Machine Learning Course - CS-433

Least Squares

Sept 24, 2024

Martin Jaggi
Last updated on: September 24, 2024

credits to Mohammad Emtiyaz Khan & Rüdiger Urbanke

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.

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.

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 =

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

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

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

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

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

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

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

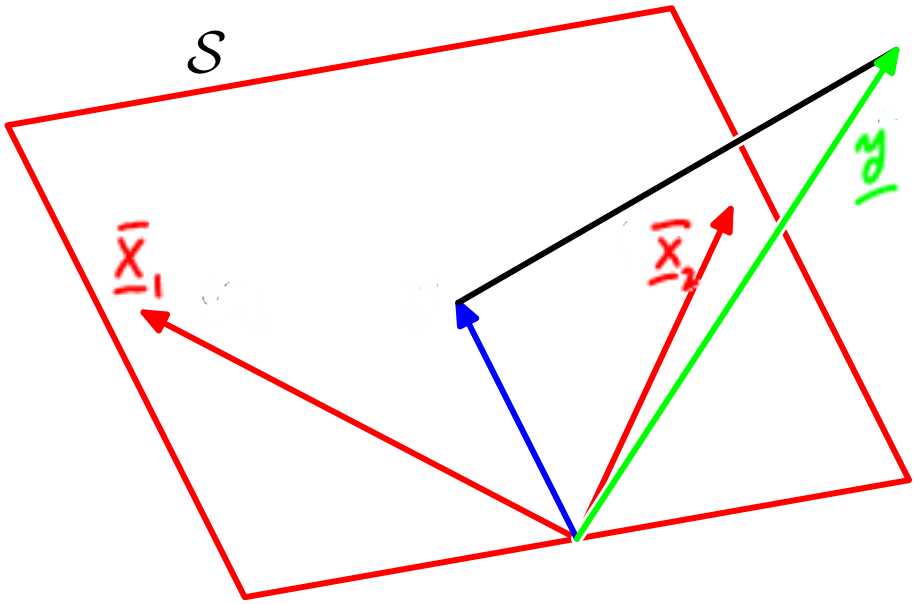
$$\mathbf{X}^\top \underbrace{(\mathbf{y} - \mathbf{X}\mathbf{w})}_{\text{error}} = \mathbf{0}.$$

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., $rank(\mathbf{X}) < D$.

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.

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

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

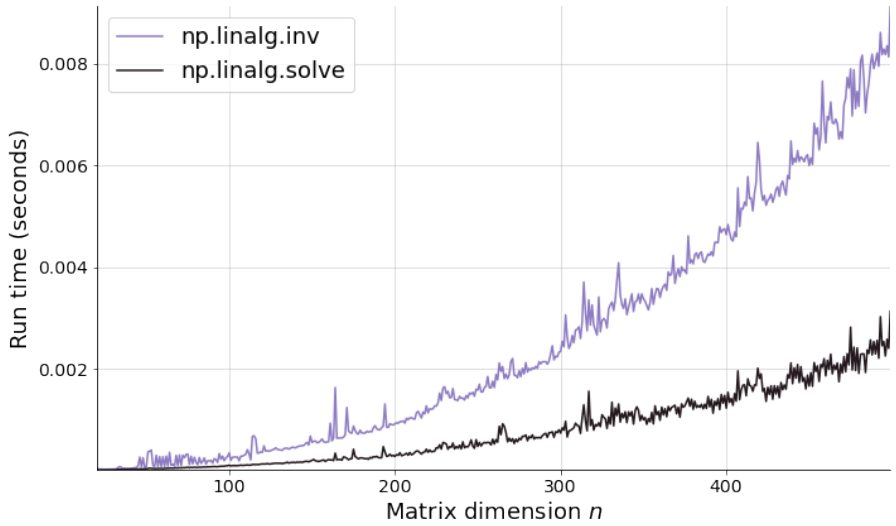
Additional Notes

Solving linear systems

There are many ways to solve a linear system $Xw = y$, but it usually involves a decomposition of the matrix X such as the QR or LU decomposition which are very robust. Matlab's backslash operator and also NumPy's linalg package implement this in just one line:

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

It is important to never invert a matrix to solve a linear system - as this would incur a cost at least three times the cost of using a linear solver. For more, see this blog post https://gregorygundersen.com/blog/2020/12/09/matrix-inversion/.



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

Closed-form solution for MAE

Can you derive closed-form solution for 1-parameter model when using the MAE cost function?

See this short article: http://www.johnmyleswhite.com/notebook/2013/03/22/modes-medians-and-means-an-unifying-perspective/.

