Regularised Regression

Vijay Jaisankar

Teaching Assistant

Made these slides with this song on repeat

Agenda

- Overview of Linear Regression
- Overfitting and underfitting
- Regularisation
- Ridge regression and Lasso regression
- Elastic-Net
- Some practical things to keep in mind

Linear Regression : Matrix Form

Credits: Dmitry Kobak

Linear Regression - Problem Statement

The model:

$$f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p.$$

It is convenient to define $x_0 \equiv 1$. Then:

$$f(x) = \vec{\beta} \cdot \vec{x} = \boldsymbol{\beta}^\top \mathbf{x} = \begin{pmatrix} \beta_0 & \cdots & \beta_p \end{pmatrix} \begin{pmatrix} x_0 \\ \vdots \\ x_p \end{pmatrix}$$

Linear Regression - Making predictions

$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta} = \begin{pmatrix} x_0^{(1)} & x_1^{(1)} & \cdots & x_p^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \cdots & x_p^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_0^{(n)} & x_1^{(n)} & \cdots & x_p^{(n)} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} = \begin{pmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(n)} \end{pmatrix}.$$

Linear Regression - Loss Function

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{\beta}^\top \mathbf{x}^{(i)}\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left([\mathbf{y}]_i - [\mathbf{X}\boldsymbol{\beta}]_i\right)^2 = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2.$$

Gradient:

$$\nabla \mathcal{L} = -\frac{2}{n}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

Gradient Descent - Essence

- Output
- Costs
- Update Weights

Overfitting and underfitting

The importance of test data

- Why not use 100% of the data provided to make the model?
- What purpose does the test data have?

The importance of letting the model converge

```python
regressor = lr.LinearRegression(X,y)

weights_gradient_descent = regressor.getParametersGradientDescent(
    learningRate = 0.00001, # Learning Rate of the Algorithm
    numInterations = 10, # Number of iterations of the Learning Algorithm
    decay = 0 # Decay of the Learning Rate of the Algorithm
)
```

Do you see anything wrong with this?

Can something be too complex for its own good?

- Can a model be too good to be true?
- Is the data exhaustive?

Putting it all together

- Overfitting *(”mugged up the data”)*
- Underfitting *(”forgot to study”)*
- Bias-Variance trade-off

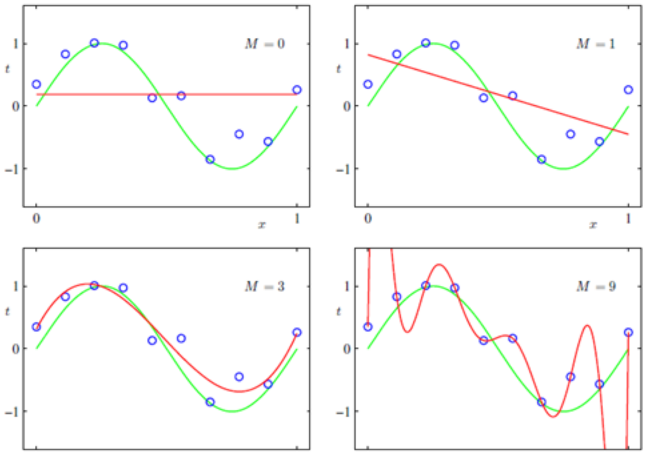
Case Study

Credits: *Pattern Recognition and Machine Learning* by Christopher Bishop

Taylor's Theorem

- How can we represent $f(x) = sin(a \cdot x)$ as a weighted sum of polynomials? *Hint: What is the title of this slide?*
- Do you see how we can model this as an instance of *Polynomial Regression?*

Fitting various Polynomials



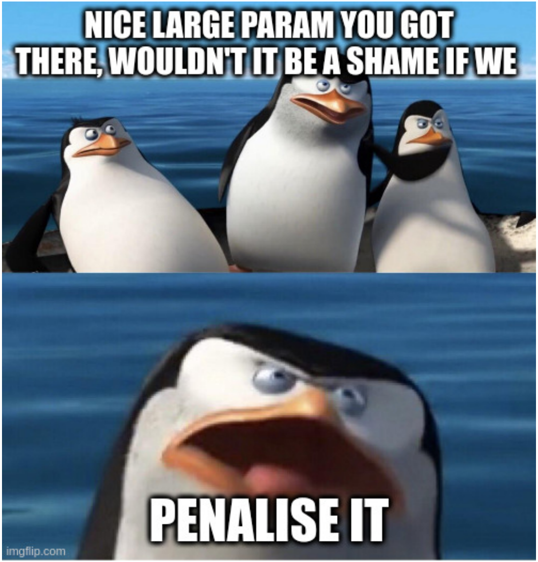
The issue with large coefficients

The issue with large coefficients

But, *why* are large parameters so bad?

- Numerical issues - overflows and such
- The more pertinent issue - how does a small change in x affect $1061800 \cdot x^6$ as opposed to $4.5 \cdot x^6$?
- Given that β represents slope, how important is it that the data does not deviate too much from its initial shape?
- **The big if** - What if the testing data deviates even slightly as compared to the training data? How do the values of β affect this loss?

Regularisation - Essence



Regularisation

Regularisation - Definition

- Regularisations are techniques used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting.

Ridge and Lasso Regression

Regularisation - new loss function

$$\mathcal{L} = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda R(\boldsymbol{\beta})$$

Here $\lambda R(\boldsymbol{\beta})$ is a *penalty* term and λ is called *regularization parameter*.

Ridge regression

Loss function:

$$\mathcal{L} = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|^2.$$

Gradient:

$$\nabla\mathcal{L} = -\frac{2}{n}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + 2\lambda\boldsymbol{\beta}.$$

Gradient descent:

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta\nabla\mathcal{L} = \boldsymbol{\beta} + \eta\frac{2}{n}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) - 2\eta\lambda\boldsymbol{\beta} =$$

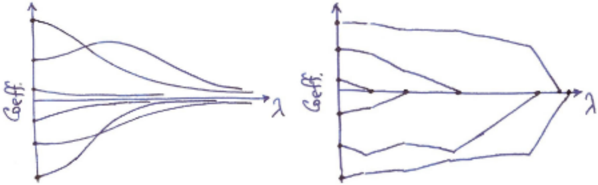
$$= \underbrace{(1 - 2\eta\lambda)}_{\text{“weight decay”}}\boldsymbol{\beta} + \eta\frac{2}{n}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

Thought Experiment

- What if $\lambda -> \infty$?
- What would that make β?
- Is this valid (if we had, say, one point or M $= 1$?)

$$\mathcal{L} = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda\|\boldsymbol{\beta}\|_1.$$

No analytic solution. But one can show that solutions are *sparse*.



Ridge

Lasso

Comparison between Lasso and Ridge Regression

- L1 regularisation cares equally about **driving down big weights to small weights**, or **driving small weights to zeros**. If you have a lot of features, and you suspect that not all of them are that important, start with Lasso.
- L2 regularisation cares more about **driving big weight to small weights**. If you only have a few features, and you are confident that all of them should be really relevant for predictions, start with Ridge.

Putting it all together

- Regularisation: Help solve overfitting by employing a smarter approach
- Ridge: "Force push"
- Lasso: "Coerce"

ElasticNet

ElasticNet - Intuition

Linear combination of **both** penalty terms.

Important stuff ahead!

Miscellaneous Points

- Validation Set
- *GridSearch*
- Choosing λ
- Regularisation for Classification
- Data transforms

Thank you