Annotated version

Machine Learning Course - CS-433

Least Squares

Oct 4, 2022

Martin Jaggi
Last updated on: October 3, 2022

credits to Mohammad Emtiyaz Khan & Rüdiger Urbanke

EPFL

Motivation

In rare cases, one can compute the optimum of the cost function analytically. Linear regression using a mean-squared error cost function is one such case. Here the solution can be obtained explicitly, by solving a linear system of equations. These equations are sometimes called the normal equations. This method is one of the most popular methods for data fitting. It is called least squares.

To derive the normal equations, we first show that the problem is convex. We then use the optimality conditions for convex functions (see the previous lecture notes on optimization). I.e., at the optimum parameter, call it $\mathbf{w}^\star$, it must be true that the gradient of the cost function is $\mathbf{0}$. I.e.,

$$\nabla \mathcal{L}(\mathbf{w}^\star) = \mathbf{0}.$$

This is a system of D equations.

Find w

$\min_w \mathcal{L}(w)$

$= \frac{1}{N}\sum_{n=1}^{N} \frac{1}{2}(y_n - w_0)^2$

$= f_w(x)$

1-param model

① $\mathcal{L}$ is convex in w? yes

② $\frac{\partial \mathcal{L}}{\partial w} = \frac{1}{N}\sum_{n=1}^{N}(y_n - w_0)(-1)$

$= w_0 - \frac{1}{N}\sum_{n=1}^{N} y_n$

$\overset{!}{=} 0$

$\Rightarrow w_0 = \bar{y}$

global solution

1st order

① ② ↳ find solution w^*

Normal Equations

Recall that the cost function for linear regression with mean-squared error is given by

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}\left(y_n - \mathbf{x}_n^\top \mathbf{w}\right)^2 = \frac{1}{2N}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}),$$

where

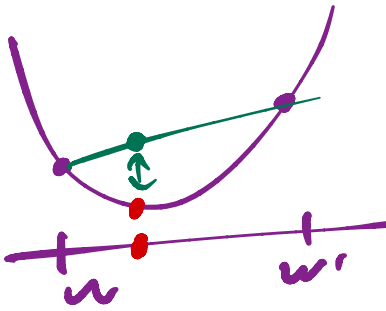
$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix}.$$

We claim that this cost function is *convex* in the $\mathbf{w}$. There are several ways of proving this:

1. Simplest way: observe that $\mathcal{L}$ is naturally represented as the sum (with positive coefficients) of the simple terms $(y_n - \mathbf{x}_n^\top \mathbf{w})^2$. Further, each of these simple terms is the composition of a linear function with a convex function (the square function). Therefore, each of these simple terms is convex and hence the sum is convex.

Convex ?

line-segment



2. Directly verify the definition, that for any $\lambda \in [0,1]$ and $\mathbf{w}, \mathbf{w}'$,

$$\mathcal{L}(\lambda\mathbf{w} + (1-\lambda)\mathbf{w}') - (\lambda\mathcal{L}(\mathbf{w}) + (1-\lambda)\mathcal{L}(\mathbf{w}')) \leq 0.$$

Computation: LHS =

$$-\frac{1}{2N}\lambda(1-\lambda)\|\mathbf{X}(\mathbf{w}-\mathbf{w}')\|_2^2, \qquad \leq 0 \quad \checkmark$$

which indeed is non-positive.

3. We can compute the second derivative (the Hessian) and show that it is positive semidefinite (all its eigenvalues are non-negative). For the present case a computation shows that the Hessian has the form

$$\frac{1}{N}\mathbf{X}^\top\mathbf{X}.$$

$\nabla^2 \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial w_i \partial w_j}\right)_{ij}$

MSE

$\nabla \mathcal{L} = -\frac{1}{N} X^T (y - Xw)$

$\nabla^2 \mathcal{L} = \frac{1}{N} X^T X$

$D \times D$ matrix

This matrix is indeed positive semidefinite since its non-zero eigenvalues are the squares of the non-zero singular values of the matrix $\mathbf{X}$.

