Machine Learning Course - CS-433

Ridge Regression and Lasso

Oct 5, 2017

Last updated: September 16, 2017

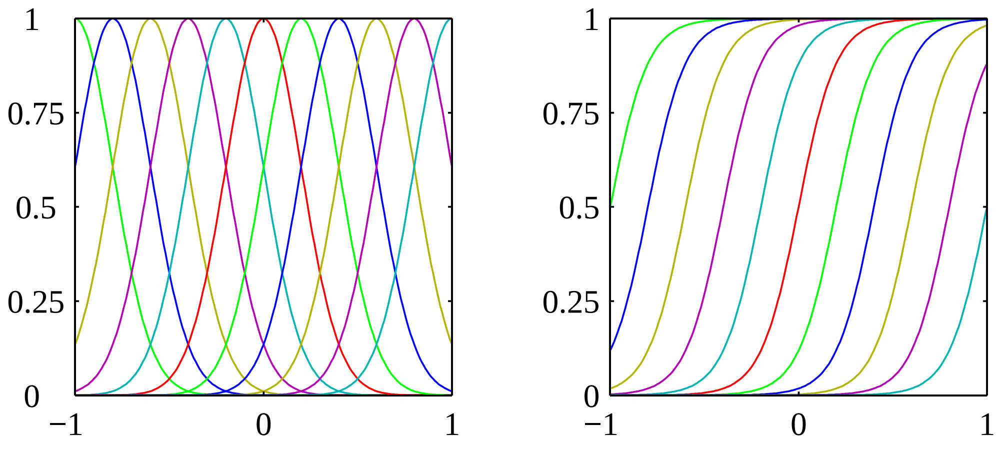
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Motivation

Linear models can be made more powerful, by constructing better features for your input data. One way is to use nonlinear basis functions.

Basis Functions

In general, we can use *basis functions* that are (known and fixed) nonlinear transformations applied to input variables. Given a D dimensional input vector $\mathbf{x}$, let us say that we have M basis functions $\phi_j(\mathbf{x})$, indexed by j.
Example of basis functions: Polynomials, Splines, Gaussians, Sigmoids, Fourier-Basis, Wavelets.



Taken from Bishop, Chapter 3

Linear Basis Function Models

The model is given as follows:

$$y_n \approx w_0 + \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}_n) = \widetilde{\boldsymbol{\phi}}(\mathbf{x}_n)^\top \mathbf{w}$$

where $\widetilde{\boldsymbol{\phi}}(\mathbf{x}_n)^\top = [1,\ \phi_1(\mathbf{x}_n),\ \phi_2(\mathbf{x}_n), \ldots, \phi_M(\mathbf{x}_n)]$

This model *is* linear in $\mathbf{w}$ but nonlinear in $\mathbf{x}$. Note that the dimensionality of $\mathbf{w}$ is now $M+1$, not D.
Defining the matrix $\widetilde{\mathbf{\Phi}}$ with $\widetilde{\boldsymbol{\phi}}(\mathbf{x}_n)^\top$ as rows,

$$\widetilde{\mathbf{\Phi}} := \begin{bmatrix} 1 & \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \dots & \phi_M(\mathbf{x}_1) \\ 1 & \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \dots & \phi_M(\mathbf{x}_2) \\ 1 & \phi_1(\mathbf{x}_3) & \phi_2(\mathbf{x}_3) & \dots & \phi_M(\mathbf{x}_3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \dots & \phi_M(\mathbf{x}_N) \end{bmatrix}$$

The least squares estimator is

$$\mathbf{w}^\star_{\text{lse}} = (\widetilde{\mathbf{\Phi}}^\top \widetilde{\mathbf{\Phi}})^{-1} \widetilde{\mathbf{\Phi}}^\top \mathbf{y}$$

Issue

The more basis functions we add the more powerful the model becomes and so we will be able to model more and more complex cases. But at the same time it is more and more likely that we overfit.

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 $\|\mathbf{w}\|_2^2 = \sum_{j=0}^{M} w_j^2$. Here the main effect is that large model weights w_j will be penalized (avoided), since we consider them "unlikely", while small ones are ok.
When $\mathcal{L}$ is MSE, this is called ridge regression:

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

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

Explicit solution for w: Differentiating and setting to zero:

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

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

Ridge Regression to Fight Ill-Conditioning

The eigenvalues of $(\widetilde{\mathbf{\Phi}}^\top\widetilde{\mathbf{\Phi}} + \lambda'\mathbf{I}_{M+1})$ are all at least λ' and so the inverse always exists. This is also referred to as lifting the eigenvalues.

