it equal 0, which leads to

$$\nabla \mathcal{J}(\boldsymbol{w}^{(k)}) + \nabla^2 \mathcal{J}(\boldsymbol{w}^{(k)})\boldsymbol{w} - \nabla^2 \mathcal{J}(\boldsymbol{w}^{(k)})\boldsymbol{w}^{(k)} = 0$$

Thus we have

$$\boldsymbol{w} = \boldsymbol{w}^{(k)} - \left[\nabla^2 \mathcal{J}(\boldsymbol{w}^{(k)})\right]^{-1} \nabla \mathcal{J}(\boldsymbol{w}^{(k)}).$$

Hence the Newton's method is of the form

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \left[\nabla^2 \mathcal{J}(\boldsymbol{w}^{(k)})\right]^{-1} \nabla \mathcal{J}(\boldsymbol{w}^{(k)}).$$

Remark

We don't need learning rate!

Remark

The Hessian $\nabla^2 \mathcal{J}(\boldsymbol{w}^{(k)})$ should be positive definite for all k if $\mathcal{J}(\boldsymbol{w}^{(k)})$ is convex.

Taylor Series

The Taylor series represents a function f as an infinite sum of terms.

Definition (Taylor Series)

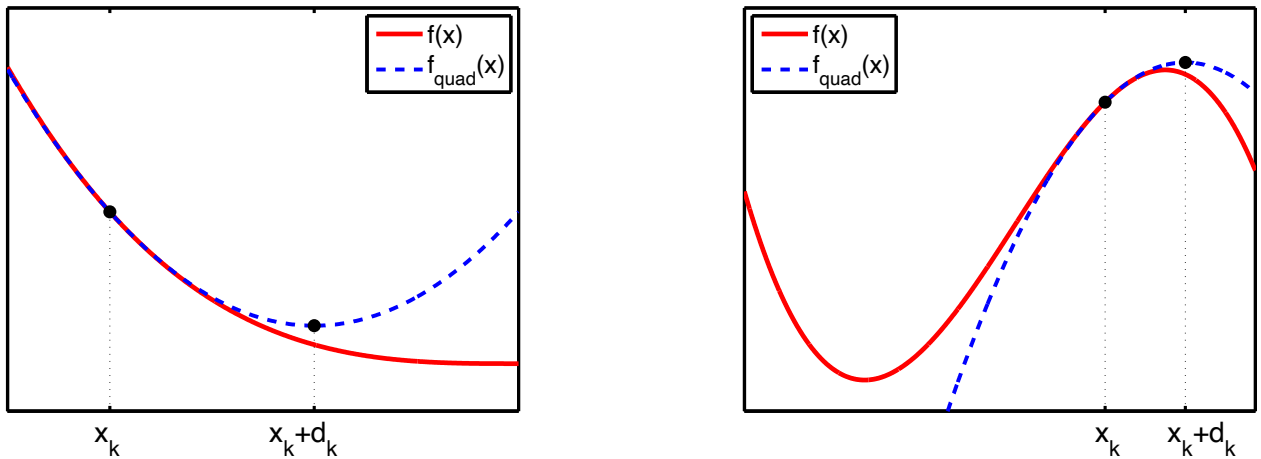
For a smooth function $f \in \mathcal{C}^\infty$, $f : \mathbb{R} \to \mathbb{R}$, the *Taylor series* of f at x_0 is defined as

$$T_\infty(x) := \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k,$$

where $f^{(k)}(x_0)$ is the kth derivative of f at x_0.

Let $T_k(x)$ is the summation up to k-th term.

Illustration of Newton's Method



(a) convex (b) non-convex

[Figure source: Murphy's book]

Newton method requires the second order derivatives + matrix inversion!

Detailed Calculation of Hessian

Calculate the Hessian for logistic regression:

$$
\begin{aligned}
\nabla^2 \mathcal{L} &= \frac{\partial}{\partial \boldsymbol{w}} [\nabla \mathcal{L}]^\top \\
&= \frac{\partial}{\partial \boldsymbol{w}} \left[\sum_{n=1}^{N} (y_n - \hat{y}_n) \mathbf{x}_n^\top \right] \\
&= -\sum_{n=1}^{N} \hat{y}_n (1 - \hat{y}_n) \mathbf{x}_n \mathbf{x}_n^\top
\end{aligned}
$$

We set the objective function $\mathcal{J}(\boldsymbol{w})$ as the negative log-likelihood:

$$\mathcal{J}(\boldsymbol{w}) = -\mathcal{L}(\boldsymbol{w}) = -\sum_{n=1}^{N} \Big[y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n) \Big].$$

Thus, the gradient and the Hessian are:

$$\nabla \mathcal{J}(\boldsymbol{w}) = -\sum_{n=1}^{N} (y_n - \hat{y}_n) \mathbf{x}_n,$$

$$\nabla^2 \mathcal{J}(\boldsymbol{w}) = \sum_{n=1}^{N} \hat{y}_n (1 - \hat{y}_n) \mathbf{x}_n \mathbf{x}_n^\top.$$

One can easily see that the Hessian is positive definite since $\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^\top$ is positive definite and $\hat{y}_n(1 - \hat{y}_n) \geq 0, \forall n$.