$\nabla^2 \mathcal{L}$ p.s.d. $\forall w$

$\Rightarrow \mathcal{L}$ convex $\checkmark$

P.S.D.: $v^T X^T X v \geq 0 \quad \forall v \in \mathbb{R}^D$

$= \|Xv\|^2$

(2)

Now where we know that the function is convex, let us find its minimum. If we take the gradient of this expression with respect to the weight vector $\mathbf{w}$ we get

$$\nabla \mathcal{L}(\mathbf{w}) = -\frac{1}{N}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}).$$

e

$\stackrel{!}{=} 0$

solve for w

If we set this expression to $\mathbf{0}$ we get the normal equations for linear regression,

$$\mathbf{X}^\top \underbrace{(\mathbf{y} - \mathbf{X}\mathbf{w})}_{\text{error}} = \mathbf{0}.$$

e

row of X^T

= feature vector of dataset

(1) + (2) $\Rightarrow$ global optimum

1st order opt. condition

$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$

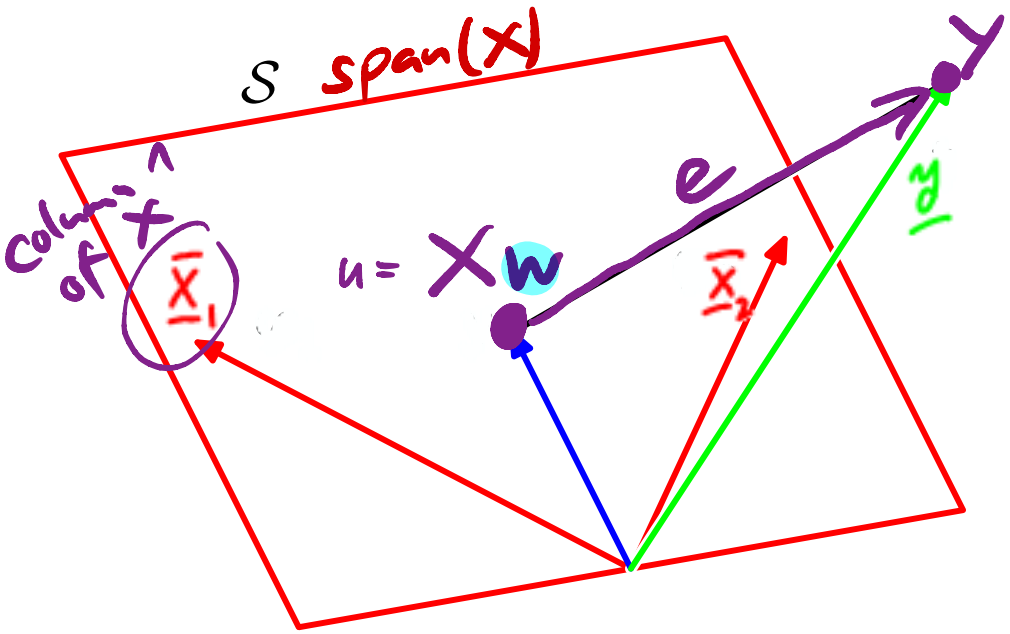
Geometric Interpretation

The error is orthogonal to all columns of $\mathbf{X}$.

The span of $\mathbf{X}$ is the space spanned by the columns of $\mathbf{X}$. Every element of the span can be written as $\mathbf{u} = \mathbf{X}\mathbf{w}$ for some choice of $\mathbf{w}$. Which element of $span(\mathbf{X})$ shall we take? The normal equations tell us that the optimum choice for $\mathbf{u}$, call it $\mathbf{u}^\star$, is that element so that $\mathbf{y} - \mathbf{u}^\star$ is orthogonal to $span(\mathbf{X})$. In other words, we should pick $\mathbf{u}^\star$ to be equal to the projection of $\mathbf{y}$ onto $span(\mathbf{X})$.

