Logistic regression and APG

DSA5103 Lecture 3

Yangjing Zhang

26-Jan-2023

NUS

Today's content

1. Logistic regression for classification
2. Ridge/lasso regularization
3. (Accelerated) proximal gradient method

Logistic regression

Classification

Binary classification:

- Email: spam/not spam
- Patient: cancer/healthy
- Student: fail/pass

We usually assign

$$\text{label} \begin{cases} 0, & \text{normal state/negative class e.g., not spam} \\ 1, & \text{abnormal state/positive class e.g., spam} \end{cases}$$

However, the label assignment can be arbitrary:

$0 =$ not spam, $1 =$ spam or $0 =$ spam, $1 =$ not spam

Data $x_i \in \mathbb{R}^p, y_i \in \{0, 1\}$, $i = 1, 2, \ldots, n$.

Classification

Multi-class classification:

- Iris flower (3 species: Setosa, Versicolor, Virginica)
- Optical character recognition

Data $x_i \in \mathbb{R}^p, y_i \in \{1, \ldots, K\}$, $i = 1, 2, \ldots, n$.

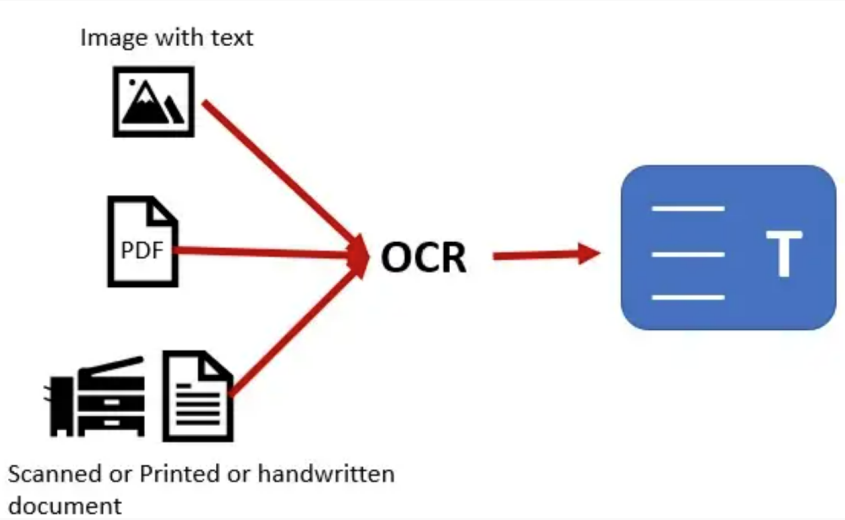


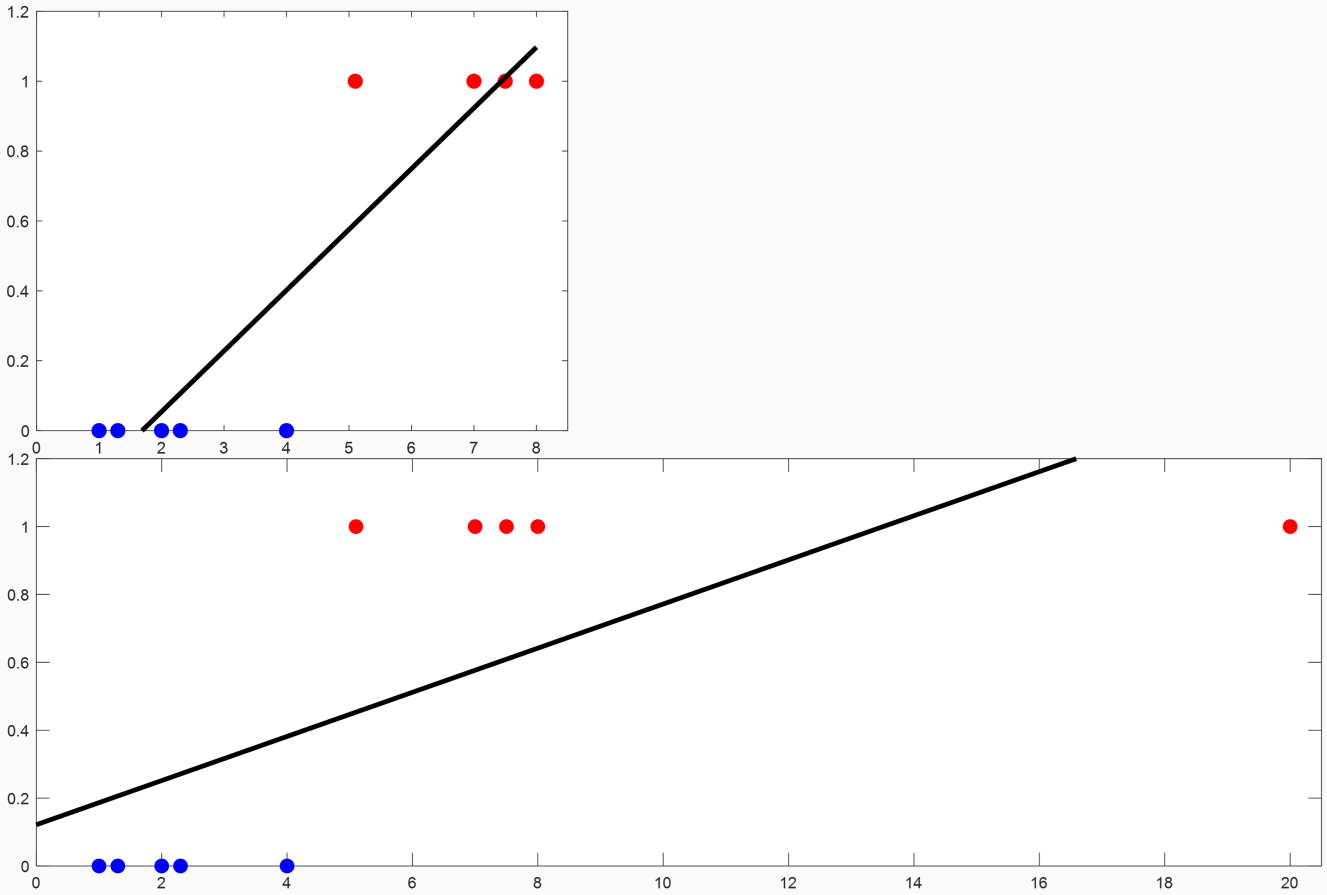
Figure 1: Convert images of text to machine-readable format. Image from internet

Linear regression for classification?

Data x_i, $y_i \in \{0, 1\}$

We fit $f(x) = \beta_0 + \beta^T x$, and predict a new input $\tilde{x}$ belong to

class 1 if $f(\tilde{x}) \geq 0.5$ class 0 if $f(\tilde{x}) < 0.5$



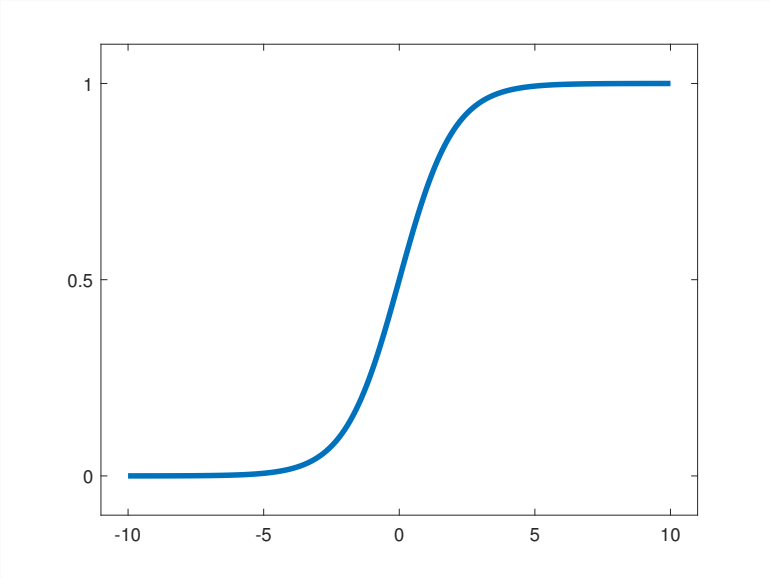
Linear regression vs. logistic regression

Linear regression

- Data $x_i,\ y_i \in \mathbb{R}$
- Fit $f(x) = \beta^T x + \beta_0 = \hat{\beta}^T \hat{x},\ \hat{\beta} = [\beta_0; \beta],\ \hat{x} = [1; x]$

