EEP 596: Adv Intro ML || Lecture 3

Dr. Karthik Mohan

Univ. of Washington, Seattle

January 10, 2023

Logistics

- Programming assignment due Saturday

Logistics

- Programming assignment due Saturday
- Lecture notes posted to the webpage

Logistics

- Programming assignment due Saturday
- Lecture notes posted to the webpage
- Conceptual 1 assigned tonight - Due next Tuesday. Refer to the lecture notes for relevant background for conceptual assignment.

Logistics

- Programming assignment due Saturday
- Lecture notes posted to the webpage
- Conceptual 1 assigned tonight - Due next Tuesday. Refer to the lecture notes for relevant background for conceptual assignment.
- Lightning Presentations - Starting from next week - Present for 5 minutes in groups of 2 at the end of Thursday's class on a topic covered so far. Can sign up for it here - This is only done once per team for the quarter.

Logistics

- Programming assignment due Saturday
- Lecture notes posted to the webpage
- Conceptual 1 assigned tonight - Due next Tuesday. Refer to the lecture notes for relevant background for conceptual assignment.
- Lightning Presentations - Starting from next week - Present for 5 minutes in groups of 2 at the end of Thursday's class on a topic covered so far. Can sign up for it here - This is only done once per team for the quarter.
- Pick a calendly slot for 1:1 ML brainstorm on any topic

Logistics

- Programming assignment due Saturday
- Lecture notes posted to the webpage
- Conceptual 1 assigned tonight - Due next Tuesday. Refer to the lecture notes for relevant background for conceptual assignment.
- Lightning Presentations - Starting from next week - Present for 5 minutes in groups of 2 at the end of Thursday's class on a topic covered so far. Can sign up for it here - This is only done once per team for the quarter.
- Pick a calendly slot for 1:1 ML brainstorm on any topic
- Any questions/thoughts?

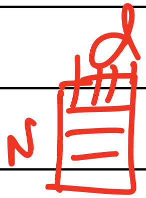
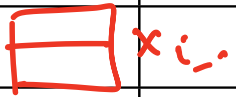
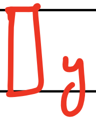
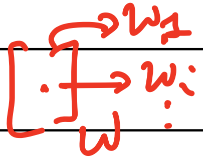
Today's class!

- Recap of Linear Regression
- Data pre-Processing
- Data Normalization
- Data Splits
- Training vs Testing
- Over-fitting & Regularization

Linear Regression Notation Recap

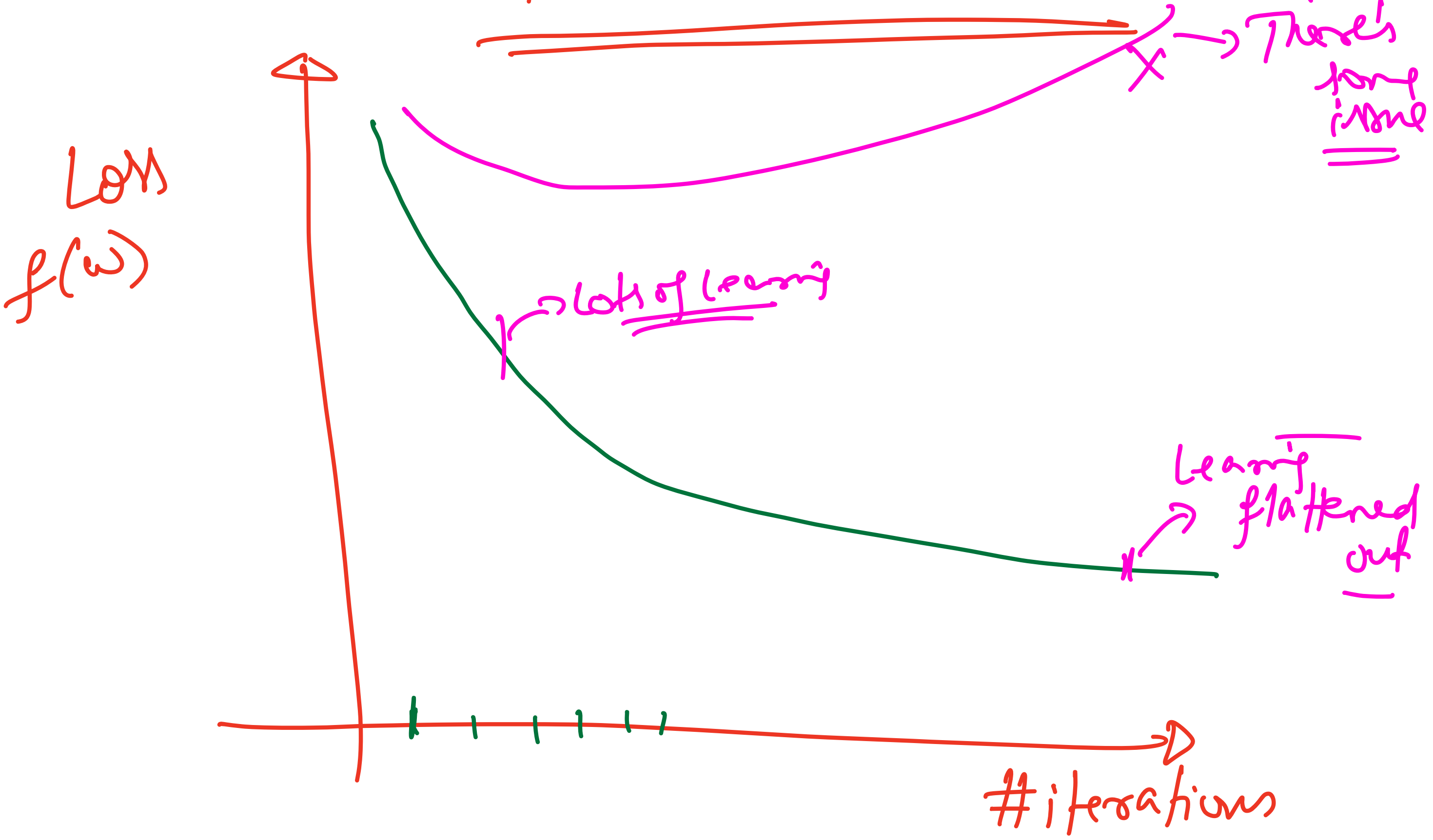
→ Euclidean Norm

$\|w\|_2 = \sqrt{w_1^2 + \dots + w_d^2}$ $\|w\|_2^2 = w_1^2 + w_2^2 + \dots + w_d^2$

Notation	Meaning
N	Number of examples (data points)
d	Feature dimension
$X \in \mathcal{R}^{N\times d}$ 	Data Matrix
$X_{i,.}$ 	*ith* row of X
$y \in \mathcal{R}^N$ 	Target vector
$w \in \mathcal{R}^d$	weight vector to learn
w_i 	*ith* element of vector w
$\hat{w} \in \mathcal{R}^d$	Learned weight vector
$\hat{y} \in \mathcal{R}^N$	Prediction vector
$\boxed{\hat{y} = X\hat{w}}$ $\hat{y}_i = X_{i,\cdot}\hat{w}$	Obtaining predictions
$f(w) = \frac{1}{2N}\|Xw - y\|_2^2$	Objective (Loss) Function
$\nabla_w f(w) = X^T(Xw - y)$	Gradient

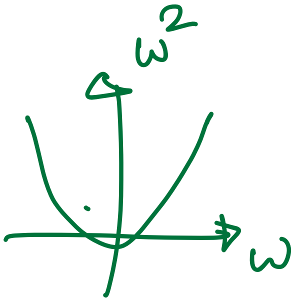
$$f(w) = \frac{1}{2N}\left[(X_{1,\cdot}\cdot w - y_1)^2 + (X_{2,\cdot}\cdot w - y_2)^2 + \dots + (X_{N,\cdot}\cdot w - y_N)^2\right]$$