Machine Learning Course - CS-433

Underfitting and Overfitting

Sept 24, 2024

Martin Jaggi
Last updated on: September 24, 2024

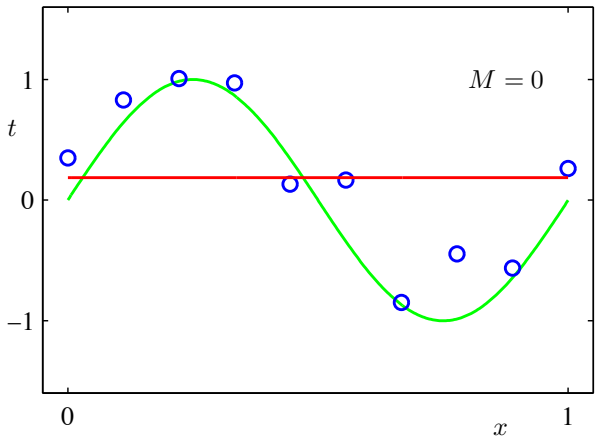
credits to Mohammad Emtiyaz Khan & Rüdiger Urbanke

Motivation

Models can be *too limited* or they can be *too rich*. In the first case we cannot find a function that is a good fit for the data in our model. We then say that we underfit. In the second case we have such a rich model family that we do not just fit the underlying function but we in fact fit the noise in the data as well. We then talk about an overfit. Both of these phenomena are undesirable. This discussion is made more difficult since all we have is data and so we do not know a priori what part is the underlying signal and what part is noise.

Underfitting with Linear Models

It is easy to see that linear models might underfit. Consider a scalar case as shown in the figure below.



The solid curve is the underlying function and the circles are the actual data. E.g., we assume that there is a scalar function $g(x)$ but that we do not observe $g(x_n)$ directly but only a noisy version of it, $y_n = g(x_n) + Z_n$, where Z_n is

the noise. The noise might be due for example to some measurement inaccuracies. The y_n are shown as blue circles. If our model family consists of only linear functions of the scalar input x, i.e., $\mathcal{H} = \{f_w(x) = wx\}$, where w is a scalar constant (the slope of the function), then it is clear that we cannot match the given function accurately, regardless of how many samples we get and how small the noise is. We therefore will underfit.

Extended/Augmented Feature Vectors From the above example it might seem that linear models are too simple to ever overfit. But in fact, linear models are highly prone to overfitting, much more so than complicated models like neural nets.
Since linear models are inherently not very rich the following is a standard "trick" to make them more powerful.
In order to increase the representational power of linear models we typically "augment" the input. E.g., if the input (feature) is one-dimensional we might add a polynomial basis (of arbitrary degree M),

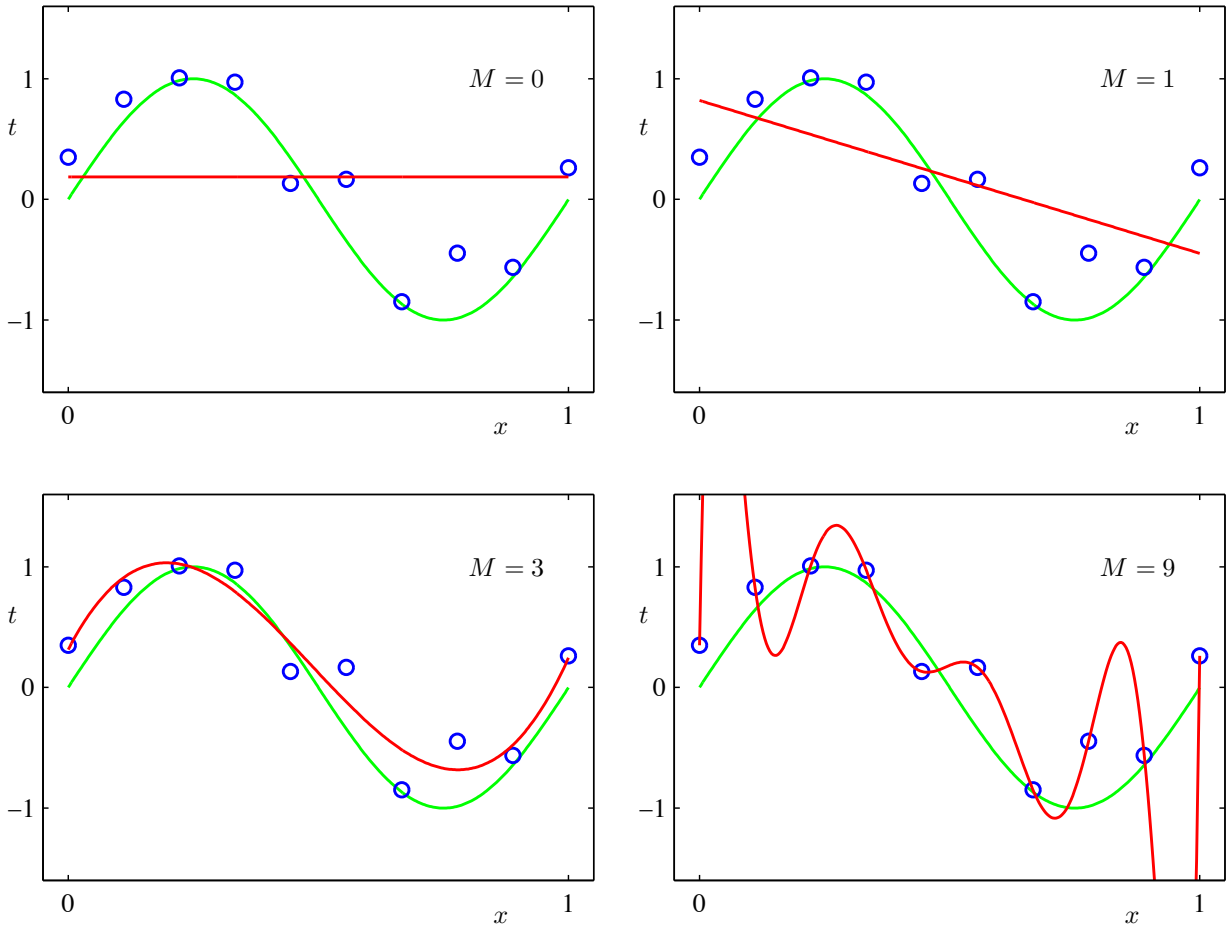
$$\boldsymbol{\phi}(x_n) := [1,\, x_n,\, x_n^2,\, x_n^3,\, \ldots,\, x_n^M]$$

