Week 4a:

Regularisation: Learning linear models that generalise

G6061: Fundamentals of Machine Learning [23/24]

Dr. Johanna Senk

UNIVERSITY OF SUSSEX

Outline

Today we'll

- Think about relaxing some of our assumption in **learning linear regression**.
- Discuss the concept of **over-fitting** and explain why it happens and how you can spot it.
- Introduce **regularisation** as a mechanism to combat overfitting, and explain it's link to Bayes' theorem.

At the end of this session you should be able to

- Understand how weighted least squares fitting works, why it can be useful, and how to implement it in `numpy`.
- Describe the concept of overfitting, and be able to spot if your model is overfitting.
- Explain what regularisation is, and how it can be applied to reduce overfitting.

UNIVERSITY OF SUSSEX

Recap: linear least squares regression

- We defined linear regression as:
  $y \approx \hat{y} = Xw$, where y=labels, $\hat{y}$ =prediction, X=input data matrix, and w=weight vector
- We defined the *cost function* as the squared prediction error:
  $\frac{1}{2N} \|X\mathbf{w} - \mathbf{y}\|^2$
- This gives us a *probabilistic interpretation* of our predictions as a normal distribution $\mathcal{N}(y; X\mathbf{w}, \sigma^2) = \mathcal{N}(y; \hat{y}, \mathbf{I}\sigma^2)$.
- This states: errors are independent & *identically distributed* across samples
  - Is this realistic? Can you think of a case when it wouldn't be?
  - How can we relax the assumption that the distribution of model errors is the same across samples?

UNIVERSITY OF SUSSEX

Weighted linear regression

- There are often times where assigning different weights on our training data pairs i.e. (x_n, y_n) might be helpful.
- This would be because we may have more or less confidence in some data points than others
  - For instance, maybe one of the hiring managers makes erratic decisions and we wanted to downweight their opinions.
  - Alternatively, another one might always make excellent decisions and we want to model that process.
- **Warning!** Blindly assuming our models are informative can exacerbate human bias and unfairness in decision making.
  - We're not covering AI fairness and bias in this module, but I'd recommend that you check out this blog post on the topic.

UNIVERSITY OF SUSSEX

Weighted linear regression

- We can modify linear least squares regression to allow σ^2 to change across training samples.
  - This is known as heteroscedastic noise.
- Our problem is *mostly* the same, we just attribute different importance to some of the data.
- Mathematically, we can still write it in *closed form*

$$\mathbf{w}^* = (X^\top W X)^{-1} X^\top W y$$

where W is a *diagonal matrix* containing the reciprocal sample variance $W_{ii} = \frac{1}{\sigma_i^2}$

```
# Given X, y and W
w_hat = np.linalg.inv(X.T@W@X)@X.T@W@y
# We can still use the psuedoinverse
w_hat = np.linalg.pinv(np.sqrt(W)@X)@np.sqrt(W)@y
y_hat = X@w_hat # predict
```

UNIVERSITY OF SUSSEX

How do I know what σ should be?

- Depends on your problem.
  - Sometimes you know the expected precision of a sensor
  - Or how erratic a human labeller is!
- You can also learn or infer it from data, but that's a more advanced use case.

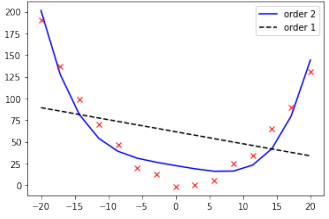
UNIVERSITY OF SUSSEX

Making linear models complicated

- Last week we saw how linear models could be extended to explain more complex relationships in the input data.