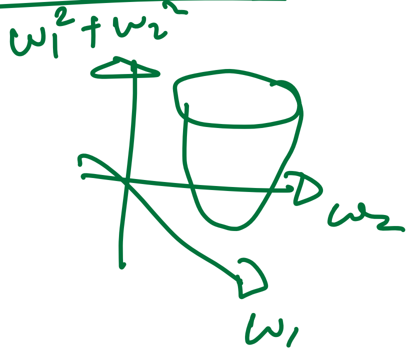
Train a ML Model



$$f(w) = \frac{1}{2} \|\overbrace{\underbrace{Xw - w}_{}}^{}\|_2^2$$

Quadratic Loss function





$$\|w\|_2^2 = \underline{w_1^2 + w_2^2 + \cdots + w_d^2}$$

Linear Regression Weights

features Target

Data pt. (x_i, y_i)

Definition

Find the best weights/parameters/coefficients w such that $X_{i,\cdot}^T w$ is as close to y_i as possible!

y_i

Linear Regression Weights

Definition

Find the best weights/parameters/coefficients w such that $X_{i,\cdot}^T w$ is as close to y_i as possible!

Mathematically

Minimize the following expression:

$$\min_w \|\underline{Xw - y}\|_2^2$$

Example of Weights for Linear Regression in the Housing Prices Data set

a) 2,2 b) 1,3 c) 3,1 d) 3,2

ICE #1

You trained a ML model with some assumptions on which attributes be numeric and which ones are categorical and obtained the following model:

$$\hat{y} = w_0 + w_1 \times \text{Sq. }\underline{\text{Footage}} + w_1 \times \underline{\text{Bedrooms}} + w_2 \times \underline{\text{Bathrooms}}+$$

$$w_{\underline{Seattle}} x_{\underline{Seattle}} + w_{\underline{Bothell}\,Bothell} + w_{\underline{Bellevue}} x_{Bellevue} + w_{\underline{Sammamish}} x_{Sammamish}$$

where

$\underline{w_0 =} 600k$, $\underline{w_1 = 1}00k$, $\underline{w_2 =} 50k$,
$w_{Seattle} = 200k, w_{Bothell} = 50k, w_{Bellevue} = 100k, w_{Sammamish} = 100k.$
How many categorical features and how many numerical features does your model have?

Example of Weights for Linear Regression in the Housing Prices Data set

Linear Model

Categorical feature

X Numerical

How could this model be improved?

You trained a ML model with some assumptions on which attributes be numeric and which ones are categorical and obtained the following model:

$$\hat{y} = \underbrace{w_0}_{\text{bias}} + w_1 \times \text{Sq. Footage} + \overset{w_2}{w_1} \times \text{Bedrooms} + \overset{w_3}{w_2} \times \text{Bathrooms} +$$

$$w_{Seattle} x_{Seattle} + w_{Bothell\,Bothell} + w_{Bellevue} x_{Bellevue} + w_{Sammamish} x_{Sammamish}$$

where

$w_1 \rightarrow$ much smaller than w_2 or w_3

$$w_0 = 600k,\ \overset{}{w}_{\underset{2}{1}} = 100k,\ w_{\underset{3}{2}} = 50k,$$

$$w_{Seattle} = 200k, w_{Bothell} = 50k, w_{Bellevue} = 100k, w_{Sammamish} = 100k.$$

Data pre-processing

Raw Data vs Pre-Processed Data

Raw data is something you download from a webpage as a "data set".
Pre-Processed data is where a sequence of transformations are applied to the data to get it into shape before its ready to go through a ML model!

Data pre-processing

Raw Data vs Pre-Processed Data

Raw data is something you download from a webpage as a “data set”. **Pre-Processed data** is where a sequence of transformations are applied to the data to get it into shape before its ready to go through a ML model!

Types of pre-processing

One is taking care of categorical variables such as location with dummy attributes (also called ‘bag of words’ model). Anything else we may need to do on the data to get good predictions?

Categorical Transformation → One Hot Encoding [0 0 0 1 0 ... 0]

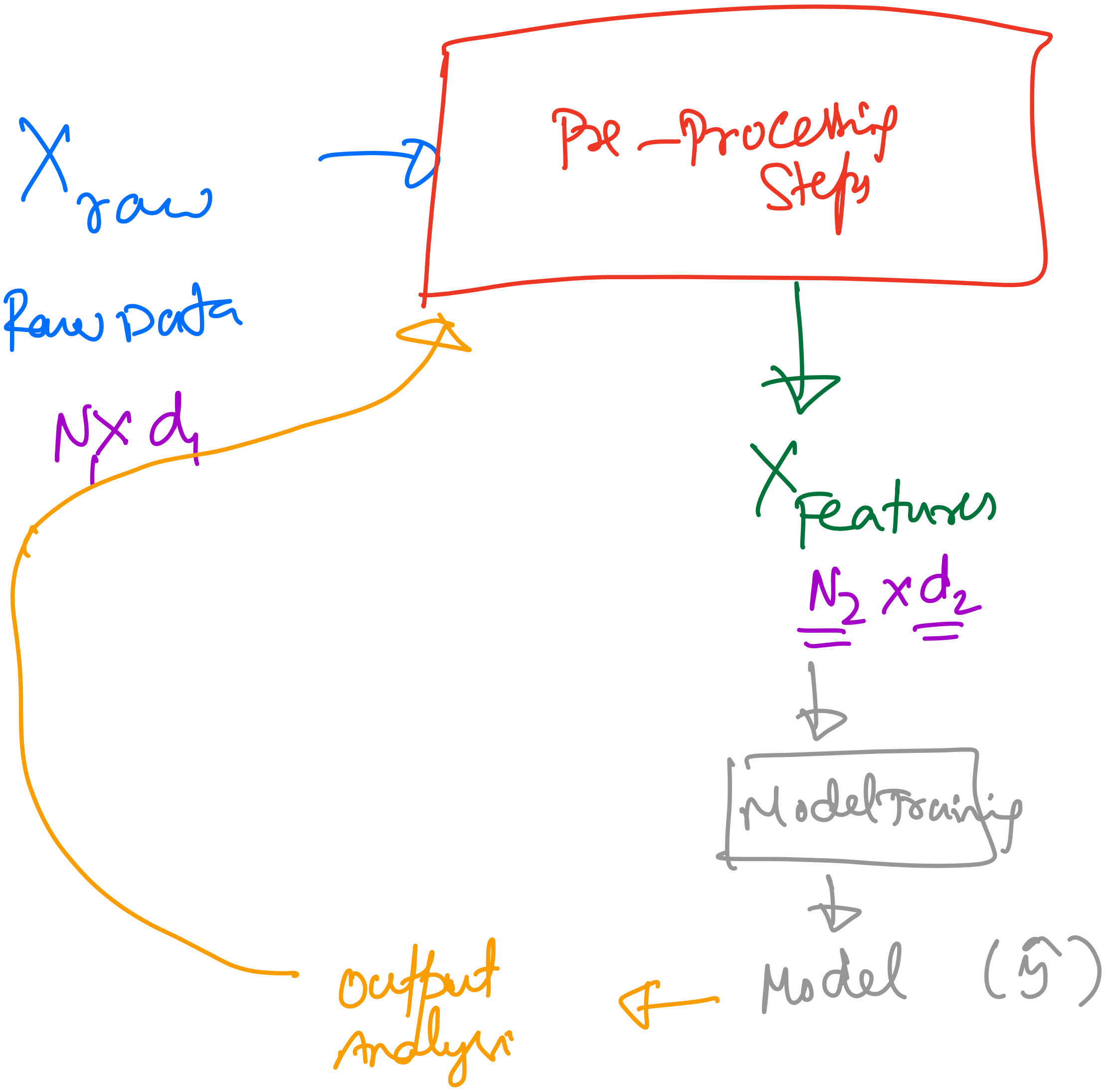
Categorical Transformation → Bag of words [0 1 0 .. 0 5 0 ... 0]

Pre-Processing

1) Cat vs Numeric Features

2) Data Imputation in case of missing values

3) Noisy Data $\begin{bmatrix}2\\2\\\vdots\\2\end{bmatrix}$



Training the Linear Regression Model

x feature

y

x_1	x_2	x_3	x_4	x_5	x_6	x_7	y

Can we use of all of data for training?