so that we end up with an extended feature vector.
We then fit a linear model to this extended feature vector $\boldsymbol{\phi}(x_n)$:

$$y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \ldots + w_M x_n^M =: \boldsymbol{\phi}(x_n)^\top \mathbf{w}.$$

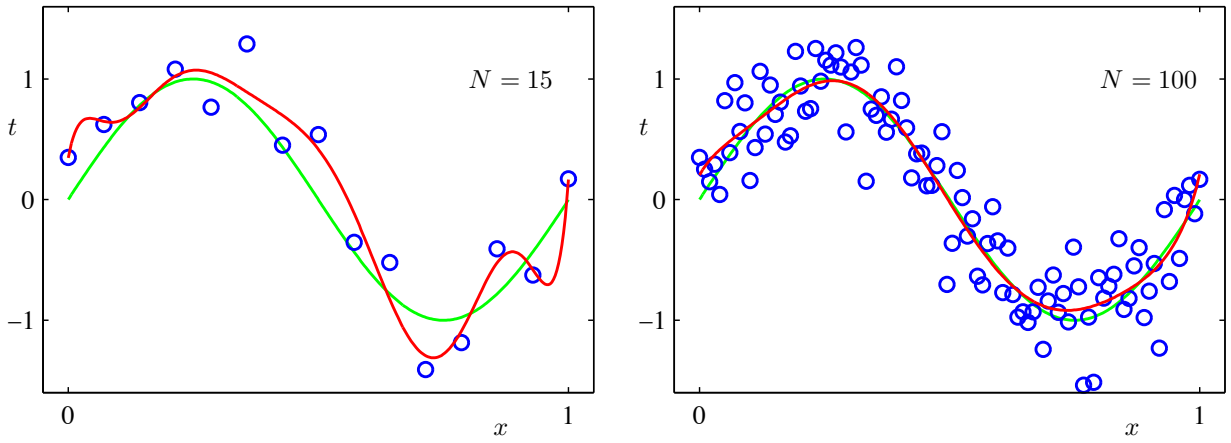
Overfitting with Linear Models

In the following four figures, circles are data points, the green line represents the "true function", and the red line is the model. The parameter M is the maximum degree in the polynomial basis.



For $M = 0$ (the model is a constant) the model is underfitting and the same is true for $M = 1$. For $M = 3$ the model fits the data fairly well and is not yet so rich as to fit in addition the small "wiggles" caused by the noise. But for $M = 9$ we now have such a rich model that it can fit every single data point and we see severe overfitting taking place. What can we do to avoid overfitting? If you increase the amount of data (increase N, but keep M fixed), overfitting

might reduce. This is shown in the following two figures where we again consider the same model complexity $M = 9$ but we have extra data ($N = 15$ or even $N = 100$).



A Word About Notation

If it is important to distinguish the original input $\mathbf{x}$ from the augmented input then we will use $\phi(\mathbf{x})$ to denote this augmented input vector. But we can consider this augmentation as part of the pre-processing, and then we might simply write $\mathbf{x}$ to denote the input. This will save us a lot of notation.

Additional Materials

Read about overfitting in the paper by Pedro Domingos (Sections 3 and 5 of "A few useful things to know about machine learning").