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^K \\ 1 & x_2 & x_2^2 & \dots & x_2^K \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^K \end{bmatrix}$$

So we can fit polynomials, or any set of basis functions:



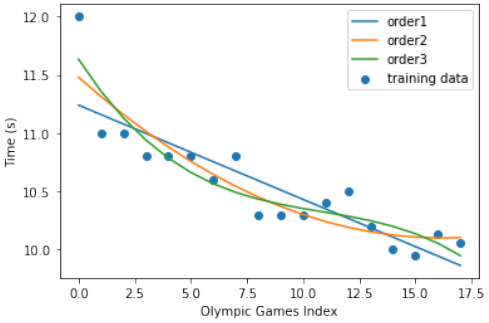
UNIVERSITY OF SUSSEX

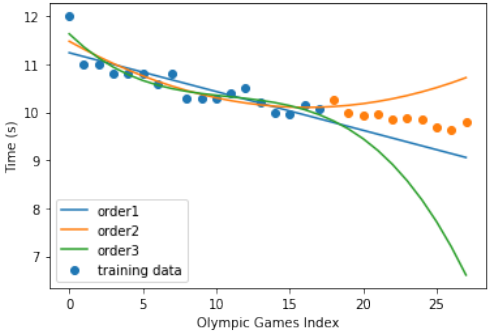
But what if our model is too complicated?

- What will it do?
- It will probably be better at describing the data you trained it on!
  - i.e. the training error will be lower.
- It will probably be worse at describing the unseen validation or test data!
  - Which means that practically speaking, it's not very useful!

UNIVERSITY OF SUSSEX

Polynomial regression



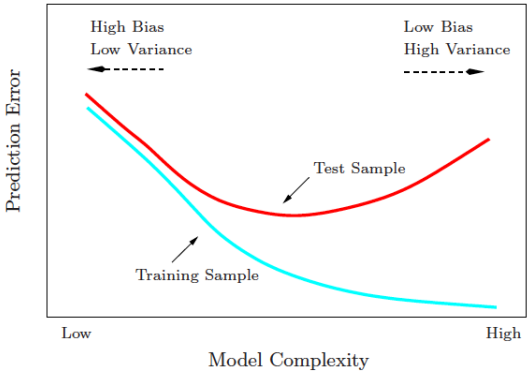


Sometimes the simplest model will be best for predicting new data.

UNIVERSITY OF SUSSEX

Training and test error as a function of model complexity

For example, the higher the degree of a polynomial, the more complex.



The Elements of Statistical Learning

UNIVERSITY OF SUSSEX

Think break

- Our linear model produces an output by

$$\hat{y} = \sum_i w_i x_i$$

- We can make the model more complicated by adding new variables to x, e.g.:

$$\hat{y} = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5$$

- What *prior knowledge* could we build into our learning to make the model less complicated?
  - How can we make a complex model act like a simple model?

UNIVERSITY OF SUSSEX

Regularisation – intuition

- Consider the 5th order polynomial model

$$\hat{f}(x; \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5$$

- Even if this model itself has 6 possible parameters to learn, we can simplify it in several ways.
- The most trivial model: $\mathbf{w} = (0, 0, 0, 0, 0, 0)$, and the model will always predict a value of zero.
- The next simplest model: set w_0 **to some non-zero value**, and the model will predict a constant w_0.
- The next next simplest model: **leave** w_0 **at its new value**, we can **set** w_1 **to some value**. The model has become more complex, it can predict a line now!

Regularisation and model complexity

- The 5th order polynomial model has more available complexity as we allow each additional parameter w_n to be given a non-zero value!
- One way to think of this, is the more non-zero weights, the more complex the model. $||\mathbf{w}||_0$
  - The L^0 norm is **not** differentiable, as it's a count, so tricky to optimise dynamically.
- Another description of complexity is: the more $\mathbf{w}$ deviates from 0, the more complex the model is.
  - This can be written as the **sum of absolute values**, $L^1 = ||\mathbf{w}||_1$.
  - But absolute value tend to be tricky to optimise, as it's not differentiable at 0.
- What might we use instead?

UNIVERSITY OF SUSSEX

Another squared term

- A squared term is helpful again.

$$||\mathbf{w}||_2^2 = \sum_{i=0}^{d} w_i^2$$

or, in a vector form,

$$\mathbf{w}^\top \mathbf{w}$$

- This gives us a description of model complexity that increases as the square of the weight values.
- Where do you think this might have come from?

UNIVERSITY OF SUSSEX

The Gaussians are back!