- Why not use all data for training ? ✗

Example: 70 : 10 : 20 Train-Val-Test data split

Choose 70% train data at random

x_1	x_2	x_3	x_4	x_5	x_6	x_7	y
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Example: 70 : 10 : 20 Train-Val-Test data split

Add 20% test data at random

x_1	x_2	x_3	x_4	x_5	x_6	x_7	y
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

2/10

Example: 70 : 10 : 20 Train-Val-Test data split

Remainder becomes validation data

x_1	x_2	x_3	x_4	x_5	x_6	x_7	y
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

→ validation

Data Splits for Machine Learning

→ on Train DataSet (70% or 80%)

- **Training** Learning parameters/weights, i.e. w for Linear Regression is called Training.

Data Splits for Machine Learning

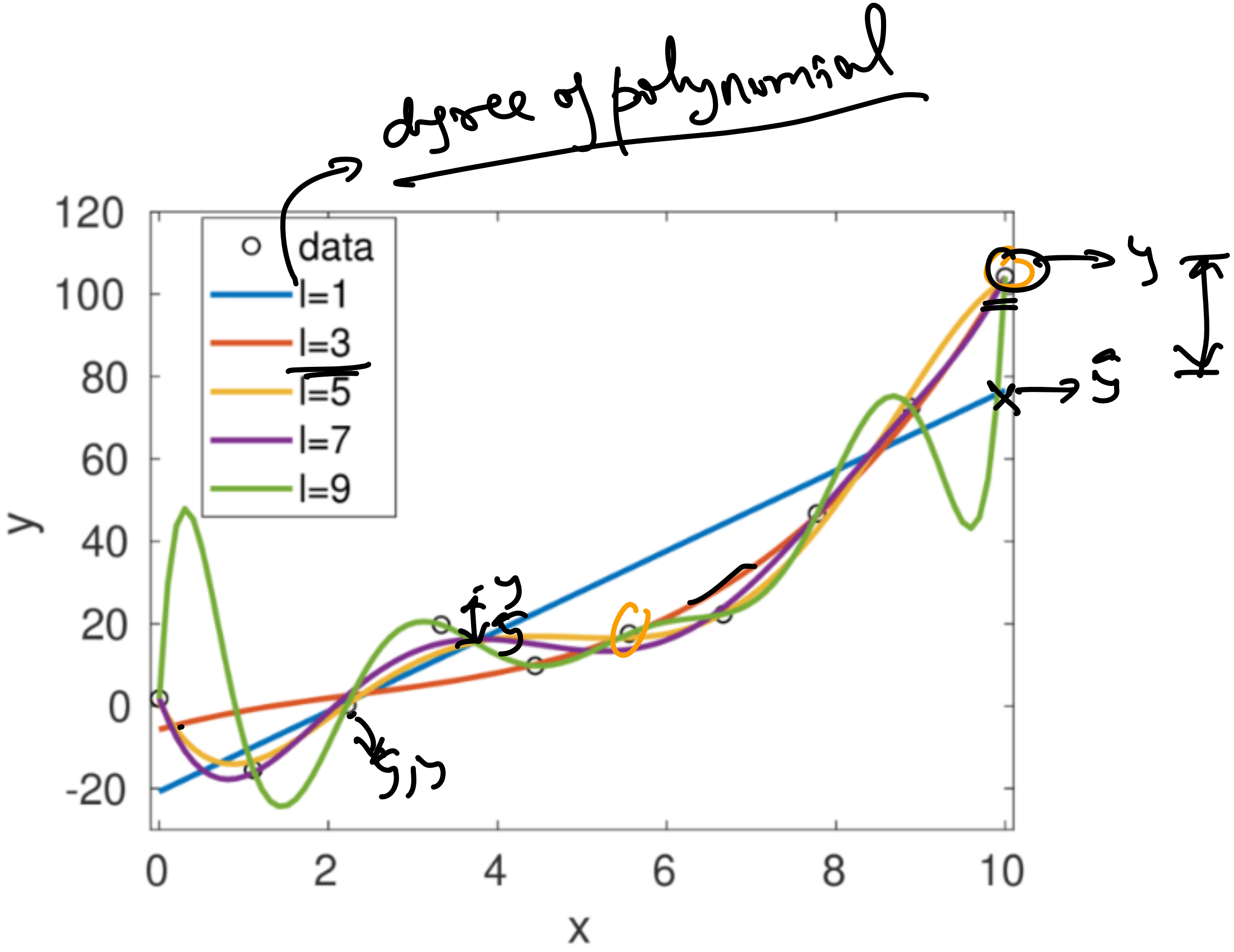
- **Training** Learning parameters/weights, i.e. w for Linear Regression is called Training.
- We don't use all data for training - Some portion of the data is kept for validation and testing.

