Linear models, regularization and selection

Nicolas Keriven
CNRS, IRISA, Rennes

ENSTA 2024

Model based approaches

Reminder on Supervised Learning

- input data $x \in \mathbb{R}^d$
- reponse y to be predicted
- training set $(x_1, x_1), \ldots, (x_n, x_n)$

In a model based approach, we seek an explicit relation between the (input) data x and the response y.
We focus here on *Discriminative models*, where we just model explicitly the conditional distribution $P(Y|X = x)$ rather than the joint distribution $P(X, Y)$

Model based approaches : Generative vs Discriminative methods

Generative methods

Deduction of $P(Y|X)$ from Bayes rule

- Linear or Quadratic Discriminant Analysis
- Naïve Bayes
- ...

Discriminative methods

Direct learning of $P(Y|X)$, e.g.

- Linear regression
- Logistic "regression" ($\leftarrow$ generalized linear model for classification tasks)
- ...

Linear model : Keep it simple !

Simple linear approach may seem overly simplistic

- true prediction functions are never linear

+ extremely useful, both conceptually and practically

Linear model : Keep it simple !

Simple linear approach may seem overly simplistic

- true prediction functions are never linear
+ extremely useful, both conceptually and practically

Practically

Gorge Box, 60' : "Essentially, all models are wrong, but some are very useful"

☞ *Simple is actually very good* : works very well in a lot of situations by capturing the main effects (which are generally the most interesting)

Linear model : Keep it simple !

Simple linear approach may seem overly simplistic

- true prediction functions are never linear
+ extremely useful, both conceptually and practically

Practically

Gorge Box, 60' : "Essentially, all models are wrong, but some are very useful"

☞ *Simple is actually very good* : works very well in a lot of situations by capturing the main effects (which are generally the most interesting)

Conceptually

Many concepts developped for the linear problem are important for a lot of the supervised learning techniques

☞ Although it is nerver correct, a linear model serves as a good and interpretable approximation of the unknown true function $f(X)$

Outline

Linear Regression Problem

Training data $x_i = [x_i^1, \ldots, x_i^d]^\top \in \mathbb{R}^d$, $y_i \in \mathbb{R}$ for $i = 1, \ldots, n$

Linear *Regression* Model

$$y_i \approx \beta_0 + \sum_{j=1}^{d} \beta_j x_i^j + \sigma \epsilon_i$$

- ϵ_i is a centered noise with unit variance ($\mathbb{E}[\epsilon_i] = 0$, $\mathrm{var}(\epsilon_i) = 1$)
- β_0 is the "intercept" (reduces to the ordinate at the origin when $d = 1$)
- $\beta = (\beta_0, \ldots, \beta_d) \in \mathbb{R}^{d+1}$ is the coefficient vector

Objective : estimation of β using the samples in the training set ← supervised learning problem

Linear Regression Problem

Training data $x_i = [x_i^1, \dots, x_i^d]^\top \in \mathbb{R}^d$, $y_i \in \mathbb{R}$ for $i = 1, \dots, n$

Linear *Regression* Model

$$y_i \approx \beta_0 + \sum_{j=1}^{d} \beta_j x_i^j + \sigma \epsilon_i$$

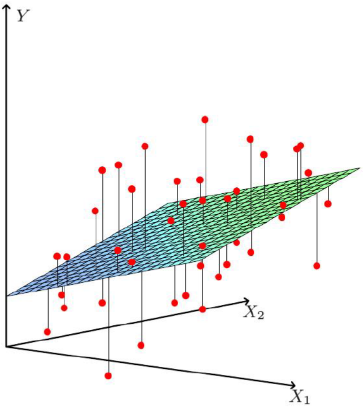
- ϵ_i is a centered noise with unit variance ($\mathbb{E}[\epsilon_i] = 0$, $\mathrm{var}(\epsilon_i) = 1$)
- β_0 is the "intercept" (reduces to the ordinate at the origin when $d = 1$)
- $\beta = (\beta_0, \dots, \beta_d) \in \mathbb{R}^{d+1}$ is the coefficient vector

Objective : estimation of β using the samples in the training set ← supervised learning problem

Remark : model linear w.r.t. β, but not necessarily linear w.r.t.

- the inputs x_i : we can add non linear predictors $h(x_1, \dots, x_p)$ in the model, e.g. x_i^2, $x_i x_j \dots$
- the outputs y_i : we can introduce a non linear link function ← generalized linear model, e.g. logistic regression

Least Squares (LS) Estimator



Linear least squares fitting with $x \in \mathbb{R}^2$

$$y_i \approx \beta_0 + \sum_{j=1}^{d} \beta_j x_i^j + \sigma \epsilon_i$$

LS estimate defined by minimizing the Residual Sum of Squares $RSS(\beta)$:

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_j \beta_j x_j^j \right)^2$$

- $\text{RSS}(\beta) \propto$ empirical risk for quadratic loss

Least Squares Estimator (Cont'd)

$$\hat{\beta} = \arg\min_{\beta} \mathrm{RSS}(\beta), \quad \text{where } \mathrm{RSS}(\beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_i^j \right)^2$$

Matrix expression of RSS

$$\mathrm{RSS}(\beta) = ||Y - X\beta||_2^2,$$

$$\text{where } Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n, \quad X = \begin{pmatrix} 1 & x_1^1 & \dots & x_1^d \\ \vdots & \vdots & & \vdots \\ 1 & x_n^1 & \dots & x_n^d \end{pmatrix} \in \mathbb{R}^{n \times (d+1)},$$

LS Estimator derivation

Gradient

$RSS(\beta)$ is a convex function, minimized by setting its gradient to 0 :

$$\nabla RSS(\beta) = 2X^\top(X\beta - Y) = 0 \Leftrightarrow (X^\top X)\beta = X^\top Y$$

LS Estimator derivation

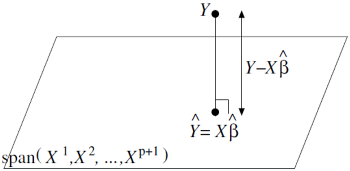
Gradient

$RSS(\beta)$ is a convex function, minimized by setting its gradient to 0 :

$$\nabla RSS(\beta) = 2X^\top(X\beta - Y) = 0 \Leftrightarrow (X^\top X)\beta = X^\top Y$$

Orthogonality principle

Another way to obtain this is to consider that $\hat{Y} = X\hat{\beta}$ is the orthogonal projection of Y onto the space spanned by the column vectors of X. Let X^j be the jth column of X



$$\begin{aligned} & \langle X^j, Y - X\beta\rangle = 0 \\ \Leftrightarrow \quad & X^\top (Y - X\beta) = 0 \\ \Leftrightarrow \quad & \left(X^\top X\right)\hat{\beta} = X^\top Y \end{aligned}$$

LS Estimator computation

In the rest of the section, assume rank $X = d + 1$, and thus $X^\top X$ is invertible.

