STATS 50: Mathematics of Sport **Spring 2019**

Week 4 – Regularized Linear Regression

Lecturer: Maxime Cauchois

Warning: *these notes may contain factual errors*

1 Regularization in statistics

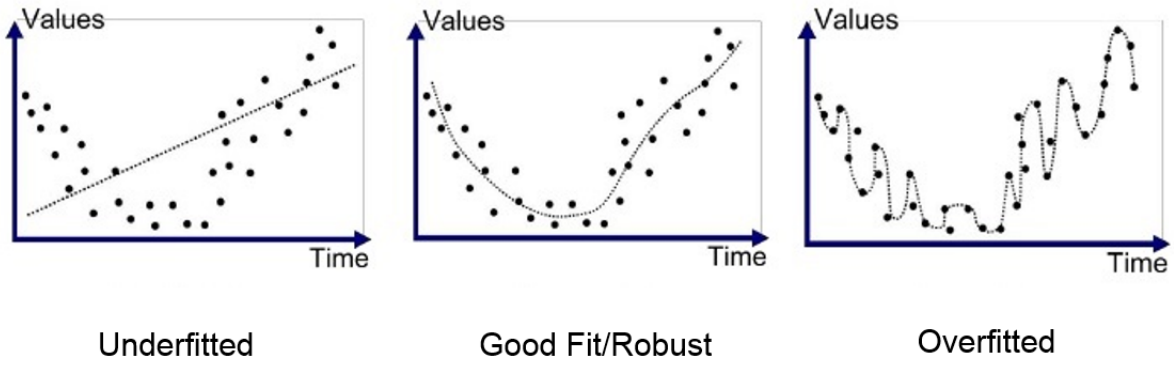

Figure 1: Regularization in statistics and machine learning

In a wide variety of problems, we are given a set of data $(x_i, y_i)_{i\in[n]}$, where $x \in \mathbb{R}^d$ represent all the potential predictors of our response y. Our goal is then to learn from our data a function $\hat{f}$ which maps x to y. In a lot of cases, we will restrict f to be linear or polynomial, and estimate its value $\hat{f}$ via least-squares, or more generally by solving an optimization problem.

However, we will sometimes observe at first that our model is overfitting to the dataset used to yield an estimate $\hat{f}$: we say that our model does not generalize well, because it performs poorly on newly came instances, and over-reliant to the instances at hand. For instance, suppose that we set our goal to predict the outcome of the next NBA playoffs, but obviously we can only rely on the regular season games or/and the previous NBA playoffs to build our predictions. Not only might the teams not display an homogeneous performance during those different periods of the season (because of several factors, such as motivation, etc) but each game itself contains some inherent noise which perturbs a potential prediction. To sum up, we have to take into account the fact that we want our model to perform well not on the data it has seen, but on data that it has not.

In figure 1, one can observe the two extreme cases that a data scientist tries to avoid when developing a new model. If the model is too simplistic for our data, we will not even be able to fit our current dataset, in which case we probably have to refine our set of possible functions $\hat{f}$, allowing more sophisticated models. On the contrary, if the model is too "high variance", meaning that the function $\hat{f}$ is not smooth enough, we might want to add some more regularization (or penalization) to our problem (3).

This standard phenomenon in data science and statistics is called *bias-variance trade-off*, and implies that one has to find the right balance when choosing our model complexity: increasing it will reduce the bias but may induce higher variance loss.

In the standard setting $(x, y) \overset{i.i.d}{\sim} \mathbb{P}$, we want to find the function $f^\star$ which minimizes:

$$\underset{f \in \mathcal{F}}{\text{minimize}}\ \mathbb{E}(L(y, f(x))) \tag{1}$$

where $\mathcal{F}$ is a set of functions (e.g. the set of linear functions from $\mathcal{X}$ to Y), and L is a loss function characterizing the quality of the prediction $f(x)$ for y. A typical example of loss function is the square loss: $L(f(x), y) = (y - f(x))^2$.

In a perfect world, if we were to know the distribution $\mathbb{P}$, that is if we knew the exact link between x and y, as well as how x is distributed in our population of interest, we would simply consider $f^\star \in \mathcal{F}$ minimizing (1), because it is by definition the best approximation of y by a function of x in our model. Of course, we dispose only of a finite number of instances, so (1) can just be approximated by an empirical version.