Logistic regression

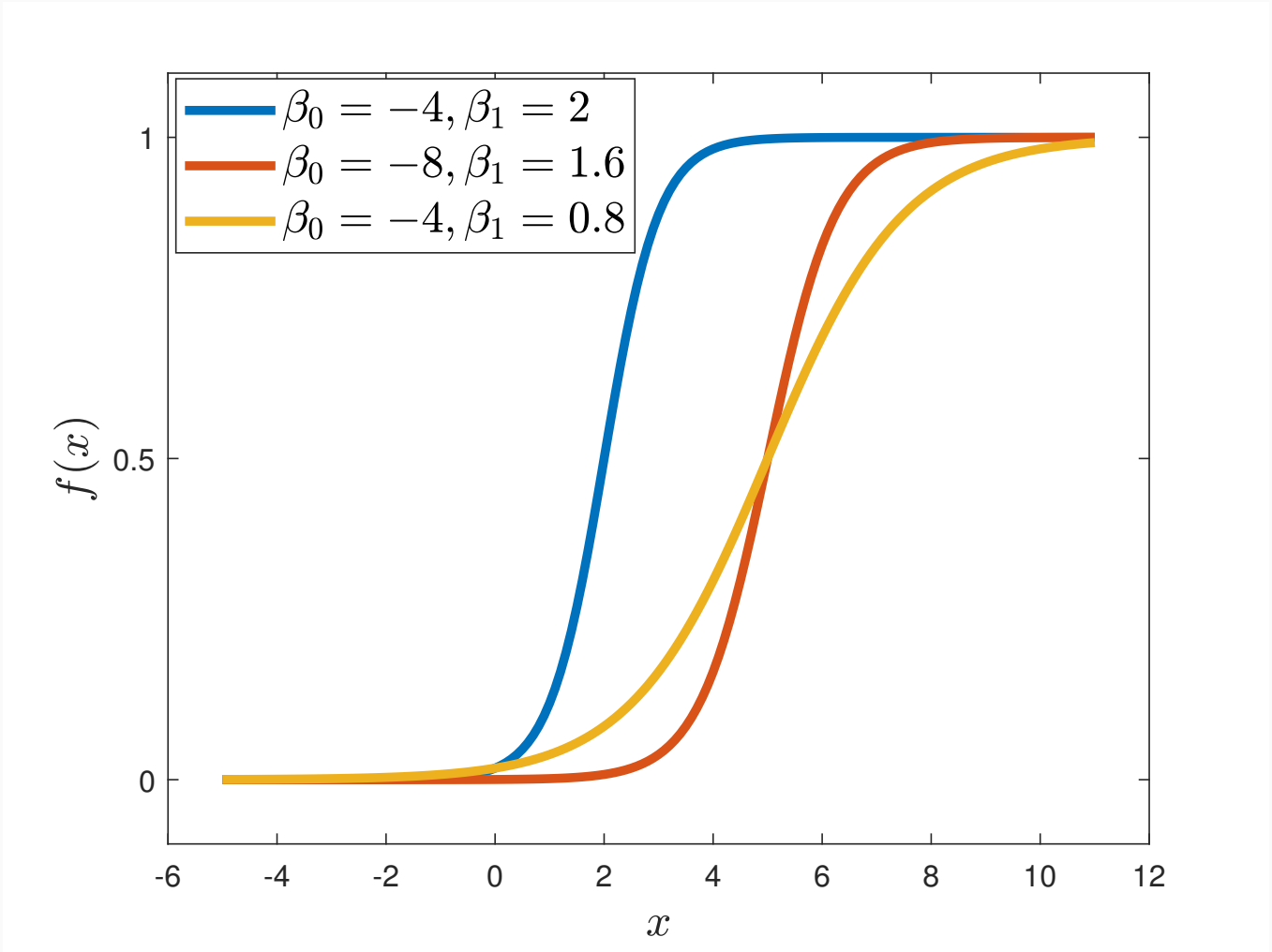
- Data $x_i,\ y_i \in \{0, 1\}$
- Fit $f(x) = g(\hat{\beta}^T \hat{x}),\ g(z) = \dfrac{1}{1 + e^{-z}}$ sigmoid/logistic function



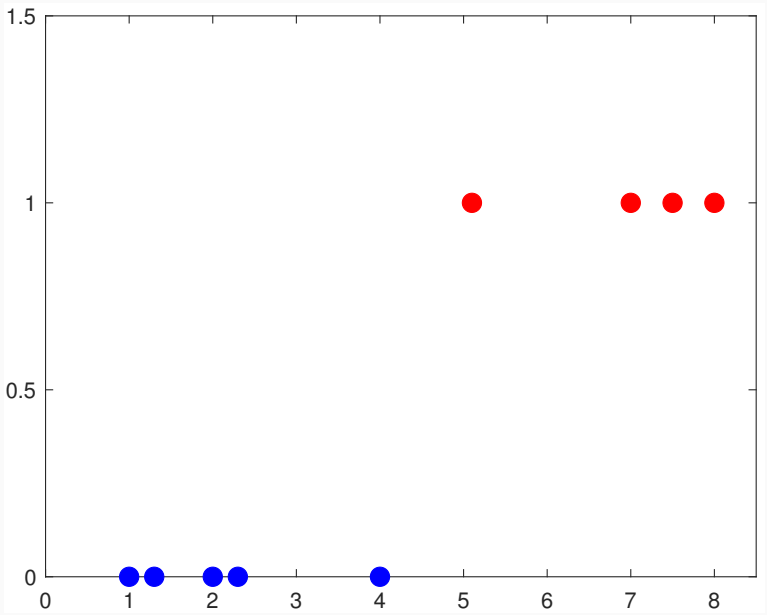
- $0 < g(z) < 1$ an increasing function
- $g(0.5) = 0.5$
- $g(z) \to 1$ as $z \to +\infty$
- $g(z) \to 0$ as $z \to -\infty$

Graph illustration

β_0, β_1 will change the shape and location of the function



Probabilistic interpretation



In logistic regression, we interpret

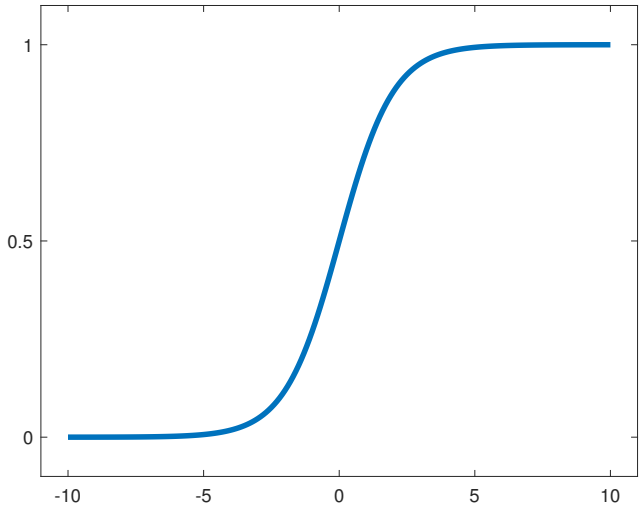
$$f(x) = \text{probability(input } x \in \text{class } 1)$$
$$= p(y = 1|x; \hat{\beta})$$

Then $1 - f(x) = \text{probability(input } x \in \text{class } 0) = p(y = 0|x; \hat{\beta})$

Logistic regression

$$f(x) = g(\hat{\beta}^T \hat{x}), \quad g(z) = \frac{1}{1+e^{-z}}$$

$$f(x) = \text{probability(input } x \in \text{class } 1)$$
$$= p(y = 1 | x; \hat{\beta})$$



Predict $y = 1$ ($x \in$ class 1) if

- $f(x) \geq 0.5$, i.e., $\hat{\beta}^T \hat{x} \geq 0$

Predict $y = 0$ ($x \in$ class 0) if

- $f(x) < 0.5$, i.e., $\hat{\beta}^T \hat{x} < 0$

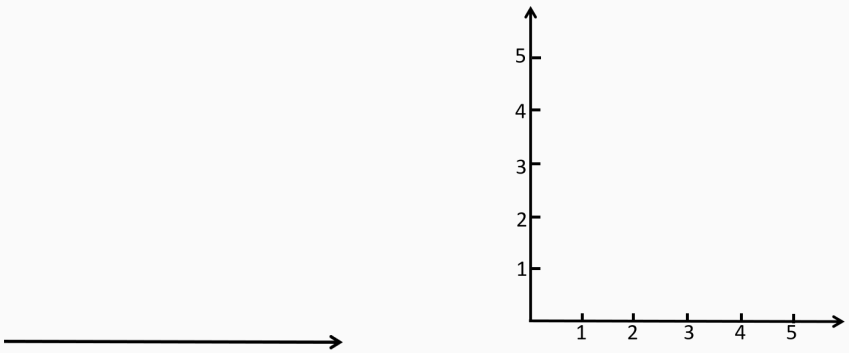
Example

(1) (One feature)

Say $\hat{\beta} = [\beta_0; \beta_1] = [-4; 2]$. Then $f(x) = g(\beta_0 + \beta_1 x)$.