Data Splits for Machine Learning

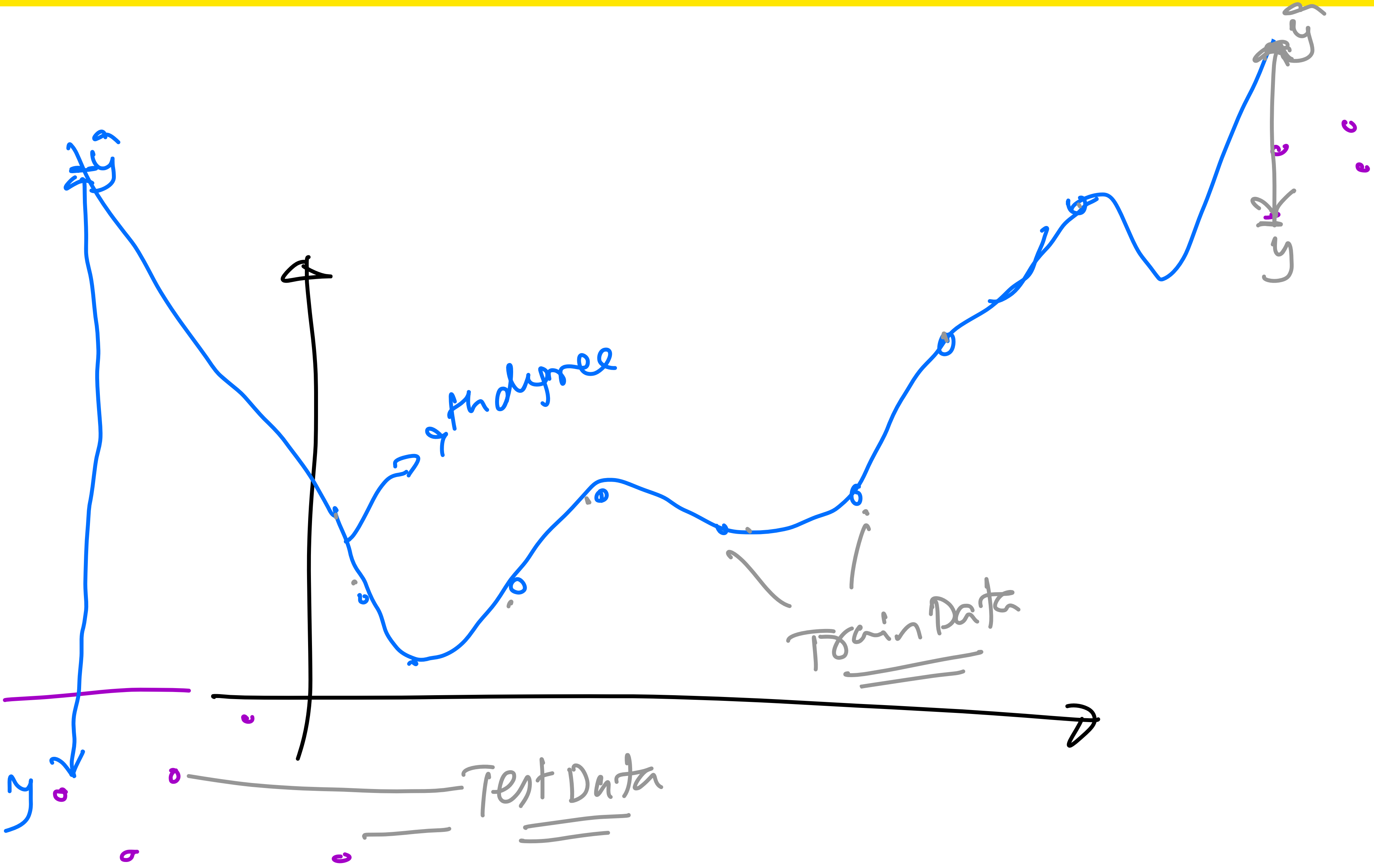
- **Training** Learning parameters/weights, i.e. w for Linear Regression is called Training.
- We don't use all data for training - Some portion of the data is kept for validation and testing.
- **Data Splits:** Usually, 80% of data is kept for training, 10% for validation and 10% for training. The splits are chosen randomly.

Let's understand the need to split data

Linear Regression → Polynomial Regression



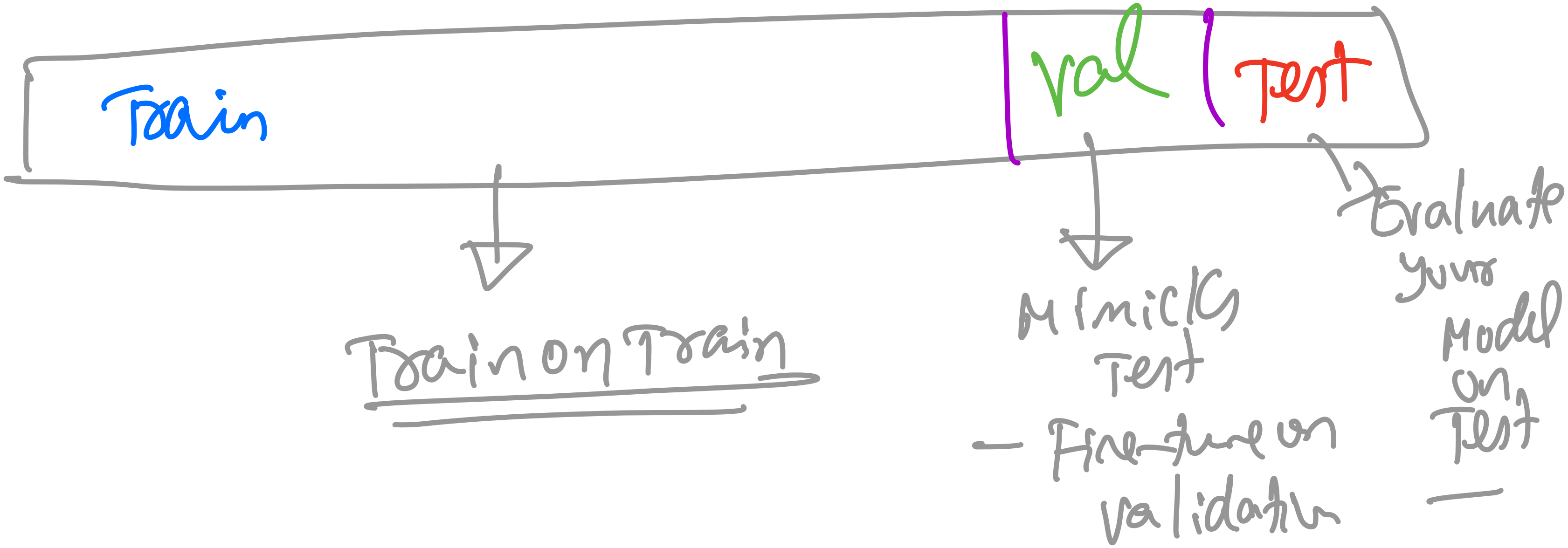
What can go wrong with polynomial regression?



The phenomenon of Overfitting

Overfitting

Overfitting is when your model performs great on training data but doesn't match up on test data. To account for overfitting, we also have a validation data set.



The phenomenon of Overfitting

Overfitting

Overfitting is when your model performs great on training data but doesn't match up on test data. To account for overfitting, we also have a validation data set.

When do we expect over-fitting? (Thumb Rule)

When the number of attributes in our model exceeds the size of the data set.

features > # data pts

⟹ might overfit!

The phenomenon of Overfitting

Overfitting

Overfitting is when your model performs great on training data but doesn't match up on test data. To account for overfitting, we also have a validation data set.

When do we expect over-fitting?

When the number of attributes in our model exceeds the size of the data set.