☞ for this, it is necessary that $n > d$

Analytical expression

Since $(X^\top X)\,\hat{\beta} = X^\top Y$,

$$\hat{\beta} = \left(X^\top X\right)^{-1} X^\top Y$$

LS Estimator computation

In the rest of the section, assume rank $X = d + 1$, and thus $X^\top X$ is invertible.

☞ for this, it is necessary that $n > d$

Analytical expression

Since $\left(X^\top X\right) \hat{\beta} = X^\top Y$,

$$\hat{\beta} = \left(X^\top X\right)^{-1} X^\top Y$$

Numerical computation in high dimension

Pb : When $d > 10^3$ or $d > 10^4$, too expansive to compute $\left(X^\top X\right)^{-1}$ ...

- Can use tricks to speed up computation of the inverse/SVD (like PCA)

☞ more efficient to use a numerical procedure to minimize the RSS and directly approximate $\hat{\beta}$! E.g. gradient descent (see next section)

LS Estimator properties

For a known X, $Y = X\beta + \sigma\varepsilon$ where $\mathbb{E}[\varepsilon] = 0_n$ and $\operatorname{cov}\varepsilon = I_n$ (only ϵ is random).

LS Estimator properties

For a known X, $Y = X\beta + \sigma\varepsilon$ where $\mathbb{E}[\varepsilon] = 0_n$ and $\operatorname{cov}\varepsilon = I_n$ (only ϵ is random).

- $\hat{\beta}$ is an unbiased estimator of β

$$\mathbb{E}[\hat{\beta}] = \mathbb{E}\left[\left(X^\top X\right)^{-1} X^\top Y\right] = \left(X^\top X\right)^{-1} X^\top X\beta + \left(X^\top X\right)^{-1} X^\top \mathbb{E}[\varepsilon] = \beta$$

LS Estimator properties

For a known X, $Y = X\beta + \sigma\varepsilon$ where $\mathbb{E}[\varepsilon] = 0_n$ and $\operatorname{cov}\varepsilon = I_n$ (only ϵ is random).

- $\hat\beta$ is an unbiased estimator of β

$$\mathbb{E}[\hat\beta] = \mathbb{E}\left[\left(X^\top X\right)^{-1} X^\top Y\right] = \left(X^\top X\right)^{-1} X^\top X\beta + \left(X^\top X\right)^{-1} X^\top \mathbb{E}[\varepsilon] = \beta$$

- Covariance

$$\operatorname{cov}\hat\beta = \mathbb{E}(\hat\beta\hat\beta^\top) = \left(X^\top X\right)^{-1} X^\top \underbrace{\operatorname{cov}(Y)}_{\sigma^2 I_n} X \left(X^\top X\right)^{-1} = \sigma^2 \left(X^\top X\right)^{-1}$$

LS Estimator properties

For a known X, $Y = X\beta + \sigma\varepsilon$ where $\mathbb{E}[\varepsilon] = 0_n$ and $\operatorname{cov}\varepsilon = I_n$ (only ϵ is random).

- $\hat{\beta}$ is an unbiased estimator of β

$$\mathbb{E}[\hat{\beta}] = \mathbb{E}\left[\left(X^\top X\right)^{-1} X^\top Y\right] = \left(X^\top X\right)^{-1} X^\top X\beta + \left(X^\top X\right)^{-1} X^\top \mathbb{E}[\varepsilon] = \beta$$

- Covariance

$$\operatorname{cov}\hat{\beta} = \mathbb{E}(\hat{\beta}\hat{\beta}^\top) = \left(X^\top X\right)^{-1} X^\top \underbrace{\operatorname{cov}(Y)}_{\sigma^2 I_n} X \left(X^\top X\right)^{-1} = \sigma^2 \left(X^\top X\right)^{-1}$$

- MSE (Power of the estimation error)

$$\mathbb{E}\left[||\hat{\beta} - \beta||^2\right] = \mathbb{E}\left[\left(\hat{\beta} - \beta\right)^\top \left(\hat{\beta} - \beta\right)\right] = \mathbb{E}\left[\operatorname{trace}\left(\hat{\beta} - \beta\right)\left(\hat{\beta} - \beta\right)^\top\right],$$

$$= \operatorname{trace}\left(\operatorname{cov}\hat{\beta}\right) = \sigma^2 \operatorname{trace}\left(\left(X^\top X\right)^{-1}\right) = \sigma^2 \sum_{j=1}^{d+1} \frac{1}{\lambda_j}$$

where $\lambda_i > 0$ are the eigenvalues of the symm. def. pos. matrix $X^\top X$. What happens for $\lambda_i \approx 0$? That is, $X^\top X$ is "**badly conditioned**"

LS Estimator properties

Noise variance estimator

An unbiased estimate of the noise variance σ^2 can be deduced as

$$\hat{\sigma}^2 = \frac{1}{n-p-1} \mathrm{RSS}\left(\hat{\beta}\right),$$

LS Estimator properties

Noise variance estimator

An unbiased estimate of the noise variance σ^2 can be deduced as

$$\hat{\sigma}^2 = \frac{1}{n-p-1}\mathrm{RSS}\left(\hat{\beta}\right),$$

Gaussian noise $\varepsilon \sim \mathcal{N}(0, I_n)$

- $\hat{\beta}$ is $\mathcal{N}\left(\beta, \sigma^2 \left(X^\top X\right)^{-1}\right)$ distributed,
- LS estimator $\hat{\beta}$ is also the maximum likelihood estimator

