Machine Learning

| Regularization

Prof. Dr. Mostafa Elhosseini

https://youtube.com/@ProfElhosseiniTechVerse

Agenda

- Parameter Sign & Magnitude: Impact on Label
- Overfitting
- Overfitting: Lots of features and little data
- Causes of Overfitting
- Addressing Overfitting
- Regularization
  - Effect on Model
  - Movie recommendations
  - Measuring how complex a model is?
  - L1 norm
  - L2 norm

Impact of Parameter Values on Labels

In a linear model:

- Target = $\boldsymbol{w_1} \times \text{Feature}_1 + \boldsymbol{w_2} \times \text{Feature}_2 + \ldots + \boldsymbol{w_n} \times \text{Feature}_n + b$
- **Increase in weight** amplifies the **positive** correlation; higher feature values lead to higher target values.
- **Negative Weights**: Increase in weight strengthens the negative correlation; higher feature values lead to lower target values.
- The model becomes **more responsive** to slight changes in the feature's value.
- Zero Weights: No impact on the target regardless of the feature value.

Impact of Parameter Values on Labels

- Price = **30**(number of rooms) + **1.5**(size) + **10**(quality of the schools) **–** **2**(age of the house) + **50**

- Why are all of the weights positive, except for the one corresponding to the age of the house?
- The reason is the other three features (number of rooms, size, and quality of the schools) are **positively correlated** to the price of the house

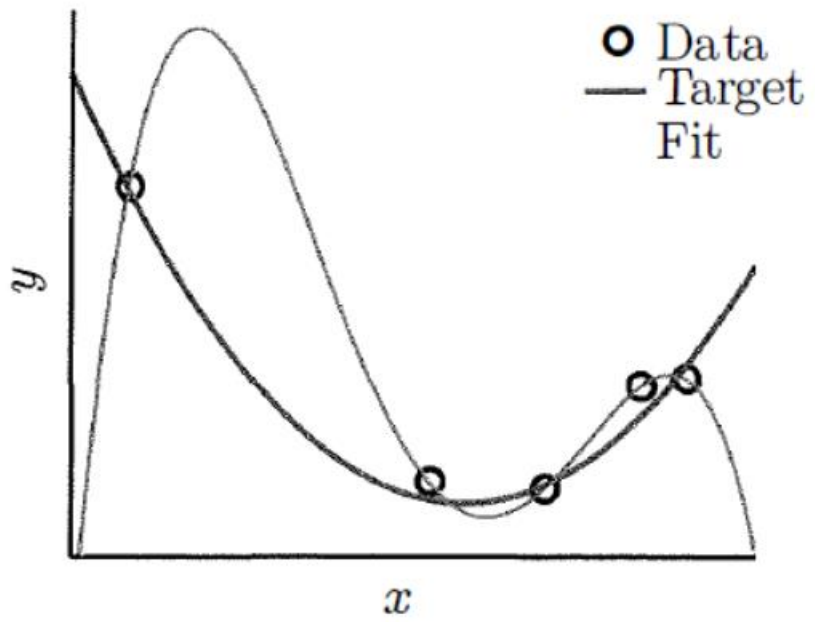
Impact of Parameter Values on Labels

- Price = 30(number of rooms) + 1.5(size) + 10(quality of the schools) – **2**(age of the house) + 50

- **Older** houses tend to be **less expensive**, the **age** feature is **negatively correlated** to the price of the house
- What if the **weight** of a feature is **zero**? This happens when a feature is **irrelevant** to the price
- In a similar way, if a feature has a very **high weight** (whether negative or positive), we interpret this as the model telling us that that feature is **important** in determining the price of the house.

