Element of Statistical Learning Note

Zichong Wang

January 9, 2026

Contents

1 Intro

While reading ESL, I felt the need to write things down—to record what I did not understand and how I eventually worked through it. This note grew out of that process. It can be regarded as a personal supplymentary material of ESL. It covers concepts from analysis, linear algebra, optimization, and computation, with a primary emphasis on statistics.

2 Chap 3: Linear Methods for Regression

2.1 Confidence region for $\hat{\beta}$

We know that $\hat{\beta} \sim \mathcal{N}(\beta, \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1})$, thus $\hat{\beta} - \beta \sim \mathcal{N}(0, \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1})$.

Since for $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$, we have $\boldsymbol{z}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{z} \sim \chi^2_k$, where $k = \text{rank}(\boldsymbol{\Sigma})$, we have

$$(\hat{\beta} - \beta)^T(\sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1})^{-1}(\hat{\beta} - \beta) \sim \chi^2_{p+1}$$

Thus,

$$\frac{1}{\sigma^2}(\hat{\beta} - \beta)^T\boldsymbol{X}^T\boldsymbol{X}(\hat{\beta} - \beta) \sim \chi^2_{p+1}$$

And have approx confidence region

$$(\hat{\beta} - \beta)^T\boldsymbol{X}^T\boldsymbol{X}(\hat{\beta} - \beta) \leq \hat{\sigma}^2\chi^2_{p+1,1-\alpha}$$

Actually, $\hat{\sigma}^2 = \frac{1}{N-p-1}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \frac{1}{N-p-1}\text{RSS}$, and $\text{RSS}/\sigma^2 \sim \chi^2_{N-p-1}$,

$$\frac{(\hat{\beta} - \beta)^T\boldsymbol{X}^T\boldsymbol{X}(\hat{\beta} - \beta)/(p+1)}{\hat{\sigma}^2} \sim F_{p+1,N-p-1}$$

2.2 What is Linear

In the context of **linear model**, we are talking about linearity in parameters, meaning that the prediction $\hat{y}$ is a linear combination of the parameters β_j. The $\boldsymbol{X}$ itself can be non-linear transformations of the original features, e.g., polynomial terms, interaction terms, etc. $y = 1/(\beta_0 + \beta_1 x)$ and $y = \beta_0 e^{\beta_1 x}$ are not linear models, since they're not linear in parameters.

In the context of **Linear estimators**, we are talking about the estimator $\hat{\theta}$ (e.g. $\hat{\beta}$) can be written as a linear combination of the observed response values y_i, i.e. $\hat{\theta} = \boldsymbol{c}^T\boldsymbol{y}$. The weight $\boldsymbol{c}$ depends only on $\boldsymbol{X}$, not on $\boldsymbol{y}$. A linear estimator **can be** a prediction at a new point, or the estimated coefficients $\hat{\beta}$ themselves.

2.3 Gauss-Markov Theorem

Why assume only know $\boldsymbol{X}$, but not $\boldsymbol{y}$?

> Note that though y_i as sample responses, are observable, the following statements and arguments including assumptions, proofs and the others assume under the only condition of knowing $\boldsymbol{X}_{i,j}$ but not y_i. — [2]

We have a *challenger* linear estimator $\tilde{\beta} = \boldsymbol{C}\boldsymbol{y}$, where $\boldsymbol{C} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T + \boldsymbol{D}$, a modification

of OLS estimator. Ensure it's unbiased:

$$\begin{aligned}\mathbb{E}(\tilde{\beta}) &= \mathbb{E}(\boldsymbol{C}\boldsymbol{y}) \\ &= \boldsymbol{C}\mathbb{E}(\boldsymbol{y}) = ((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T + \boldsymbol{D})\boldsymbol{X}\beta \\ &= \beta + \boldsymbol{D}\boldsymbol{X}\beta \\ &= \beta\end{aligned}$$

Meaning $\boldsymbol{D}\boldsymbol{X} = 0$.

Now, compute the variance:

$$\begin{aligned}\mathrm{Var}(\tilde{\beta}) &= \mathrm{Var}(\boldsymbol{C}\boldsymbol{y}) \\ &= \boldsymbol{C}\mathrm{Var}(\boldsymbol{y})\boldsymbol{C}^T \\ &= \sigma^2\boldsymbol{C}\boldsymbol{C}^T \\ &= \sigma^2((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T + \boldsymbol{D})((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T + \boldsymbol{D})^T \\ &= \sigma^2((\boldsymbol{X}^T\boldsymbol{X})^{-1} + \boldsymbol{D}\boldsymbol{D}^T) \\ &= \mathrm{Var}(\hat{\beta}) + \sigma^2\boldsymbol{D}\boldsymbol{D}^T \\ &\geq \mathrm{Var}(\hat{\beta})\end{aligned}$$

2.4 QR decomposition

Any real squared matrix $\boldsymbol{A}$ can be decomposed as $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$, where $\boldsymbol{Q}$ is orthogonal matrix ($\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$), and $\boldsymbol{R}$ is upper triangular matrix.

Any rectangular matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ ($m \geq n$), we can decompose it as $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$, where $\boldsymbol{Q}$ is $m \times m$ orthogonal matrix, and $\boldsymbol{R}$ is $m \times n$ upper triangular matrix (the last $m - n$ rows are all zero). It can be regarded as $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R} = [\boldsymbol{Q}_1 \quad \boldsymbol{Q}_2]\begin{bmatrix}\boldsymbol{R}_1 \\ \boldsymbol{0}\end{bmatrix} = \boldsymbol{Q}_1\boldsymbol{R}_1$, where $\boldsymbol{Q}_1 \in \mathbb{R}^{m\times n}, \boldsymbol{Q}_2 \in \mathbb{R}^{m\times(m-n)}, \boldsymbol{R}_1 \in \mathbb{R}^{n\times n}$ which is upper triangular.

If $\boldsymbol{A}$ have k linearly independent columns, then first k columns of $\boldsymbol{Q}$ form an orthonormal basis of the column space of $\boldsymbol{A}$. The fact that any column k of $\boldsymbol{A}$ only depends on the first k columns of $\boldsymbol{Q}$ corresponds to the triangular form of $\boldsymbol{R}$.

QR decomposition can be calculated using Gram-Schmidt process, or using Householder reflections. In practice, Householder reflections are more stable and efficient.

2.4.1 Application to Least Squared

In linear least squares problems, we aim to find a vector $\boldsymbol{x}$ that minimizes the Euclidean norm of the residual for an overdetermined system $\boldsymbol{A}\boldsymbol{x} \approx \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $m \geq n$. The goal is to solve:

$$\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$$

Using the Full QR Decomposition, we substitute $\boldsymbol{A} = \boldsymbol{Q}\begin{bmatrix}\boldsymbol{R}_1 \\ \boldsymbol{0}\end{bmatrix}$:

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 = \left\|\boldsymbol{Q}\begin{bmatrix}\boldsymbol{R}_1 \\ \boldsymbol{0}\end{bmatrix}\boldsymbol{x} - \boldsymbol{b}\right\|_2$$

Since multiplying by an orthogonal matrix $\boldsymbol{Q}$ preserves the Euclidean norm, we can left-multiply the entire expression by $\boldsymbol{Q}^T$:

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 = \left\| \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{x} - \boldsymbol{Q}^T \boldsymbol{b} \right\|_2$$

If we partition $\boldsymbol{Q}^T\boldsymbol{b}$ into two components—$\boldsymbol{c}_1 \in \mathbb{R}^n$ and $\boldsymbol{c}_2 \in \mathbb{R}^{m-n}$:

$$\left\| \begin{bmatrix} \boldsymbol{R}_1\boldsymbol{x} - \boldsymbol{c}_1 \\ -\boldsymbol{c}_2 \end{bmatrix} \right\|_2^2 = \|\boldsymbol{R}_1\boldsymbol{x} - \boldsymbol{c}_1\|_2^2 + \|\boldsymbol{c}_2\|_2^2$$

To minimize the total error, we must make the first term zero. The least squares solution is found by solving the square, upper-triangular system:

$$\boldsymbol{R}_1\boldsymbol{x} = \boldsymbol{c}_1$$

Since $\boldsymbol{R}_1$ is upper triangular, we can efficiently solve this system using back substitution, **that's how we solve linear equations manually in algebra class**! The remaining term $\|\boldsymbol{c}_2\|_2$ represents the minimum residual norm (the "error" of the fit).

