Applied Data Science

Session 18: Regularized Regression

Dr. Soharab Hossain Shaikh

1

Note

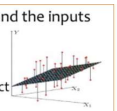
- In the slides we have used the following to denote the model parameters/coefficients (Θ, β, w) to be learnt during training.
- The hypothesis function is parameterized by these terms.
  - Θ_i or β_i or w_i denote individual parameters.
  - $\boldsymbol{\Theta}$ or $\boldsymbol{\beta}$ or $\mathbf{w}$ denote the vector of parameters.
- These terms have been **<u>used interchangeably</u>** in the slides.

2

Probabilistic Interpretation of LMS

- Let us assume that the target variable and the inputs are related by the equation:

$$y_i = \theta^T \mathbf{x}_i + \varepsilon_i$$



where ε is an error term of unmodeled effect noise

- Now assume that ε follows a Gaussian $N(0,\sigma)$, then we have:

$$p(y_i \mid x_i;\theta) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i-\theta^T\mathbf{x}_i)^2}{2\sigma^2}\right)$$

- By independence assumption:

$$L(\theta) = \prod_{i=1}^{n} p(y_i \mid x_i;\theta) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{\sum_{i=1}^{n}(y_i-\theta^T\mathbf{x}_i)^2}{2\sigma^2}\right)$$

3

Probabilistic Interpretation of LMS

- Hence the log-likelihood is:

$$l(\theta) = n\log\frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2}\frac{1}{2}\sum_{i=1}^{n}(y_i-\theta^T\mathbf{x}_i)^2$$

- Do you recognize the last term?

Yes it is: $$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\theta - y_i)^2$$

- Thus under independence assumption, LMS is equivalent to MLE of θ !

4

Non-linear Basis Function

- So far we only used the observed values $x_1, x_2, \ldots$
- However, linear regression can be applied in the same way to **functions** of these values
  - Eg: to add a term w x_1x_2 add a new variable $z=x_1x_2$ so each example becomes: x_1, x_2, …. z
- As long as these functions can be directly computed from the observed values the parameters are still linear in the data and the problem remains a multi-variate linear regression problem

$$y = w_0 + w_1 x_1^2 + \ldots + w_k x_k^2 + \varepsilon$$

5

Non-linear Basis Functions

- What type of functions can we use?
- A few common examples:

- Polynomial: $\phi_j(x) = x^j$ for $j=0 \ldots n$

- Gaussian: $\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$

- Sigmoid: $\phi_j(x) = \frac{1}{1+\exp(-s_j x)}$

- Logs: $\phi_j(x) = \log(x+1)$

Any function of the input values can be used. The solution for the parameters of the regression remains the same.

6

General Linear Regression Problem

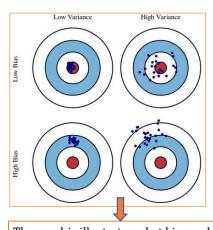
- Using our new notations for the basis function linear regression can be written as

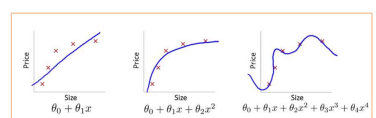
$$y = \sum_{j=0}^{n} w_j \, \phi_j(\mathbf{x})$$

- Where $\phi_j(\mathbf{x})$ can be either x_j for multivariate regression or one of the non-linear basis functions we defined
- … and $\phi_0(\mathbf{x})=1$ for the intercept term

7

Bias and Variance of a ML Model



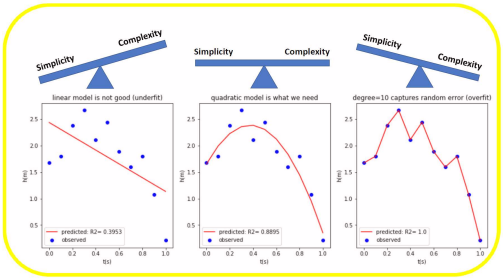


High bias (underfit)	"Just right"	High variance (overfit)
d=1	d=2	d=4
Over simplified	Just-right – the optimum	Over complicated

The graphic illustrates what bias and variance are.
Imagine the bull's-eye is the true population parameter that we are estimating, β, and the shots at it are the values of our estimates resulting from four different estimators - low bias and variance, high bias and variance, and the combinations thereof.