- Again, the squared term comes from a Gaussian/Normal distribution! which has a log-probability $\log p(\mathbf{w}|\mu, \sigma^2) \propto \frac{(\mathbf{w}-\mu)^\top(\mathbf{w}-\mu)}{\sigma^2}$
  - in our case $\mu = 0$
- But this time we can think of it slightly differently, and see this as a *prior* distribution on the values of $\mathbf{w}$.
  - This means it expresses our preferences on what values $\mathbf{w}$ should take, before we've seen any data.
  - Instead of σ^2 describing noise, it's describing the *flexibility* that we're giving our model to choose various values for $\mathbf{w}$. As our preference is for $\mathbf{w}$ to be 0, we set $\mu = 0$ for most cases.

UNIVERSITY OF SUSSEX

Maximum Likelihood

- When we discussed linear regression, we expressed it probabilistically saying our signal was corrupted with Gaussian noise.

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \mathcal{N}(\mathbf{y}|\mathbf{x}^\top \mathbf{w}, \sigma^2)$$

- This equation is the *likelihood* of observing $\mathbf{y}$ given that we've seen $\mathbf{x}$ and $\mathbf{w}$.
- When we found the best values of $\mathbf{w}$ before, we were doing Maximum Likelihood estimation!

UNIVERSITY OF SUSSEX

Maximum-A-Posteriori

- Now we're adding regularisation as a prior distribution, we're actually doing a form of *Bayesian inference!*
- Bayes' rule can be written as:

$$p(\mathbf{w}|\mathbf{x},\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x},\mathbf{w})p(\mathbf{w})$$

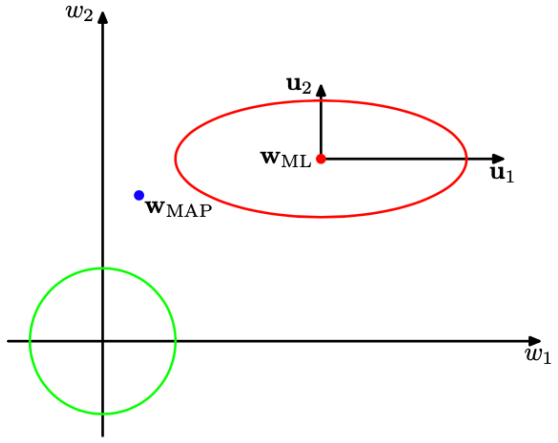
- taking logs on both sides

$$\log \underbrace{p(\mathbf{w}|\mathbf{x},\mathbf{y})}_{posterior} \propto \log \underbrace{p(\mathbf{y}|\mathbf{x},\mathbf{w})}_{likelihood} + \log \underbrace{p(\mathbf{w})}_{prior}$$

- So by *adding* the 2 log probabilities together and optimising that, we are finding the best values for $\mathbf{w}$ after seeing the prior and the data.
- This is called the *posterior* and finding the best value for $\mathbf{w}$ is called *Maximum-a-posteriori*.

UNIVERSITY OF SUSSEX

MAP vs. ML



A geometric perspective. w_{map} is between w_{ml} and the prior.

Taken from Bishop: Pattern recongition and machine learning

UNIVERSITY OF SUSSEX

Regularisation

- So rather than just minimising a loss function $\mathcal{L}(y, \hat{f}(x; \mathbf{w}))$, for example our residual error $\frac{1}{2N} \sum_{n=1}^{N} (y^n - \hat{f}(x^n; \mathbf{w}))^2$.
- We can minimise a **regularised** loss $\mathcal{L}'$ by adding together our loss $\mathcal{L}$ and a term penalising over-complexity:

$$\underset{\mathbf{w}}{\text{minimise}} \quad \mathcal{L}'(y, \hat{f}(x; \mathbf{w})) = \underset{\mathbf{w}}{\text{minimise}} \quad \left\{ \mathcal{L}(y, \hat{f}(x; \mathbf{w})) + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w} \right\}$$

  The **free (hyper-)** parameter $\lambda = \frac{1}{\sigma^2}$ controls the trade-off between penalising not fitting the data well ($\mathcal{L}$) and penalising overly complex models ($\mathbf{w}^\top \mathbf{w}$)
- Now we have
  - Regularised linear regression: $\mathcal{L}' = \mathcal{L}_{\text{squared loss}} + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w}$
  - Regularised logistic regression: $\mathcal{L}' = \mathcal{L}_{\text{logistic loss}} + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w}$
  - Regularised . . .: $\mathcal{L}' = \mathcal{L}_{\ldots} + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w}$