2.4.2 About orthogonal matrix

Let $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ be an orthogonal matrix. Write $\boldsymbol{Q}$ in terms of its column vectors:

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \cdots & \boldsymbol{q}_n \end{bmatrix}, \qquad \boldsymbol{q}_i^\top \boldsymbol{q}_j = \delta_{ij}.$$

Then

$$\boldsymbol{Q}^\top = \begin{bmatrix} \boldsymbol{q}_1^\top \\ \boldsymbol{q}_2^\top \\ \vdots \\ \boldsymbol{q}_n^\top \end{bmatrix}.$$

Computation of $\boldsymbol{Q}^\top\boldsymbol{Q}$.

$$\boldsymbol{Q}^\top\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{q}_1^\top \\ \boldsymbol{q}_2^\top \\ \vdots \\ \boldsymbol{q}_n^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \cdots & \boldsymbol{q}_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_1^\top\boldsymbol{q}_1 & \boldsymbol{q}_1^\top\boldsymbol{q}_2 & \cdots & \boldsymbol{q}_1^\top\boldsymbol{q}_n \\ \boldsymbol{q}_2^\top\boldsymbol{q}_1 & \boldsymbol{q}_2^\top\boldsymbol{q}_2 & \cdots & \boldsymbol{q}_2^\top\boldsymbol{q}_n \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{q}_n^\top\boldsymbol{q}_1 & \boldsymbol{q}_n^\top\boldsymbol{q}_2 & \cdots & \boldsymbol{q}_n^\top\boldsymbol{q}_n \end{bmatrix}.$$

Using $\boldsymbol{q}_i^\top\boldsymbol{q}_j = \delta_{ij}$,

$$\boldsymbol{Q}^\top\boldsymbol{Q} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} = \boldsymbol{I}.$$

Computation of $\boldsymbol{Q}\boldsymbol{Q}^\top$.

$$\boldsymbol{Q}\boldsymbol{Q}^\top = \begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \cdots & \boldsymbol{q}_n \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_1^\top \\ \boldsymbol{q}_2^\top \\ \vdots \\ \boldsymbol{q}_n^\top \end{bmatrix} = \boldsymbol{q}_1\boldsymbol{q}_1^\top + \boldsymbol{q}_2\boldsymbol{q}_2^\top + \cdots + \boldsymbol{q}_n\boldsymbol{q}_n^\top = \sum_{k=1}^{n} \boldsymbol{q}_k\boldsymbol{q}_k^\top.$$

For any $\boldsymbol{x} \in \mathbb{R}^n$,

$$(\boldsymbol{Q}\boldsymbol{Q}^\top)\boldsymbol{x} = \sum_{k=1}^{n} \boldsymbol{q}_k(\boldsymbol{q}_k^\top \boldsymbol{x}) = \boldsymbol{x},$$

since $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n\}$ is an orthonormal basis of $\mathbb{R}^n$. Hence

$$\boldsymbol{Q}\boldsymbol{Q}^\top = \boldsymbol{I}.$$

Therefore, for an orthogonal matrix $\boldsymbol{Q}$,

$$\boldsymbol{Q}^\top\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^\top = \boldsymbol{I}.$$

2.4.3 SVD and orthogonal matrix

There are two kinds of SVD: full SVD and reduced SVD. Let $\boldsymbol{X} \in \mathbb{R}^{N\times p}$, and nomally $N > p$.

- **Full SVD** in standard linear algebra.

$$\boldsymbol{X} = \boldsymbol{U}_{\text{full}}\boldsymbol{D}_{\text{full}}\boldsymbol{V}^T$$

  - $\boldsymbol{U}_{\text{full}} \in \mathbb{R}^{N\times N}$ is orthogonal matrix, whose columns are eigenvectors of $\boldsymbol{X}\boldsymbol{X}^T$.
  - $\boldsymbol{D}_{\text{full}} \in \mathbb{R}^{N\times p}$ is a rectangular diagonal matrix, and the last $N-p$ rows are all zero.
  - $\boldsymbol{V} \in \mathbb{R}^{p\times p}$ is orthogonal matrix, whose columns are eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$, and are basis of row space of $\boldsymbol{X}$.

- **Reduced SVD** in linear regression. We simply ignore the zero parts of $\boldsymbol{U}$ and $\boldsymbol{D}$.

  $\boldsymbol{D}$ and $\boldsymbol{U}$ can be seen as

$$\boldsymbol{D}_{\text{full}} = \begin{bmatrix} d_1 & 0 & \ldots \\ 0 & \ddots & \\ \ldots & & d_p \\ \hline 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} p \text{ (Non-zero part)} \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} N-p \text{ (All zero part)} \end{matrix}$$

$$\mathbf{U}_{\text{full}} = \begin{bmatrix} \mathbf{U}_1 & | & \mathbf{U}_2 \end{bmatrix}$$

  Thus,

$$\begin{aligned} \mathbf{U}_{full} \cdot \mathbf{D}_{full} &= \begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{D}_p \\ \mathbf{0} \end{bmatrix} \\ &= \mathbf{U}_1 \cdot \mathbf{D}_p + \mathbf{U}_2 \cdot \mathbf{0} \\ &= \mathbf{U}_1 \cdot \mathbf{D}_p \end{aligned}$$

  That's the reduced SVD:

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$$

  where $\boldsymbol{U} \in \mathbb{R}^{N\times p}$ with orthonormal columns, $\boldsymbol{D} \in \mathbb{R}^{p\times p}$ diagonal with positive entries, and $\boldsymbol{V} \in \mathbb{R}^{p\times p}$ orthogonal.

In regression, we usually use reduced SVD, and $\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{H} \neq \boldsymbol{I}$, but $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$. For $\boldsymbol{y} \in \mathbb{R}^N$, $\boldsymbol{U}^T\boldsymbol{y}$ map $\boldsymbol{y}$ from $\mathbb{R}^N$ to $\mathbb{R}^p$, which is the coefficients in the basis of columns of $\boldsymbol{U}$. And $\boldsymbol{U}\boldsymbol{U}^T\boldsymbol{y}$ project $\boldsymbol{y}$ onto the column space of $\boldsymbol{X}$.

2.5 Multiple testing in forward selection

ESL page 60:

> Other more traditional packages base the selection on F -statistics, adding "significant" terms, and dropping "non-significant" terms. These are out of fashion, since they do not take proper account of the multiple testing issues.

Assume we have p candidate features, and already selected k features. When considering adding a new feature, we are actually performing $p-k$ hypothesis tests (each test for one feature). Even the rest $p-k$ features are all noise, with significance level α, we still have a probability of $1-(1-\alpha)^{p-k}$ to incorrectly add at least one noise feature.

2.6 Ridge regression

Answer questions: why two forms are equivalent? Why not equivariant under scaling of the inputs? What is a good practice for it? df of ridge? In the case of orthogonal inputs, why $\hat{\beta}^{\text{ridge}} = \hat{\beta}/(1+\lambda)$?