If we finally obtain an estimate $\hat{f}$ of a function, $\hat{f}$ is itself a random function depending on the samples $(x_i, y_i)_{i \in [n]}$ we observed. Now, if we are given a new example $x \in \mathcal{X}$ to predict (fixed here), we can estimate the level of suboptimality of our estimator when L is the squared loss by:

$$\mathbb{E}\left[(f(x) - \hat{f}(x))^2\right] = \overbrace{\mathbb{E}\left[(f(x) - \mathbb{E}(\hat{f}(x)))^2\right]}^{\text{Bias}} + \overbrace{\text{Var}(\hat{f}(x))}^{\text{Variance}} \tag{2}$$

If $\hat{f}$ is too noisy, the variance part will be large, which will hurt our prediction. On the other hand, if $\hat{f}$ is chosen among a set too small to account for the complexity of the true model, the bias term will tend to increase. To find the correct balance, we often penalize models which are too "complicated", because **generalization to new data is the our main desiderata**. In other terms, we usually solve the following problem:

$$\underset{\hat{f} \in \hat{\mathcal{F}}}{\text{minimize}}\ \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{f}(x_i)) + \mathcal{R}(\hat{f}) \tag{3}$$

where $\hat{\mathcal{F}}$ is a set of "acceptable" functions, L stands for the loss function which characterizes the distance between our estimate $\hat{f}(x_i)$ and y_i in order to make it as small as possible (and fit our data), and $\mathcal{R}(\hat{f})$ represents our penalization term, which will basically grow large when $\hat{f}$ becomes too "noisy" and overfits our data.

By performing the minimization written in (3), we can see that we are making a trade-off between two antagonist forces. On the one hand, we would like to constrain all the $\hat{f}(x_i)$ to be equal to y_i so as to cancel the first term of our loss. On the other hand, the second terms prevents $\hat{f}$ from being too erratic and encourages us to find smoother functions to represent our problem. To draw a parallel with the figure 1, the first term of our loss pushes $\hat{f}$ towards the left, while the second terms pulls $\hat{f}$ on the left. The "art" of machine learning is then to find the correct trade-off between both so as to end up in the middle.

2 Regularization in linear regression

We now come back to the more narrow setting of linear regression, where we assume that the function $\hat{f}$ mapping x to y is linear, parameterized by some $\theta \in \mathbb{R}^d$ such that:

$$\hat{f}_\theta(x) = \theta^T x$$

We previously studied the simple least-squares scenario, where we only try to minimize over all θ:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^T x_i)^2 = \frac{1}{n}\|Y - X\theta\|_2^2 \tag{4}$$

where $X = (x_1^T, ..., x_n^T)$ and $Y = (y_1, ..., y_n)$ are concatenated versions of our data.

We can observe that the standard least-squares problem adds no regularization term to our problem. However, in many scenarios, such as Bradley-Terry or Rasch models which will be studied in the next few weeks, we are going to add penalization terms and solve the following problem:

$$\underset{\theta}{\text{minimize}}\ \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^T x_i)^2 + \mathcal{R}_\lambda(\theta) \tag{5}$$

where $\mathcal{R}_\lambda$ is a regularization term which essentially takes large values when θ itself is large. Its magnitude is often parameterized by some $\lambda > 0$, which allows to mitigate the effects of regularization and control the bias-variance trade-off that we make: as λ grows, we penalize large values of θ even more so that we drag the value of θ towards 0, whereas if $\lambda \to 0$, we effectively remove any regularization term and allow θ to be close to its initial least-squares solution.

Adding a regularization term has essentially two practical main reasons:

- We avoid any identifiability issue, in other terms we ensure that there is a single solution θ to our problem (5). Indeed, in (4), there are several cases in which two different values of θ achieve the same mean squared error, in which case it is not obvious at first sight which one we should choose. We will see in the next lectures that this case can appear more often than not, and is strongly connected with the underlying model we have decided to apply to our data.

- We sometimes just want to encourage small values of θ, as in the case of regression to the mean! Indeed, suppose that each θ_j represents some kind of deviation from the average for the player j, then we previously saw that shrinking every estimate towards 0 was often quantitatively preferable to the more naive estimate: the same phenomenon appears here.

3 Ridge Regression

There are two main types of regularization used in linear regression: the Lasso or l_1 penalty (see [1]), and the ridge or l_2 penalty (see [2]). Here, we will rather focus on the latter, despite the growing trend in machine learning in favor of the former.