Machine Learning Course - CS-433

Maximum Likelihood

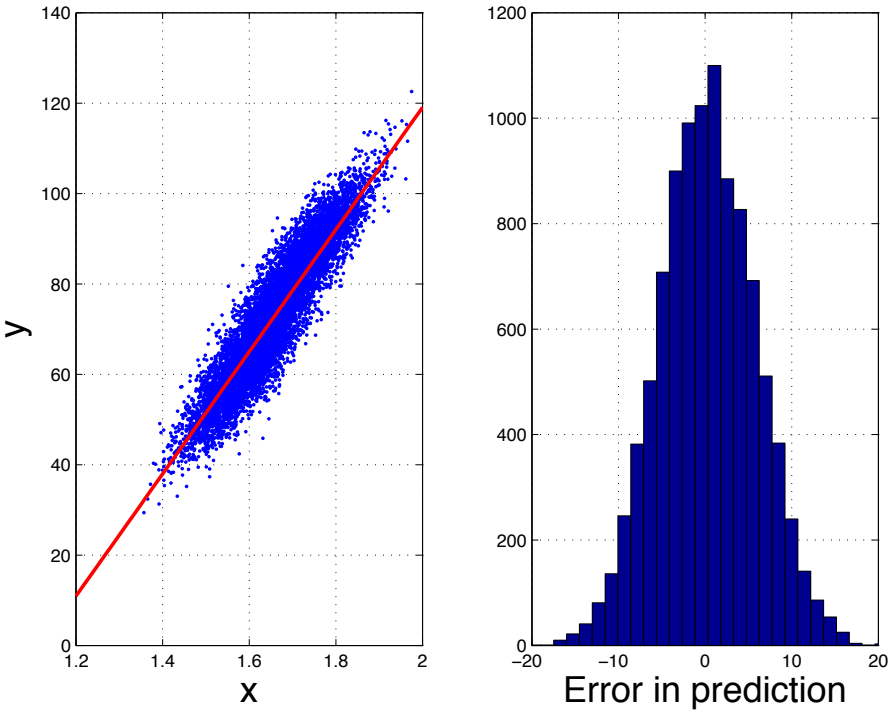
Sept 25, 2024

Martin Jaggi
Last updated on: September 24, 2024

credits to Mohammad Emtiyaz Khan & Rüdiger Urbanke

Motivation

In the previous lecture 3a we arrived at the least-squares problem in the following way: we postulated a particular cost function (square loss) and then, given data, found that model that minimizes this cost function. In the current lecture we will take an alternative route. The final answer will be the same, but our starting point will be probabilistic. In this way we find a second interpretation of the least-squares problem.



Gaussian distribution and independence

Recall the definition of a Gaussian random variable in $\mathbb{R}$ with mean μ and variance σ^2. It has a density of

$$p(y \mid \mu, \sigma^2) = \mathcal{N}(y \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y-\mu)^2}{2\sigma^2}\right].$$

In a similar manner, the density of a Gaussian random vector with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ (which must be a positive semi-definite matrix) is

$$\mathcal{N}(\mathbf{y} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D \, det(\boldsymbol{\Sigma})}} \exp\left[-\frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right].$$

Also recall that two random variables X and Y are called *independent* when $p(x, y) = p(x)p(y)$.

A probabilistic model for least-squares

We assume that our data is generated by the model,

$$y_n = \mathbf{x}_n^\top \mathbf{w} + \epsilon_n,$$

where the ϵ_n (the noise) is a zero-mean Gaussian random variable with variance σ^2 and the noise that is added to the various samples is independent of each other, and independent of the input. Note that the model $\mathbf{w}$ is unknown.

Therefore, given N samples, the likelihood of the data vector $\mathbf{y} = (y_1, \cdots, y_N)$ given the input $\mathbf{X}$ (each row is one input) and the model $\mathbf{w}$ is equal to

$$p(\mathbf{y} \,|\, \mathbf{X}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n \,|\, \mathbf{x}_n, \mathbf{w}) = \prod_{n=1}^{N} \mathcal{N}(y_n \,|\, \mathbf{x}_n^\top \mathbf{w}, \sigma^2).$$

The probabilistic view point is that we should maximize this likelihood over the choice of model $\mathbf{w}$. I.e., the "best" model is the one that maximizes this likelihood.

Defining cost with log-likelihood

Instead of maximizing the likelihood, we can take the logarithm of the likelihood and maximize it instead. Expression is called the log-likelihood (LL).

$$\mathcal{L}_{\text{LL}}(\mathbf{w}) := \log\, p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2 + \text{cnst}.$$

Compare the LL to the MSE (mean squared error)

$$\mathcal{L}_{\text{LL}}(\mathbf{w}) = -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2 + \text{cnst}$$

$$\mathcal{L}_{\text{MSE}}(\mathbf{w}) = \frac{1}{2N} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2$$

Maximum-likelihood estimator (MLE)

It is clear that maximizing the LL is equivalent to minimizing the MSE:

$$\arg\min_{\mathbf{w}} \mathcal{L}_{\text{MSE}}(\mathbf{w}) = \arg\max_{\mathbf{w}} \mathcal{L}_{\text{LL}}(\mathbf{w}).$$

This gives us another way to design cost functions.

MLE can also be interpreted as finding the model under which the observed data is most likely to have been generated from (probabilistically). This interpretation has some advantages that we discuss now.