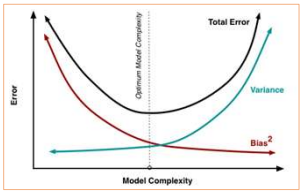
8

Bias Variance Tradeoff



9

Regularized Regression



- The **OLS** estimator has the desired property of being **unbiased**.
- However, it can have a **huge variance**. Specifically, this happens when:
- The predictor variables are highly correlated with each other;
- There are many predictors. if number pf predictors approaches total number of samples then variance approaches infinity.
- The general solution to this is: **reduce variance at the cost of introducing some bias**.
- This approach is called **regularization** and is almost always beneficial for the predictive performance of the model.

10

Regularized Regression

- As the model complexity, which in the case of linear regression can be thought of as the number of predictors, increases, estimates' variance also increases, but the bias decreases.
- The unbiased OLS would place us on the right-hand side of the picture, which is far from optimal.
- That's why we regularize: to lower the variance at the cost of some bias, thus moving left on the plot, towards the optimum.

11

Regularization

- Penalize large coefficients (*controlling the importance of a feature*)
- Making some coefficients zero (e.g. Lasso) – the feature vanishes (effect of *feature selection* and *dimension reduction*)
- Controls model overfitting by not giving too much importance to any of the features. e.g. Ridge makes coefficients small.
- Feature selection also helps the model be trained on relatively less number of features (lower dimensional feature vectors). Faster training – training possible with relatively less amount of data.

12

Ridge Regression

Ridge regression penalizes **sum of squared** coefficients (L2 penalty).

- Adds an **L2 regularizer** to Linear Regression

$$J_{\mathrm{RR}}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \boxed{\lambda ||\boldsymbol{\theta}||_2^2}$$

$$= \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)})^2 + \boxed{\lambda \sum_{k=1}^{K}\theta_k^2}$$

prefers parameters close to zero

- Bayesian interpretation: MAP estimation with a **Gaussian prior** on the parameters

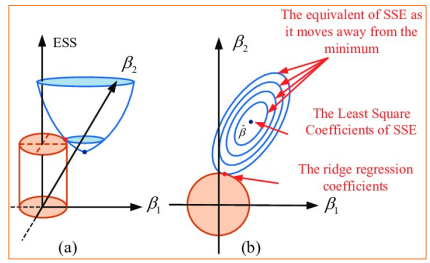
$$\boldsymbol{\theta}^{MAP} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(y^{(i)}|\mathbf{x}^{(i)}) + \log p(\boldsymbol{\theta})$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, J_{\mathrm{RR}}(\boldsymbol{\theta})$$

where $p(\boldsymbol{\theta}) \sim \mathcal{N}(0, \frac{1}{\lambda})$

13

Ridge Regression



14

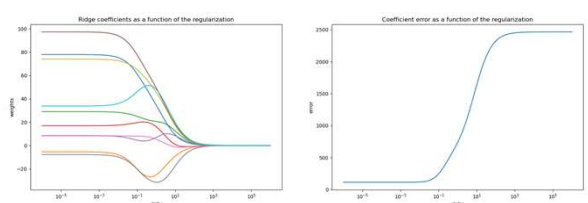
Ridge Regression

- The $\boldsymbol{\lambda}$ parameter is the regularization penalty.
- Notice that:
- As $\lambda \to 0$ $\quad \theta_{ridge} \to \theta_{OLS}$
- As $\lambda \to \infty$ $\quad \theta_{ridge} \to 0$

- So, setting λ to 0 is the same as using the OLS, while the larger its value, the stronger is the coefficients' size penalized.
- **as $\boldsymbol{\lambda}$ becomes larger, the variance decreases, and the bias increases**.
- How much bias are we willing to accept in order to decrease the variance? In other words, what is the optimal value for λ?