Overfitting in Linear Regression

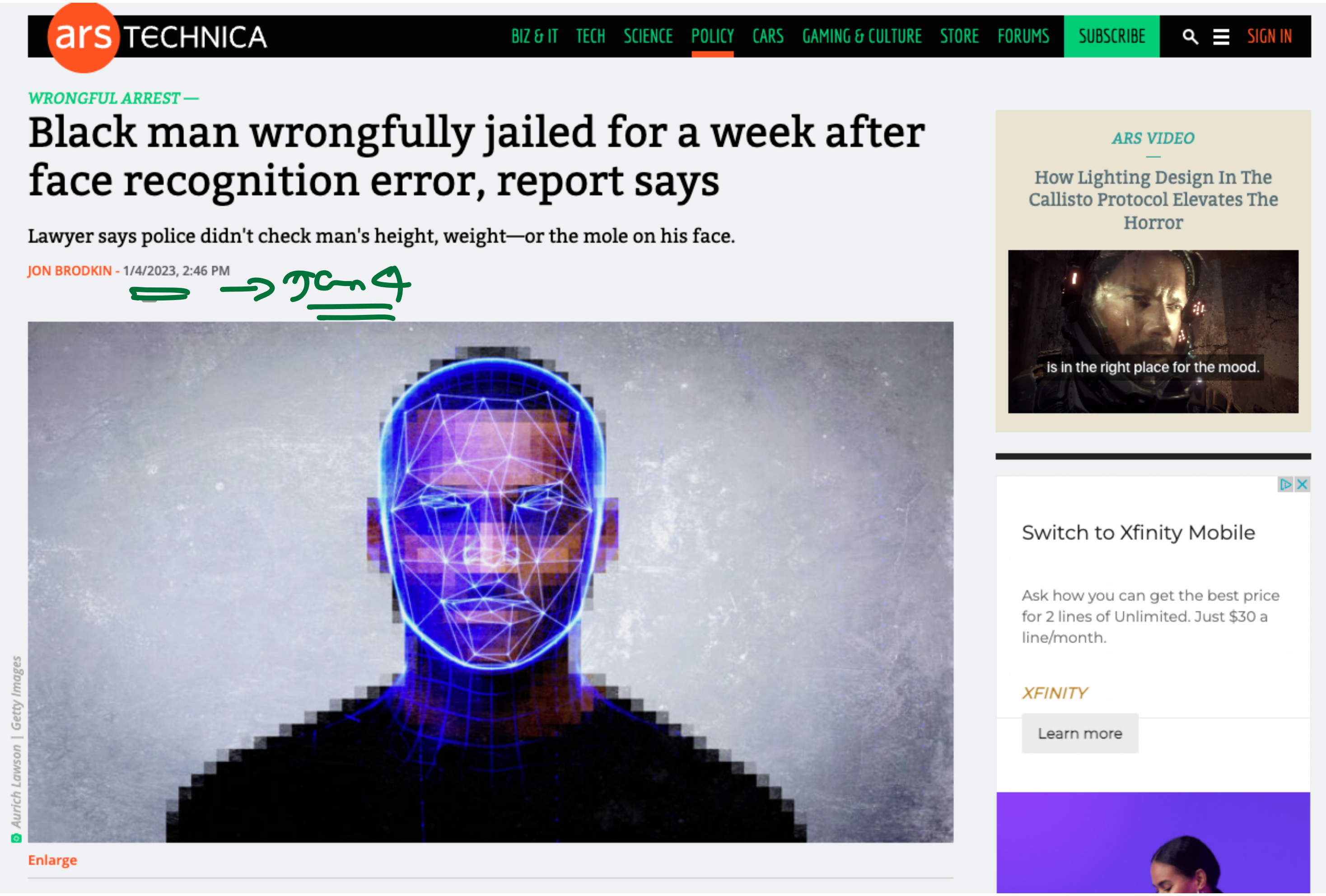
In terms of the data matrix, X: $\#$ rows $<<$ $\#$ columns

Breakout #1

Salary prediction from Resume

A Tech company would like to automate salary prediction based on resumes and profiles when it comes to giving a job offer for successful candidates. You as a ML engineer on the team decide to gather past data from Human Resources (HR). The HR team hands out to you a dataset that consists of anonymized job offer details of current employees where specifically, you have the resume for each anonymized employee and their negotiated job offer. You train a regression model that takes in attributes from the resume and predicts the offer salary. As you test the model on your test data sets - You dig in a little deeper on the weights that are learned by the model. You discover that the weight attributed to employees with the male geneder is higher than those of the female gender. This gets you wondering on what's happening with the ML model? Discuss in your groups - a) What attributes/features would you have used in the ML model b) What may be some reasons for the bias that might be happening and how would you mitigate it?

Recent Example of Model Bias



WRONGFUL ARREST —

Black man wrongfully jailed for a week after face recognition error, report says

Lawyer says police didn't check man's height, weight—or the mole on his face.

JON BRODKIN - 1/4/2023, 2:46 PM → Jan 4

Aurich Lawson | Getty Images

Enlarge

ARS VIDEO

How Lighting Design In The Callisto Protocol Elevates The Horror

News Article

Over-fitting, Model Bias, Robustness

Over-ftting

When the model "over-fits" to the data - Perhaps there wasn't enough data or there was too many parameters in the model (e.g. higher degree polynomial)

Model Bias

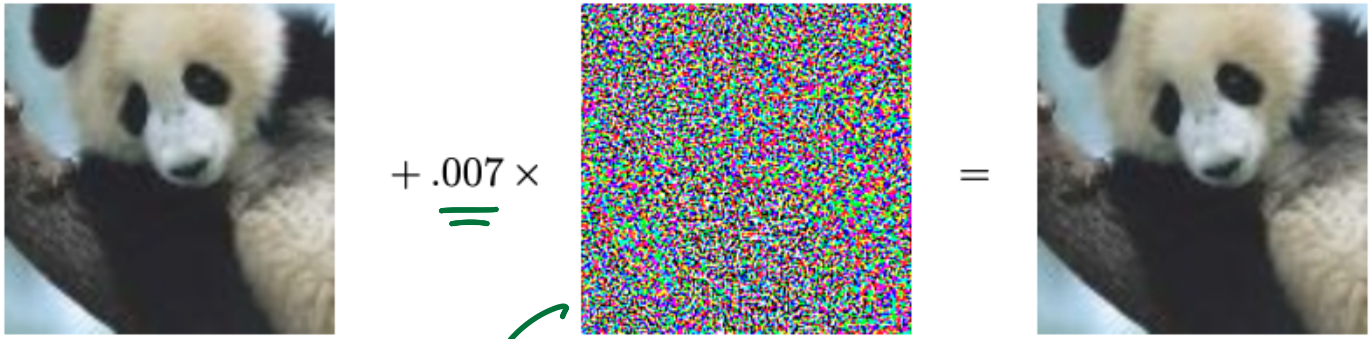
The model ends up having a bias in its predictions. Model bias can be due to over-fitting or it can also be due to data bias (e.g. resume $\rightarrow$ salary prediction example we looked at earlier)

Model Robustness

A model is said to be not so robust when small perturbations to the data can lead to very different results. Model robustness may or may not be connected to over-fitting. Certainly if model has seen only few examples of a particular data, it maybe more prone to robustness issues.

Model Robustness Example

classification |GoogleNet (cv model)



$\boldsymbol{x}$

"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

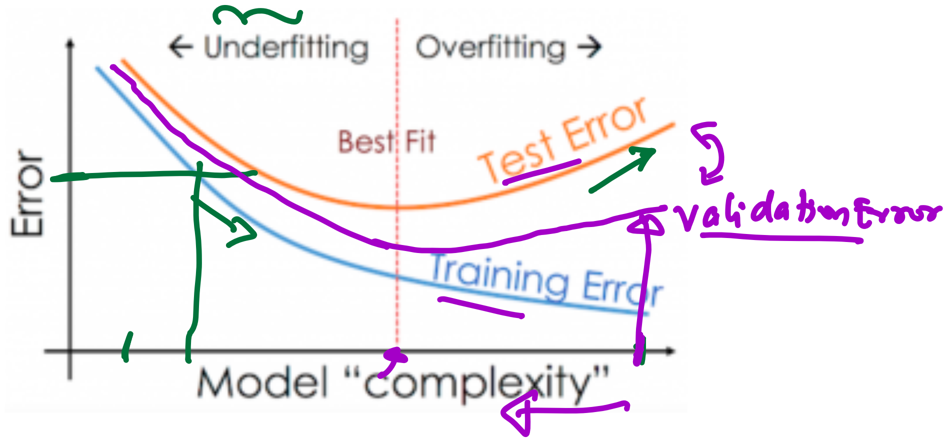
"nematode"
8.2% confidence

$\boldsymbol{x} +$
$\epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

ICLR 2015 reference paper

The figure to remember for over-fitting!



Polynomial Fits :- Degree Higher ↑ → More complexity

Example: 70 : 10 : 20 Train-Val-Test data split

Remainder becomes validation data

x_1	x_2	x_3	x_4	x_5	x_6	x_7	y
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

Data Splits for Machine Learning

- **Training** Learning parameters/weights, i.e. w for Linear Regression is called Training.

Data Splits for Machine Learning

- **Training** Learning parameters/weights, i.e. w for Linear Regression is called Training.
- We don't use all data for training - Some portion of the data is kept for validation and testing.