Proof: Write the singular-value decomposition of $\widetilde{\mathbf{\Phi}}^\top\widetilde{\mathbf{\Phi}}$ as $\mathbf{USU}^\top$. We then have

$$\begin{aligned}\widetilde{\mathbf{\Phi}}^\top\widetilde{\mathbf{\Phi}} + \mathbf{I}_{M+1} &= \mathbf{USU}^\top + \lambda'\mathbf{UI}_{M+1}\mathbf{U}^\top \\ &= \mathbf{U}[\mathbf{S} + \lambda'\mathbf{I}_{M+1}]\mathbf{U}^\top.\end{aligned}$$

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

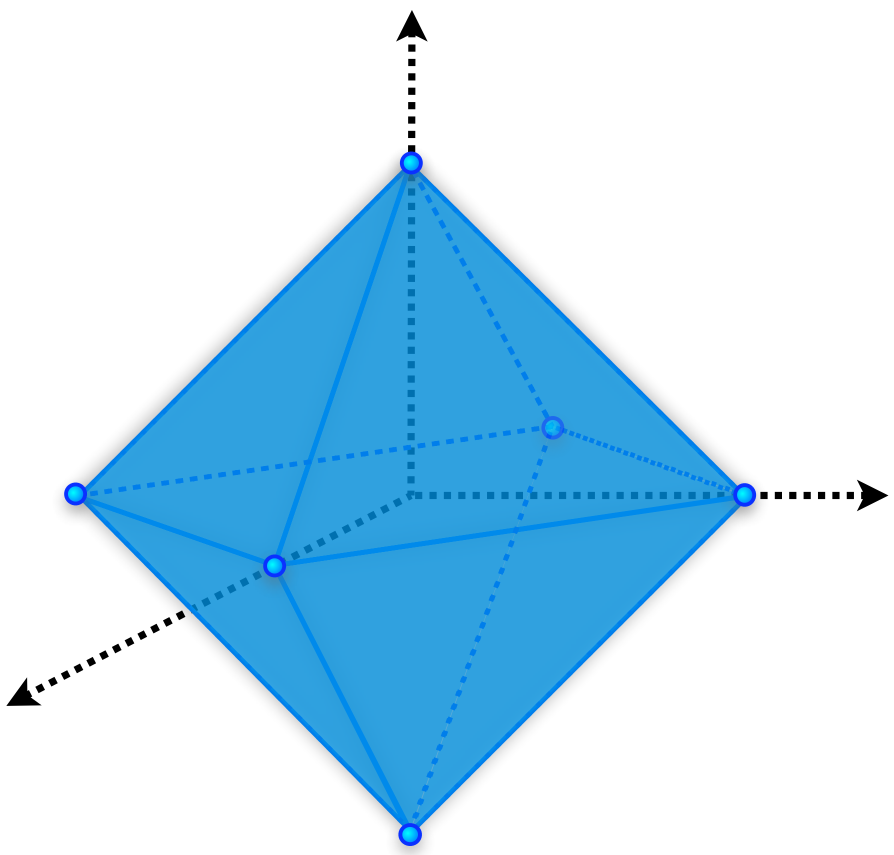
L_1-Regularization: The Lasso

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:

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

where

$$\|\mathbf{w}\|_1 := \sum_{i=0}^{M}|w_i|.$$



The figure above shows a "ball" of constant L_1 norm. To keep things simple assume that $\widetilde{\mathbf{\Phi}}^\top \widetilde{\mathbf{\Phi}}$ is invertible. Then the set

$$\{\mathbf{w} : \|\mathbf{y} - \widetilde{\mathbf{\Phi}}^\top \mathbf{w}\|^2 = \alpha\}$$

is an ellipsoide and this ellipsoide simply scales around it's origin as we change α. The optimum solution is not gotten by scaling both, this ellipsoid as well the constant L_1 "ball", so that they both "just touch" in such a way that the sum of the two contributions is minimum. Because of the geometry of the L_1 ball it is now quite likely that the point where they touch is on one of the lower-dimensional faces, i.e., the solution is sparse.
In turn, sparsity is desirable, since it leads to a "simple" model.

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 $\tau > 0$).

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

The following picture illustrates this.

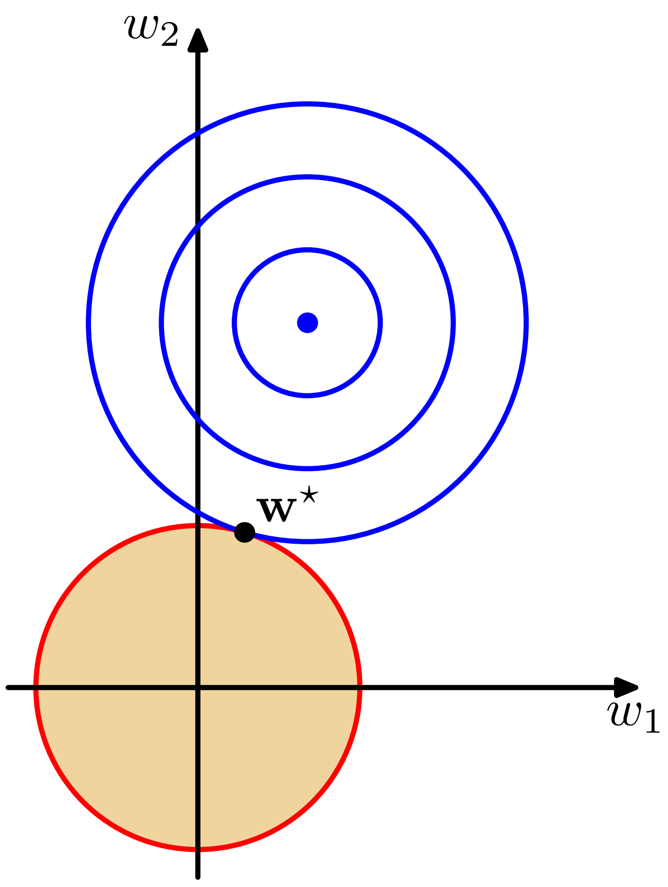


Figure 1: Geometric interpretation of Ridge Regression. Blue lines indicating the level sets of the MSE cost function.