Outline

Reminder on Steepest descent, aka gradient descent

We can define the criterion to be minimized as $J(\beta)$ here $J(\beta) = \frac{1}{2}\mathrm{RSS}(\beta)$

Gradient descent (steepest descent)

- Ubiquitous iterative procedure based on the observation that $J(\beta)$ decreases fastest in the direction of the negative gradient of $J(\cdot)$ at β, ie $-\nabla J(\beta)$.
- Powers most modern ML framework ("training" has become synonymous for "applying iterative optimization")
- Extremely perfected in modern (Python) librairies, with automatic computation of gradients ! (autodiff)

Algorithm

- Starts at β^0
- For $t = 1, \ldots, T$,

$$\beta_{t+1} = \beta_t - \eta_t \nabla_\beta J(\beta_t),$$

- $\eta_t \in \mathbb{R}_+$ is the learning rate
- If $\beta^t \to \beta^\infty$, the eq. above becomes $\nabla J(\beta^\infty) = 0$!

Stochastic Gradient Descent (SGD)

Consider the minimization of a function that can be decomposed as a sum over the sample :

$$J(\beta) = \frac{1}{n} \sum_i J_i(\beta)$$

- Eg, any empirical risk such as the RSS, where $J_i(\beta) = L(\beta, x_i, y_i)$

Pb : For large datasets (high n), the full gradient $\nabla_\beta J$ is often too expensive to compute

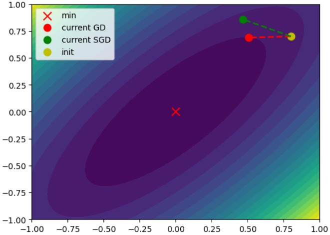
☞ stochastic approximation of the batch gradient to decrease the computational burden

Stochastic gradient descent (SGD)

At each iteration t :

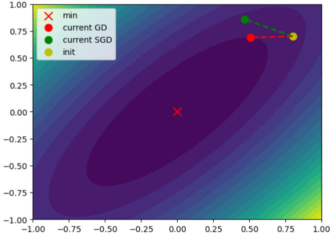
- Sample b indices $i_1, \ldots, i_b$ between 1 and n
- Approximate $J(\beta) \approx J_b(\beta) = \frac{1}{b} \sum_j J_{i_j}(\beta)$
- Take a gradient step $\beta^{t+1} = \beta^t - \eta_t \nabla J_b(\beta)$

SGD : properties



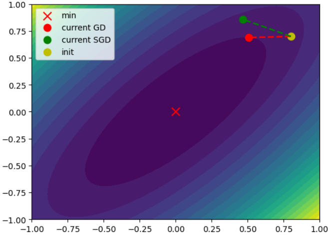
- $i_1, \ldots, i_b$ is called a mini-batch. $b = 1$ is "pure" SGD, $b = n$ is regular GD (also called full-batch GD). Often $b = 32$, 64 or 128 (convenient for GPUs), it is an important hyperparameter (crossval).

SGD : properties



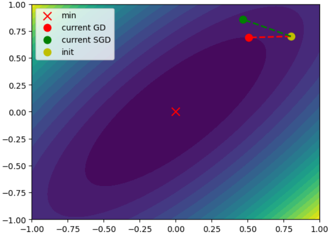
- $i_1, \ldots, i_b$ is called a mini-batch. $b = 1$ is "pure" SGD, $b = n$ is regular GD (also called full-batch GD). Often $b = 32$, 64 or 128 (convenient for GPUs), it is an important hyperparameter (crossval).
- When sampling without replacement, a full pass is called an epoch

SGD : properties



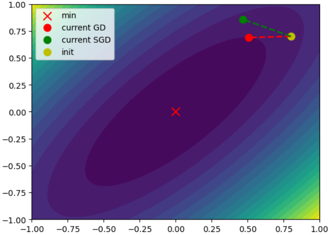
- $i_1, \ldots, i_b$ is called a mini-batch. $b = 1$ is "pure" SGD, $b = n$ is regular GD (also called full-batch GD). Often $b = 32$, 64 or 128 (convenient for GPUs), it is an important hyperparameter (crossval).
- When sampling without replacement, a full pass is called an epoch
- When b increases, SGD becomes smoother, but more memory-intensive.
  - !! : often in practice, since batches are usually paralleled over GPUs, a larger b can mean a faster epoch. But memory might become an issue.

SGD : properties



- $i_1, \ldots, i_b$ is called a mini-batch. $b = 1$ is "pure" SGD, $b = n$ is regular GD (also called full-batch GD). Often $b = 32$, 64 or 128 (convenient for GPUs), it is an important hyperparameter (crossval).
- When sampling without replacement, a full pass is called an epoch
- When b increases, SGD becomes smoother, but more memory-intensive.
  - !! : often in practice, since batches are usually paralleled over GPUs, a larger b can mean a faster epoch. But memory might become an issue.
- if η_k is not too big, each GD step is ensured to decrease $J(\beta)$, that is not the case for SGD (but almost !)

SGD : properties



- $i_1, \ldots, i_b$ is called a mini-batch. $b = 1$ is "pure" SGD, $b = n$ is regular GD (also called full-batch GD). Often $b = 32$, 64 or 128 (convenient for GPUs), it is an important hyperparameter (crossval).
- When sampling without replacement, a full pass is called an epoch
- When b increases, SGD becomes smoother, but more memory-intensive.
  !! : often in practice, since batches are usually paralleled over GPUs, a larger b can mean a faster epoch. But memory might become an issue.
- if η_k is not too big, each GD step is ensured to decrease $J(\beta)$, that is not the case for SGD (but almost !)
- SGD + "autodiff" are the workhorses of deep learning

Outline

Limitations of Least Squares Estimators (LSE)

Recall : $\hat{\beta} = (X^\top X)^{-1} X^\top Y$

Problem

When $\text{rank}(X) < d$, or when X has singular values close to zero, then $X^\top X$ is no more invertible, or ill conditioned (eigenvalues close to zero)...