Regularised linear regression

$$\mathcal{L}' = \underbrace{\frac{1}{2N}\langle X\mathbf{w}-\mathbf{y}, X\mathbf{w}-\mathbf{y}\rangle}_{\text{residual error (see linear regression ecture)}} + \frac{\lambda}{2}\underbrace{\mathbf{w}^\top\mathbf{w}}_{\text{penalty term}}$$

Taking partial derivatives of $\mathcal{L}'$ with respect to $\mathbf{w}$ and set it to zero

$$\nabla_{\mathbf{w}}\mathcal{L}' = \frac{1}{N}X^\top(X\mathbf{w}-\mathbf{y}) + \lambda\mathbf{w} = 0$$

The regularised linear regression solution is

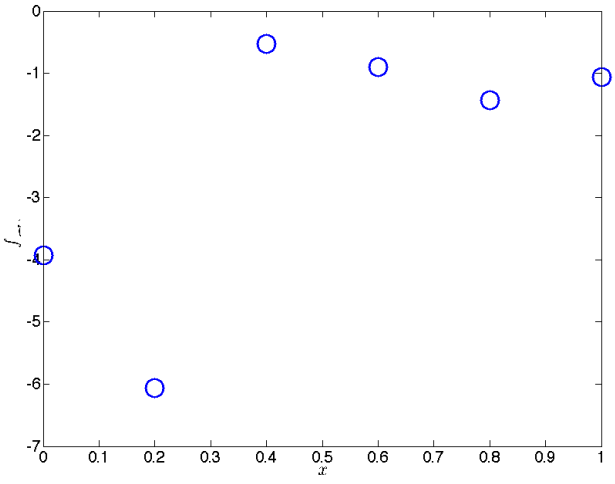
$$\mathbf{w}_{\text{regularised}} = (X^\top X + N\lambda I)^{-1}X^\top\mathbf{y} \quad \text{where } I = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & & \ddots & \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}\Bigg\}$$

To be contrasted to our **non**-regularised version of linear regression

$$\mathbf{w}_{\text{non-regularised}} = (X^\top X)^{-1}X^\top\mathbf{y}$$

Regularisation in action

Our data with just 6 data points: $(0, -4), (0.2, -6), \ldots$



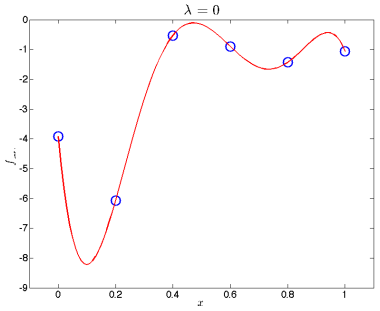
UNIVERSITY OF SUSSEX

Regularisation in action

- We use the 5th order polynomial model

$$\hat{f}(x; \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5$$

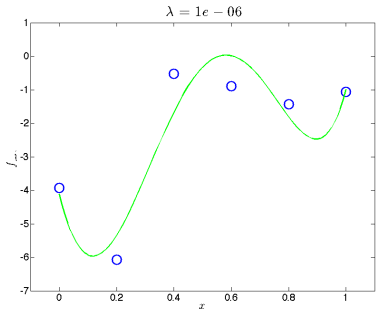
- We set $\lambda = 0 \rightarrow$ we recover the **non**-regularised version of linear regression



$$\underset{\mathbf{w}}{\text{minimise}} \quad \left\{ \mathcal{L}(y, \hat{f}(x; \mathbf{w})) + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w} \right\}$$

Regularisation in action

- We set $\lambda = 1e-06 \rightarrow$ the model follows the general shape of the exact 5th order polynomial but without as much variability and is further away from the data points.

