When learning fails: underfitting, overfitting

Why can learning fail?

- Distribution change
- Underfitting
- Overfitting

Training data

- We train the system with **training data**.

Sqft	Price
3000	$400,000
2000	$300,000
1500	$250,000

- What is the simplest way to return the correct y value?

Test data

- We are not interested in the values for the training data, we want predictions for the values we don't already know!

Sqft	Price
3200	?
1900	?
1450	?

- We want our model to **generalize** to data that it didn't previously seen!

Generalization

- Is this even possible?
- It is a deep theoretical question, beyond the scope of this class but...
- If the training data and test data are completely unrelated, learning cannot work.
- The simplest assumption we can make is that the **test data is drawn from the same probability distribution as the training data**.

Distribution drift

Sqft	Price
3200	$3,200,000
1900	$2,400,000
1450	$1,200,000

- What just happened?
- The training data was for house prices in Orlando in 2005, the test data was for San Francisco in 2020. They are not the same distribution!

Distribution drift

- The distribution shifted between the training and testing
- Some reasons:
  - We apply it to a new model (different geographic area)
  - Changes in time
  - Changes in the underlying problem
- Trying to predict COVID cases was a notorious case where none of the predictions ever worked, as the problem kept shifting!
- Also known as distribution shift, data drift, covariate shift etc.

Distribution drift

- Major problem, no obvious solution.
- We should at least be aware of it, and detect it.
  - Then we know that we cannot trust our predictions
- Sometimes, if we understand the nature of the shift we can compensate for it in the output (eg. inflation, or price variations due to local taxes)
- Sometimes, we can retrain our model with relatively small amount of new data.

Underfitting

- Draw figure here.

Underfitting

- If the hypothesis function family cannot represent the patterns of the data, we are **underfitting**
- Even if we find the minimum of the loss function within the given family, we still don't have a good prediction.
- Solution: find a **more expressive** hypothesis.

Overfitting

- Draw figure here (training data + test data)

Overfitting

- What happened?
- The loss is very low on the training data, but high on test data.
- This is a deep theoretical problem…
  - The training data is where you get your information from…
  - The loss function is what describes what you want…
  - The test data is unknown at training time…
  - So, how do you even approach this problem?

Some ways to avoid overfitting

- "Your model is too expressive"
  - Solution: use "less expressive" models are less likely to overfit.
    - Rule of thumb: number of parameters $<<$ number of training data point
  - Historically, this was a dominant view...
  - But these days, we are using neural networks that are **very** expressive
- "Your data is too noisy"
  - Some of the features in the training data are very noisy - the model overfit on them, so what you learn is the noise, not the underlying pattern
  - Solution: manually select which features to keep, do not include noisy ones in training

Some ways to avoid overfitting

- "Regularization"
  - Change the loss function by adding one of more **regularization terms**
  - These express preferences about the parameters we are looking for
  - Keep all the features, keep the expressivity
  - But you will need to settle that the part of the loss function describing the error might not be optimized

Example: regularization for linear regression

- Unregularized least squares

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left(y - (\theta_n x_n^{(i)} + \ldots \theta_0 x_0^{(i)})\right)^2$$

- Regularized least squares

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left(y - (\theta_n x_n^{(i)} + \ldots \theta_0 x_0^{(i)})\right)^2 + \lambda\sum_{i=1}^{n}\theta_i^2$$

What does regularization do?

- It penalizes the models that have a θ_i component to have a magnitude that is too big (either in positive or negative direction)
  - For instance, it prevents the model focusing on only one feature
- As this is square of length, it is called L_2-regularization
  - It has a set of other fancy names: ridge regression, weight decay, Tikhonov regularization,
- Other regularization techniques are possible: L_1-regularization (with the absolute values), combinations etc.
- Does it break stochastic gradient descent?
  - NO! It is differentiable in $\boldsymbol{\theta}$
- DRAW FIGURE
- Secret insight: regularization usually expresses some kind of **prior knowledge** about the model