Properties of MLE

MLE is a *sample* approximation to the *expected log-likelihood*:

$$\mathcal{L}_{\text{LL}}(\mathbf{w}) \approx \mathbb{E}_{p(y,\mathbf{x})}\big[\log p(y \mid \mathbf{x}, \mathbf{w})\big]$$

MLE is consistent, i.e., it will give us the correct model assuming that we have a sufficient amount of data. (can be proven under some weak conditions)

$$\mathbf{w}_{\text{MLE}} \longrightarrow^{p} \mathbf{w}_{\text{true}} \quad \text{in probability}$$

The MLE is asymptotically normal, i.e.,

$$(\mathbf{w}_{\text{MLE}} - \mathbf{w}_{\text{true}}) \longrightarrow^{d} \frac{1}{\sqrt{N}}\mathcal{N}(\mathbf{w}_{\text{MLE}} \mid \mathbf{0}, \mathbf{F}^{-1}(\mathbf{w}_{\text{true}}))$$

where $\mathbf{F}(\mathbf{w}) = -\mathbb{E}_{p(\mathbf{y})}\left[\frac{\partial^2 \mathcal{L}}{\partial \mathbf{w} \partial \mathbf{w}^\top}\right]$ is the Fisher information.

MLE is efficient, i.e. it achieves the Cramer-Rao lower bound.

$$\text{Covariance}(\mathbf{w}_{\text{MLE}}) = \mathbf{F}^{-1}(\mathbf{w}_{\text{true}})$$

Another example

We can replace Gaussian distribution by a Laplace distribution.

$$p(y_n \,|\, \mathbf{x}_n, \mathbf{w}) = \frac{1}{2b} e^{-\frac{1}{b}|y_n - \mathbf{x}_n^\top \mathbf{w}|}$$

Machine Learning Course - CS-433

Regularization: Ridge Regression and Lasso

Sept 25, 2024

Martin Jaggi
Last updated on: September 24, 2024

credits to Mohammad Emtiyaz Khan & Rüdiger Urbanke

Motivation

We have seen that by augmenting the feature vector we can make linear models as powerful as we want. Unfortunately this leads to the problem of overfitting. *Regularization* is a way to mitigate this undesirable behavior.

We will discuss regularization in the context of linear models, but the same principle applies also to more complex models such as neural nets.

Regularization

Through regularization, we can penalize complex models and favor simpler ones:

$$\min_{\mathbf{w}} \quad \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w})$$

The second term Ω is a regularizer, measuring the complexity of the model given by $\mathbf{w}$.

L_2-Regularization: Ridge Regression

The most frequently used regularizer is the standard Euclidean norm (L_2-norm), that is

$$\Omega(\mathbf{w}) = \lambda \|\mathbf{w}\|_2^2$$

where $\|\mathbf{w}\|_2^2 = \sum_i w_i^2$. Here the main effect is that large model weights w_i will be penalized (avoided), since we consider them "unlikely", while small ones are ok. When $\mathcal{L}$ is MSE, this is called ridge regression:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} \left[y_n - \mathbf{x}_n^\top \mathbf{w} \right]^2 \; + \; \lambda \|\mathbf{w}\|_2^2$$

Least squares is a special case of this: set $\lambda := 0$.

Explicit solution for $\mathbf{w}$: Differentiating and setting to zero:

$$\mathbf{w}^\star_{\text{ridge}} = (\mathbf{X}^\top \mathbf{X} + \lambda' \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$$

(here for simpler notation $\frac{\lambda'}{2N} = \lambda$)

Ridge Regression to Fight Ill-Conditioning

The eigenvalues of $(\mathbf{X}^\top\mathbf{X} + \lambda'\mathbf{I})$ are all at least λ' and so the inverse always exists. This is also referred to as *lifting the eigenvalues*.

Proof: Write the Eigenvalue decomposition of $\mathbf{X}^\top\mathbf{X}$ as $\mathbf{USU}^\top$. We then have

$$\begin{aligned}\mathbf{X}^\top\mathbf{X} + \lambda'\mathbf{I} &= \mathbf{USU}^\top + \lambda'\mathbf{UIU}^\top \\ &= \mathbf{U}[\mathbf{S} + \lambda'\mathbf{I}]\mathbf{U}^\top.\end{aligned}$$

We see now that every Eigenvalue is "lifted" by an amount λ'.