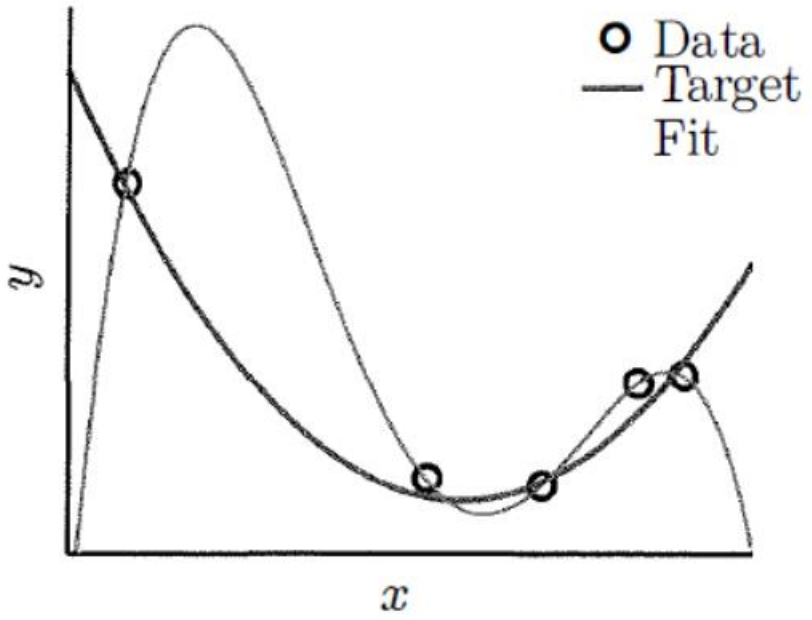
Overfitting:

- **Overfitting** occurs when a model learns the training data too well, capturing noise or random fluctuations in the data rather than the underlying patterns
  - Model performs well on the training data, but it does not generalize well



Overfitting:

- Complex model uses its additional degrees of freedom to 'learn' the noise
- The little noise in the data has misled the learning
- If you have lots of features and little data – **overfitting** can be a problem



Causes of Overfitting

- Too Complex Model: Models with too many parameters relative to the amount of training data are prone to overfitting.
- Insufficient Data: When the training dataset is small, the model may find it easier to memorize the data rather than generalize from it.
- Noise in Data: If the training data contains noise or irrelevant patterns, the model may learn these as if they are genuine features.

Addressing Overfitting

- Simplifying the Model
- **Cross-Validation**: Use techniques like cross-validation to assess model performance on different subsets of the data, helping to identify overfitting.
- More Data
- **Feature Selection**: Choose relevant features and eliminate irrelevant ones to reduce the risk of overfitting.
- Learning Curves
- **Regularization**: Apply regularization techniques to penalize overly complex models, discouraging them from fitting the noise.

Regularization

- **Regularization** is a technique used in machine learning to **prevent overfitting** and **improve model generalization**.

- We don’t need to train several models. We simply train the model once, but **during the training**, we try to not only improve the model’s **performance** but also reduce its **complexity**

- **Adds a penalty to the loss function** used to train the model. This penalty discourages overly complex models by **increasing the cost** of having **large weights** (coefficients).

Regularization

- simple way to **regularize** a polynomial model is to reduce the number of **polynomial degrees**
- The core idea behind regularization is to prevent overfitting by **adding a penalty** for complexity to the model's loss function
- Improve generalization to new, unseen data

Effect on Model

- **Reduces overfitting** by making the model less sensitive to the fluctuations in the training data.

- Helps in **selecting features** by **shrinking** coefficients, making the model simpler and more **interpretable** (especially with **L1 regularization**).

- Widespread Use: Used in various machine learning models, including linear regression, logistic regression, and neural networks, to maintain robustness and prevent overfitting.

Example - Movie recommendations

- Imagine that we have a **movie streaming website**, and we are trying to build a **recommender system**.
- For simplicity, imagine that we have only **10 movies**: M1, M2, …, M10. A new movie, **M11, comes out**, and we'd like to build a **linear regression** model to **recommend movie 11** based on the **previous 10**.