Causes

- redundant or nearly-collinear predictors : useless features
- high dimensional problem where $d \approx n$ (or $d > n$) : $\text{rank}(X) \leq \min(d, n)$.

Limitations of Least Squares Estimators (LSE)

Recall : $\hat{\beta} = (X^\top X)^{-1} X^\top Y$

Problem

When $\mathrm{rank}(X) < d$, or when X has singular values close to zero, then $X^\top X$ is no more invertible, or ill conditioned (eigenvalues close to zero)...

Causes

- redundant or nearly-collinear predictors : useless features
- high dimensional problem where $d \approx n$ (or $d > n$) : $\mathrm{rank}(X) \leq \min(d, n)$.

Effects

No single, or stable, solution for $\hat{\beta}$

- high variance of $\hat{\beta}$ as an eigenvalue λ_i of $X^\top X$ is close to zero ($||\hat{\beta}|| \to +\infty$ as $\lambda_i \to 0$),
- true error rate $\sigma^2 \sum_i \frac{1}{\lambda_i^2}$ explodes since a small perturbation in the training set yields a substantially different estimate $\hat{\beta}$ and prediction rule $\hat{y} = x^\top \hat{\beta}$

☞ over-fitting problem : the bias is good (zero), but the variance is very high !

Instability of LSE : Deconvolution illustration

- $y \in \mathbb{R}^q$ with $q = 256^2$, $\beta \in \mathbb{R}^d$ with $d = 256^2$. X is a blurring operator.

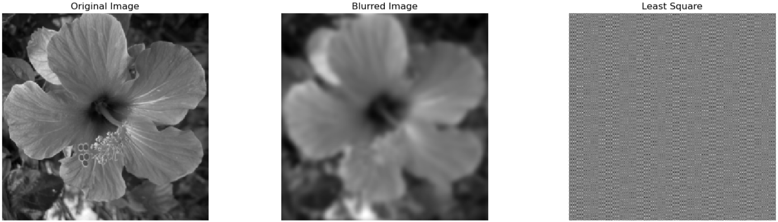
Original Image

Blurred Image

$\beta \leftarrow$ original image $\quad$ $y = X\beta \leftarrow$ blurred image

Instability of LSE : Deconvolution illustration

- $y \in \mathbb{R}^q$ with $q = 256^2$, $\beta \in \mathbb{R}^d$ with $d = 256^2$. X is a blurring operator.



$\beta \leftarrow$ original image $\quad y = X\beta \leftarrow$ blurred image $\quad \hat{\beta} = (X^\top X)^{-1} X^\top y$

Due to the bad conditioning of $X^\top X$ (e.v. close to zero), the noise (here numerical round-off errors) is multiplied by an almost infinite gain, and the estimated coefficients $\hat{\beta}_j$ explode to $\pm\infty$!

Regularization methods

Supplementary materials

- Prof. A. Ihler short (8mn) and educational video https://www.youtube.com/watch?v=sO4ZirJh9ds
- Wikipedia page https://en.wikipedia.org/wiki/Regularized_least_squares#Specific_examples
- Scikit-learn nice documentation with examples (can stop just before section 1.1.4) https://scikit-learn.org/stable/modules/linear_model.html

Regularization : shrinkage

Idea : introducing a little bias in the estimation of β may lead to a substantial decrease in variance and, hence, in the true error rate

Regularization : shrinkage

Idea : introducing a little bias in the estimation of β may lead to a substantial decrease in variance and, hence, in the true error rate

Penalized regression

Regularize the estimation problem by introducing a penalization term for β

$$\widetilde{\beta} = \arg\min_{\beta} \left[\mathrm{RSS}(\beta) + \lambda\,\mathrm{Pen}(\beta)\right]$$

Regularization : shrinkage

Idea : introducing a little bias in the estimation of β may lead to a substantial decrease in variance and, hence, in the true error rate

Penalized regression

Regularize the estimation problem by introducing a penalization term for β

$$\widetilde{\beta} = \arg\min_{\beta} \left[\mathrm{RSS}(\beta) + \lambda\, \mathrm{Pen}(\beta)\right]$$

- $\mathrm{RSS}(\beta)$ is the *fidelity term* to the training set : it promotes Y close to $X\beta$
- $\mathrm{Pen}(\beta)$ is the *a priori* to regularize the solution : it promotes "desirable properties" of β
- $\lambda > 0$ is the penalization coefficient, it tunes the balances between the two terms. Standard practice is to use cross-validation to estimate an optimal λ for the test

Table of Contents

Ridge regression

Since we want to avoid $\beta \to \infty$, let's penalize its (squared) norm :

$$\mathrm{Pen}(\beta) \equiv \beta^\top \beta = ||\beta||_2^2, \quad \leftarrow \textit{Tychonov} \text{ regularization (40's)}$$

$\widetilde{\beta}$ is thus obtained by minimizing

$$\mathrm{RSS}(\beta) + \lambda \, \mathrm{Pen}(\beta) = \|Y - X\beta\|^2 + \lambda\|\beta\|^2$$

Putting the gradient to 0, we obtain the Ridge estimator :
$\tilde{\beta} = (X^\top X + \lambda I)^{-1} X^\top Y$

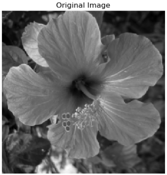
Remark

similar to LSE, with an additional 'ridge' on the diagonal of $X^\top X$

- $X^\top X + \lambda I$ has all its eigenvalues greater than $\lambda > 0$, $\leftarrow$ ensures that $\widetilde{\beta}$ is always defined, and stable for large enough λ
- ☞ when $\lambda \to 0$, then $\widetilde{\beta} \to \hat{\beta}$ (over-fitting risk),
- ☞ when $\lambda \to +\infty$, then $\widetilde{\beta} \to 0$ (under-fitting)

Ridge Regression : deconvolution illustration

- $y \in \mathbb{R}^n$ with $n = 256^2$, $\beta \in \mathbb{R}^p$ with $d = 256^2$,
- $X \in \mathbb{R}^{n \times p} \leftarrow$ sized $(256^2) \times (256^2)$ matrix...