Ridge regression shrinks the regression coefficients by imposing a penalty on their size:

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

and is equivalent to

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$$
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 \le t$$

And there is a one-to-one correspondence between λ and t.

2.6.1 How to use ridge regression

Apparently, ridge regression is not equivariant under scaling of the inputs. For example, using OLS, measuring x in meters or in centimeters will not change the predictions, since the latter coefficients will just 100 times of the first. However, in ridge regression, the penalty term $\lambda \sum_{j=1}^{p} \beta_j^2$ will be affected by the scale of x_j. Which means, using centimeters instead of meters will make the penalty on β_j 10000 times larger, leading to different solutions. Thus, it's important to standardize the features (zero mean and unit variance) before applying ridge.

Usually, we calculate μ and σ from training set, and use them to standardize both training and test sets. Scaler can be regarded as part of the model, not data cleaning!

2.6.2 Equivalence of two forms

First, take a review of **KKT conditions**.

Condider a minimization problem with both equality and inequality constraints:

$$\begin{aligned} &\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ \text{subject to } &g_i(\boldsymbol{x}) \leq 0, && i = 1, \dots, m \\ &h_j(\boldsymbol{x}) = 0, && j = 1, \dots, l \end{aligned}$$

And the lagrangian function is:

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}) + \sum_{j=1}^{l} \mu_j h_j(\boldsymbol{x})$$

If $\boldsymbol{x}^*$ is a local minimum, then there exist multipliers $\lambda_i^* \geq 0$ and μ_j^* such that the following conditions hold:

- **Stationary**
$$\nabla_x L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = 0$$
- **Primal feasibility**
$$g_i(\boldsymbol{x}^*) \leq 0, \quad i = 1, \dots, m$$
$$h_j(\boldsymbol{x}^*) = 0, \quad j = 1, \dots, l$$
The gradient can be regarded as the force to push a particle, primal stationary means the force of $\partial f(\boldsymbol{x}^*)$ is balanced by a linear sum of forces from constraints.
- **Dual feasibility**
$$\lambda_i^* \geq 0, \quad i = 1, \dots, m$$
All the $\partial g_i(\boldsymbol{x}^*)$ forces must be one-sided, pointing inwards into the feasible set for $\boldsymbol{x}$.
- **Complementary slackness**
$$\lambda_i^* g_i(\boldsymbol{x}^*) = 0, \quad i = 1, \dots, m$$
The force only activated when the particle is on the boundary of feasible set.

In ridge regression, we have:

$$\mathcal{L} = \|\boldsymbol{Y} - \boldsymbol{X}\beta\|_2^2 + \alpha(\|\beta\|_2^2 - t)$$

According to stationary condition:

$$\nabla_\beta \mathcal{L} = -2\boldsymbol{X}^T(\boldsymbol{Y} - \boldsymbol{X}\beta) + 2\alpha\beta = 0$$

On the other hand, solving the unconstrained form is

$$\nabla_\beta \left(\|\boldsymbol{Y} - \boldsymbol{X}\beta\|_2^2 + \lambda\|\beta\|_2^2 \right) = -2\boldsymbol{X}^T(\boldsymbol{Y} - \boldsymbol{X}\beta) + 2\lambda\beta = 0$$

Thus, if we set $\lambda = \alpha$, the two forms are equivalent.

One step further, according to complementary slackness:

$$\alpha(\|\beta\|_2^2 - t) = 0$$

From unconstrained form, given λ, we can solve β, denote $\beta(\lambda)$, and define $t(\lambda) = \|\beta(\lambda)\|_2^2$. Then, $\beta(\lambda)$ is also the solution of constrained form with $t = t(\lambda)$ (apparently it's on the boundary, and the coefficient $\alpha = \lambda$).

From the constrained form, given t, we can also solve β, denote $\beta(t)$.

- If $\|\beta(t)\|_2^2 < t$, then according to complementary slackness, $\alpha = 0$, which means no penalty, $\lambda = 0$, back to OLS.
- If $\|\beta(t)\|_2^2 = t$, the boundary is effective, correspond to some $\alpha > 0$, and $\lambda = \alpha$.

2.6.3 Bayesian view

Assume prior $\boldsymbol{\beta} \sim \mathcal{N}(0, \tau^2 \boldsymbol{I})$, and likelihood $\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{X}\beta, \sigma^2 \boldsymbol{I})$. Then the posterior is:

$$\begin{aligned} p(\beta|\boldsymbol{Y}) &\propto p(\boldsymbol{Y}|\beta)p(\beta) \\ &\propto \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{Y} - \boldsymbol{X}\beta\|_2^2\right) \exp\left(-\frac{1}{2\tau^2}\|\beta\|_2^2\right) \\ &\propto \exp\left(-\frac{1}{2}\left(\frac{1}{\sigma^2}\|\boldsymbol{Y} - \boldsymbol{X}\beta\|_2^2 + \frac{1}{\tau^2}\|\beta\|_2^2\right)\right) \end{aligned}$$

Maximizing the posterior is equivalent to minimizing the negative log-posterior. The object is equivalent to ridge regression with $\lambda = \sigma^2/\tau^2$.

However, it seems that from bayesian view, we don't have a hard constrain on the size of β, just a prior that β is likely to be small. But the constrained form of ridge regression directly enforce $\|\beta\|_2^2 \leq t$. Is it contradictory?

No. Actually, the posterior distribution of β still have infinite support, meaning that β can still be large with small probability. Our MAP estimate of β is the **mode** of the posterior, it's just a **point estimate**. This point estimator can still satisfy the hard constraint $\|\beta\|_2^2 \leq t$ for some t.

In ridge regression, since the posterior is symmetric gaussian distribution, the mean and mode are same. However, in lasso and other priors, we usually use **mode**, **not mean** in common Bayes estimate.

2.6.4 SVD in ridge

The solution of ridge regression is:

$$\hat{\beta}^{\text{ridge}} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

When $\boldsymbol{X}^T\boldsymbol{X}$ is singular (not full rank), OLS estimator is not defined, and adding $\lambda\boldsymbol{I}$ ensures that $\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}$ is positive definite and invertible.

Take reduced SVD of $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$, $r = \text{Rank}(\boldsymbol{X})$, where $\boldsymbol{U} \in \mathbb{R}^{N\times r}, \boldsymbol{D} \in \mathbb{R}^{r\times r}, \boldsymbol{V} \in \mathbb{R}^{p\times r}$. Then,

$$\begin{aligned} \boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I} &= \boldsymbol{V}\boldsymbol{D}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T + \lambda\boldsymbol{I} \\ &= \boldsymbol{V}\boldsymbol{D}^T\boldsymbol{D}\boldsymbol{V}^T + \lambda\boldsymbol{V}\boldsymbol{V}^T \\ &= \boldsymbol{V}(\boldsymbol{D}^2 + \lambda\boldsymbol{I})\boldsymbol{V}^T \end{aligned}$$

must be full rank and invertible.