Here is an alternative proof. Recall that for a symmetric matrix $\mathbf{A}$ we can also compute eigenvalues by looking at the so-called Rayleigh ratio,

$$R(\mathbf{A}, \mathbf{v}) = \frac{\mathbf{v}^\top\mathbf{A}\mathbf{v}}{\mathbf{v}^\top\mathbf{v}}.$$

Note that if $\mathbf{v}$ is an eigenvector with eigenvalue λ then the Rayleigh coefficient indeed gives us λ. We can find the smallest and largest eigenvalue by minimizing and maximizing this coefficient. But note that if we apply this to the symmetric matrix $\mathbf{X}^\top\mathbf{X} + \lambda'\mathbf{I}$ then for any vector $\mathbf{v}$ we have

$$\frac{\mathbf{v}^\top(\mathbf{X}^\top\mathbf{X} + \lambda'\mathbf{I})\mathbf{v}}{\mathbf{v}^\top\mathbf{v}} \geq \frac{\lambda'\mathbf{v}^\top\mathbf{v}}{\mathbf{v}^\top\mathbf{v}} = \lambda'.$$

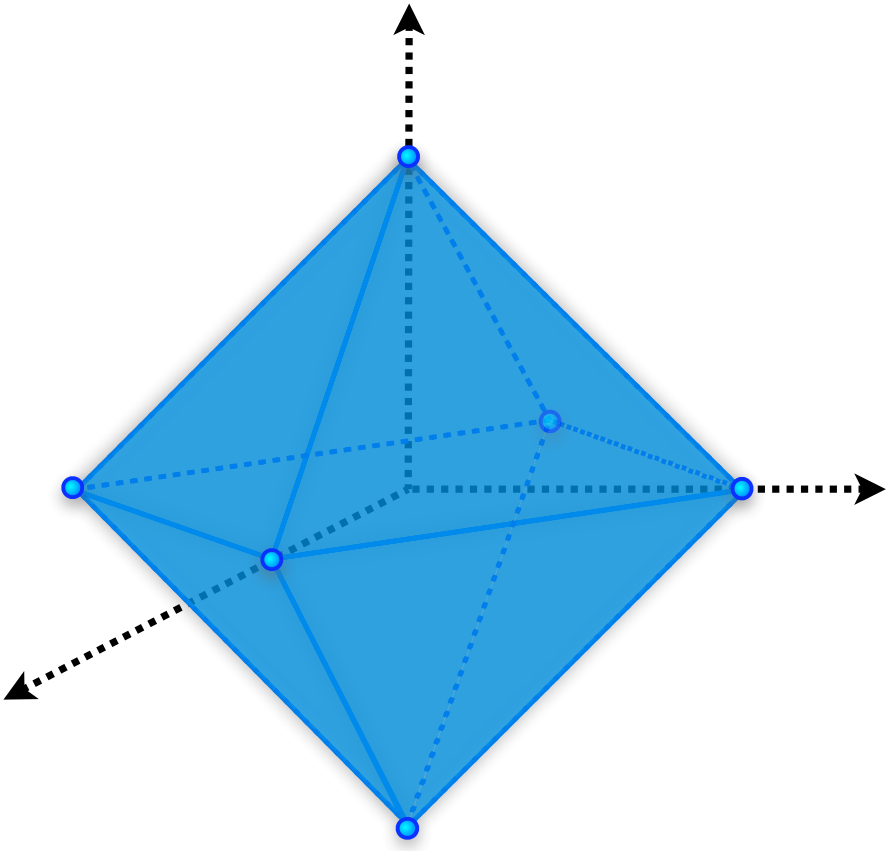
L_1-Regularization: The Lasso

As an alternative measure of the complexity of the model, we can use a different norm. A very important case is the L_1-norm, leading to L_1-regularization. In combination with the MSE cost function, this is known as the Lasso:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} [y_n - \mathbf{x}_n^\top \mathbf{w}]^2 \; + \; \lambda \, \|\mathbf{w}\|_1$$

where

$$\|\mathbf{w}\|_1 := \sum_i |w_i|.$$



The figure above shows a "ball" of constant L_1 norm. To keep things simple assume that $\mathbf{X}^\top \mathbf{X}$ is invertible. We claim that in this case the set

$$\{\mathbf{w} : \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 = \alpha\} \tag{1}$$

is an ellipsoid and this ellipsoid simply scales around its origin as we change α. We claim that for the L_1-regularization the optimum solution is likely going to be sparse (only has few non-zero components) compared to the case where we use L_2-regularization.

Why is this the case? Assume that a genie tells you the L_1-norm of the optimum solution. Draw the L_1-ball with that norm value (think of 2D to visualize it). So now you know that the optimal point is somewhere on the surface of this "ball". Further you know that there are ellipsoids, all with the same mean and rotation that describes the equal error surfaces incurred by the first term. The optimum solution is where the "smallest" of these ellipsoids just touches the L_1-ball. Due to the geometry of this ball this point is more likely to be on one of the "corner" points. In turn, sparsity is desirable, since it leads to a "simple" model.