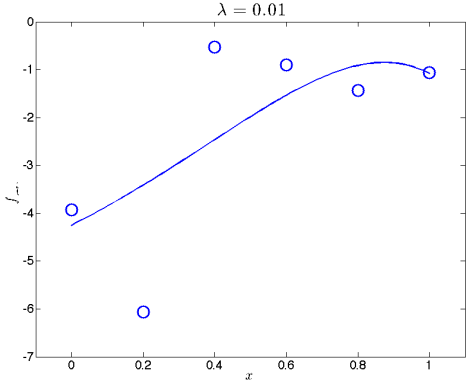


$$\underset{\mathbf{w}}{\text{minimise}} \quad \left\{ \mathcal{L}(y, \hat{f}(x; \mathbf{w})) + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w} \right\}$$

UNIVERSITY OF SUSSEX

Regularisation in action

- We set $\lambda = 0.01 \rightarrow$ the model becomes less complex.

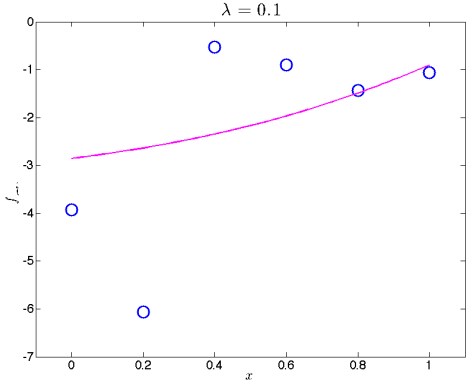


$$\underset{\mathbf{w}}{\text{minimise}} \quad \left\{ \mathcal{L}(y, \hat{f}(x; \mathbf{w})) + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w} \right\}$$

UNIVERSITY OF SUSSEX

Regularisation in action

- We set $\lambda = 0.1 \rightarrow$ the model becomes even less complex.



$$\underset{\mathbf{w}}{\text{minimise}} \quad \left\{ \mathcal{L}(y, \hat{f}(x; \mathbf{w})) + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w} \right\}$$

UNIVERSITY OF SUSSEX

Regularisation in summary

$$\underset{\mathbf{w}}{\text{minimise}} \quad \left\{ \mathcal{L}(y, \hat{f}(x; \mathbf{w})) + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w} \right\}$$

- **Larger** λ, higher regularisation:
  too large, we will not capture any useful trends in the data

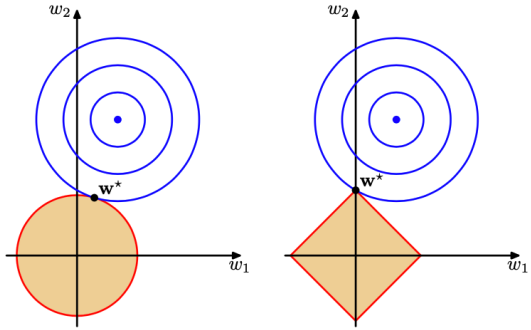
- **Smaller** λ, lower regularisation:
  too small, our function will likely be too complex

More regularization tends to cause less overfitting.

UNIVERSITY OF SUSSEX

Other forms of regularisation

- LASSO (Least Absolute Shrinkage and Selection Operator) regularisation places a penalty on the absolute values of the weights $||\mathbf{w}||_1 = \sum_i |\mathbf{w}_i|$.



- Elastic net combines ridge regression and LASSO: $\lambda_1||\mathbf{w}||_1 + \lambda_2||\mathbf{w}||_2$
  - you need to balance the two terms though.

Taken from Bishop: Pattern recongition and machine learning

UNIVERSITY OF SUSSEX

Regularised linear models in sklearn

- Several options available in the sklearn.linear_models.
- Please investigate the documentation.
- Some regularisation types may use alternative optimisation approaches, and be much slower.

UNIVERSITY OF SUSSEX

Summary and outlook

- We've talked about how to incoporate how much we trust particular data samples in fitting linear regression.
- We've also introduced the concept of over-fitting, which we want to avoid.
- Finally, we covered *regularisation*, which enables us to encourage our models to be simpler.

Next lecture:

- We're still missing how to choose our hyper-parameters λ!
- Next time we'll cover Model selection: how to choose the model and it's hyper-parameters from data.

UNIVERSITY OF SUSSEX