It's also easy to see that in the case of orthogonal inputs (i.e., $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}$), the ridge estimates are just a scaled version of the least squares estimates, that is

$$\hat{\beta}^{\text{ridge}} = \frac{1}{1+\lambda}\hat{\beta}$$

Further more,

$$\begin{aligned} \boldsymbol{X}\beta^{\text{OLS}} &= \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y} \\ &= \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T(\boldsymbol{V}\boldsymbol{D}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T)^{-1}\boldsymbol{V}\boldsymbol{D}^T\boldsymbol{U}^T\boldsymbol{Y} \\ &= \boldsymbol{U}\boldsymbol{D}\boldsymbol{D}^{-2}\boldsymbol{D}^T\boldsymbol{U}^T\boldsymbol{Y} \\ &= \boldsymbol{U}\boldsymbol{U}^T\boldsymbol{Y} \end{aligned}$$

and

$$\begin{aligned}\boldsymbol{X}\beta^{\text{ridge}} &= \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X}+\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{Y}\\ &= \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T(\boldsymbol{V}(\boldsymbol{D}^2+\lambda\boldsymbol{I})\boldsymbol{V}^T)^{-1}\boldsymbol{V}\boldsymbol{D}^T\boldsymbol{U}^T\boldsymbol{Y}\\ &= \boldsymbol{U}\boldsymbol{D}(\boldsymbol{D}^2+\lambda\boldsymbol{I})^{-1}\boldsymbol{D}^T\boldsymbol{U}^T\boldsymbol{Y}\\ &= \boldsymbol{U}\text{diag}\left(\frac{d_1^2}{d_1^2+\lambda},\frac{d_2^2}{d_2^2+\lambda},\dots,\frac{d_r^2}{d_r^2+\lambda}\right)\boldsymbol{U}^T\boldsymbol{Y}\end{aligned}$$

In both way, $\boldsymbol{U}^T\boldsymbol{Y}$ gets the coordinates of $\boldsymbol{Y}$ in the basis of columns of $\boldsymbol{U}$ (the principal components of $\boldsymbol{X}$), and then send back to original space using $\boldsymbol{U}$, but ridge regression shrinks the coordinates. It's easy to see the directions with larger variance have lower effect, and those with small variance shrink most.

Similar to OLS, the degree of freedom of ridge is defined as

$$\text{df}(\lambda) = \text{tr}(\boldsymbol{H}(\lambda)) = \sum_{j=1}^{p}\frac{d_j^2}{d_j^2+\lambda}$$

But actually it's because the definition of df is

$$\text{df}(\hat{\boldsymbol{y}}) = \frac{1}{\sigma^2}\text{Cov}(\hat{y}_i, y_i)$$

indicating how much the prediction $\hat{y}_i$ change if data y_i have a tiny change. If the change is big, then model is 'reciting' the data point, model is complex, taking lots of df. If the change is small, meaning model is smooth, little affected by single data point. Model is simple, df is low.

For those prediction is a linear transformation of $\boldsymbol{y}$ (OLS, ridge), the df is the trace of hat matrix.

2.6.5 Revisit PCA

For a matrix $\boldsymbol{X}\in\mathbb{R}^{n\times p}$, the mean is

$$\mu := \frac{1}{n}\boldsymbol{X}^T\boldsymbol{1}\in\mathbb{R}^p$$

The centered matrix is

$$\boldsymbol{X}_c := \boldsymbol{X} - \boldsymbol{1}\mu^T$$

centering is to put the center of data cloud in origin, otherwise PCs are not purely about variance, but also about the shift from origin.

The SVD of $\boldsymbol{X}_c$ is $\boldsymbol{X}_c = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$, and since it's centered, the covariance is

$$\boldsymbol{S} = \frac{1}{n-1}\boldsymbol{X}_c^T\boldsymbol{X}_c = \frac{1}{n-1}\boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}^T$$

apparently, each column of $\boldsymbol{V}$ is the eigenvector of $\boldsymbol{S}$: $v_k = \boldsymbol{V}_{:,k}\in\mathbb{R}^p$, and project origin p dim feature into one dim.

Thus, to get a k dim representation, we need to take first k column of $\boldsymbol{V}$, which is an encoder:

$$\boldsymbol{Z}_k = \boldsymbol{X}_c\boldsymbol{V}_k = \boldsymbol{U}_k\boldsymbol{\Sigma}_k$$

The first principal component direction $z_1 = \boldsymbol{X}_c v_1$ has **largest** sample variance among all normalized linear combinations of the columns of $\boldsymbol{X}_c$, the variance is

$$\begin{aligned}\mathrm{Var}(z_1) &= \frac{1}{n-1} z_1^T z_1 = \frac{1}{n-1}(\boldsymbol{X}_c v_1)^T(\boldsymbol{X}_c v_1) \\ &= \frac{1}{n-1} v_1^T \boldsymbol{X}_c^T \boldsymbol{X}_c v_1 \\ &= v_1^T \boldsymbol{S} v_1 \\ &= v_1^T \lambda_1 v_1 \\ &= \lambda_1 = \frac{\sigma_1^2}{n-1}\end{aligned}$$

So the variance of the first PC scores is exactly the top eigenvalue of the covariance matrix.

And if we need to reconstruct the origin matrix, just decode in the same way:

$$\hat{\boldsymbol{X}}_{\boldsymbol{c}} = \boldsymbol{Z}_k \boldsymbol{V}_k^T, \quad \hat{\boldsymbol{X}} = \hat{\boldsymbol{X}}_{\boldsymbol{c}} + \mathbf{1}\mu^T$$

for a single sample:

$$\hat{x} = \boldsymbol{V}_k z = \boldsymbol{V}_k(\boldsymbol{V}_k^T z)$$

Thus, $\boldsymbol{V}$ is just orthogonal basis of the feature space. It maps centered data in to its space, and get the coordinates.

From an auto encoder's view, it is

$$\min_{\boldsymbol{W} \in \mathbb{R}^{p \times k}} \|\boldsymbol{X}_c - \boldsymbol{X}_c \boldsymbol{W} \boldsymbol{W}^\top\|_F^2 \quad \text{s.t.} \quad \boldsymbol{W}^\top \boldsymbol{W} = \boldsymbol{I}$$

It seems we are talking about $\boldsymbol{V}$ all the time. What about $\boldsymbol{U}$? Well, he is the basis of the sample space. Since we don't care about sample space that much, we just don't talk about it that much.

2.7 LASSO

2.7.1 Subgradient and subdifferential

For a convex function $f : \mathbb{R} \to \mathbb{R}$, a number g is a subgradient of f at x_0 if for all x,

$$f(x) \geq f(x_0) + g \cdot (x - x_0)$$

So the line with slope g touching $(x_0, f(x_0))$ lies below the whole function.

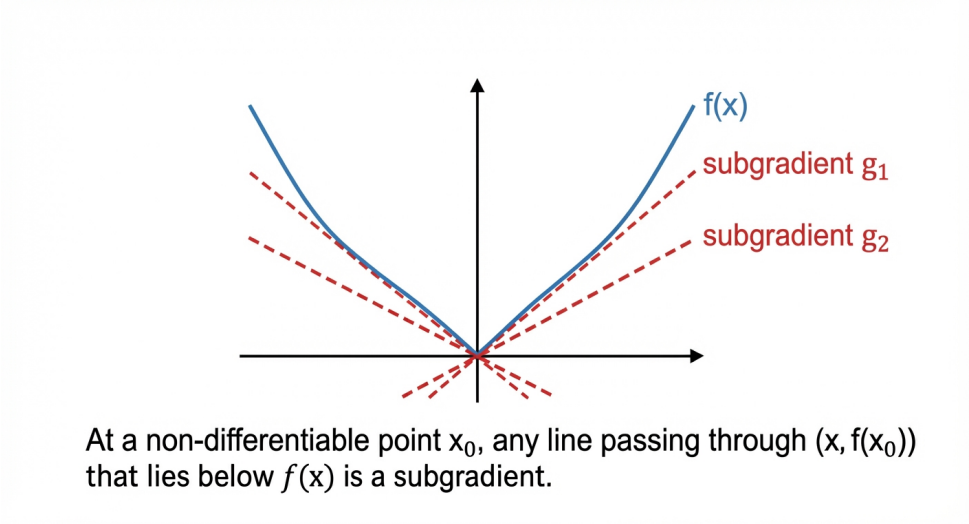


Figure 1: Illustration of subgradient.