How do we see the claim that (1) describes and ellipsoid? First look at $\alpha = \|\mathbf{X}\mathbf{w}\|^2 = \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w}$. This is a quadratic form. Let $\mathbf{A} = \mathbf{X}^\top \mathbf{X}$. Note that $\mathbf{A}$ is a symmetric matrix and by assumption it has full rank. If $\mathbf{A}$ is a diagonal matrix with strictly positive elements a_i along the diagonal then this describes the equation

$$\sum_i a_i \mathbf{w}_i^2 = \alpha,$$

which is indeed the equation for an ellipsoid. In the general case, $\mathbf{A}$ can be written as (using the SVD) $\mathbf{A} = \mathbf{U}\mathbf{B}\mathbf{U}^T$, where $\mathbf{B}$ is a diagonal matrix with strictly positive entries. This then corresponds to an ellipsoid with rotated axes. If we now look at $\alpha = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$, where $\mathbf{y}$ is in the column space of $\mathbf{X}$ then we can write it as $\alpha = \|\mathbf{X}(\mathbf{w}_0 - \mathbf{w})\|^2$ for a suitable chosen $\mathbf{w}_0$ and so this corresponds to a shifted ellipsoid. Finally, for the general case, write $\mathbf{y}$ as $\mathbf{y} = \mathbf{y}_\parallel + \mathbf{y}_\perp$, where $\mathbf{y}_\parallel$ is the component of $\mathbf{y}$ that lies in the subspace spanned by the columns of $\mathbf{X}$ and $\mathbf{y}_\perp$ is the component that

is orthogonal. In this case

$$\begin{aligned}\alpha &= \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 \\ &= \|\mathbf{y}_{\|} + \mathbf{y}_{\perp} - \mathbf{X}\mathbf{w}\|^2 \\ &= \|\mathbf{y}_{\perp}\|^2 + \|\mathbf{y}_{\|} - \mathbf{X}\mathbf{w}\|^2 \\ &= \|\mathbf{y}_{\perp}\|^2 + \|\mathbf{X}(\mathbf{w}_0 - \mathbf{w})\|^2.\end{aligned}$$

Hence this is then equivalent to the equation $\|\mathbf{X}(\mathbf{w}_0 - \mathbf{w})\|^2 = \alpha - \|\mathbf{y}_{\perp}\|^2$, proving the claim. From this we also see that if $\mathbf{X}^{\top}\mathbf{X}$ is not full rank then what we get is not an ellipsoid but a cylinder with an ellipsoidal cross-section.

Additional Notes

Other Types of Regularization

Popular methods such as shrinkage, dropout and weight decay (in the context of neural networks), early stopping of the optimization are all different forms of regularization.

Another view of regularization: The ridge regression formulation we have seen above is similar to the following constrained problem (for some $\tau > 0$).

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2, \qquad \text{such that } \|\mathbf{w}\|_2^2 \leq \tau$$

The following picture illustrates this.

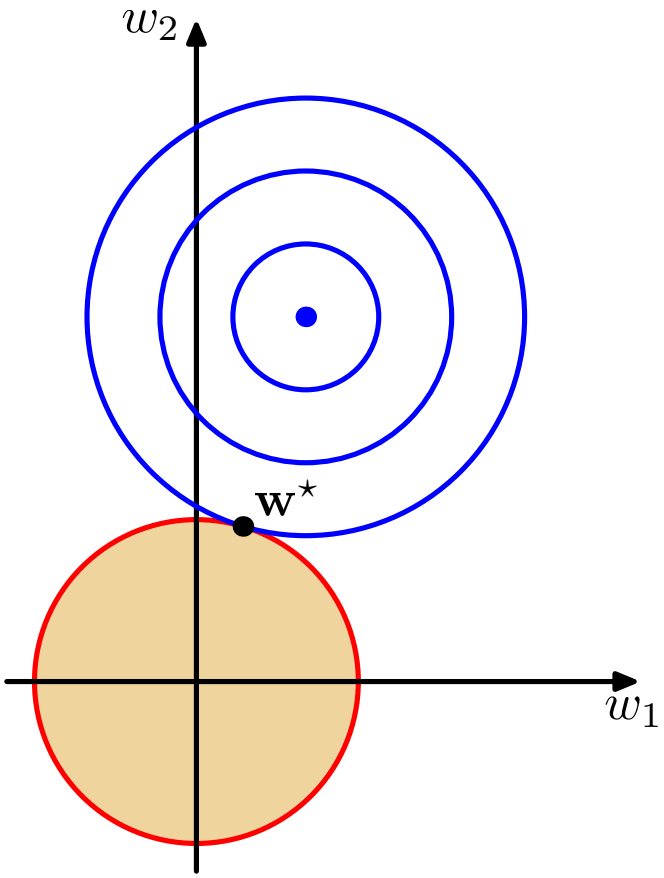


Figure 1: Geometric interpretation of Ridge Regression. Blue lines indicate the level sets of the MSE cost function.

For the case of using L_1 regularization (known as the Lasso, when used with MSE) we analogously consider

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n^\top \mathbf{w})^2, \qquad \text{such that } \|\mathbf{w}\|_1 \leq \tau$$

This forces some of the elements of $\mathbf{w}$ to be strictly 0 and therefore enforces sparsity in the model (some features will not be used since their coefficients are zero).

- Why does L_1 regularizer enforce sparsity? *Hint:* Draw a picture similar to the above, and locate the optimal solution.
- Why is it good to have sparsity in the model? Is it going to be better than least-squares? When and why?