$\beta \leftarrow$ original image

$y = X\beta \leftarrow$ blurred image

Ridge Regression : deconvolution illustration

- $y \in \mathbb{R}^n$ with $n = 256^2$, $\beta \in \mathbb{R}^p$ with $d = 256^2$,
- $X \in \mathbb{R}^{n\times p} \leftarrow$ sized $(256^2) \times (256^2)$ matrix...

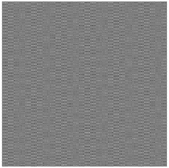
Original Image

$\beta \leftarrow$ original image

Blurred Image

$y = X\beta \leftarrow$ blurred image

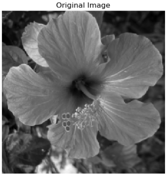
Least Square



$\hat{\beta}=(X^\top X)^{-1}X^\top y \leftarrow$ LS estimate

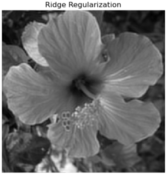
Ridge Regression : deconvolution illustration

- $y \in \mathbb{R}^n$ with $n = 256^2$, $\beta \in \mathbb{R}^p$ with $d = 256^2$,
- $X \in \mathbb{R}^{n \times p} \leftarrow$ sized $(256^2) \times (256^2)$ matrix...

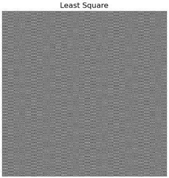


$\beta \leftarrow$ original image

$y = X\beta \leftarrow$ blurred image



$\widetilde{\beta} = (X^\top X + \lambda I)^{-1} X^\top y \leftarrow$ ridge estimate



$\hat{\beta} = (X^\top X)^{-1} X^\top y \leftarrow$ LS estimate

Kernel Ridge Regression

- Consider $d > n$. Computing

$$\hat{\beta} = (X^\top X + \lambda Id)^{-1} X^\top Y$$

still costs $O(d^3)$ to compute.

Kernel Ridge Regression

- Consider $d > n$. Computing

$$\hat{\beta} = (X^\top X + \lambda Id)^{-1} X^\top Y$$

still costs $O(d^3)$ to compute.

- However, as in PCA, we can invert a matrix of size n instead. From the first order condition on the gradient, we know that β is of the form $\beta = X^\top \alpha$. Replacing in the gradient, we get

$$\alpha = (XX^\top + \lambda Id)^{-1} Y = (K + \lambda Id)^{-1} Y$$

where $K = [\langle x_i, x_j \rangle]_{ij}$

- A prediction becomes $h(x) = \beta^\top x = \sum_i \alpha_i \langle x_i, x \rangle$

Kernel Ridge Regression

- Consider $d > n$. Computing

$$\hat{\beta} = (X^\top X + \lambda Id)^{-1} X^\top Y$$

still costs $O(d^3)$ to compute.

- However, as in PCA, we can invert a matrix of size n instead. From the first order condition on the gradient, we know that β is of the form $\beta = X^\top \alpha$. Replacing in the gradient, we get

$$\alpha = (XX^\top + \lambda Id)^{-1} Y = (K + \lambda Id)^{-1} Y$$

where $K = [\langle x_i, x_j \rangle]_{ij}$

- A prediction becomes $h(x) = \beta^\top x = \sum_i \alpha_i \langle x_i, x \rangle$

- Everything only depends on the dot product ! We can apply the kernel trick and replace it with a psd kernel $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$

KRR : illustration

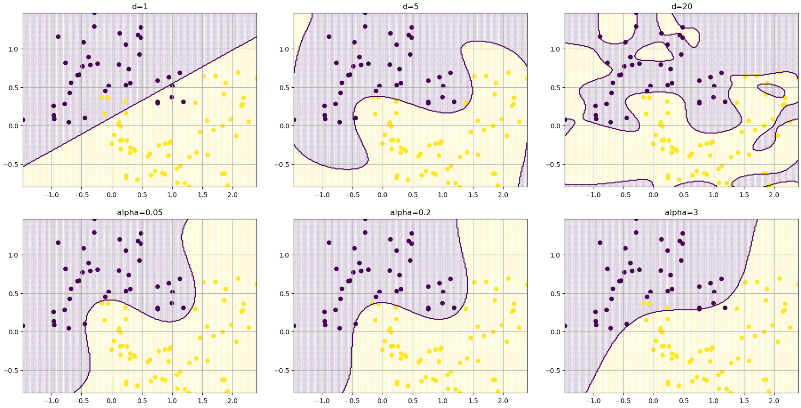


Table of Contents

Regularization by promoting sparsity

Sparse representations/approximations

A representation, or an approximation, is said to be sparse when most of the coefficients are zero. Having β sparse $\Leftrightarrow$ some features are not useful for regression. Sparse methods are feature selection methods.

Regularization by promoting sparsity

Sparse representations/approximations

A representation, or an approximation, is said to be sparse when most of the coefficients are zero. Having β sparse $\Leftrightarrow$ some features are not useful for regression. Sparse methods are feature selection methods.

'Bet on Sparsity' principle

Sparsity is a good option in high dimension !

- if the sparsity assumption *does not hold*, there are infinitely many β for which $Y = X\beta$ when $d > n$. The problem is ill-posed.
- but if sparsity holds, then it is often possible to recover the true β

Regularization by promoting sparsity

Sparse representations/approximations

A representation, or an approximation, is said to be sparse when most of the coefficients are zero. Having β sparse $\Leftrightarrow$ some features are not useful for regression. Sparse methods are feature selection methods.

'Bet on Sparsity' principle

Sparsity is a good option in high dimension !

- if the sparsity assumption *does not hold*, there are infinitely many β for which $Y = X\beta$ when $d > n$. The problem is ill-posed.
- but if sparsity holds, then it is often possible to recover the true β
- Sparsity is the basis of compressed sensing (90's, 00's) : a very beautiful theory saying that ill-posed problems are sometimes solvable, and the NP-hard procedure to solve them can be approximated by an efficient one that gives the same solution. Hugely influential in many areas.