Ridge regression uses the l_2 norm of θ as a penalization term, solving the following problem:

$$\underset{\theta}{\text{minimize}}\left\{\frac{1}{n}\|Y - X\theta\|_2^2 + \lambda\|\theta\|_2^2\right\} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^T x_i)^2 + \lambda\sum_{j=1}^{d}\theta_j^2 \tag{6}$$

For each $\lambda > 0$, we get an estimate $\hat{\theta}_\lambda$ of our parameter, which can be computed using the following closed formula:

$$\hat{\theta}_\lambda = \underset{\theta}{\text{argmin}}\ \frac{1}{n}\left\{\|Y - X\theta\|_2^2 + \lambda\|\theta\|_2^2\right\} = (X^T X/n + \lambda I)^{-1} X^T Y$$

We see that the expression we get is quite close to the least-squares expression, which should not be so surprising if we consider that we actually also solved a least-squares problem. Indeed, solving (6) comes down to artificially adding d data points $(\tilde{x}_i, \tilde{y}_i)_{1 \leq i \leq d}$ with $\tilde{x}_{ij} = \sqrt{\lambda}$ if $i = j$ and 0 otherwise, and $\tilde{y}_i = 0$. In other terms, when we replace a linear regression model by a ridge regression one, it implies that we do not necessarily trust our model too much, and we prefer adding some "false" data points so as not to rely too much on our n data points.

One final discussion remains open nonetheless. It is almost immediate to see that the amount of confidence we put in our linear model is inversely proportional to the magnitude of λ, leading to the following question: how should we choose λ?

4 Cross Validation

Ideally, we would like to choose λ so as to end up with a model which generalizes as well as possible on future data. However, we also know that any information that we would get using already processed data would be biased, so we have to come up with a scheme which allows us to evaluate our model on some "uncontaminated" data. Here is an algorithm which allows us to do that, and is called K-fold cross validation.

Data: (X, Y) where $X \in \mathbb{R}^{n \times d}$ and $Y \in \mathbb{R}^n$
Result: λ_{opt}
Randomly split your data (X, Y) into K equal parts $(X^{(1)}, Y^{(1)}), ...(X^{(K)}, Y^{(K)})$.
for $k = 1, ..., K$ **do**

- Solve (6) and obtain $\hat{\theta}_\lambda^{(k)}$ for several values of λ, using all but the k-fold of your data
- Use each $\hat{\theta}_\lambda^{(k)}$ to make predictions on the k-fold of your data (previously retained from the training part), ie compute $X^{(k)}\hat{\theta}_\lambda^{(k)}$ for all possible values of λ
- Get a mean squared error on the k-fold for each value of λ, comparing your predictions to the actual values $Y^{(k)}$

For each λ, compute the average mean-squared error over all K-folds, and return λ with the lowest one.

end

Algorithm 1: K-fold cross-validation

The value of λ returned by the K-fold CV algorithm is the one more susceptible of generalizing best, and should therefore be the one we eventually use for prediction purposes.

5 Regularized Linear Regression in R

In R, there are some built-in packages which can run cross validation for us! To fit a regularized linear regression model in R, you will need to have installed (the first time) and loaded (every time) the `glmnet` package. To fit the model with cross validation to choose , use the `cv.glmnet()` function:

```
> mod_ridge = cv.glmnet(X, y, alpha = 0, intercept = FALSE, standardize = FALSE)
```

To use the model to predict score differentials, use the `predict()` function:

```
> pred = predict(mod_ridge, X, s = 'lambda.min')
```

To extract the estimated $\hat{\theta}$ from the model, it is easiest to again use the `predict()` function (and the fact that if $X = I$, then the prediction is $\hat{y} = I\hat{\theta} = \hat{\theta}$):

```
> theta = predict(mod_ridge, diag(ncol(X)), s = 'lambda.min')
```

The `cv.glmnet()` function takes care of cross validation. The one thing you want to check is that the default range of λ values is sufficient. To view a diagnostic plot, use the `plot()` function:

```
> plot(mod_ridge)
```

References

[1] Tibshirani, Robert. "Regression shrinkage and selection via the lasso." Journal of the Royal Statistical Society. Series B (Methodological) (1996): 267-288.

[2] Hoerl, Arthur E., and Robert W. Kennard. "Ridge regression: Biased estimation for nonorthogonal problems." Technometrics 12.1 (1970): 55-67.