Ridge Regression as MAP estimator

Recall that classic *least-squares linear regression* can be interpreted as the maximum likelihood estimator:

$$
\begin{aligned}
\mathbf{w}_{\text{lse}} &\overset{(a)}{=} \arg\min_{\mathbf{w}} && -\log p(\mathbf{y}, \mathbf{X}|\mathbf{w}) \\
&\overset{(b)}{=} \arg\min_{\mathbf{w}} && -\log p(\mathbf{X}|\mathbf{w}) p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \\
&\overset{(c)}{=} \arg\min_{\mathbf{w}} && -\log p(\mathbf{X}) p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \\
&\overset{(d)}{=} \arg\min_{\mathbf{w}} && -\log p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \\
&\overset{(e)}{=} \arg\min_{\mathbf{w}} && -\log \left[\prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \mathbf{w}) \right] \\
&\overset{(f)}{=} \arg\min_{\mathbf{w}} && -\log \left[\prod_{n=1}^{N} \mathcal{N}(y_n \,|\, \mathbf{x}_n^\top \mathbf{w}, \sigma^2) \right] \\
&= \arg\min_{\mathbf{w}} && -\log \left[\prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y_n - \mathbf{x}_n^\top \mathbf{w})^2} \right] \\
&= \arg\min_{\mathbf{w}} && -N \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \sum_{n=1}^{N} \frac{1}{2\sigma^2}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 \\
&= \arg\min_{\mathbf{w}} && \frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathbf{x}_n^\top \mathbf{w})^2
\end{aligned}
$$

In step (a) on the right we wrote down the negative of the log of the likelihood. The maximum likelihood criterion choses that parameter $\mathbf{w}$ that minimizes this quantity (i.e., maximizes the likelihood). In step (b) we factored the likelihood. The usual assumption is that the choice of the input samples $\mathbf{x}_n$ does not depend on the model parameter (which only influces the output given the input. Hence, in step (c) we removed the conditioning. Since the factor $p(\mathbf{X})$ does not depend on $\mathbf{w}$, i.e., is a constant wrt to $\mathbf{w}$) we can remove it. This is done in step (d). In step (e) we used the assumption that the samples are iid. In step (f) we then used our assumption that the samples have the form $y_n = \mathbf{w}_n^\top \mathbf{w} + Z_n$,

where Z_n is a Gaussian noise with mean zero and variance σ_2. The rest is calculus.

Ridge regression has a very similar interpretation. Now we start with the posterior $p(\mathbf{w}|\mathbf{X}, \mathbf{y})$ and chose that parameter $\mathbf{w}$ that maximizes this posterior. Hence this is called the maximum-a-posteriori (MAP) estimate. As before, we take the log and add a minus sign and minimize instead. In order to compute the posterior we use Bayes law and we assume that the components of the weight vector are iid Gaussians with mean zero and variance $\frac{1}{\lambda}$.

$$
\begin{aligned}
\mathbf{w}_{\text{ridge}} &= \arg\min_{\mathbf{w}} && -\log p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \\
&\overset{(a)}{=} \arg\min_{\mathbf{w}} && -\log \frac{p(\mathbf{y}, \mathbf{X}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}, \mathbf{X})} \\
&\overset{(b)}{=} \arg\min_{\mathbf{w}} && -\log p(\mathbf{y}, \mathbf{X}|\mathbf{w})p(\mathbf{w}) \\
&\overset{(c)}{=} \arg\min_{\mathbf{w}} && -\log p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w}) \\
&= \arg\min_{\mathbf{w}} && -\log \left[p(\mathbf{w}) \prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \mathbf{w}) \right] \\
&= \arg\min_{\mathbf{w}} && -\log \left[\mathcal{N}(\mathbf{w} \mid 0, \tfrac{1}{\lambda}\mathbf{I}) \prod_{n=1}^{N} \mathcal{N}(y_n \mid \mathbf{x}_n^\top \mathbf{w}, \sigma^2) \right] \\
&= \arg\min_{\mathbf{w}} && -\log \left[\frac{1}{(2\pi\frac{1}{\lambda})^{D/2}} e^{-\frac{\lambda}{2}\|\mathbf{w}\|^2} \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y_n - \mathbf{x}_n^\top \mathbf{w})^2} \right] \\
&= \arg\min_{\mathbf{w}} && \sum_{n=1}^{N} \frac{1}{2\sigma^2}(y_n - \mathbf{x}_n^\top \mathbf{w})^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2.
\end{aligned}
$$

In step (a) we used Bayes' law. In step (b) and (c) we eliminated quantities that do not depend on $\mathbf{w}$.

Regularization as prior information

Based on the previous derivation of ridge regression, we can more generally see regularization as encoding any kind of prior information we have and thus it can be understood as a compressed form of data, in this sense, using regularization is equivalent to adding data which helps reduce overfitting.