Predict $y = 1$ if

$$\beta_0 + \beta_1 x = -4 + 2x \geq 0, \text{ i.e., } x \geq 2$$



(2) (Two features)

Say $\hat{\beta} = [\beta_0; \beta_1; \beta_2] = [-4; 2; 1]$. Then $f(x) = g(\beta_0 + \beta_1 x_1 + \beta_2 x_2)$.

Predict $y = 1$ if

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = -4 + 2x_1 + x_2 \geq 0, \text{ i.e., } 2x_1 + x_2 \geq 4$$

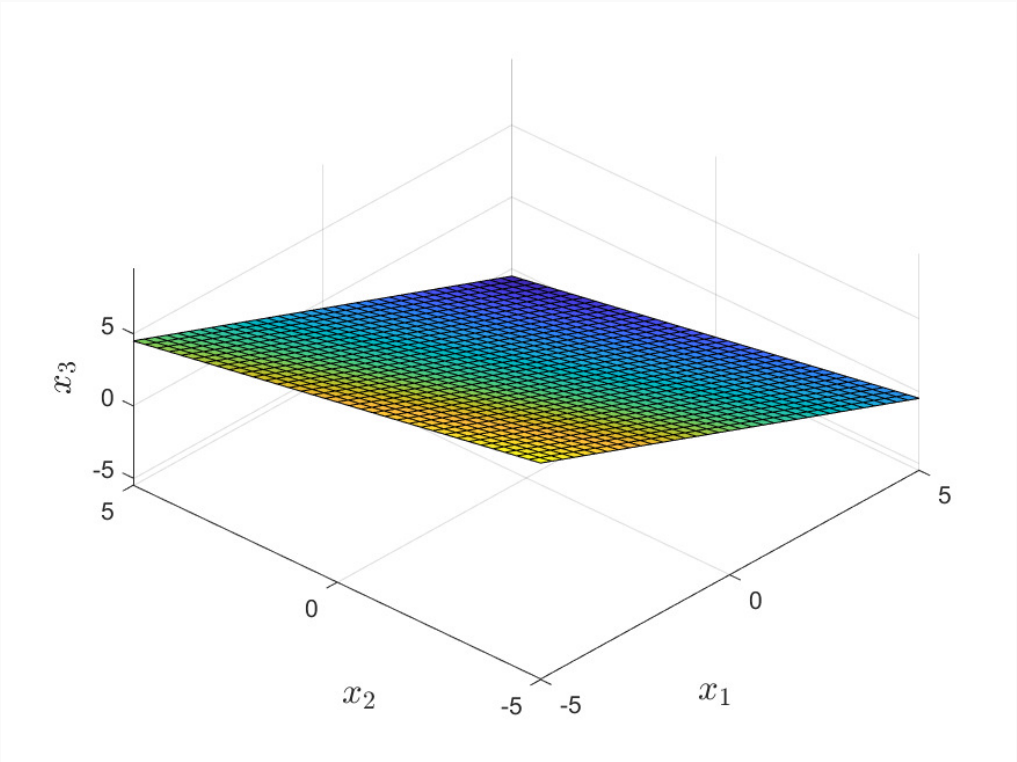
Example

(3) (Three features)

Say $\hat{\beta} = [\beta_0; \beta_1; \beta_2; \beta_3] = [-4; 2; 1; 2]$. Then $f(x) = g(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3)$.

Predict $y = 1$ if

$$\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_3 x_3 = -4+2x_1+x_2+2x_3 \geq 0, \text{ i.e., } 2x_1+x_2+2x_3 \geq 4$$



Decision boundary

The set of all $x \in \mathbb{R}^p$ such that

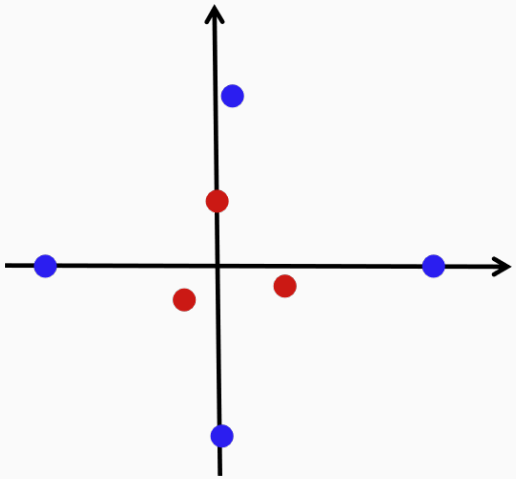
$$\beta_0 + \beta^T x = 0$$

is called the decision boundary between classes 0 and 1.

The logistic regression has a linear decision boundary; it is

- a point when $p = 1$
- a line when $p = 2$
- a plan when $p = 3$
- in general a $(p-1)$-dimensional subpace

Feature expansion



x_{i1} x_{i2}

x_{i1} x_{i2} x_{i1}^2 x_{i2}^2 $x_{i1}x_{i2}$

Say $\hat{\beta} = [\beta_0; \beta_1; \ldots; \beta_5] = [-4; 0; 0; 1; 1; 0]$. The decision boundary is

$$-4 + x_1^2 + x_2^2 = 0$$

Maximum likelihood estimation

- Data (x_i, y_i), $i = 1, 2, \ldots, n$, $x_i \in \mathbb{R}^p$, $y_i \in \{0, 1\}$.
- The likelihood of a single training example (x_i, y_i) is

$$\text{probability}(x_i \in \text{class } y_i)$$

$$= \begin{cases} p(y_i = 1 | x_i; \hat{\beta}) = f(x_i), & \text{if } y_i = 1 \\ 1 - p(y_i = 1 | x_i; \hat{\beta}) = 1 - f(x_i), & \text{if } y_i = 0 \end{cases}$$

$$= \Big[f(x_i) \Big]^{y_i} \Big[1 - f(x_i) \Big]^{1 - y_i}$$

- Hope the likelihood is close to 1 for every training example (x_i, y_i)

Maximum likelihood estimation

- Assume independence of the training samples, the likelihood is

$$\prod_{i=1}^{n} \Big[f(x_i) \Big]^{y_i} \Big[1 - f(x_i) \Big]^{1-y_i}$$

- Want to find $\hat{\beta}$ to maximize the log-likelihood. Note that maximizing a (positive) function is the same as maximizing the log of a function (because log is monotone increasing)

Maximum likelihood estimation

If $f(x) > 0$ for all x,

$$\max_x f(x) \iff \max_x \log(f(x))$$

since log is monotone increasing, i.e., $\log(x) \geq \log(y)$ if $x \geq y > 0$

$$\begin{aligned} & x^* \text{ is a global maximizer of } f(\cdot) \\ \iff & f(x^*) \geq f(x) \, \forall \, x \\ \iff & \log(f(x^*)) \geq \log(f(x)) \, \forall \, x \\ \iff & x^* \text{ is a global maximizer of } \log(f(\cdot)) \end{aligned}$$

Similar arguments work for local minimizer.

Note that the likelihood is always positive and we can take log.

Cost function

Cost function[1]

$$L(\hat{\beta}) = -\log\left(\prod_{i=1}^{n}\Big[f(x_i)\Big]^{y_i}\Big[1-f(x_i)\Big]^{1-y_i}\right)$$
$$= -\sum_{i=1}^{n} y_i \log(f(x_i)) + (1-y_i)\log(1-f(x_i))$$

For a particular training example (x_i, y_i), the cost is

$$-y_i \log(f(x_i)) - (1-y_i)\log(1-f(x_i))$$
$$= \begin{cases} -\log(f(x_i)), & \text{if } y_i = 1 \\ -\log(1-f(x_i)), & \text{if } y_i = 0 \end{cases}$$