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

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

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 the picture similar to above for Lasso, 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 least-squares can be interpreted as the maximum likelihood estimator.

$$\mathbf{w}_{\text{lse}} = \arg\max_{\mathbf{w}} \quad \log\left[\prod_{n=1}^{N}\mathcal{N}(y_n \,|\, \mathbf{x}_n^\top \mathbf{w}, \sigma^2)\right]$$

Ridge regression has a very similar interpretation:

$$\mathbf{w}_{\text{ridge}} = \arg\max_{\mathbf{w}} \quad \log\left[\prod_{n=1}^{N}\mathcal{N}(y_n \,|\, \mathbf{x}_n^\top \mathbf{w}, \sigma^2) \cdot \mathcal{N}(\mathbf{w} \,|\, 0, \tfrac{1}{\lambda}\mathbf{I})\right]$$

This is called a Maximum-a-posteriori (MAP) estimate.

Plug in the definition of the Gaussian distribution, and take the log, to see that you obtain the Ridge Regression optimization problem.

MAP Estimate and Bayes' Rule

In general, a MAP estimate maximizes the product of the likelihood *and* the prior (on the model), as opposed to a maximum likelihood estimate which maximizes only the former.

$$\mathbf{w}_{\text{lik}} = \arg\max_{\mathbf{w}} \; p(\mathbf{y} \,|\, \mathbf{w}, \mathbf{X}, \boldsymbol{\lambda}) \tag{1}$$

$$\mathbf{w}_{\text{MAP}} = \arg\max_{\mathbf{w}} \; p(\mathbf{y} \,|\, \mathbf{w}, \mathbf{X}, \boldsymbol{\lambda}) \cdot p(\mathbf{w} \,|\, \boldsymbol{\theta}) \tag{2}$$

Here $p(\mathbf{y} \,|\, \mathbf{w}, \mathbf{X}, \boldsymbol{\lambda})$ defines the *likelihood* of observing the data $\mathbf{y}$ given $\mathbf{X}, \mathbf{w}$ and some likelihood parameters $\boldsymbol{\lambda}$.
Similarly, $p(\mathbf{w} \,|\, \boldsymbol{\lambda})$ is the *prior* distribution. This incorporates our prior knowledge about $\mathbf{w}$. Using the Bayes rule,

$$p(A, B) = p(A|B)\, p(B) = p(B|A)\, p(A) \tag{3}$$

we can rewrite the MAP estimate, as follows:

$$\mathbf{w}_{\text{MAP}} = \arg\max_{\mathbf{w}} \; p(\mathbf{w} \,|\, \mathbf{y}, \mathbf{X}, \boldsymbol{\lambda}, \boldsymbol{\theta}) \tag{4}$$

Therefore, the MAP estimate is the maximum of the posterior distribution, which explains the name.

Can Linear Models Really Overfit?

Yes! #2: In the previous chapter on overfitting, we have seen an example of overfitting for linear regression, when using polynomial basis functions. Here is another example:
Let us try simple linear regression on the weather in Lausanne. Our prediction target y_n is 1 if day n was a sunny day, and 0 if it rained. First try, we feed only on input feature to the model, $x_n \in \mathbb{R}$ being the temperature on that day. Linear regression will do poorly in this case. (Note the data matrix $\mathbf{X}$ in this case is of dimension $n \times 1$).
For a second try, consider a simple feature expansion. In addition to the temperature recorded in x_{n0} for each day n, we add new features $x_{ni} := 1$ if $i = n$ and zero otherwise. This is called an indicator feature, for each datapoint indicating the 'active' day n corresponding to the datapoint. (Note the data matrix $\mathbf{X}$ in this case is of dimension $n \times (1 + n)$).
Now multi-linear regression can with 100% accuracy fit the weather of all days perfectly! Simply set $w_0 := 0$, and $w_i := 1$ if day i was a sunny day, and zero otherwise. However, this model $\mathbf{w}$

- has perfect accuracy on all past days n
- ignores all reasonable features ($w_0 = 0$ for temperature x_{n0})
- completely overfits,

and is totally useless, since it only 'learned' meaningless features, that is from the features x_{ni} for $1 \leq i \leq n$.
While the above is an artificial example, similar phenomena happen in many practical applications to various degrees of severity, especially once a large number of features is available. This again illustrates that regularization is necessary in practice, for most models, to avoid overfitting.
(Even in our artificial example, regularization would actually be beneficial, and cause the reasonable feature x_{n0} to be *also* used, i.e. $w_0 \neq 0$. Try it out in Python if you're interested.)