15

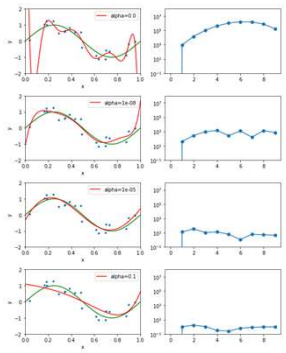
Ridge Coefficients as a function of the L2 Regularization



Regularization strength = $\boldsymbol{\lambda}$ in the previous equation.
In Sklearn this parameter is called *alpha*.

16

Model Coefficients Shrinkage with Ridge



17

Lasso Regression

Least Absolute Shrinkage and Selection Operator (LASSO)

- Adds an **L1 regularizer** to Linear Regression

$$J_{\text{LASSO}}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \boxed{\lambda||\boldsymbol{\theta}||_1}$$

$$= \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)})^2 + \boxed{\lambda\sum_{k=1}^{K}|\theta_k|}$$

yields sparse parameters (exact zeros)

- Bayesian interpretation: MAP estimation with a **Laplace prior** on the parameters

$$\boldsymbol{\theta}^{MAP} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(y^{(i)}|\mathbf{x}^{(i)}) + \log p(\boldsymbol{\theta})$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, J_{\text{LASSO}}(\boldsymbol{\theta})$$

where $p(\boldsymbol{\theta}) \sim \text{Laplace}(0, f(\lambda))$

18

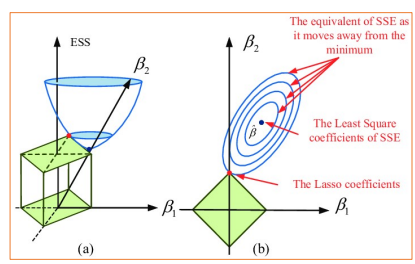
Lasso Regression

- Lasso, or Least Absolute Shrinkage and Selection Operator, is quite similar conceptually to ridge regression.
- It also adds a penalty for non-zero coefficients, but unlike ridge regression which penalizes sum of squared coefficients (L2 penalty), lasso penalizes the sum of their absolute values (L1 penalty).
- As a result, for high values of λ, many coefficients are exactly zeroed under lasso, which is never the case in ridge regression.

19

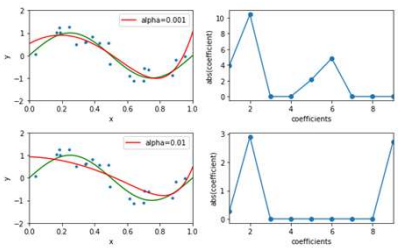
Lasso Regression

Lasso tends to generate *sparser* solutions than a quadratic regularizer.



20

Model Coefficients Shrinkage with Lasso



Regularization strength = λ in the previous equation.

In Sklearn this parameter is called *alpha*.

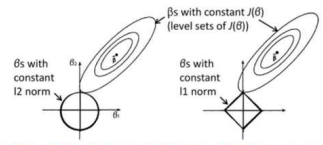
21

Ridge vs. Lasso Regression

$$\min_{\beta}(\mathbf{X}\beta - \mathbf{Y})^T(\mathbf{X}\beta - \mathbf{Y}) + \lambda \text{pen}(\beta) = \min_{\beta} J(\beta) + \lambda \text{pen}(\beta)$$

Ridge Regression: $\text{pen}(\beta) = \|\beta\|_2^2$

Lasso: $\text{pen}(\beta) = \|\beta\|_1$



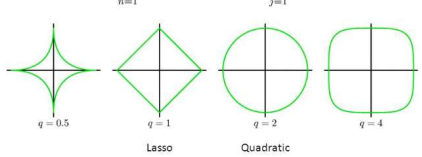
Lasso (l1 penalty) results in sparse solutions – vector with more zero coordinates
Good for high-dimensional problems – don't have to store all coordinates!