- We have a **dataset of 100 users**. For each user, we have **10 features**, which are the times (in seconds) that the user has watched each of the original 10 movies

Example - Movie recommendations

- A **movie streaming website**,
- Only **10 movies**: M1, M2, …, M10.
- We'd like to build a **linear regression** model to **recommend movie 11** based on the **previous 10**.
- We have a **dataset of 100 users**.
- For each user, we have **10 features**, which are the times (in seconds) that the user has watched each of the original 10 movies
- The **label** for each user is the **amount of time** the user watched movie **11**

Example - Movie recommendations

- Given that the model is a **linear regression** model, the equation

- $\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + w_7x_7 + w_8x_8 + w_9x_9 + w_{10}x_{10} + b$

Example - Movie recommendations

- Given that the model is a **linear regression** model, the equation
- $\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + w_7x_7 + w_8x_8 + w_9x_9 + w_{10}x_{10} + b$

- **Model 1**: $\hat{y} = 2x_3 + 1.4x_7 - 0.5x_9 + 4$
- **Model 2**: $\hat{y} = 22x_1 - 103x_2 - 14x_3 + 109x_4 - 93x_5 + 203x_6 + 87x_7 - 55x_8 + 378x_9 - 25x_{10} + 8$

- **Model 2** seems a bit **complicated**, and it may be the **one overfitting**

Example - Movie recommendations

- **Model 2**: $\hat{y} = 22x_1 - 103x_2 - 14x_3 + 109x_4 - 93x_5 + 203x_6 + 87x_7 - 55x_8 + 378x_9 - 25x_{10} + 8$

- Model 2 seems a bit complicated, and it may be the one overfitting. It's unlikely that the time that a user watched **movie 2** needs to be **multiplied by –103** and then added to other numbers to obtain the prediction. This may **fit the data well**, but it **definitely** looks like it's **memorizing** the data instead of learning it.

Example - Movie recommendations

- Model 1: $\hat{y} = 2x_3 + 1.4x_7 - 0.5x_9 + 4$

- **Model 1**, in contrast, **looks much simpler**, and it gives us some interesting information.
- From the fact that **most of the coefficients are zero**, except those for movies 3, 7, and 9, it tells us that the only **three movies** that are **related to movie 11** are those three movies.

- Furthermore, from the fact that the **coefficients of movies 3 and 7** are **positive**, the model tells us that **if a user watched movie 3 or movie 7, then they are likely to watch movie 11**. Because the coefficient of **movie 9 is negative**, then if the **user watched movie 9**, they **are not likely to watch movie 11**

Example - Movie recommendations

- Our goal is to **have a model like model 1** and to avoid models like model 2.

- But unfortunately, if **model 2 produces a smaller error** than model 2, then running the linear regression algorithm will select model 2 instead.
- What can we do? Here is where **regularization** comes to the rescue.
- The first thing we **need is a measure that tells us that model 2 is much more complex** than model 1.

Measuring how complex a model is?

- let's look at models 1 and 2 from the previous section and try to come up with some formula that is **low for model 1 and high for model 2**.

- Notice that a model with **more coefficients**, or coefficients of **higher value**, tends to be **more complex**. Therefore, any formula that matches this will work, such as the following:

  - The sum of the absolute values of the coefficients
  - The sum of the squares of the coefficients

Measuring how complex a model is?

- L1 norm: The sum of the absolute values of the coefficients
- L2 norm: The sum of the squares of the coefficients
- We use **absolute values and squares** to **get rid** of the negative coefficients; otherwise, large negative numbers will cancel out with large positive numbers, and we may end up with a small value for a very complex model.
- The **bias in the models is not included** in the L1 and L2 norm. Why? Well, the bias in the model is precisely the number of seconds that we expect a user to watch movie 11 if they haven't watched any of the previous 10 movies. This number **is not associated with the complexity of the model**; therefore, we leave it alone

Measuring how complex a model is?