The set of all such slopes is the subdifferential:

$$\partial f(x_0) = \{g : f(x) \geq f(x_0) + g \cdot (x - x_0), \forall x\}$$

Thus, for $f(x) = |x|$,

$$\partial |x| = \begin{cases} \{1\} & \text{if } x > 0 \\ \{-1\} & \text{if } x < 0 \\ [-1, 1] & \text{if } x = 0 \end{cases}$$

When $x = 0$, the subdifferential is a interval!

Common way to compute Lasso is ISAT in 4.4.1.

2.7.2 Why shrink to zero

Consider LASSO objective function

$$\min_{\beta} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1$$

the solution $\hat{\boldsymbol{\beta}}$ satisfies

$$\mathbf{0} \in \partial J(\hat{\beta}) \in \mathbb{R}^p$$

take one feature, and define $\boldsymbol{r} = \boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}$

$$0 \in -X_j^T \boldsymbol{r} + \lambda \partial \hat{\beta}_j$$

which means, if $X_j^T \boldsymbol{r} \in [-\lambda, \lambda]$, $\hat{\beta}_j$ will be shrink to zero.

What does a low value of $\mathbf{x}_j^T \mathbf{r}$ actually indicate? It signifies that the feature has a low correlation with the current residual. In other words, this feature is not useful for reducing the prediction error, even if it contains 'new' or unique information.

We must distinguish this from **feature-to-feature correlation** (multicollinearity):

- **Redundancy (High Feature Correlation):** Consider predicting weight using height. Height in 'cm' and 'inches' are perfectly correlated. Lasso chooses only one because adding the second provides no *additional* help in reducing the residual.
- **Irrelevance (Low Residual Correlation):** Consider the feature 'eating carrots'. This is uncorrelated with height (it brings independent information), but it has no relationship with weight. Since it cannot lower the residual ($\boldsymbol{x}_{carrots}^T \boldsymbol{r} \approx 0$), Lasso abandons it.

2.7.3 Soft thresholding

When design matrix $\boldsymbol{X}$ is orthogonal, i.e. $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}$, the OLS solution become $\hat{\boldsymbol{\beta}}^{\text{OLS}} = \boldsymbol{X}^T\boldsymbol{Y}$, and LASSO becomes

$$\begin{aligned}
\hat{\boldsymbol{\beta}}^{\text{Lasso}} &= \underset{\beta}{\text{argmin}} \, \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \\
&= \underset{\beta}{\text{argmin}} \, \|\boldsymbol{X}^T\boldsymbol{Y} - \boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \\
&= \underset{\beta}{\text{argmin}} \, \|\hat{\boldsymbol{\beta}}^{\text{OLS}} - \boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1
\end{aligned}$$

Let the gradient to zero,

$$\frac{\partial}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \hat{\boldsymbol{\beta}}^{\text{OLS}} + \lambda \cdot \text{sign}(\boldsymbol{\beta}) = 0$$

$$\begin{aligned}
\Rightarrow \quad \hat{\boldsymbol{\beta}}^{\text{Lasso}} &= \hat{\boldsymbol{\beta}}^{\text{OLS}} - \lambda \cdot \text{sign}(\hat{\boldsymbol{\beta}}^{\text{Lasso}}) \\
&= \text{sign}(\hat{\boldsymbol{\beta}}^{\text{OLS}}) \max\{0, |\hat{\boldsymbol{\beta}}^{\text{OLS}}| - \lambda\} \\
&= S_\lambda(\hat{\boldsymbol{\beta}}^{\text{OLS}})
\end{aligned}$$

Here, S_λ is the soft-thresholding operator.

3 Chap 4: Linear methods for classification

Generally, there are two ways to model classification problem: hard boundary $\delta_k(x)$ or posterior probabilities $\Pr(G = k \mid X = x)$. If eithor $\delta_k(x)$ or $\Pr(G = k \mid X = x)$ are linear in x, then the dicision boundary is linear.

3.1 Linear Regression of an Indicator Matrix

Let $\boldsymbol{Y}$ be the $N \times K$ indicator matrix where each row has exactly one 1. Let $\boldsymbol{X}$ be the $N \times (p+1)$ design matrix, including a leading column of ones for the intercept.

We fit a linear model to **each column** of $\boldsymbol{Y}$ simultaneously using Least Squares. The objective is to minimize the Frobenius norm of the residual matrix:

$$\min_{\boldsymbol{B}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{B}\|_F^2$$

The solution $\hat{\boldsymbol{B}}$ is given by the normal equation:

$$\hat{\boldsymbol{B}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

For a new input vector $\boldsymbol{x} = (1, x_1, \ldots, x_p)^T$, the vector of fitted values (class scores) is:

$$\hat{\boldsymbol{f}}(\boldsymbol{x})^T = \boldsymbol{x}^T\hat{\boldsymbol{B}} = \boldsymbol{x}^T(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

Proof: Why $\sum \hat{f}_k(\boldsymbol{x}) = 1$

We want to calculate the sum of the predicted values across all K classes. Let $\mathbf{1}_K$ be a column vector of K ones, and $\mathbf{1}_N$ be a column vector of N ones.

The sum can be written as the dot product of the prediction vector and $\mathbf{1}_K$:

$$\sum_{k=1}^{K} \hat{f}_k(\boldsymbol{x}) = \hat{\boldsymbol{f}}(\boldsymbol{x})^T\mathbf{1}_K = \boldsymbol{x}^T(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T \underbrace{(\boldsymbol{Y}\mathbf{1}_K)}_{\text{step (a)}}$$

Step (a): Since each row of the indicator matrix $\boldsymbol{Y}$ sums to exactly 1 (one-hot encoding), multiplying $\boldsymbol{Y}$ by $\mathbf{1}_K$ results in a vector of ones:

$$\boldsymbol{Y}\mathbf{1}_K = \mathbf{1}_N$$

Substituting this back:

$$\sum_{k=1}^{K} \hat{f}_k(\boldsymbol{x}) = \boldsymbol{x}^T \underbrace{(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbf{1}_N}_{\text{step (b)}}$$

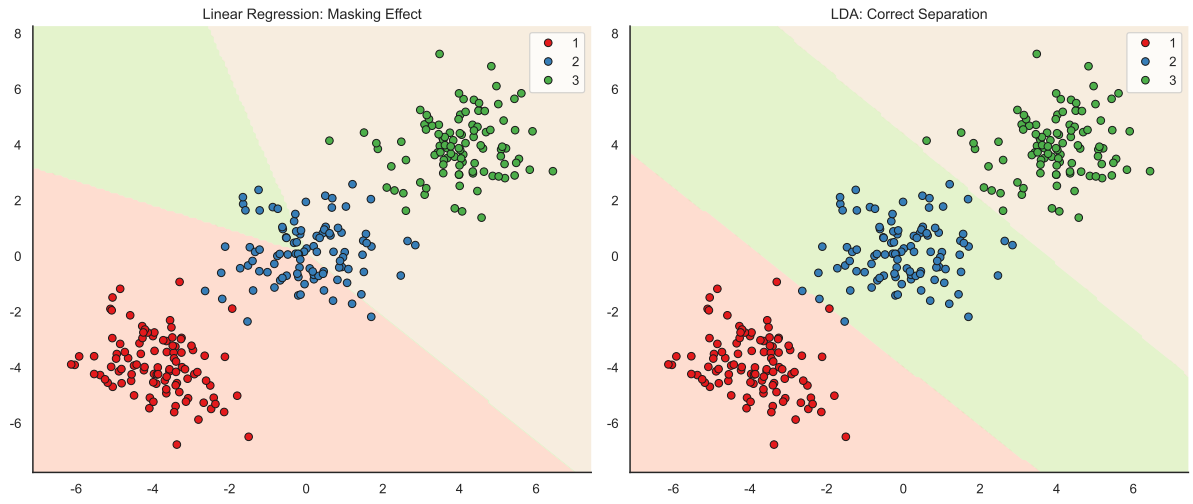
Step (b): The term $(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbf{1}_N$ represents the OLS regression coefficients obtained when regressing the vector $\mathbf{1}_N$ onto $\boldsymbol{X}$. Since $\boldsymbol{X}$ contains a column of ones (the intercept), the constant vector $\mathbf{1}_N$ lies perfectly in the column space of $\boldsymbol{X}$. The coefficients that strictly reconstruct $\mathbf{1}_N$ are 1 for the intercept and 0 for all other features. Let $\boldsymbol{e}_1 = (1, 0, \ldots, 0)^T$. Then:

$$(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbf{1}_N = \boldsymbol{e}_1$$

Conclusion: Since the input vector is $\boldsymbol{x} = (1, x_1, \ldots, x_p)^T$, its dot product with $\boldsymbol{e}_1$ picks out the first element (the intercept):

$$\sum_{k=1}^{K} \hat{f}_k(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{e}_1 = 1$$

Masking effect



4 Optimization methods

I read [1] for convex optimization methods.

4.1 Norms of matrices

Mathematically, a function $f : \mathbb{K}^{m\times n} \to \mathbb{R}$ is a matrix norm if for all matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{K}^{m\times n}$ and scalars α, it satisfies:

- Non-negativity: $\|\boldsymbol{A}\| \geq 0$ (and $\|\boldsymbol{A}\| = 0 \iff \boldsymbol{A} = \boldsymbol{0}$).
- Scalar Property: $\|\alpha\boldsymbol{A}\| = |\alpha|\|\boldsymbol{A}\|$.
- Triangle Inequality: $\|\boldsymbol{A} + \boldsymbol{B}\| \leq \|\boldsymbol{A}\| + \|\boldsymbol{B}\|$.

Most useful matrix norms also satisfy the submultiplicative property:

$$\|\boldsymbol{AB}\| \leq \|\boldsymbol{A}\|\|\boldsymbol{B}\|$$

Two kinds of norms: Induced (operator) norms and Frobenius norm (not induced).

4.1.1 Induced norm

An **induced norm** is defined by how much the matrix $\boldsymbol{A}$ stretches a vector $\boldsymbol{x}$ under a specific vector p-norm.

$$\|\boldsymbol{A}\|_p = \max_{\|\boldsymbol{x}\|_p=1} \|\boldsymbol{Ax}\|_p$$

The "Big Three" Induced Norms

- L_1 Norm (Maximum Column Sum):

$$\|\boldsymbol{A}\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{m} |a_{ij}|$$

- L_∞ Norm (Maximum Row Sum):

$$\|\boldsymbol{A}\|_\infty = \max_{1\leq i\leq m} \sum_{j=1}^{n} |a_{ij}|$$

- L_2 Norm (Spectral Norm):

$$\|\boldsymbol{A}\|_2 = \sqrt{\lambda_{\max}(\boldsymbol{A}^H\boldsymbol{A})} = \sigma_{\max}(\boldsymbol{A})$$

  where $\sigma_{\max}$ is the largest singular value of $\boldsymbol{A}$.

Here are proofs

- L_∞ **Norm (Maximum Row Sum)**

  Step 1: Upper Bound. For any vector $\boldsymbol{x}$ such that $\|\boldsymbol{x}\|_\infty = 1$, we have $|x_j| \leq 1$ for all j. For any row i:

  $$|(\boldsymbol{A}\boldsymbol{x})_i| = \left|\sum_{j=1}^{n} a_{ij}x_j\right| \leq \sum_{j=1}^{n} |a_{ij}||x_j| \leq \sum_{j=1}^{n} |a_{ij}|$$

  Thus, $\|\boldsymbol{A}\boldsymbol{x}\|_\infty = \max_i |(\boldsymbol{A}\boldsymbol{x})_i| \leq \max_i \sum_{j=1}^{n} |a_{ij}|$.

  Step 2: Attainability. Let k be the row index where $\sum_j |a_{ij}|$ is maximized. Define $\boldsymbol{x}$ such that $x_j = \operatorname{sgn}(a_{kj})$ if $a_{kj} \neq 0$ and $x_j = 1$ otherwise. Then $\|\boldsymbol{x}\|_\infty = 1$ and $|(\boldsymbol{A}\boldsymbol{x})_k| = \sum_j |a_{kj}|$, achieving the bound.

- L_1 **Norm (Maximum Column Sum)**

  Step 1: Upper Bound. For $\|\boldsymbol{x}\|_1 = 1$, we expand the L_1 norm of the product:

  $$\|\boldsymbol{A}\boldsymbol{x}\|_1 = \sum_{i=1}^{m} \left|\sum_{j=1}^{n} a_{ij}x_j\right| \leq \sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}||x_j| = \sum_{j=1}^{n} |x_j| \left(\sum_{i=1}^{m} |a_{ij}|\right)$$

  here, for finite and non-negative summation, we can always switch the order.

  Since $\sum_i |a_{ij}| \leq \max_j \sum_i |a_{ij}|$, we have:

  $$\|\boldsymbol{A}\boldsymbol{x}\|_1 \leq \left(\max_{1\leq j\leq n} \sum_{i=1}^{m} |a_{ij}|\right) \sum_{j=1}^{n} |x_j| = \max_{1\leq j\leq n} \sum_{i=1}^{m} |a_{ij}|$$

  Step 2: Attainability. Let k be the index of the column with the maximum absolute sum. Set $\boldsymbol{x} = \boldsymbol{e}_k$ (the k-th canonical basis vector). Then $\|\boldsymbol{x}\|_1 = 1$ and $\|\boldsymbol{A}\boldsymbol{x}\|_1 = \sum_i |a_{ik}|$, achieving the bound.

- L_2 **Norm (Spectral Norm)**

  Step 1: Upper Bound. Consider the square of the norm $\|\boldsymbol{A}\boldsymbol{x}\|_2^2 = \boldsymbol{x}^H\boldsymbol{A}^H\boldsymbol{A}\boldsymbol{x}$. Let $\boldsymbol{M} = \boldsymbol{A}^H\boldsymbol{A}$, which is Hermitian and positive semi-definite. By the spectral theorem, for any $\boldsymbol{x}$ with $\|\boldsymbol{x}\|_2 = 1$:

  $$\boldsymbol{x}^H\boldsymbol{M}\boldsymbol{x} \leq \lambda_{\max}(\boldsymbol{M})\|\boldsymbol{x}\|_2^2 = \lambda_{\max}(\boldsymbol{A}^H\boldsymbol{A})$$

  Taking the square root, we find $\|\boldsymbol{A}\boldsymbol{x}\|_2 \leq \sqrt{\lambda_{\max}(\boldsymbol{A}^H\boldsymbol{A})}$.

  Step 2: Attainability. Choose $\boldsymbol{x}$ to be the normalized eigenvector of $\boldsymbol{A}^H\boldsymbol{A}$ corresponding to $\lambda_{\max}$. Then $\boldsymbol{x}^H\boldsymbol{A}^H\boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}^H(\lambda_{\max}\boldsymbol{x}) = \lambda_{\max}\|\boldsymbol{x}\|_2^2 = \lambda_{\max}$, achieving the bound.