Data Splits for Machine Learning

- **Training** Learning parameters/weights, i.e. w for Linear Regression is called Training.
- We don't use all data for training - Some portion of the data is kept for validation and testing.
- **Data Splits:** Usually, 80% of data is kept for training, 10% for validation and 10% for training. The splits are chosen randomly.

Data Splits for Machine Learning

- **Training** Learning parameters/weights, i.e. w for Linear Regression is called Training.
- We don't use all data for training - Some portion of the data is kept for validation and testing.
- **Data Splits:** Usually, 80% of data is kept for training, 10% for validation and 10% for training. The splits are chosen randomly.
- Why not use all data for training ?

Data Splits for Machine Learning

- **Training** Learning parameters/weights, i.e. w for Linear Regression is called Training.
- We don't use all data for training - Some portion of the data is kept for validation and testing.
- **Data Splits:** Usually, 80% of data is kept for training, 10% for validation and 10% for training. The splits are chosen randomly.
- Why not use all data for training ? ✓ (Eval on unseen Data)
- Why not just have **train** and **test** data? What's the point of validation data set? (Account for overfitting)

Understanding over-fitting better

- Idea is that there maybe many solutions that fit the data - So pick the solution wisely!

Understanding over-fitting better

- Idea is that there maybe many solutions that fit the data - So pick the solution wisely!
- ✓ Polynomial regression is definitely prone to over-fitting for high-degree polynomials

Understanding over-fitting better

- Idea is that there maybe many solutions that fit the data - So pick the solution wisely!
- Polynomial regression is definitely prone to over-fitting for high-degree polynomials
- Overfitting can also happen with linear regression!! How?

Understanding over-fitting better

- Idea is that there maybe many solutions that fit the data - So pick the solution wisely!
- Polynomial regression is definitely prone to over-fitting for high-degree polynomials
- Overfitting can also happen with linear regression!! How?
- Consider the linear system $\underline{Xw = y}$. This system is under-determined when $\underline{\underline{N < d}}$ (number of examples ¡ feature dimension)

Understanding over-fitting better

- Idea is that there maybe many solutions that fit the data - So pick the solution wisely!
- Polynomial regression is definitely prone to over-fitting for high-degree polynomials
- Overfitting can also happen with linear regression!! How?
- Consider the linear system $Xw = y$. This system is under-determined when $N < d$ (number of examples ¡ feature dimension)
- Infinitely many solutions when $\underline{\underline{N < d}}$!

Understanding over-fitting better

- ICE #2: Find a solution for $\underline{\underline{1^T w = 1}}$ where $w \in \mathcal{R}^2$.

$$\downarrow$$

$$\underline{\underline{w_1 + w_2}} = 1$$

$$\left.\begin{array}{cc} 0 & 1 \\ 0.5 & 0.5 \\ 1 & 0 \\ 0.3 & 0.7 \\ \vdots & \end{array}\right\} \rightarrow$$

Understanding over-fitting better

- ICE #2: Find a solution for $1^T w = 1$ where $w \in \mathcal{R}^2$.
- Unique solution when $N > d$ and when X has full rank!

Understanding over-fitting better

- ICE #2: Find a solution for $1^T w = 1$ where $w \in \mathcal{R}^2$.
- Unique solution when $N > d$ and when X has full rank!
- Over-fitting happens when number of data points comparable to the number of attributes/features (order)

Understanding over-fitting better

- ICE #2: Find a solution for $1^T w = 1$ where $w \in \mathcal{R}^2$.
- Unique solution when $N > d$ and when X has full rank!
- Over-fitting happens when number of data points comparable to the number of attributes/features (order)
- `Solution A` to overfitting: Increase number of examples so that $\underline{\underline{N}} >> d$

Understanding over-fitting better

- ICE #2: Find a solution for $1^T w = 1$ where $w \in \mathcal{R}^2$.
- Unique solution when $N > d$ and when X has full rank!
- Over-fitting happens when number of data points comparable to the number of attributes/features (order)
- `Solution A` to overfitting: Increase number of examples so that $N >> d$
- `Solution B`: Decrease number of features so that $\underline{d << N}$

Understanding over-fitting better

- ICE #2: Find a solution for $1^T w = 1$ where $w \in \mathcal{R}^2$.
- Unique solution when $N > d$ and when X has full rank!
- Over-fitting happens when number of data points comparable to the number of attributes/features (order)
- `Solution A` to overfitting: Increase number of examples so that $N >> d$
- `Solution B`: Decrease number of features so that $d << N$
- `Solution C`: Regularization! (Perhaps accomplish B as well along the way)

Regularization in Linear Regression

Large weights

Is a sign of over-fitting. (People have been seeing an RMSE of $1e + 20$ on the housing prices data set). With large weights - Small changes in feature value can throw the predictions off.

Regularization in Linear Regression

Large weights

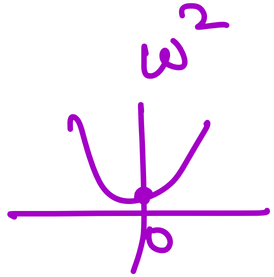
Is a sign of over-fitting. (People have been seeing an RMSE of $1e+20$ on the housing prices data set). With large weights - Small changes in feature value can throw the predictions off.

l_2 Regularization - Ridge Regression

Regularized loss (objective function):

$$\min_w \frac{1}{2}\underbrace{\overbrace{\|Xw-y\|_2^2}^{\to \text{Linear Regren}}}_{\textcircled{1}} + \underbrace{\overbrace{\lambda\|w\|_2^2}}_{\textcircled{2}}$$

Ridge Regression



Regularization in Linear Regression

Large weights

Is a sign of over-fitting. (People have been seeing an RMSE of $1e+20$ on the housing prices data set). With large weights - Small changes in feature value can throw the predictions off.

l_2 Regularization - Ridge Regression

Regularized loss (objective function):

$$\min_{w} \frac{1}{2}\|Xw-y\|_2^2+\lambda\underline{\underline{\|w\|_2^2}}$$

l_1 Regularization - ~~Ridge Regression~~ Lasso

Regularized loss (objective function):

sparse features/

sparsity

$$\|w\|_1 = |w_1|+|w_2| + |w_3| + - - + |w_d|$$

$$\min_{w} \frac{1}{2}\|Xw-y\|_2^2+\lambda\|w\|_1$$

Elastic net

"parameters"

→ sparse

$$\min_{w} \frac{1}{2}\underbrace{\|Xw-y\|_2^2}_{\text{Fit}} + \underline{\lambda_1}\|w\|_1 + \underline{\lambda_2}\|w\|_2^2$$

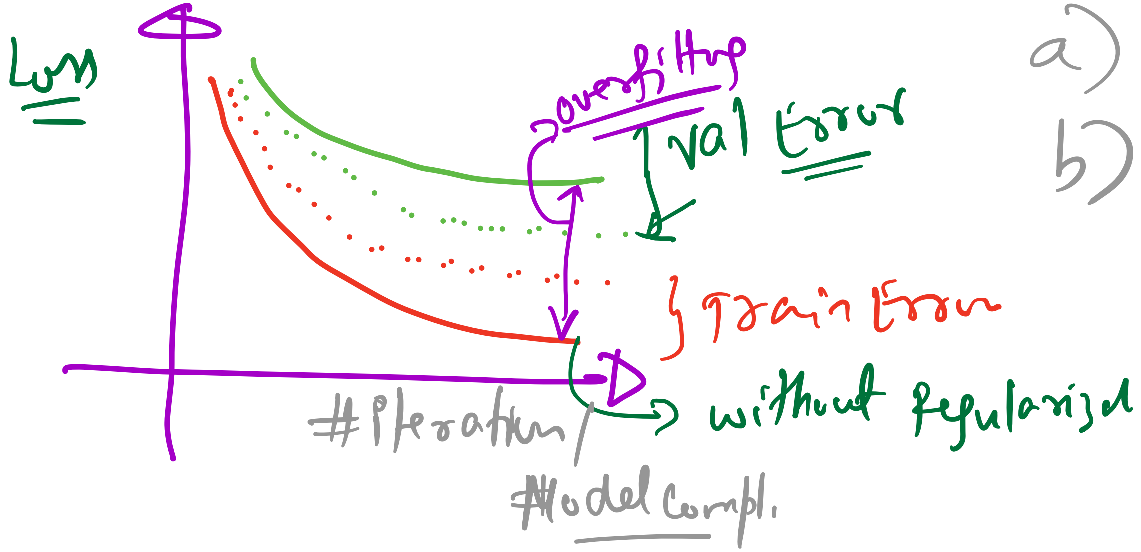
Prevent large weights

"Hyper parameters"