22

Regularized Least Squares

With a more general regularizer, we have

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$



23

Ridge vs. Lasso Regression

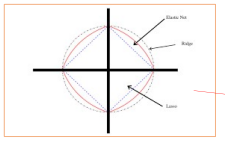
- Often neither one is overall better.
- Lasso can set some coefficients to zero, thus performing variable selection, while ridge regression cannot.
- Both methods allow to use correlated predictors, but they solve multicollinearity issue differently:
  - In ridge regression, the coefficients of correlated predictors are similar;
  - In lasso, one of the correlated predictors has a larger coefficient, while the rest are (nearly) zeroed.
- Lasso tends to do well if there are a small number of significant parameters and the others are close to zero (ergo: when only a few predictors actually influence the response).
- Ridge works well if there are many large parameters of about the same value (ergo: when most predictors impact the response).
- However, in practice, we don't know the true parameter values, so the previous two points are somewhat theoretical. Just run cross-validation to select the more suited model for a specific case.
- Or... combine the two!

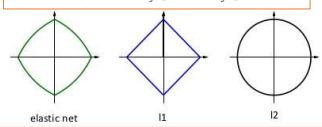
24

Elastic Net

- Originally proposed for feature selection
  - to group correlated features together
- Trade-off between sparsity and security against **l2-norm** attacks

$$\text{minimize}\left\{SSE + \lambda_1 \sum_{j=1}^{p} \beta_j^2 + \lambda_2 \sum_{j=1}^{p} |\beta_j|\right\}$$





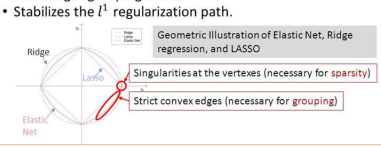
25

Elastic Net

- *Elastic Net* penalize the size of the regression coefficients based on both their l^1 norm and their l^2 norm :

$$argmin_\beta \sum_i (y_i - \boldsymbol{\beta}'\boldsymbol{x}_i)^2 + \lambda_1 \sum_{k=1}^{K} |\beta_k| + \lambda_2 \sum_{k=1}^{K} \beta_k^2$$

- The l^1 norm penalty generates a sparse model.
- The l^2 norm penalty:
  - Removes the limitation on the number of selected variables.
  - Encourages grouping effect.
  - Stabilizes the l^1 regularization path.



26

Elastic Net

- Elastic Net first emerged as a result of critique on lasso, whose variable selection can be too dependent on data and thus unstable. The solution is to combine the penalties of ridge regression and lasso to get the best of both worlds.
- Elastic Net aims at minimizing a loss function that involves both L1 and L2 penalty terms.

$$\textbf{Penalty Term} = \lambda\left(\frac{1-\alpha}{2}\sum_{j=1}^{m}\hat{\beta}_j^2 + \alpha\sum_{j=1}^{m}|\hat{\beta}_j|\right)$$

Here λ is the regularization parameter – controls the degree of regularization.
α is the parameter controlling the influence of the L1 vs. L2 penalty.
β_j are the model parameters/coefficients.

27

Optimal value of λ

- $\boldsymbol{\lambda}$ is a hyperparameter o the model.
- To choose $\boldsymbol{\lambda}$ through cross-validation, you should choose a set of values (denote by p in the algorithm) of λ to test, split the dataset into K folds, and follow this algorithm:

- for p in 1:P:
- for k in 1:K:
- keep fold k as hold-out data
- use the remaining folds and $\lambda = \lambda_p$ to estimate $\hat{\beta}_{ridge}$
- predict hold-out data: $y_{test,k} = X_{test,k}\hat{\beta}_{ridge}$
- compute a sum of squared residuals: $SSR_k = \|y - y_{test,k}\|^2$
- end for k
- average SSR over the folds: $SSR_p = \frac{1}{K}\sum_{k=1}^{K} SSR_k$
- end for p
- choose optimal value: $\lambda_{opt} = argmin_p SSR_p$

28

Summary

- If your linear model contains many predictor variables or if these variables are correlated, the standard OLS parameter estimates have large variance, thus making the model unreliable.
- To counter this, you can use regularization - a technique allowing to decrease this variance at the cost of introducing some bias. Finding a good bias-variance trade-off allows to minimize the model's total error.
- There are three popular regularization techniques, each of them aiming at decreasing the size of the coefficients:
  - Ridge Regression, which penalizes sum of squared coefficients (L2 penalty).
  - Lasso Regression, which penalizes the sum of absolute values of the coefficients (L1 penalty).
  - Elastic Net, a convex combination of Ridge and Lasso.
- The size of the respective penalty terms can be tuned via cross-validation to find the model's best fit.