4.1.2 Frobenius norm

This is the most common entry-wise norm, analogous to the Euclidean distance for vectors.

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{\operatorname{tr}(\boldsymbol{A}^H\boldsymbol{A})}$$

Where tr is the trace and $\boldsymbol{A}^H$ is the conjugate transpose. It is also equal to the square root of the sum of the squares of the singular values:

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{\min(m,n)} \sigma_i^2}$$

4.1.3 Schatten norm

Schatten norms apply a p-norm to the singular values σ_i of the matrix. Let $\sigma(\boldsymbol{A})$ be the vector of singular values:

$$\|\boldsymbol{A}\|_p = \left(\sum_{i=1}^{\min(m,n)} \sigma_i^p \right)^{1/p}$$

- $p = 1$ (Nuclear Norm): $\|\boldsymbol{A}\|_* = \sum \sigma_i$. Often used in machine learning for rank minimization.
- $p = 2$ (Frobenius Norm): $\|\boldsymbol{A}\|_F = \sqrt{\sum \sigma_i^2}$.
- $p = \infty$ (Spectral Norm): $\|\boldsymbol{A}\|_2 = \sigma_{\max}$.

4.2 Gradient descent

Assume f is convex and differentiable, with $\mathrm{dom}(f) = \mathbb{R}^n$, and additionally that ∇f is Lipschitz continuous with constant $L > 0$,

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L \|x - y\|_2, \quad \forall x, y$$

when the stepsize $t \leq 1/L$, we have

$$f(x^{(k)}) - f^* \leq \frac{\|x^{(0)} - x^*\|^2}{2tk}$$

indicating a convergence rate $\mathcal{O}(1/k)$. In another word, to achieve precision ϵ, needs about $\mathcal{O}(1/\epsilon)$ times of iteration.

4.2.1 Descent Lemma

Express the difference between two points

$$f(\boldsymbol{y}) = f(\boldsymbol{x}) + \int_0^1 \nabla f(\boldsymbol{x} + \tau(\boldsymbol{y} - \boldsymbol{x}))^T (\boldsymbol{y} - \boldsymbol{x})\, d\tau$$

then introduce the gradient at $\boldsymbol{x}$

$$f(\boldsymbol{y}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{y} - \boldsymbol{x}) + \int_0^1 \left(\nabla f(\boldsymbol{x} + \tau(\boldsymbol{y} - \boldsymbol{x})) - \nabla f(\boldsymbol{x})\right)^T (\boldsymbol{y} - \boldsymbol{x})\, d\tau$$

apply Cauchy-Schwarz Inequality,

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{y} - \boldsymbol{x}) + \int_0^1 \|\nabla f(\boldsymbol{x} + \tau(\boldsymbol{y} - \boldsymbol{x})) - \nabla f(\boldsymbol{x})\| \cdot \|\boldsymbol{y} - \boldsymbol{x}\|\, d\tau$$

Using the L-Lipschitz property, we know $\|\nabla f(\boldsymbol{x}+\tau(\boldsymbol{y}-\boldsymbol{x})) - \nabla f(\boldsymbol{x})\| \leq L\|\tau(\boldsymbol{y}-\boldsymbol{x})\| = L\tau\|\boldsymbol{y}-\boldsymbol{x}\|$. Substituting this back:

$$\begin{aligned} f(\boldsymbol{y}) &\leq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{y} - \boldsymbol{x}) + \int_0^1 L\tau \|\boldsymbol{y} - \boldsymbol{x}\|^2\, d\tau \\ &= f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{y} - \boldsymbol{x}) + \frac{1}{2} L \|\boldsymbol{y} - \boldsymbol{x}\|^2 \end{aligned}$$

4.2.2 Convergence rate

In gradient descent update $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - t\nabla f(\boldsymbol{x}^{(k)})$ with step size $t \leq 1/L$:

$$\begin{aligned} f(\boldsymbol{x}^{(k+1)}) &\leq f(\boldsymbol{x}^{(k)}) + \nabla f(\boldsymbol{x}^{(k)})^T(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}) + \frac{L}{2}\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|^2 \\ &= f(\boldsymbol{x}^{(k)}) - t\|\nabla f(\boldsymbol{x}^{(k)})\|^2 + \frac{Lt^2}{2}\|\nabla f(\boldsymbol{x}^{(k)})\|^2 \\ &= f(\boldsymbol{x}^{(k)}) + (\frac{Lt^2}{2} - t)\|\nabla f(\boldsymbol{x}^{(k)})\|^2 \end{aligned}$$

Since $t \leq 1/L$, we have $-t + \frac{Lt^2}{2} \leq -\frac{t}{2}$. Thus:

$$f(\boldsymbol{x}^{(k+1)}) \leq f(\boldsymbol{x}^{(k)}) - \frac{t}{2}\|\nabla f(\boldsymbol{x}^{(k)})\|^2$$

which ensures **descent**.

By the convexity of f, we have $f(\boldsymbol{x}^*) \geq f(\boldsymbol{x}^{(k)}) + \nabla f(\boldsymbol{x}^{(k)})^T(\boldsymbol{x}^* - \boldsymbol{x}^{(k)})$, which rearranges to:

$$f(\boldsymbol{x}^{(k)}) - f^* \leq \nabla f(\boldsymbol{x}^{(k)})^T(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*)$$

plus the previous one, and get

$$f(\boldsymbol{x}^{(k+1)}) - f^* \leq \nabla f(\boldsymbol{x}^{(k)})^T(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*) - \frac{t}{2}\|\nabla f(\boldsymbol{x}^{(k)})\|^2$$

What's next? We want to know the relationship between the descent of $f(\boldsymbol{x}^{(k)} - f^*)$, and the descent of $\boldsymbol{x}^{(k)} - \boldsymbol{x}^*$, so consider

$$\begin{aligned} \|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^*\|^2 &= \|(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*) - t\nabla f(\boldsymbol{x}^{(k)})\|^2 \\ &= \|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^2 - 2t\nabla f(\boldsymbol{x}^{(k)})^T(\boldsymbol{x}^{(k)} - \boldsymbol{x}^*) + t^2\|\nabla f(\boldsymbol{x}^{(k)})\|^2 \end{aligned}$$

thus,

$$f(\boldsymbol{x}^{(k+1)}) - f^* \leq \frac{1}{2t}\left(\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^2 - \|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^*\|^2\right)$$

Summing from $i = 0$ to $k - 1$:

$$\sum_{i=0}^{k-1}(f(\boldsymbol{x}^{(i+1)}) - f^*) \leq \frac{1}{2t}\left(\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|^2 - \|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^2\right) \leq \frac{\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|^2}{2t}$$

and we know $k(f(\boldsymbol{x}^{(k)}) - f^*) \leq \sum_{i=0}^{k-1}(f(\boldsymbol{x}^{(i+1)}) - f^*)$, we get

$$f(\boldsymbol{x}^{(k)}) - f^* \leq \frac{\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|^2}{2tk}$$

4.2.3 Another view of GD

If we approximate the function around $\boldsymbol{x}$ by replacing the Hessian $\nabla^2 f(\boldsymbol{x})$ with $\frac{1}{t}\boldsymbol{I}$:

$$\boldsymbol{x}^+ = \arg\min_{\boldsymbol{z}}\left(f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{z} - \boldsymbol{x}) + \frac{1}{2t}\|\boldsymbol{z} - \boldsymbol{x}\|_2^2\right)$$

where t is the step size. Surprisingly, taking the derivative of the right-hand side with respect to $\boldsymbol{z}$ and setting it to zero gives the gradient descent update:

$$\nabla f(\boldsymbol{x}) + \frac{1}{t}(\boldsymbol{z} - \boldsymbol{x}) = 0 \implies \boldsymbol{x}^+ = \boldsymbol{x} - t\nabla f(\boldsymbol{x})$$

From this view, the standard gradient descent assume the Hessian matrix is same.