ICE #3

Quad Loss = $\frac{1}{2}\|Xw-y\|_2^2$

Which of the two figures is the right one for a regularized Linear Regression Model?



a) Dotted Lines

b) Solid Lines

ICE #4

Ridge Regression

$$\min_{\underline{w}} \overbrace{\frac{1}{2}\|Xw - y\|_2^2 + \lambda\|w\|_2^2}$$

What happens to $\hat{w}$ as $\lambda \to \infty$?

a) $\hat{w} = (X^T X)^{-1}(X^T y)$

b) $\hat{w} = \frac{1}{\sqrt{\lambda}}[1, 1, \ldots, 1]$

c) $\hat{w} = [\underline{0, 0,} \ldots, 0]$

Summary so far

- Linear Regression finds a line of best fit through the data.
- R^2 measure determines the goodness of fit.
- Usually multiple good attributes are needed for a good prediction and a good fit. → Feature Engg.
- Data pre-processing. Categorical attributes are handled through creation of dummy attributes and in addition normalizing of the attributes brings all attributes on the same scale for regression.
- We have a closed form/analytical solution for Linear Regression, but for large data sets, gradient descent algorithm (iterative) gets used for scalability reasons.
- We don't use all of a data set for training. A portion of data is kept for validation and testing. This is to prevent over-fitting and also for fair evaluation purposes.
- The data set split is usually $80 - 10 - 10$ or $70 - 10 - 20$ (train-val-test).

Summary so far

- Over-fitting happens when we have fewer data points as compared to the number of attributes or features.
- Over-fitting can be taken care off by increasing data-set size, decreasing number of attributes or through regularization strategies

Coding Pointers for Assignment 1

- **Pandas** library in Python is good for data pre-processing before training your Linear Regression model

Coding Pointers for Assignment 1

- **Pandas** library in Python is good for data pre-processing before training your Linear Regression model
- **Dummy attributes** for categorical variables can also be added in through `pandas.get_dummies()` method.

Coding Pointers for Assignment 1

- **Pandas** library in Python is good for data pre-processing before training your Linear Regression model
- **Dummy attributes** for categorical variables can also be added in through `pandas.get_dummies()` method.
- Use **Scikit-learn** for implementing Linear Regression. Options for ridge regression and lasso available.

Coding Pointers for Assignment 1

- **Pandas** library in Python is good for data pre-processing before training your Linear Regression model
- **Dummy attributes** for categorical variables can also be added in through `pandas.get_dummies()` method.
- Use **Scikit-learn** for implementing Linear Regression. Options for ridge regression and lasso available.
- **Hyper-parameter tuning** needed on validation data set to get the “best” model that has least amount of over-fitting!

Lasso and Sparsity

- Lasso tends to "sparsify" $\hat{w}$ as λ increases.

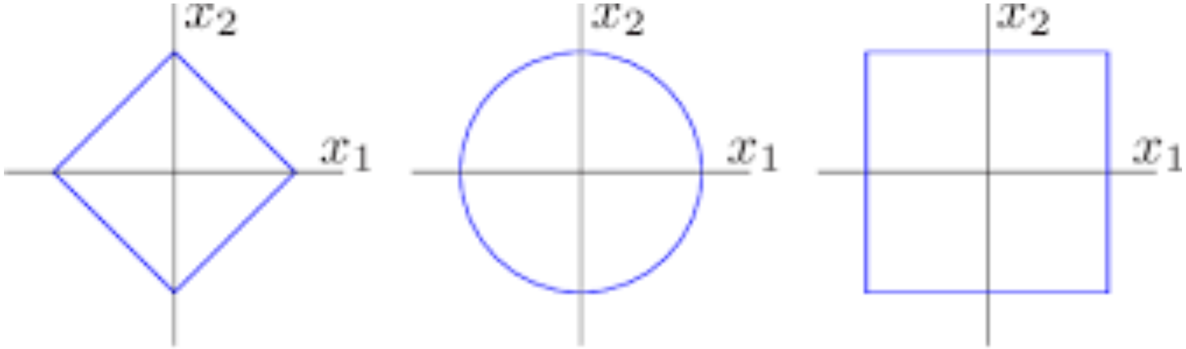
Lasso and Sparsity

- Lasso tends to "sparsify" $\hat{w}$ as λ increases.
- If you have too many features and want to reduce them either to prevent over-fitting or keep model simple - You can either do "feature selection" heuristics or use Lasso

Lasso and Sparsity

- Lasso tends to “sparsify” $\hat{w}$ as λ increases.
- If you have too many features and want to reduce them either to prevent over-fitting or keep model simple - You can either do “feature selection” heuristics or use Lasso
- Lasso also leads to explainable machine learning models. Why?

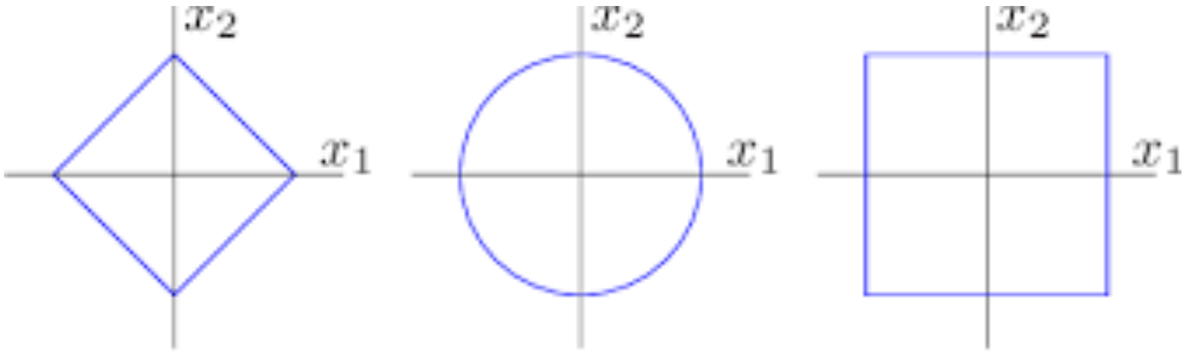
Norm balls and their role in Regularization