[1] We use natural logarithms in logistic regression

Understand the cost function

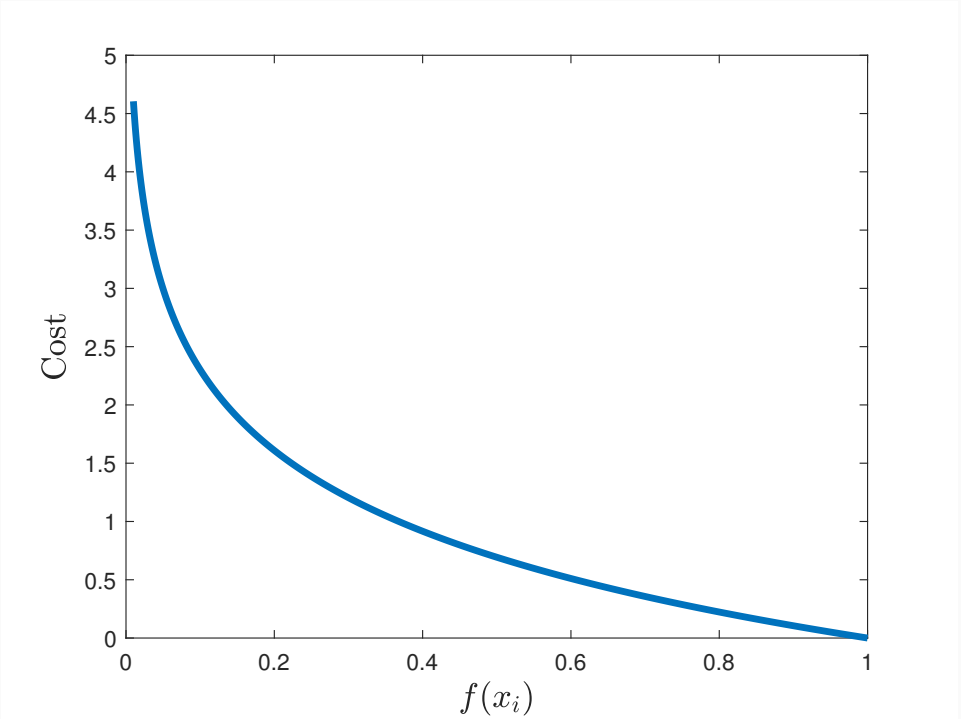
cost $= -\log(f(x_i))$, when $y_i = 1$

- $f(x_i) = 1$, cost $= 0$ ("perfect" scenario $\Rightarrow$ zero cost)
- $f(x_i) = 0.9$, cost $= 0.11$ ("good" scenario $\Rightarrow$ small cost)
- $f(x_i) = 0.1$, cost $= 2.3$ ("bad" scenario $\Rightarrow$ large cost)

Understand the cost function

cost $= -\log(f(x_i))$, when $y_i = 1$

- $f(x_i) = 1$, cost $= 0$ ("perfect" scenario $\Rightarrow$ zero cost)
- $f(x_i) = 0.9$, cost $= 0.11$ ("good" scenario $\Rightarrow$ small cost)
- $f(x_i) = 0.1$, cost $= 2.3$ ("bad" scenario $\Rightarrow$ large cost)



Understand the cost function

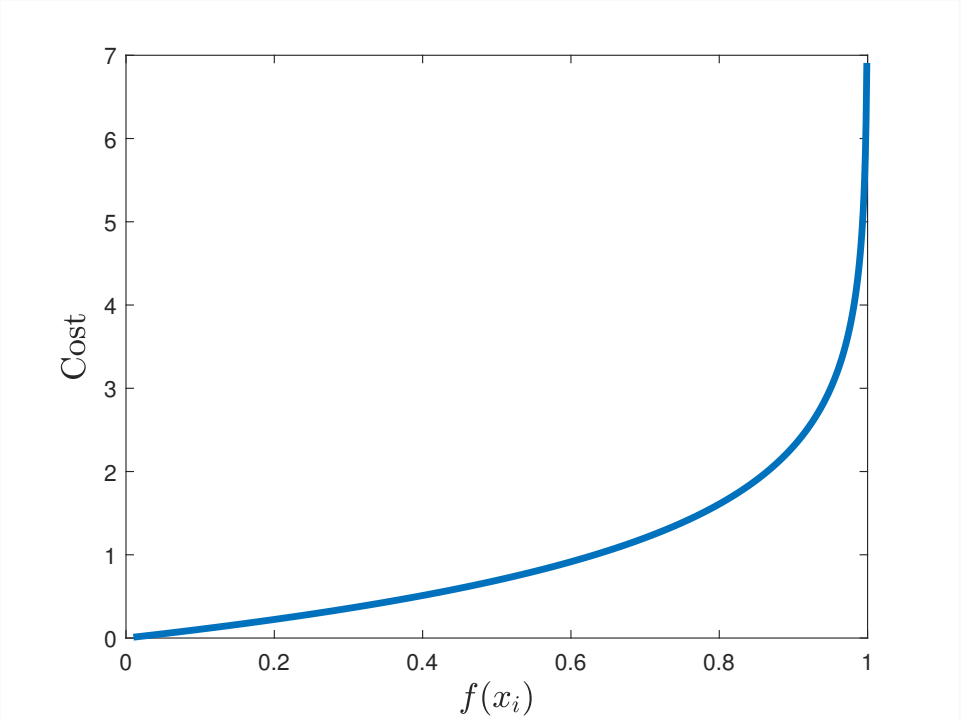
cost $= -\log(1 - f(x_i))$, when $y_i = 0$

- $f(x_i) = 0$, cost $= 0$ ("perfect" scenario $\Rightarrow$ zero cost)
- $f(x_i) = 0.1$, cost $= 0.11$ ("good" scenario $\Rightarrow$ small cost)
- $f(x_i) = 0.9$, cost $= 2.3$ ("bad" scenario $\Rightarrow$ large cost)

Understand the cost function

cost $= -\log(1 - f(x_i))$, when $y_i = 0$

- $f(x_i) = 0$, cost $= 0$ ("perfect" scenario $\Rightarrow$ zero cost)
- $f(x_i) = 0.1$, cost $= 0.11$ ("good" scenario $\Rightarrow$ small cost)
- $f(x_i) = 0.9$, cost $= 2.3$ ("bad" scenario $\Rightarrow$ large cost)



Simply the cost function

$$L(\hat{\beta}) = L(\beta_0, \beta) \stackrel{*}{=} \sum_{i=1}^{n} \log(1 + e^{\beta_0 + \beta^T x_i}) - y_i(\beta_0 + \beta^T x_i)$$

Derivation*. Recall that $f(x) = \frac{1}{1+e^{-\hat{\beta}^T \hat{x}}}$ and

$$\begin{aligned} L(\hat{\beta}) &= -\sum_{i=1}^{n} y_i \log(f(x_i)) + (1 - y_i) \log(1 - f(x_i)) \\ &= -\sum_{i=1}^{n} y_i \log\left(\frac{f(x_i)}{1 - f(x_i)}\right) + \log(1 - f(x_i)) \end{aligned}$$

It remains prove two equalities:

1. $\log\left(\frac{f(x_i)}{1-f(x_i)}\right) = \hat{\beta}^T \hat{x}_i$
2. $\log(1 - f(x_i)) = -\log(1 + e^{-\hat{\beta}^T \hat{x}_i})$

Simply the cost function

1. $$\log\left(\frac{f(x_i)}{1-f(x_i)}\right) = \log\left(\frac{\frac{1}{1+e^{-\hat{\beta}^T\hat{x}_i}}}{1-\frac{1}{1+e^{-\hat{\beta}^T\hat{x}_i}}}\right) = \log\left(\frac{1}{1+e^{-\hat{\beta}^T\hat{x}_i}-1}\right)$$
$$= \log(e^{\hat{\beta}^T\hat{x}_i}) = \hat{\beta}^T\hat{x}_i$$

2. $$\log(1-f(x_i)) = \log\left(1-\frac{1}{1+e^{-\hat{\beta}^T\hat{x}_i}}\right) = \log\left(\frac{1+e^{-\hat{\beta}^T\hat{x}_i}-1}{1+e^{-\hat{\beta}^T\hat{x}_i}}\right)$$
$$= \log\left(\frac{1}{1+e^{\hat{\beta}^T\hat{x}_i}}\right) = -\log\left(1+e^{\hat{\beta}^T\hat{x}_i}\right)$$

Gradient of the cost function

- Cost function

$$L(\beta_0, \beta_1, \ldots, \beta_p) = \sum_{i=1}^{n} \log(1 + e^{\beta_0 + \beta^T x_i}) - y_i(\beta_0 + \underbrace{\beta^T x_i}_{\| \; \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}})$$

Gradient of the cost function

- Cost function

$$L(\beta_0, \beta_1, \ldots, \beta_p) = \sum_{i=1}^{n} \log(1 + e^{\beta_0 + \beta^T x_i}) - y_i(\beta_0 + \underbrace{\beta^T x_i}_{\| \; \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}})$$