Lasso ('least absolute shrinkage and selection operator') estimator

Idea : penalize β by its number of non-zero coefficients, the "ℓ_0 pseudo-norm" $\|\beta\|_0 = \sharp\{\beta_i \neq 0\}$. But the optimization becomes NP-hard !

Definition

Lasso : replace the ℓ_0 norm by its "convex approximation" : the ℓ_1 norm $\|\beta\|_1 = \sum_i |\beta_i|$.

$$\widetilde{\beta}_{\mathrm{lasso}} = \arg\min_{\beta} \left[\mathrm{RSS}(\beta) + \lambda \,||\beta||_1\right],$$

- unlike Ridge regression no analytical expression of $\widetilde{\beta}_{\mathrm{lasso}}$
- One could apply GD, but $\|\cdot\|_1$ is technically not differentiable ! (in $\beta_i = 0$)
- One could still apply GD and cross fingers (like ReLU in deep learning)... but we never obtain exactly $\beta_i = 0$, which is precisely what we are interested in ! !
- Dedicated convex optimization methods for Lasso, eg ISTA, LARS, FISTA...

Lasso : geometric interpretation

Why does the ℓ_1 norm still promotes sparsity ?

Lasso : geometric interpretation

Why does the ℓ_1 norm still promotes sparsity ?

Consider the following problem : $\min_{\|\beta\|_p \leq \tau} \mathrm{RSS}(\beta)$, where $p = 1$ for Lasso and $p = 2$ for Ridge. (It is not the exact same formulation, but they can be shown to be mostly equivalent).

Lasso : geometric interpretation

Why does the ℓ_1 norm still promotes sparsity ?

Consider the following problem : $\min_{\|\beta\|_p \leq \tau} \mathrm{RSS}(\beta)$, where $p = 1$ for Lasso and $p = 2$ for Ridge. (It is not the exact same formulation, but they can be shown to be mostly equivalent).

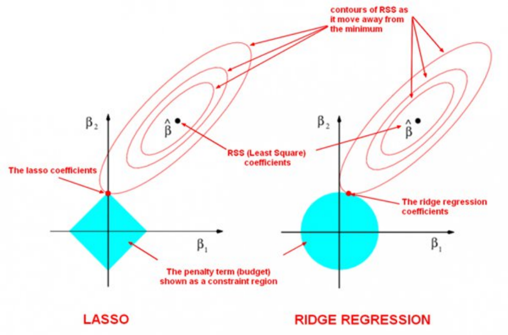
Exo : Now, draw the level sets of the RSS (ellipsoids), and the constraint regions. The solution is given by the ellipsoid that "touches" the region.

Lasso : geometric interpretation

Why does the ℓ_1 norm still promotes sparsity ?

Consider the following problem : $\min_{\|\beta\|_p \leq \tau} \mathrm{RSS}(\beta)$, where $p = 1$ for Lasso and $p = 2$ for Ridge. (It is not the exact same formulation, but they can be shown to be mostly equivalent).

Exo : Now, draw the level sets of the RSS (ellipsoids), and the constraint regions. The solution is given by the ellipsoid that "touches" the region.



Scale your data !

- Linear models (w/o regularization) are invariant under the scaling of the variables : the prediction function is unchanged.
- Regularized linear models are not due to the penalty term : scaling of the variables matters !

☞ the variables that have the greatest magnitudes are favoured (same problem for distance based ML methods s.t. K-NN, SVM, ...)

Practical advices

- If the variables are in different units, scaling each is strongly recommended.
- If they are in the same units, you might or might not scale the variables (depend on your problem)

Usual scaling methods

- normalization in $[0, 1]$: $\tilde{x}_i = \dfrac{x_i - \min_i}{\max_i - \min_i}$
- standardization to get zero mean and unit variance : $\tilde{x}_i = \dfrac{x_i - \mu_i}{\sigma_i}$

Table of Contents

Application : prostate data

Stamey et al. (1989) study to examine the association between prostate specific antigen (PSA) and several clinical measures that are potentially associated with PSA in men. Objective is to predict the Log PSA (supervised regression problem) from eight variables

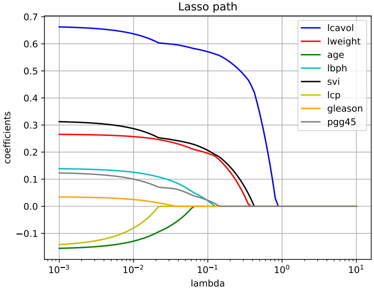
- lcavol : Log cancer volume
- lweight : Log prostate weight
- age : The man's age
- lbph : Log of the amount of benign hyperplasia
- svi : Seminal vesicle invasion ; 1=Yes, 0=No
- lcp : Log of capsular penetration
- gleason : Gleason score
- pgg45 : Percent of Gleason scores 4 or 5

Application : prostate data

Lasso estimate (ℓ_1-penalization) : $\widetilde{\beta}(\lambda) = \arg\min_\beta \mathrm{RSS}(\beta) + \lambda||\beta||_1$,

Lasso path : We can plot the estimated variable coeffs $\widetilde{\beta}(\lambda)_j$ vs λ

- For large λ all the coefficients ar zeros ($||\widetilde{\beta}(\lambda)||_1 = 0$)
- When $\lambda \searrow$ then $||\widetilde{\beta}(\lambda)||_1 \nearrow$: most significant variables sequentially enters the model (non-zero coeffs)



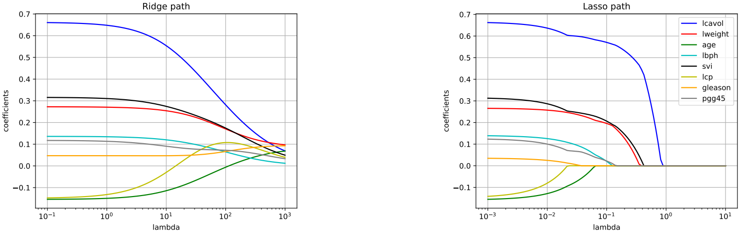
Choosing λ

- small $\lambda \to$ overfitting
- large $\lambda \to$ underfitting
- cross-validation estimation of λ yields $\lambda = 0.21$