(a) ℓ_1-ball (b) ℓ_2-ball (c) ℓ_∞-ball

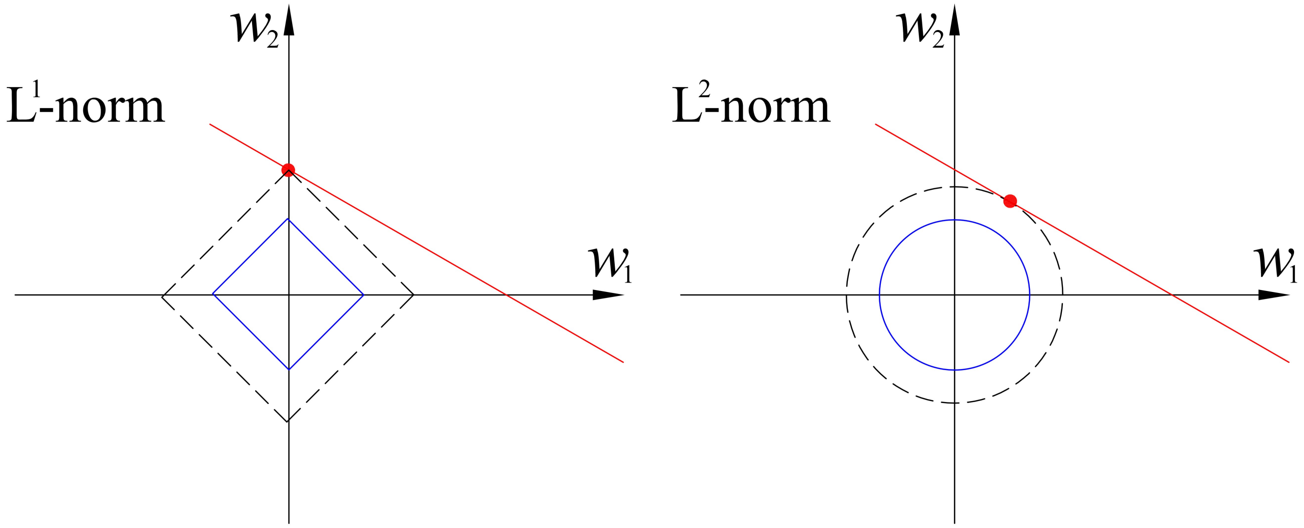
Fig. 1. The unit norm balls

Norm balls and their role in Regularization



(a) ℓ_1-ball (b) ℓ_2-ball (c) ℓ_∞-ball

Fig. 1. The unit norm balls



Two ends of the regularization spectrum

The extremes

When $\lambda = 0$, we have $\hat{w} = (X^T X)^{-1} X^T y$. When $\lambda = \infty$, we get $\hat{w} = 0$.

Two ends of the regularization spectrum

The extremes

When $\lambda = 0$, we have $\hat{w} = (X^T X)^{-1} X^T y$. When $\lambda = \infty$, we get $\hat{w} = 0$.

Middle Path

If one end is over-fitting, and the other end is under-fitting. Hyper-parameter tuning on λ, gives us the middle path and the best hyper-parameter to "minimize" validation error.

Hyper-parameter Tuning

Grid Search

Search across a grid of hyper-parameters. Example for λ - Perhaps choose $0, 10^{-4}, 10^{-3}, 10^{-2}, \ldots$ to search across.

Hyper-parameter Tuning

Grid Search

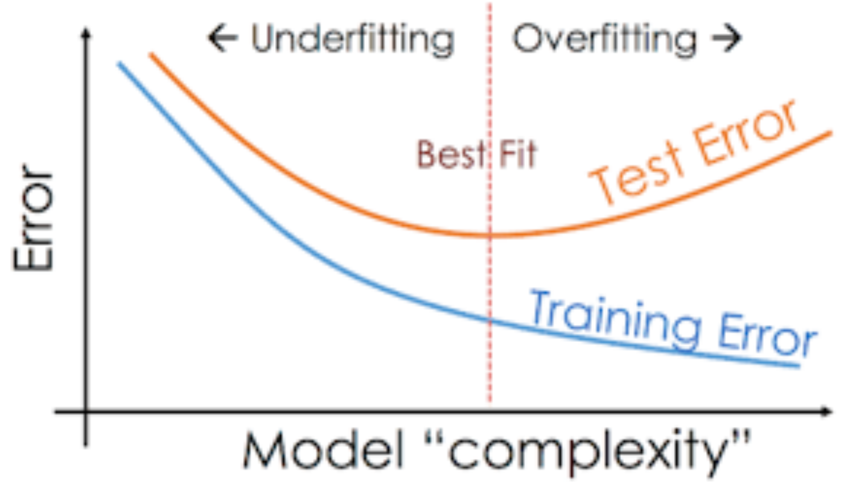
Search across a grid of hyper-parameters. Example for λ - Perhaps choose $0, 10^{-4}, 10^{-3}, 10^{-2}, \ldots$ to search across.

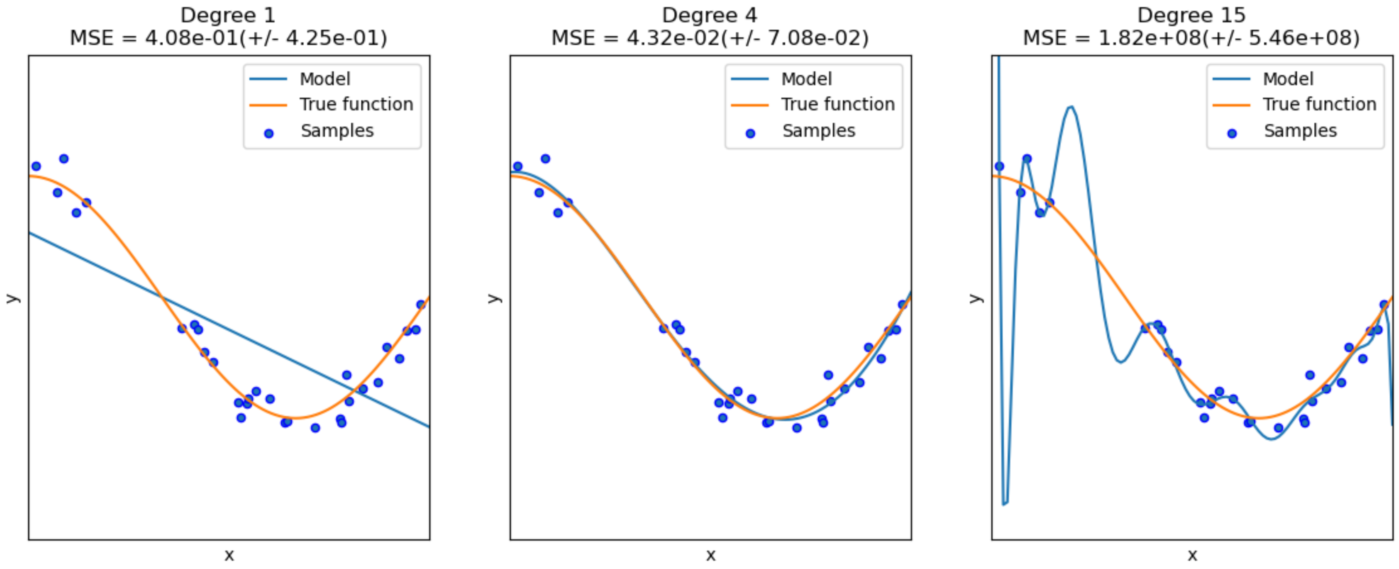
Train vs Validation

Even if Training is a 'Regularized Loss' (e.g. Ridge Regression), we minimize the following Loss on Validation data:

$$\frac{1}{2}\|Xw - y\|_2^2$$

Over-fitting - Tale of polynomials


← Underfitting
Overfitting →
Best Fit
Test Error
Training Error
Error
Model "complexity"


Degree 1
MSE = 4.08e-01(+/- 4.25e-01)
Degree 4
MSE = 4.32e-02(+/- 7.08e-02)
Degree 15
MSE = 1.82e+08(+/- 5.46e+08)
Model
True function
Samples
x
y