- Calculate

$$\frac{\partial}{\partial \beta_0} L = \sum_{i=1}^{n} \left(\frac{1}{1 + e^{-(\beta_0 + \beta^T x_i)}} - y_i \right) = \sum_{i=1}^{n} (f(x_i) - y_i)$$

$$\frac{\partial}{\partial \beta_1} L = \sum_{i=1}^{n} \left(\frac{1}{1 + e^{-(\beta_0 + \beta^T x_i)}} - y_i \right) x_{i1} = \sum_{i=1}^{n} (f(x_i) - y_i)\, x_{i1}$$

$$\frac{\partial}{\partial \beta_2} L = \sum_{i=1}^{n} \left(\frac{1}{1 + e^{-(\beta_0 + \beta^T x_i)}} - y_i \right) x_{i2} = \sum_{i=1}^{n} (f(x_i) - y_i)\, x_{i2}$$

$$\vdots$$

$$\frac{\partial}{\partial \beta_p} L = \sum_{i=1}^{n} \left(\frac{1}{1 + e^{-(\beta_0 + \beta^T x_i)}} - y_i \right) x_{ip} = \sum_{i=1}^{n} (f(x_i) - y_i)\, x_{ip}$$

Linear regression vs. logistic regression

Linear regression

$$L = \frac{1}{2}\sum_{i=1}^{n}(\beta^T x_i + \beta_0 - y_i)^2$$

Gradient

$$\frac{\partial}{\partial \beta_0} L = \sum_{i=1}^{n}(\beta^T x_i + \beta_0 - y_i)$$

$$= \sum_{i=1}^{n}(f(x_i) - y_i)$$

$$\frac{\partial}{\partial \beta_j} L = \sum_{i=1}^{n}(\beta^T x_i + \beta_0 - y_i)x_{ij}$$

$$= \sum_{i=1}^{n}(f(x_i) - y_i)x_{ij}$$

for $j = 1, 2, \ldots, p$

Logistic regression $L =$

$$\sum_{i=1}^{n}\log(1 + e^{\beta_0 + \beta^T x_i}) - y_i(\beta_0 + \beta^T x_i)$$

Gradient

$$\frac{\partial}{\partial \beta_0} L = \sum_{i=1}^{n}(\frac{1}{1 + e^{-(\beta_0 + \beta^T x_i)}} - y_i)$$

$$= \sum_{i=1}^{n}(f(x_i) - y_i)$$

$$\frac{\partial}{\partial \beta_j} L = \sum_{i=1}^{n}(\frac{1}{1 + e^{-(\beta_0 + \beta^T x_i)}} - y_i)x_{ij}$$

$$= \sum_{i=1}^{n}(f(x_i) - y_i)x_{ij}$$

for $j = 1, 2, \ldots, p$

Solution may not exist

The solution (global minimizer) of the minimization problem

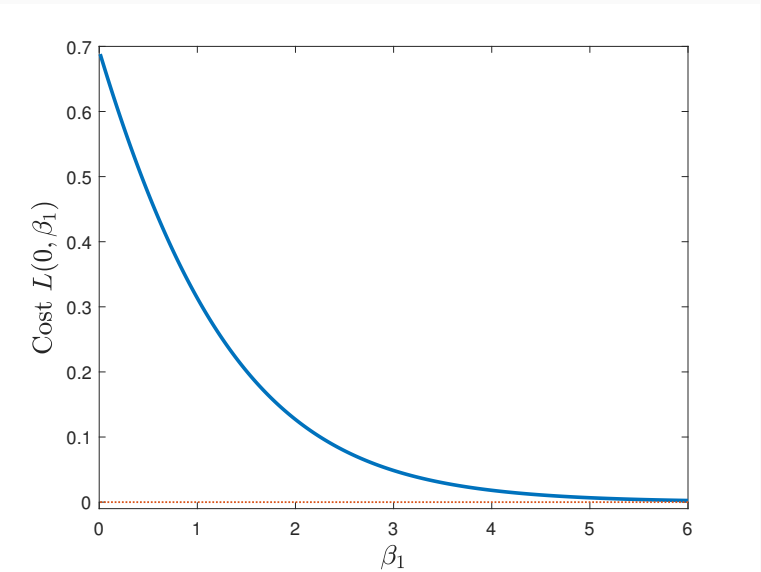
$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \quad \sum_{i=1}^{n} \log(1 + e^{\beta_0+\beta^T x_i}) - y_i(\beta_0 + \beta^T x_i)$$

may not exist. (Regularization will help solve this issue)

Example. $n = 1$, $x_1 = -1$, $y_1 = 0$. Then the cost function

$$L(\beta_0, \beta_1) = \log(1 + e^{\beta_0 - \beta_1})$$

We can see that $\min L = 0$. However this value cannot be attained.



Multi-class classification: one-vs-rest

Idea: transfer multi-class classification to multiple binary classification problems

Data: $x_i \in \mathbb{R}^p, y_i \in \{1, 2, \ldots, K\}$, $i = 1, 2, \ldots, n$.

For each $k \in \{1, 2, \ldots, K\}$

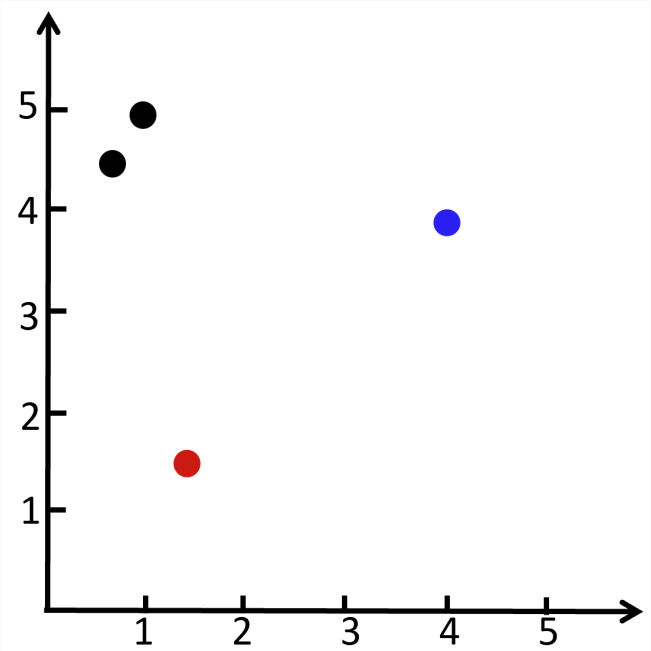
1. Construct a new label $\tilde{y}_i = 1$ if $y_i = k$ and $\tilde{y}_i = 0$ otherwise
2. Learn a binary classifier f_k with data $x_i, \tilde{y}_i$

Multi-class classifier predicts class k where k achieves the maximal value

$$\max_{k \in \{1,2,\ldots,K\}} f_k(x)$$

Multi-class classification: one-vs-rest

feature 1	feature 2	label y
1	5	1
0.8	4.5	1
1.5	1.5	2
4	4	3



Say for a new input $\tilde{x}$, we have $f_1(\tilde{x}) = 0.8$, $f_2(\tilde{x}) = 0.1$, $f_3(\tilde{x}) = 0.6$. Then we say

$\tilde{x}$ belongs to class 1 with probability 80%

$\tilde{x}$ belongs to class 2 with probability 10%

$\tilde{x}$ belongs to class 3 with probability 60%

and we predict it belongs to class 1.

Ridge/lasso regularization

Over-fitting

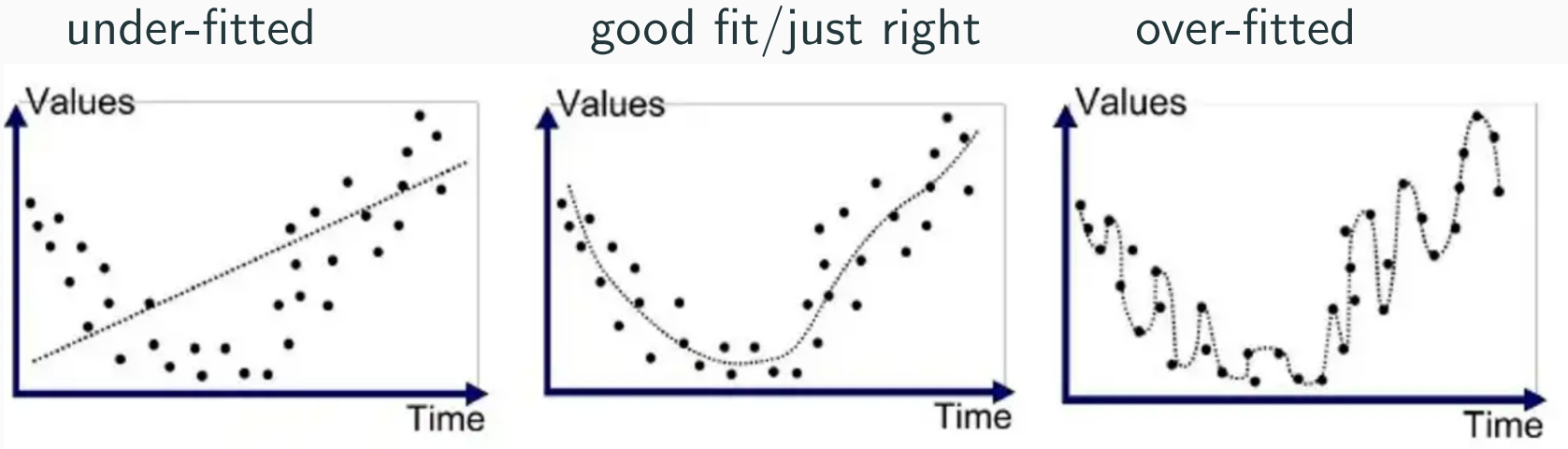


Figure 2: Image from internet

- Under-fitting: a model is too simple and does not adequately capture the underlying structure of the data
- Over-fitting: a model is too complicated and contains more parameters that can be justified by the data; it does not generalize well from training data to test data
- Good fit: a model adequately learns the training data and generalizes well to test data

Ridge regularization

In linear/logistic regression, over-fitting occurs frequently. Regularization will make the model simpler and works well for most of the regression/classification problems.