4.3 Subgradient descent

Here we assume function is G Lipschitz continuous: $|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq G\|\boldsymbol{x} - \boldsymbol{y}\|_2$, which bounds the norm of subgradient $\|\boldsymbol{g}\|_2 \leq G$.

The subgradient update rule is $\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} - t_k \boldsymbol{g}^{(k-1)}$, where $\boldsymbol{g}^{(k-1)} \in \partial f(\boldsymbol{x}^{(k-1)})$.

Analyze the squared distance to the optimal point $\boldsymbol{x}^*$:

$$\begin{aligned}\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|_2^2 &= \|\boldsymbol{x}^{(k-1)} - t_k \boldsymbol{g}^{(k-1)} - \boldsymbol{x}^*\|_2^2 \\ &= \|\boldsymbol{x}^{(k-1)} - \boldsymbol{x}^*\|_2^2 - 2t_k(\boldsymbol{g}^{(k-1)})^T(\boldsymbol{x}^{(k-1)} - \boldsymbol{x}^*) + t_k^2\|\boldsymbol{g}^{(k-1)}\|_2^2\end{aligned}$$

From the definition of a subgradient: $(\boldsymbol{g}^{(k-1)})^T(\boldsymbol{x}^{(k-1)} - \boldsymbol{x}^*) \geq f(\boldsymbol{x}^{(k-1)}) - f(\boldsymbol{x}^*)$. Substituting this and $\|\boldsymbol{g}\|_2 \leq G$:

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|_2^2 \leq \|\boldsymbol{x}^{(k-1)} - \boldsymbol{x}^*\|_2^2 - 2t_k(f(\boldsymbol{x}^{(k-1)}) - f^*) + t_k^2 G^2 \tag{$*$}$$

Summing from $i = 1$ to k and noting $\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|_2^2 \geq 0$:

$$0 \leq R^2 - 2\sum_{i=1}^{k} t_i(f(\boldsymbol{x}^{(i-1)}) - f^*) + G^2 \sum_{i=1}^{k} t_i^2$$

where $R = \|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|_2$. Rearranging for the best iterate $f(\boldsymbol{x}_{best}^{(k)}) = \min_{i=0,\ldots,k} f(\boldsymbol{x}^{(i)})$:

$$f(\boldsymbol{x}_{best}^{(k)}) - f^* \leq \frac{R^2 + G^2 \sum_{i=1}^{k} t_i^2}{2\sum_{i=1}^{k} t_i}$$

Thus, if $\sum_{i=1}^{k} t_i^2 < \infty$ and $\sum_{i=1}^{k} t_i = \infty$, subgradient can converge to optimal.

For a fixed step size $t_i = t$:

$$f(\boldsymbol{x}_{best}^{(k)}) - f^* \leq \frac{R^2}{2kt} + \frac{G^2 t}{2}$$

To achieve an error of ϵ, we set $t = \epsilon/G^2$. This requires:

$$k \geq \frac{R^2 G^2}{\epsilon^2}$$

Thus, the subgradient method has a convergence rate of $\mathcal{O}(1/\epsilon^2)$, which is slower than the $\mathcal{O}(1/\epsilon)$ rate of gradient descent.

When optimal value f^* is known, minimizing equotion ($*$) get Polyak step sizes:

$$t_k = \frac{f(\boldsymbol{x}^{(k-1)}) - f^*}{\|\boldsymbol{g}^{(k-1)}\|_2^2}$$

4.4 Proximal Gradient Descent

Although subgradient methods can solve non-differentiable optimization problems, it's slow $\mathcal{O}(1/\epsilon^2)$. Thus, we try write it as a composite function

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) + h(\boldsymbol{x})$$

where g is convex and differentiable, h is convex but not necessarily differentiable.

For smooth g, use quadratic approximation, and for non-smooth h, just keep it.

$$\begin{aligned}\boldsymbol{x}^+ &= \underset{\boldsymbol{z}}{\operatorname{argmin}} \left(g(\boldsymbol{x}) + \nabla g(\boldsymbol{x})^T(\boldsymbol{z}-\boldsymbol{x}) + \frac{1}{2t}\|\boldsymbol{z}-\boldsymbol{x}\|_2^2 \right) + h(\boldsymbol{z}) \\ &= \underset{\boldsymbol{z}}{\operatorname{argmin}} \left(\frac{1}{2t}\|\boldsymbol{z} - (\boldsymbol{x} - t\nabla g(\boldsymbol{x}))\|_2^2 + h(\boldsymbol{z}) \right)\end{aligned}$$

The first term make $\boldsymbol{z}$ stay close to the gradient update of g, while also make h small (second term).

Define **proximal mapping**:

$$\operatorname{prox}_{h,t}(\boldsymbol{v}) = \underset{\boldsymbol{z}}{\operatorname{argmin}} \left(\frac{1}{2t}\|\boldsymbol{z}-\boldsymbol{v}\|_2^2 + h(\boldsymbol{z}) \right)$$

then the procedure of proximal gradient descent is, for $k = 1, 2, 3, \ldots,$

$$\boldsymbol{v} = \boldsymbol{x}^{(k-1)} - t_k \nabla g(\boldsymbol{x}^{(k-1)})$$

$$\boldsymbol{x}^{(k)} = \operatorname{prox}_{h,t_k}(\boldsymbol{v})$$

The name of *proximal* comes from $\|\boldsymbol{z}-\boldsymbol{v}\|$ term, it makes the new solution $\boldsymbol{z}$ near input point $\boldsymbol{v}$

Proximal gradient descent have same convergence rate as gradient descent, but need to consider prox cost...

4.4.1 ISTA

Iterative Shrinkage-Thresholding Algorithm (ISTA) is used for Lasso 2.7.

$$\min \frac{1}{2}\|\boldsymbol{y} - X\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$$

$$\operatorname{prox}_{\lambda\|\cdot\|_1,t}(\boldsymbol{v}) = \underset{\boldsymbol{z}}{\operatorname{argmin}} \left(\frac{1}{2t}\|\boldsymbol{z}-\boldsymbol{v}\|_2^2 + \lambda\|\boldsymbol{z}\|_1 \right)$$

The solution is soft-thresholding operator, denote $S_{\lambda t}(\boldsymbol{v})$

$$[S_{\lambda t}(\boldsymbol{v})]_i = \begin{cases} v_i - \lambda t & \text{if } v_i > \lambda t \\ 0 & \text{if } |v_i| \le \lambda t \\ v_i + \lambda t & \text{if } v_i < -\lambda t \end{cases}$$

In practice, iteratively do

- Residual gradient update: $\boldsymbol{v} = \boldsymbol{\beta} + tX^T(\boldsymbol{y} - X\boldsymbol{\beta})$
- Soft-thresholding truncate: $\boldsymbol{\beta}^+ = S_{\lambda t}(\boldsymbol{v})$

4.5 Stochastic gradient descent

References

[1] Ryan Tibshirani. *Convex Optimization (10-725/36-725)*. Carnegie Mellon University course website. Accessed: 2026-01-03. 2019. URL: https://www.stat.cmu.edu/~ryantibs/convexopt/.

[2] Wikipedia contributors. *Gauss–Markov theorem — Wikipedia, The Free Encyclopedia*. Accessed: 2025-12-29. 2025. URL: https://en.wikipedia.org/wiki/Gauss%E2%80%93Markov_theorem.