- Model 1: $\hat{y} = 2x_3 + 1.4x_7 - 0.5x_9 + 4$
- Model 2: $\hat{y} = 22x_1 - 103x_2 - 14x_3 + 109x_4 - 93x_5 + 203x_6 + 87x_7 - 55x_8 + 378x_9 - 25x_{10} + 8$

L1 norm:

- Model 1: |2| + |1.4| + |–0.5| = 3.9
- **Model 2**: |22| + |–103| + |–14| + |109| + |–93| + |203| + |87| + |–55| + |378| + |–25| = 1,089

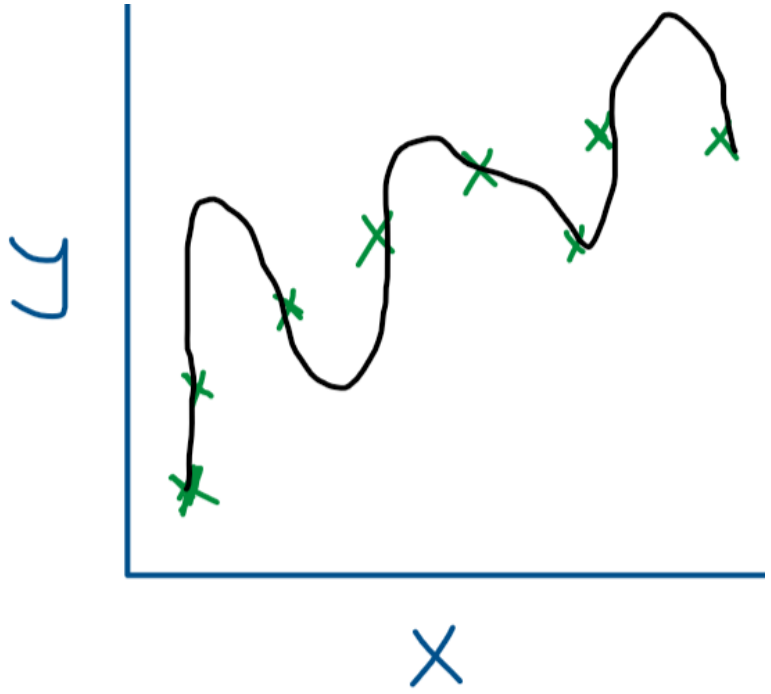
Recall: cost function

$$J = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(X^{(i)}) - y^{(i)}\right)^2$$

- You parameter can be updated in any way, just to lower the MSE value, and take care…
  - The larger your parameters become, the higher the chance your model overfit the data

How to regularize?

- Penalize and make some of the θ parameters really small
  - Like θ_3, θ_4
  - $\boldsymbol{J = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(X^{(i)}) - y^{(i)}\right)^2 + 1000\,\theta_3{}^2 + 1000\,\theta_4{}^2}$
  - The only way to minimize this function is to make $\boldsymbol{\theta_3, \theta_4}$ very small
  - So here we end up with $\boldsymbol{\theta_3}$ and $\boldsymbol{\theta_4}$ being close to zero
  - Now we approximately have a quadratic equation
  - $y = \theta_0 + \theta_1 x + \theta_2 x^2$



$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

How to regularize?

- Smaller values for parameters corresponds to a simpler hypothesis
  - You get rid of some of the terms
- A simpler hypothesis is less prone to overfitting, but…
- we do not know what are the high order terms
  - How do we choose the ones to shrink?
- It is better to **shrink all parameters**

Regularization Types

- For a linear model, regularization is typically achieved by constraining the weights of the model.
  - Ridge Regression
  - Lasso Regression
  - Elastic Net,

Regularization Types

- **L1** Regularization (Lasso): Adds a penalty equivalent to the **absolute value** of the magnitude of the coefficients. Encourages sparsity, which can **set some coefficients to zero**.

- L2 Regularization (Ridge): Adds a penalty equivalent to the **square of the magnitude** of the coefficients. This discourages large coefficients but does not set them to zero.

L1 Regularization (Lasso)

- Next Lecture