- Ridge regularization:

$$\lambda \|\beta\|^2 = \lambda \sum_{j=1}^{p} \beta_j^2$$

  λ: regularization parameter, $\|\beta\|^2$: regularizer

- It is differentiable. It forces β_j's to be small
- Extreme case: suppose λ is a huge number, it will push all β_j's to be zero and the model will be naive

Ridge regularized problems

- Logistic regression + ridge regularization (Gradient methods can be used, a solution exists)

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \quad \sum_{i=1}^{n} \log(1 + e^{\beta_0+\beta^T x_i}) - y_i(\beta_0 + \beta^T x_i) + \lambda \sum_{j=1}^{p} \beta_j^2$$

- Linear regression + ridge regularization (Apply either normal equation or gradient methods)

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \quad \frac{1}{2} \sum_{i=1}^{n} (\beta^T x_i + \beta_0 - y_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Normal equation for ridge regularized linear regression

$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

For simplicity, we assume[2] $\beta_0 = 0$

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{n}(\beta^T x_i - y_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \frac{1}{2}\|X\beta - Y\|^2 + \lambda\|\beta\|^2$$

Compute the gradient

$$\text{Gradient} = X^T(X\beta - Y) + 2\lambda\beta$$

Normal equation (setting gradient to be zero)

$$(2\lambda I + X^T X)\beta = X^T Y$$

$$\Rightarrow \quad \beta = (2\lambda I + X^T X)^{-1} X^T Y$$

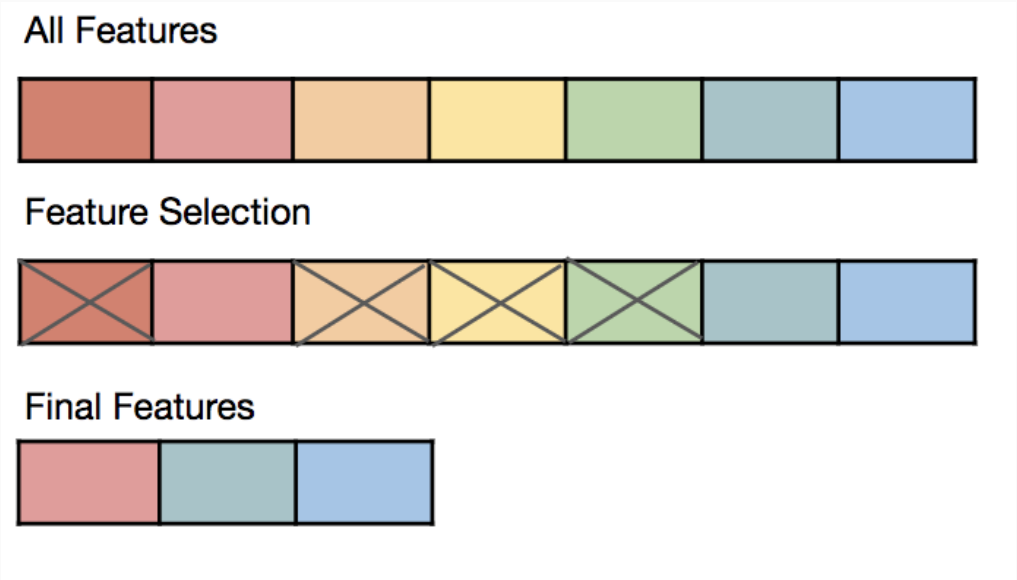
[2]Note that the intercept should be zero $\beta_0 = 0$ if the data is standardized

Lasso regularization

- Lasso (**L**east **A**bsolute **S**hrinkage and **S**election **O**perator) regularization:

$$\lambda \|\beta\|_1 = \lambda \sum_{j=1}^{p} |\beta_j|$$

- It is non-differentiable. It forces some β_j's to be exactly zero
- It can be used for feature selection (model selection). It selects important features (removing non-informative or redundant features)
- When λ is larger, less features will be selected



Lasso regularized problems

- Logistic regression + lasso regularization (Gradient methods is no longer applicable)

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \quad \sum_{i=1}^{n} \log(1+e^{\beta_0+\beta^T x_i}) - y_i(\beta_0+\beta^T x_i) + \lambda \sum_{j=1}^{p} |\beta_j|$$

- Linear regression + lasso regularization (Gradient methods is no longer applicable)

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{n} (\beta^T x_i + \beta_0 - y_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

- In the following, we always assume $\beta_0 = 0$. Note that the intercept should be zero $\beta_0 = 0$ if the data is standardized.

Lasso regularized problems

- Logistic regression + lasso regularization (Gradient methods is no longer applicable)

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \quad \sum_{i=1}^{n} \log(1+e^{\beta_0+\beta^T x_i}) - y_i(\beta_0+\beta^T x_i) + \lambda \sum_{j=1}^{p} |\beta_j|$$

- Linear regression + lasso regularization (Gradient methods is no longer applicable)

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{n}(\beta^T x_i + \beta_0 - y_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

- In the following, we always assume $\beta_0 = 0$. Note that the intercept should be zero $\beta_0 = 0$ if the data is standardized.

Given feature matrix $X \in \mathbb{R}^{n\times p}$ and response vector $Y \in \mathbb{R}^p$, the famous lasso problem [3]

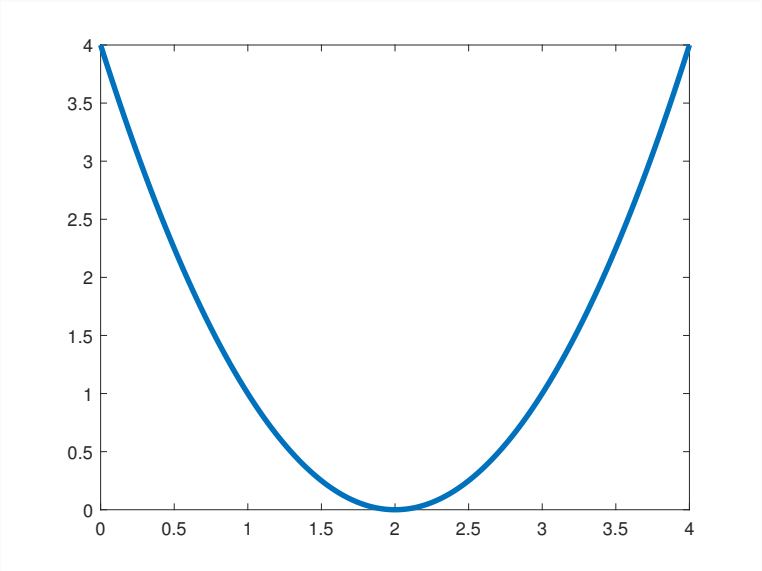
$$\boxed{\underset{\beta\in\mathbb{R}^p}{\text{minimize}} \quad \frac{1}{2}\|X\beta - Y\|^2 + \lambda\|\beta\|_1}$$