The following figure illustrates this:

(taken from Bishop's book)



Least Squares

The matrix $\mathbf{X}^\top\mathbf{X} \in \mathbb{R}^{D\times D}$ is called the Gram matrix. If it is invertible, we can multiply the normal equation by the inverse of the Gram matrix from the left to get a closed-form expression for the minimum:

$$\mathbf{w}^\star = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}.$$

We can use this model to predict a new value for an unseen datapoint (test point) $\mathbf{x}_m$:

$$\hat{y}_m := \mathbf{x}_m^\top\mathbf{w}^\star = \mathbf{x}_m^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}.$$

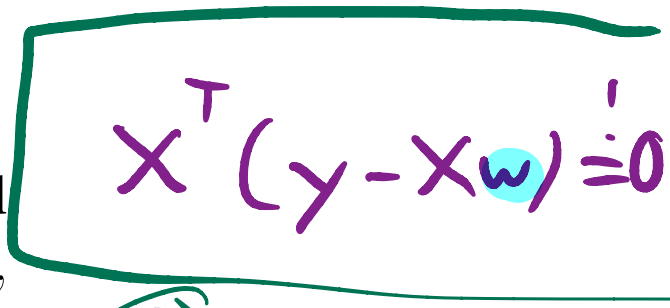
Invertibility and Uniqueness

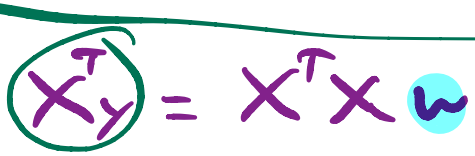
Note that the Gram matrix $\mathbf{X}^\top\mathbf{X} \in \mathbb{R}^{D\times D}$ is invertible if and only if $\mathbf{X}$ has full column rank, or in other words $rank(\mathbf{X}) = D$.

Proof: To see this assume first that $rank(\mathbf{X}) < D$. Then there exists a non-zero vector $\mathbf{u}$ so that $\mathbf{X}\mathbf{u} = \mathbf{0}$. It follows that $\mathbf{X}^\top\mathbf{X}\mathbf{u} = \mathbf{0}$, and so $rank(\mathbf{X}^\top\mathbf{X}) < D$. Therefore, $\mathbf{X}^\top\mathbf{X}$ is not invertible.
Conversely, assume that $\mathbf{X}^\top\mathbf{X}$ is not invertible. Hence, there exists a non-zero vector $\mathbf{v}$ so that $\mathbf{X}^\top\mathbf{X}\mathbf{v} = \mathbf{0}$. It follows that

$$\mathbf{0} = \mathbf{v}^\top\mathbf{X}^\top\mathbf{X}\mathbf{v} = (\mathbf{X}\mathbf{v})^\top(\mathbf{X}\mathbf{v}) = \|\mathbf{X}\mathbf{v}\|^2.$$

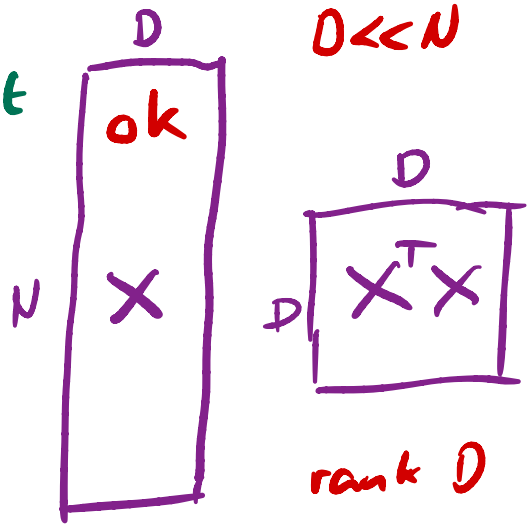
This implies that $\mathbf{X}\mathbf{v} = \mathbf{0}$, i.e., $rank(\mathbf{X}) < D$.