$\Rightarrow$ only 3 predictors enter the model to predict PSA : lcavol, svi, lweight

Application : prostate data

Comparison of ridge and lasso estimators



Path of the penalized coefficients as a function of λ

- Ridge estimates are smooth functions of λ, with coefficients that are never stuck at zero.
- Shrinkage effect : the larger λ, the more the coefficients are shrunken toward 0 for both penalties
- For small λ, thus large $||\widetilde{\beta}(\lambda)||$, both estimator becomes equivalent (convergence toward LSE)

Outline

Table of Contents

Discriminative model for classification : $Y \in \mathcal{Y} \leftarrow$ discrete set

Discriminative model

For a given $X = x$, we want to model directly

$$\mathbb{P}(Y = k | X = x)$$

for each value of the class label $k \in \mathcal{Y}$

- do not require to specify the marginal distribution of the inputs X
- the loss functions are generally more complicated than regression !

Discriminative model for classification : $Y \in \mathcal{Y} \leftarrow$ discrete set

Discriminative model

For a given $X = x$, we want to model directly

$$\mathbb{P}(Y = k|X = x)$$

for each value of the class label $k \in \mathcal{Y}$

- do not require to specify the marginal distribution of the inputs X
- the loss functions are generally more complicated than regression !

Model-based classification rule

We predict the class with the highest probability

$$\hat{Y} = \arg\max_k \mathbb{P}(Y = k|X = x)$$

- If we had access to the true $\mathbb{P}(Y = k|X = x)$, this is the optimal rule for misclassication rate referred to as *Bayes Classifier*

How can we use linear **regression** to model a probability $\mathbb{P}(Y = k|X = x)$?

Linear model for classification : Logistic regression (LR)

Classification problem $Y \in \mathcal{Y} \leftarrow$ discrete set

Binary classification problem : $\mathcal{Y} = \{1, 2\}$

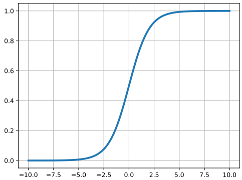
Consider the following model

$$\mathbb{P}(Y_i = 1 | X_i = x_i) = \phi(x_i^\top \beta) = \frac{\exp(x_i^\top \beta)}{1 + \exp(x_i^\top \beta)},$$

where

- $x_i = (1, x_i^1, \dots, x_i^d)^\top \in \mathbb{R}^{d+1} \leftarrow$ intercept term included by default,
- ϕ is the logistic function : maps a real value to a probability between 0 and 1

$$\begin{aligned} \phi \ : \mathbb{R} &\rightarrow (0, 1) \\ u &\mapsto \frac{\exp u}{1+\exp u} = \frac{1}{1+\exp(-u)}. \end{aligned}$$



LR is a generalized linear model

We still linearly combine the features to obtain the desired quantity.

Consider

- $p_i \equiv \Pr(Y_i = 1 | X_i = x_i) = \phi(x_i^\top \beta)$
- ϕ^{-1} : $p \in (0,1) \mapsto \log \frac{p}{1-p} \in \mathbb{R}$ is the logit function

Generalized linear model

- Linear equation w.r.t. β :

$$\text{logit}\,(p_i) = x_i^\top \beta,$$

- + additional nonlinear constraint (proba sum to 1) :

$$\Pr(Y_i = 2 | X_i = x_i) = 1 - p_i = \frac{1}{1 + \exp\left(x_i^\top \beta\right)}$$

Logistic regression for classification in K classes

Multiclass problem : $\mathcal{Y} = \{1, 2, \ldots, K\}$

LR can be easily extented to *multinomial* LR :

- The output of the linear model is $K-1$-dimensional : $\beta \in \mathbb{R}^{d\times(K-1)}$ and $v = \beta^\top x_i \in \mathbb{R}^{K-1}$, and normalization $v_K = 1$ (the model does not change up to an additive constant)

Logistic regression for classification in K classes

Multiclass problem : $\mathcal{Y} = \{1, 2, \dots, K\}$

LR can be easily extented to *multinomial* LR :

- The output of the linear model is $K-1$-dimensional : $\beta \in \mathbb{R}^{d\times(K-1)}$ and $v = \beta^\top x_i \in \mathbb{R}^{K-1}$, and normalization $v_K = 1$ (the model does not change up to an additive constant)
- It is transformed into a probability distribution that sum to one through $\mathbb{P}(Y = c|X = x) = \frac{e^{v_c}}{\sum_p e^{v_p}}$

Logistic regression for classification in K classes

Multiclass problem : $\mathcal{Y} = \{1, 2, \dots, K\}$

LR can be easily extented to *multinomial* LR :

- The output of the linear model is $K-1$-dimensional : $\beta \in \mathbb{R}^{d \times (K-1)}$ and $v = \beta^\top x_i \in \mathbb{R}^{K-1}$, and normalization $v_K = 1$ (the model does not change up to an additive constant)
- It is transformed into a probability distribution that sum to one through $\mathbb{P}(Y = c|X = x) = \frac{e^{v_c}}{\sum_p e^{v_p}}$

Equivalently,

$$
\begin{aligned}
\log \frac{\Pr(Y_i=1|X_i=x_i)}{\Pr(Y_i=K|X_i=x_i)} &= x_i^\top \beta_1 \\
\log \frac{\Pr(Y_i=2|X_i=x_i)}{\Pr(Y_i=K|X_i=x_i)} &= x_i^\top \beta_2, \\
&\vdots \\
\log \frac{\Pr(Y_i=K-1|X_i=x_i)}{\Pr(Y_i=K|X_i=x_i)} &= x_i^\top \beta_{K-1},
\end{aligned}
$$