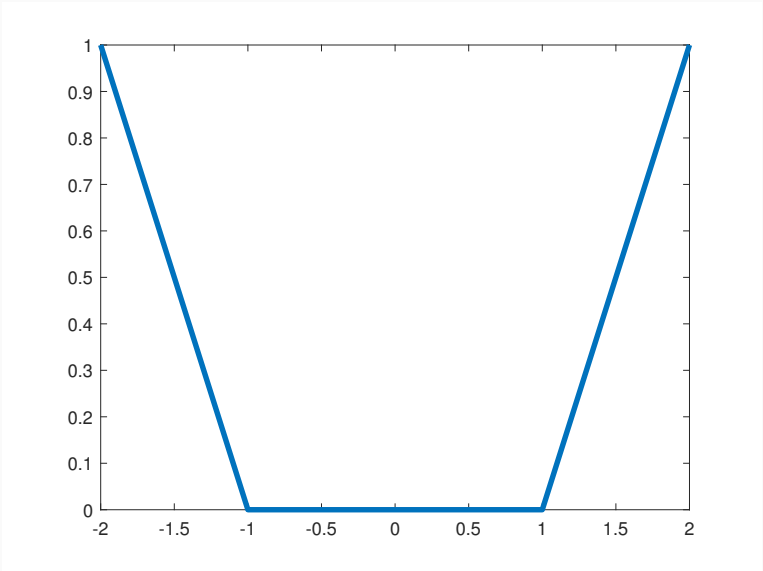
(Accelerated) proximal gradient method

Notation

$\langle x, y \rangle = x^T y$, for two vectors $x, y \in \mathbb{R}^p$

$\arg\min_x f(x)$ denotes the solution set of x for which $f(x)$ attains its minimum (argument of the minimum)





Left: $\min_x f(x) = 0$, $\arg\min_x f(x) = \{2\}$

Right: $\min_x f(x) = 0$, $\arg\min_x f(x) = [-1, 1]$

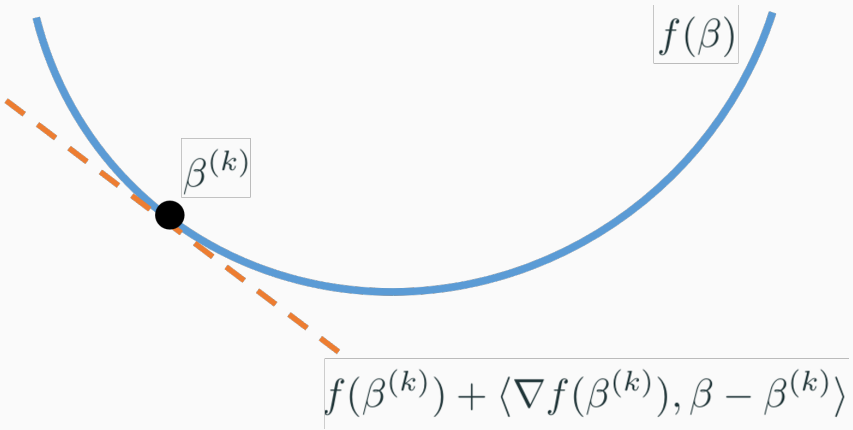
A proximal point view of gradient methods

To minimize a differentiable function $\min_\beta f(\beta)$

$$\beta^{(k+1)} = \beta^{(k)} - \alpha_k \nabla f(\beta^{(k)})$$

The gradient step can be written equivalently as

$$\beta^{(k+1)} = \arg\min_\beta \{ \underbrace{f(\beta^{(k)}) + \langle \nabla f(\beta^{(k)}), \beta - \beta^{(k)} \rangle}_{\text{linear approximation}} + \underbrace{\frac{1}{2\alpha_k} \|\beta - \beta^{(k)}\|^2}_{\text{proximal term}} \}$$



Moreau envelope and proximal mapping

For any[3] convex function f, we define

- Moreau envelope (Moreau-Yosida regularization) of f at x

$$M_f(x) = \min_y \left\{ f(y) + \frac{1}{2}\|y - x\|^2 \right\}$$

- Proximal mapping of f at x

$$P_f(x) = \arg\min_y \left\{ f(y) + \frac{1}{2}\|y - x\|^2 \right\}$$

- $M_f(x)$ is differentiable (Moreau envelope is a way to smooth a possibly non-differentiable convex function)
- $P_f(x)$ exists and is unique
- $M_f(x) \leq f(x)$
- $\arg\min f(x) = \arg\min M_f(x)$

[3] To be precise $f : \mathbb{R}^n \to (-\infty, \infty]$ should be a closed proper convex function

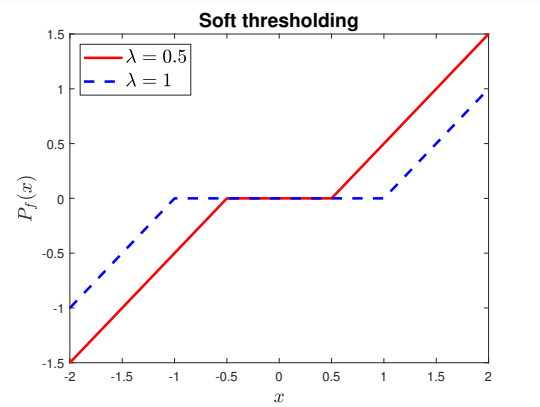
Example

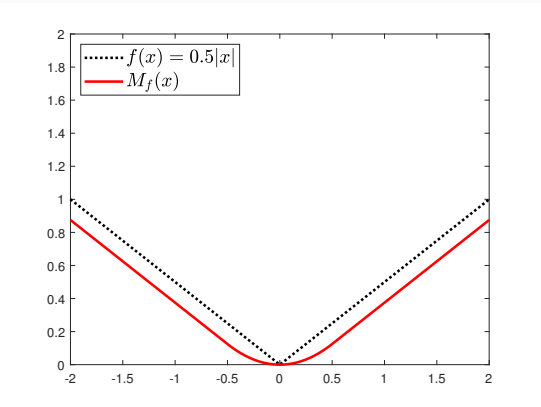
Let $f(x) = \lambda|x|$, $x \in \mathbb{R}$. Then its Moreau envelope is (known as Huber function)

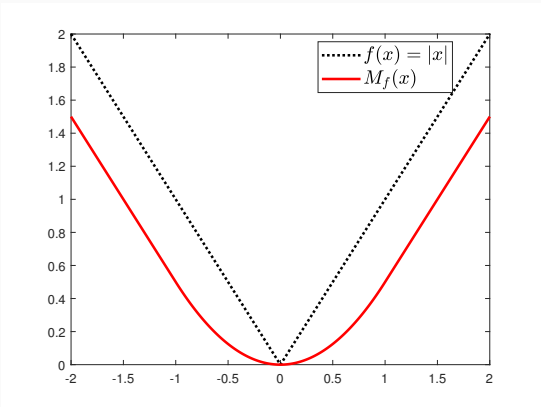
$$M_f(x) = \begin{cases} \frac{1}{2}x^2, & |x| \le \lambda \\ \lambda|x| - \frac{\lambda^2}{2}, & |x| > \lambda \end{cases}$$

Its proximal mapping is (known as soft thresholding)

$$P_f(x) = \operatorname{sign}(x) \max\{|x| - \lambda, 0\}$$







Soft thresholding

The soft thresholding operator $S : \mathbb{R}^n \to \mathbb{R}^n$ is defined as

$$S_\lambda(x) = \begin{bmatrix} \operatorname{sign}(x_1)\max\{|x_1| - \lambda, 0\} \\ \operatorname{sign}(x_2)\max\{|x_2| - \lambda, 0\} \\ \vdots \\ \operatorname{sign}(x_n)\max\{|x_n| - \lambda, 0\} \end{bmatrix}$$

for any $x \in \mathbb{R}^n$ and $\lambda > 0$.

Example. Given

$$x = \begin{bmatrix} 1.5 \\ -0.4 \\ 3 \\ -2 \\ 0.8 \end{bmatrix}, \quad S_{0.5}(x) = \begin{bmatrix} 1 \\ 0 \\ 2.5 \\ -1.5 \\ 0.3 \end{bmatrix}, \quad S_2(x) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Moreau envelope and proximal mapping

- $M_f(\cdot)$ is always differentiable even though f is non-differentiable
- $P_f(\cdot)$ is important in many optimization algorithms (e.g., accelerated proximal gradient methods introduced later)
- For many widely used regularizaters, $P_f(\cdot)$ and $M_f(\cdot)$ have explicit expression

Optimizing composite functions

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \quad f(\beta) + g(\beta)$$

- $f(\beta)$ is convex and differentiable
- $g(\beta)$ is convex and non-differentiable
- For example, in lasso,

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \quad \underbrace{\frac{1}{2}\|X\beta - Y\|^2}_{=f(\beta)} + \underbrace{\lambda\|\beta\|_1}_{=g(\beta)}$$

- Since $g(\beta)$ is non-differentiable, we cannot apply gradient methods

Proximal gradient step

$$\underset{\beta \in \mathbb{R}^p}{\text{minimize}} \quad f(\beta) + g(\beta)$$

Gradient step (suppose $g(\beta)$ disappears):

$$\beta^{(k+1)} = \beta^{(k)} - \alpha_k \nabla f(\beta^{(k)})$$

which can be written equivalently as

$$\beta^{(k+1)} = \arg\min_{\beta} \left\{ f(\beta^{(k)}) + \langle \nabla f(\beta^{(k)}), \beta - \beta^{(k)} \rangle + \frac{1}{2\alpha_k} \|\beta - \beta^{(k)}\|^2 \right\}$$