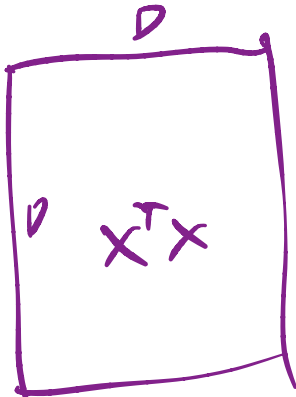


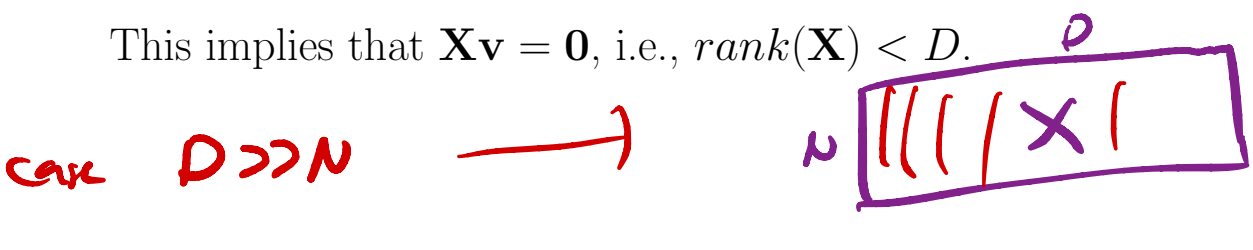


D×D

solve linear system

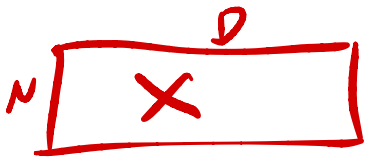






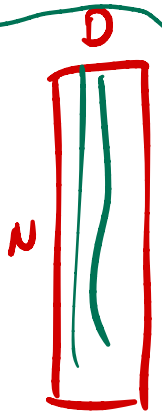
Rank Deficiency and Ill-Conditioning

Unfortunately, in practice, $\mathbf{X}$ is often rank deficient.



- If $D > N$, we always have $rank(\mathbf{X}) < D$ (since row rank = col. rank)
- If $D \leq N$, but some of the columns $\mathbf{x}_{:d}$ are (nearly) collinear, then the matrix is ill-conditioned, leading to numerical issues when solving the linear system.

over-parameterized
$\Rightarrow$ X^TX not invertible
$\Rightarrow w^*$ not unique



$\Rightarrow$ rank < D

Can we solve least squares if $\mathbf{X}$ is rank deficient? Yes, using a linear system solver.

$$\underset{D\times D}{X^TX}\ \underset{D}{w} = \underset{D}{X^Ty}$$

condition number $= \frac{\lambda_{max}(X^TX)}{\lambda_{min}(X^TX)}$

zero if rank-deficient

Summary of Linear Regression

We have studied three types of methods:

1. Grid Search
2. Iterative Optimization Algorithms
   (Stochastic) Gradient Descent
3. Least squares
   closed-form solution, for linear MSE

compute X^TX — inverse

computational cost = $\mathcal{O}(N\cdot D^2 + D^3)$

Additional Notes

Closed-form solution for MAE

Can you derive closed-form solution for 1-parameter model when using MAE cost function?
See this short article: http://www.johnmyleswhite.com/notebook/2013/03/22/modes-medians-and-means-an-unifying-perspective/.

Implementation

There are many ways to solve a linear system, but using the QR decomposition is one of the most robust ways. Matlab's backslash operator and also NumPy's linalg package implement this in just one line:

```
w = np.linalg.solve(X, y)
```

For a robust implementation, see Sec. 7.5.2 of Kevin Murphy's book.