Logistic Regression: IRLS

- Newton's update has the form

$$\Delta \boldsymbol{w} = -\underbrace{\left[\sum_{n=1}^{N} \hat{y}_n(1-\hat{y}_n)\mathbf{x}_n\mathbf{x}_n^\top\right]^{-1}}_{\text{inverse of Hessian}} \underbrace{\left[-\sum_{n=1}^{N}(y_n-\hat{y}_n)\mathbf{x}_n\right]}_{\text{gradient}}$$

- Newton's update reduces to iterative re-weighted least squares (IRLS):

$$\Delta \boldsymbol{w} = (\mathbf{X}\mathbf{S}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{S}\mathbf{b},$$

where

$$\mathbf{S} = \begin{bmatrix} \hat{y}_1(1-\hat{y}_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \hat{y}_N(1-\hat{y}_N) \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \frac{y_1-\hat{y}_1}{\hat{y}_1(1-\hat{y}_1)} \\ \vdots \\ \frac{y_1-\hat{y}_N}{\hat{y}_N(1-\hat{y}_N)} \end{bmatrix}$$

Alternatively we write

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + (\mathbf{X}\mathbf{S}_k\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{S}_k\mathbf{b}_k$$

where $\mathbf{S}_k$ and $\mathbf{b}_k$ are computed using $\boldsymbol{w}_k$.
That is,

$$\begin{aligned}
\boldsymbol{w}_{k+1} &= \boldsymbol{w}_k + (\mathbf{X}\mathbf{S}_k\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{S}_k\mathbf{b}_k \\
&= (\mathbf{X}\mathbf{S}_k\mathbf{X}^\top)^{-1}\left[(\mathbf{X}\mathbf{S}_k\mathbf{X}^\top)\boldsymbol{w}_k + \mathbf{X}\mathbf{S}_k\mathbf{b}_k\right] \\
&= (\mathbf{X}\mathbf{S}_k\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{S}_k\left[\mathbf{X}^\top\boldsymbol{w}_k + \mathbf{b}_k\right].
\end{aligned}$$

Algorithm Outline

Algorithm 1 IRLS

Input: $\{(\mathbf{x}_n, y_n) \mid n = 1, \ldots, N\}$

Initialize $\mathbf{w} = \mathbf{0}$ and $w_0 = \log(\overline{y}/(1-\overline{y}))$
repeat
  for $n = 1, \ldots, N$ **do**
    Compute $\eta_n = \mathbf{w}^\top \mathbf{x}_n + w_0$
    Compute $\widehat{y}_n = \sigma(\eta_n)$
    Compute $s_n = \widehat{y}_n(1 - \widehat{y}_n)$
    Compute $z_n = \eta_n + \frac{y_n - \widehat{y}_n}{s_n}$
  end for
  Construct $\mathbf{S} = \mathrm{diag}(s_{1:N})$
  Update $\mathbf{w} = \left(\mathbf{X}\mathbf{S}\mathbf{X}^\top\right)^{-1}\mathbf{X}\mathbf{S}\mathbf{z}$
until convergence
return w

Summary

Classification

- Issues when using linear regression for classification
- A systemic formulation: logistic regression
  - The optimality of Bayes decision rule
  - Logistic regression and probabilistic model
- A systemic solver: gradient-based optimization
  - Gradient decent/ascent
  - Newton method (IRLS for logistic regression)

Multiclass Logistic Regression

- Model input-output by a softmax transformation of linear functions of input features:

$$p(y_n = k \mid x_n) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{j=1}^K \exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}$$

- The log-likelihood is given by

$$\sum_{n=1}^N \sum_{k=1}^K y_{kn} \log p(y_n = k \mid x_n)$$

where $y_{kn} = \mathbb{1}[y_n = k]$.

Further Readings

- Textbook: Chapter 10 (Logistic Regression)
- Textbook: Chapter 8.3 (Optimization: Second-order methods)