Proximal gradient step:

$$\beta^{(k+1)} = \arg\min_{\beta} \left\{ f(\beta^{(k)}) + \langle \nabla f(\beta^{(k)}), \beta - \beta^{(k)} \rangle + g(\beta) + \frac{1}{2\alpha_k} \|\beta - \beta^{(k)}\|^2 \right\}$$

Proximal gradient step

Proximal gradient step:

$$\beta^{(k+1)} = \arg\min_{\beta} \left\{ f(\beta^{(k)}) + \langle \nabla f(\beta^{(k)}), \beta - \beta^{(k)} \rangle + g(\beta) + \frac{1}{2\alpha_k} \|\beta - \beta^{(k)}\|^2 \right\}$$

After ignoring constant terms and completing the square, the above step can be written equivalently as

$$\begin{aligned} \beta^{(k+1)} &= \arg\min_{\beta} \left\{ \frac{1}{2\alpha_k} \left\| \beta - \left(\beta^{(k)} - \alpha_k \nabla f(\beta^{(k)}) \right) \right\|^2 + g(\beta) \right\} \\ &= P_{\alpha_k g} \left(\beta^{(k)} - \alpha_k \nabla f(\beta^{(k)}) \right) \end{aligned}$$

Proximal gradient step

Proximal gradient step:

$$\beta^{(k+1)} = \arg\min_{\beta} \left\{ f(\beta^{(k)}) + \langle \nabla f(\beta^{(k)}), \beta - \beta^{(k)} \rangle + g(\beta) + \frac{1}{2\alpha_k} \|\beta - \beta^{(k)}\|^2 \right\}$$

After ignoring constant terms and completing the square, the above step can be written equivalently as

$$\begin{aligned} \beta^{(k+1)} &= \arg\min_{\beta} \left\{ \frac{1}{2\alpha_k} \left\| \beta - \left(\beta^{(k)} - \alpha_k \nabla f(\beta^{(k)}) \right) \right\|^2 + g(\beta) \right\} \\ &= P_{\alpha_k g} \left(\beta^{(k)} - \alpha_k \nabla f(\beta^{(k)}) \right) \end{aligned}$$

Derivation* completing the square

$$\begin{aligned} & \langle \nabla f(\beta^{(k)}), \beta \rangle + \frac{1}{2\alpha_k} \|\beta - \beta^{(k)}\|^2 \\ =& \langle \nabla f(\beta^{(k)}) - \frac{1}{\alpha_k} \beta^{(k)}, \beta \rangle + \frac{1}{2\alpha_k} \|\beta\|^2 + \text{constant} \\ =& \frac{1}{\alpha_k} \langle \alpha_k \nabla f(\beta^{(k)}) - \beta^{(k)}, \beta \rangle + \frac{1}{2\alpha_k} \|\beta\|^2 + \text{constant} \end{aligned}$$

Proximal gradient methods

Algorithm (Proximal gradient (PG) method)

Choose $\beta^{(0)}$, constant step length $\alpha > 0$. Set $k \leftarrow 0$

repeat until convergence

$$\beta^{(k+1)} = P_{\alpha g}\left(\beta^{(k)} - \alpha \nabla f(\beta^{(k)})\right)$$

$$k \leftarrow k + 1$$

end(repeat)

return $\beta^{(k)}$

Proximal gradient methods

(Informal) in convex problems (f and g are convex), iteration complexity of PG method is $O(\frac{1}{k})$:

$$f(\beta^{(k)}) + g(\beta^{(k)}) - \underbrace{\min_{\beta \in \mathbb{R}^p} f(\beta) + g(\beta)}_{\text{optimal value}} \leq O\left(\frac{1}{k}\right)$$

If adopting stopping condition

$$f(\beta^{(k)}) + g(\beta^{(k)}) - \text{optimal value} \leq 10^{-4}$$

we need $O(10^4)$ PG iterations

Nesterov's accelerated method

Nesterov's idea: include a momentum term for acceleration

$$\bar{\beta}^{(k)} = \beta^{(k)} + \underbrace{\frac{t_k - 1}{t_{k+1}}\left(\beta^{(k)} - \beta^{(k-1)}\right)}_{\text{momentum term}}$$

$$\beta^{(k+1)} = P_{\alpha g}\left(\bar{\beta}^{(k)} - \alpha \nabla f(\bar{\beta}^{(k)})\right)$$

$\{t_k\}$ is a positive sequence such that $t_0 = t_1 = 1$, $t_{k+1}^2 - t_{k+1} \leq t_k$

- e.g., $t_k = 1, \forall\, k$. In this case, momentum term $= 0$, it is reduced to PG without acceleration
- e.g., $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2} \Rightarrow t_0 = 1, t_1 = 1, t_2 = \frac{1+\sqrt{5}}{2}, \ldots$
- there are many other sequences satisfying the condition
- Next we simply take $t_0 = t_1 = 1, t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$

Accelerated proximal gradient methods

Algorithm (Accelerated proximal gradient (APG) method)

Choose $\beta^{(0)}$, constant step length $\alpha > 0$. Set $t_0 = t_1 = 1$, $k \leftarrow 0$

repeat until convergence

$$\bar{\beta}^{(k)} = \beta^{(k)} + \frac{t_k - 1}{t_{k+1}}(\beta^{(k)} - \beta^{(k-1)})$$

$$\beta^{(k+1)} = P_{\alpha g}\left(\bar{\beta}^{(k)} - \alpha \nabla f(\bar{\beta}^{(k)})\right)$$

$$k \leftarrow k + 1$$

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$$

end(repeat)

return $\beta^{(k)}$

The algorithmic framework follows from [1]. It is built based on Nesterov's accelerated method in 1983 [2]

Accelerated proximal gradient methods

(Informal) in convex problems (f and g are convex), iteration complexity of APG method is $O(\frac{1}{k^2})$:

$$f(\beta^{(k)}) + g(\beta^{(k)}) - \underbrace{\min_{\beta \in \mathbb{R}^p} f(\beta) + g(\beta)}_{\text{optimal value}} \leq O\left(\frac{1}{k^2}\right)$$

If adopting stopping condition

$$f(\beta^{(k)}) + g(\beta^{(k)}) - \text{optimal value} \leq 10^{-4}$$

we need $O(10^2)$ PG iterations

Accelerated proximal gradient methods

- Backtracking line search is also applicable for finding step length α_k.
- For simplicity, we take a constant step length. It should satisfy $\alpha \in (0, \frac{1}{L})$, where L is Lipschitz constant[4] of $\nabla f(\cdot)$ (typically unknown)
- APG methods enjoy the same computational cost per iteration as PG methods.
- Iteration complexity: APG $O(\frac{1}{k^2})$; PG $O(\frac{1}{k})$

[4] It means $\|\nabla f(\beta) - \nabla f(\hat{\beta})\| \leq L\|\beta - \hat{\beta}\|$ for all β and $\hat{\beta}$

APG for lasso

$$\underset{\beta\in\mathbb{R}^p}{\text{minimize}} \quad \underbrace{\frac{1}{2}\|X\beta - Y\|^2}_{=f(\beta)} + \underbrace{\lambda\|\beta\|_1}_{=g(\beta)} \quad \text{lasso problem}$$

Then $\nabla f(\beta) = X^T(X\beta - Y)$ with Lipschitz constant $L = \lambda_{\max}(X^TX)$.
Choose step length $\alpha = 1/L$. APG iterations:

$$\bar{\beta}^{(k)} = \beta^{(k)} + \frac{t_k - 1}{t_{k+1}}\left(\beta^{(k)} - \beta^{(k-1)}\right)$$
$$\beta^{(k+1)} = S_{\lambda/L}\left(\bar{\beta}^{(k)} - \frac{1}{L}X^T(X\bar{\beta}^{(k)} - Y)\right)$$

APG is also applicable for "logistic regression + lasso regularization"

Restart

- A strategy to speed up APG is to restart the algorithm after a fixed number of iterations
- using the latest iterate as the starting point of the new round of APG iteration
- a reasonable choice is to perform restart every 100 or 200 iterations

References i

A. Beck and M. Teboulle.

A fast iterative shrinkage-thresholding algorithm for linear inverse problems.

SIAM journal on imaging sciences, 2(1):183–202, 2009.

Y. E. Nesterov.

A method for solving the convex programming problem with convergence rate O$(1/k^2)$.

In Dokl. Akad. Nauk SSSR,, volume 269, pages 543–547, 1983.

R. Tibshirani.

Regression shrinkage and selection via the lasso.

Journal of the Royal Statistical Society: Series B (Methodological), 58(1):267–288, 1996.