☞ $K-1$ equations + sum-to-one constraint on the probabilities

Table of Contents

Parameter estimation

Since we model the distribution of $Y|X$, the log-likelihood expresses as

$$\begin{aligned}\ell(\beta) &= \sum_{i=1}^{n} \log p_{X,Y}(x_i, y_i|\beta), \\ &= \sum_{i=1}^{n} \log p_{Y|X}(y_i|x_i, \beta) + \sum_{i=1}^{n} \log p(x_i), \quad \leftarrow \text{Bayes rule}\end{aligned}$$

where the second term is a constant that does not depend on β

Parameter estimation

Since we model the distribution of $Y|X$, the log-likelihood expresses as

$$\begin{aligned}\ell(\beta) &= \sum_{i=1}^{n} \log p_{X,Y}(x_i, y_i|\beta), \\ &= \sum_{i=1}^{n} \log p_{Y|X}(y_i|x_i, \beta) + \sum_{i=1}^{n} \log p(x_i), \quad \leftarrow \text{Bayes rule}\end{aligned}$$

where the second term is a constant that does not depend on β

Maximum likelihood estimator

Maximizing the log-likelihood w.r.t $\beta \Leftrightarrow$ Maximizing the conditional log-likelihood $\beta \mapsto \sum_{i=1}^{n} \log p_{Y|X}(y_i|x_i, \beta)$

- no analytical expression of the ML estimator,
- iterative procedure : Stochastic Gradient Descent especially for regularized estimation, Newton-Raphson...
- Rk : regularized estimator defined as $\widetilde{\beta} = \arg\min_\beta -\ell(\beta) + \lambda \mathrm{Pen}(\beta)$: same idea, reduce variance (Tikhonov) or promote sparsity (ℓ_1)

Table of Contents

Application : South African coronary heart disease (CHD) [1]

A retrospective sample of males in a coronary heart-disease (CHD) high-risk region of the Western Cape, South Africa.

Matrix of the predictor scatterplots

Each plot $\equiv$ pair of risk factors. Here 7 predictors :

- *sbp* : systolic blood pressure,
- *tobacco* : cumulative tobacco consumption (kg),
- *ldl* : $\sim$ cholesterol,
- *famhist* : family history of heart disease (Present, Absent)
- *obesity* : quantitative indicator,
- *alcohol* : current alcohol consumption
- *age* : age at onset

Response : CHD event (case) or not (control). 160 cases / 302 controls

1. https ://github.com/empathy87/The-Elements-of-Statistical-Learning-Python-Notebooks/blob/master/examples/South African Heart Disease.ipynb

Application : South African CHD (Cont'd)

Logistic regression fit of CHD events

	Coefficient	Std. Error	Z score
(Intercept)	−4.130	0.964	−4.285
sbp	0.006	0.006	1.023
tobacco	0.080	0.026	3.034
ldl	0.185	0.057	3.219
famhist	0.939	0.225	4.178
obesity	−0.035	0.029	-1.187
alcohol	0.001	0.004	0.136
age	0.043	0.010	4.184

- A Z score ($\equiv$ Coeff / Std. Error) > 2 in absolute value is significant at the 5% level (under Gaussian noise assumption)

Must be interpreted with caution !

- systolic blood pressure (sbp) is not significant !
- nor is obesity (conversely, < 0 coefficient) !

$\rightarrow$ result of the strong correlations between the predictors : **over-fitting** issue !

Application : South African CHD (Cont'd) with greedy selection procedure

Model selection : greedy backward procedure

To prevent from over-fitting, find the variables that are sufficient for explaining the CHD outputs (example of greedy algorithm)

- drop the least significant predictor, and refit the model
- repeat until no further terms can be dropped ← backward selection

	Coefficient	Std. Error	Z score
`(Intercept)`	−4.204	0.498	−8.45
`tobacco`	0.081	0.026	3.16
`ldl`	0.168	0.054	3.09
`famhist`	0.924	0.223	4.14
`age`	0.044	0.010	4.52

Interpretations

- Tobacco is measured in total lifetime usage in kilograms, with a median of 1kg for the controls and 4.1kg for the cases
- An increase of 1kg ⇒ increase of the CHD proba of $\exp(0.081) = 1.084$ or 8.4% (confidence interval at 95% $[1.03, 1.14]$)

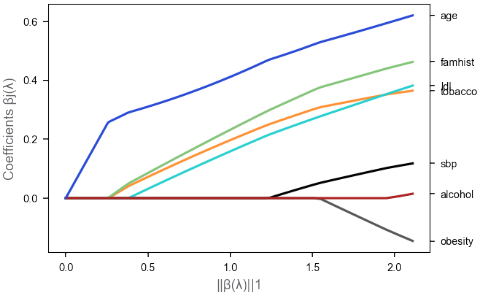
Application : South African CHD with lasso selection procedure

Model selection : ℓ_1 penalization (Lasso type method)

$$\widetilde{\beta}(\lambda) = \arg\min_{\beta} -\ell(\beta) + \lambda||\beta||_1,$$

$\rightarrow$ function of λ where less significant variables are explicitly discarded

Path of the des coefficients ℓ_1-penalized coefficients as a function of $||\hat{\beta}(\lambda)||_1$



Choosing λ

- large $||\widetilde{\beta}(\lambda)||_1$ (small λ) $\rightarrow$ over-fitting
- small $||\widetilde{\beta}(\lambda)||_1$ (large λ) $\rightarrow$ under-fitting
- $0.43 \leq ||\widetilde{\beta}(\lambda)||_1 \leq 1.3$ $\rightarrow$ 4 same predictors than backward selection procedure

Outline

Conclusions

Generalized Linear Models

Learning of the prediction rule based on a model of Y given X

☞ Linear regression, Logistic regression

Properties

- Simplicity : useful to capture the main effects
- Interpretability
- Efficient numerical procedures for large or high-dimensional data

Conclusions on Regularization for linear models

Regularization procedures are essential tools for data analysis, especially for big datasets involving many predictors, to

- prevent for over-fitting,
- better interpret the relations between the variables,
- improve the prediction performance

Shrinkage procedures

- ℓ_2 (ridge) regularization promotes the simplicity : shrink all the coefficients toward 0
- ℓ_1 (lasso) regularization promotes the simplicity+sparsity : shrink all the coefficients toward 0 + coefficients of non-signicant enough variables exactly equal to 0
  - But more difficult to compute !
- useful to capture the main effects and to interpret the relations between the variables

☞ concepts that extend to non-linear methods, e.g. neural nets