29

How to *build* a Regularized Regression Model in *Scikit-Learn*

30

Ridge Regressor in Scikit-Learn

scikit-learn

sklearn.linear_model.Ridge

class sklearn.linear_model.**Ridge**(*alpha=1.0, *, fit_intercept=True, normalize=False, copy_X=True, max_iter=None, tol=0.001, solver='auto', random_state=None*) [source]

- Build an estimator model like **Ridge()**
- Use **fit()** function to ***Train*** the model with **Training Dataset**
- Use **predict()** function to ***Test/Evaluate*** the model with **Test Dataset**
- Use **predict()** function to make ***prediction/inference*** on **New Unseen Data**

31

Example Code

```
>>> from sklearn.linear_model import Ridge
>>> import numpy as np
>>> n_samples, n_features = 10, 5
>>> rng = np.random.RandomState(0)
>>> y = rng.randn(n_samples)
>>> X = rng.randn(n_samples, n_features)
>>> clf = Ridge(alpha=1.0)
>>> clf.fit(X, y)
Ridge()
```

32

Lasso Regressor in Scikit-Learn

scikit-learn

sklearn.linear_model.Lasso

class sklearn.linear_model.Lasso(*alpha=1.0, *, fit_intercept=True, normalize=False, precompute=False, copy_X=True, max_iter=1000, tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic'*) [source]

- Build an estimator model like **Lasso()**
- Use **fit()** function to ***Train*** the model with **Training Dataset**
- Use **predict()** function to ***Test/Evaluate*** the model with **Test Dataset**
- Use **predict()** function to make ***prediction/inference*** on **New Unseen Data**

33

Example Code

```
>>> from sklearn import linear_model
>>> clf = linear_model.Lasso(alpha=0.1)
>>> clf.fit([[0,0], [1, 1], [2, 2]], [0, 1, 2])
Lasso(alpha=0.1)
>>> print(clf.coef_)
[0.85 0.  ]
>>> print(clf.intercept_)
0.15...
```

34

ElasticNet Regressor in Scikit-Learn

scikit-learn

sklearn.linear_model.ElasticNet

class sklearn.linear_model.ElasticNet(*alpha=1.0, *, l1_ratio=0.5, fit_intercept=True, normalize=False, precompute=False, max_iter=1000, copy_X=True, tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic'*) [source]

- Build an estimator model like **ElasticNet()**
- Use **fit()** function to ***Train*** the model with **Training Dataset**
- Use **predict()** function to ***Test/Evaluate*** the model with **Test Dataset**
- Use **predict()** function to make ***prediction/inference*** on **New Unseen Data**

35

Example Code

```
>>> from sklearn.linear_model import ElasticNet
>>> from sklearn.datasets import make_regression
```

```
>>> X, y = make_regression(n_features=2, random_state=0)
>>> regr = ElasticNet(random_state=0)
>>> regr.fit(X, y)
ElasticNet(random_state=0)
>>> print(regr.coef_)
[18.83816048 64.55968825]
>>> print(regr.intercept_)
1.451...
>>> print(regr.predict([[0, 0]]))
[1.451...]
```

36

References

- Christopher Bishop; *"Pattern Recognition and Machine Learning"*; Springer.
- Trevor Hastie, Robert Tibshirani, Jerome Friedman; "*The Elements of Statistical Learning: Data Mining, Inference, and Prediction*"; Springer.

37

38

Go to the Coding Demo…

39

To be continued in the next session…..

40