Basics of Linear regression and Ridge regression

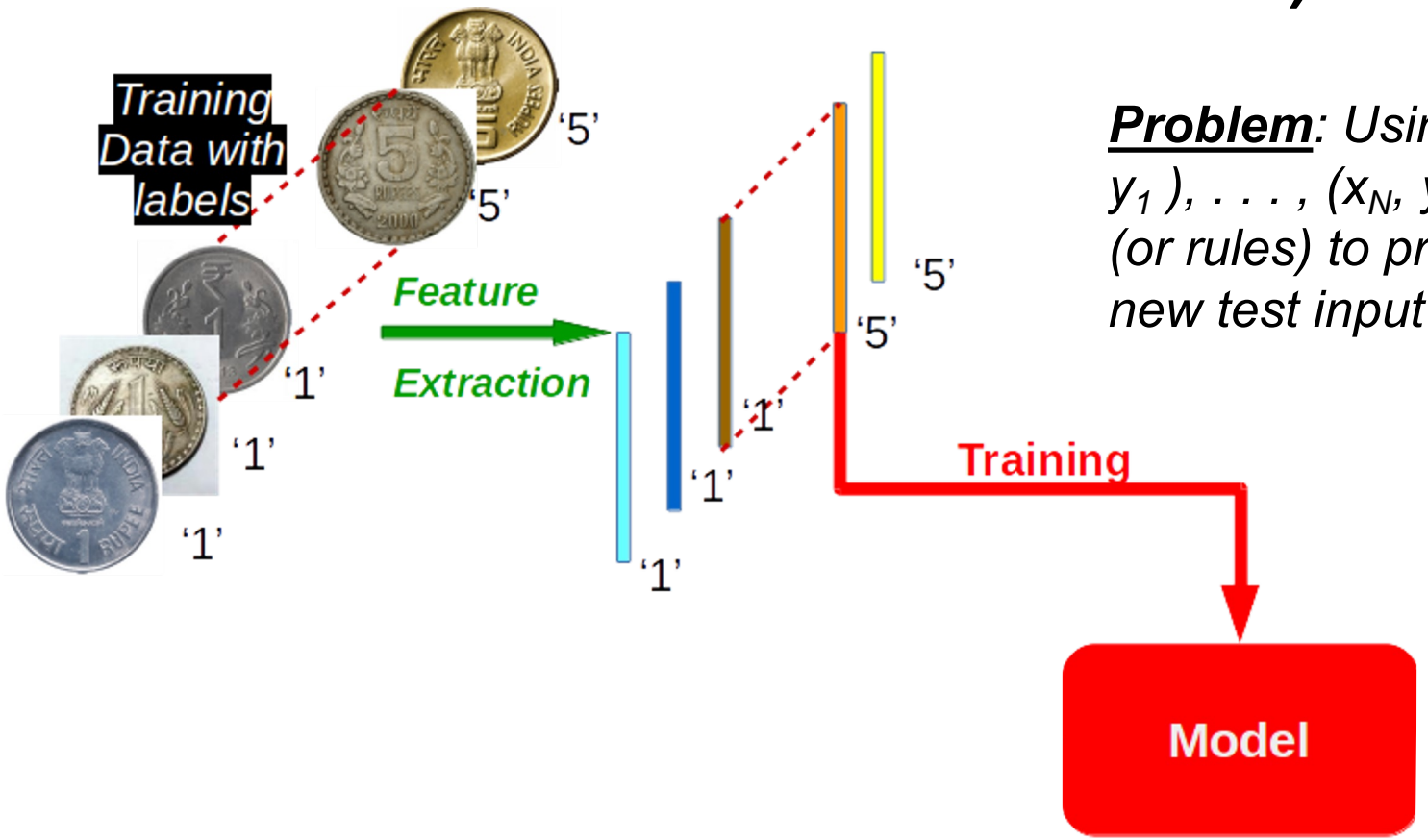
By: Dr. Puneet Gupta

The pillars of DL

Supervised learning

Unsupervised learning

Supervised Learning (in classification)

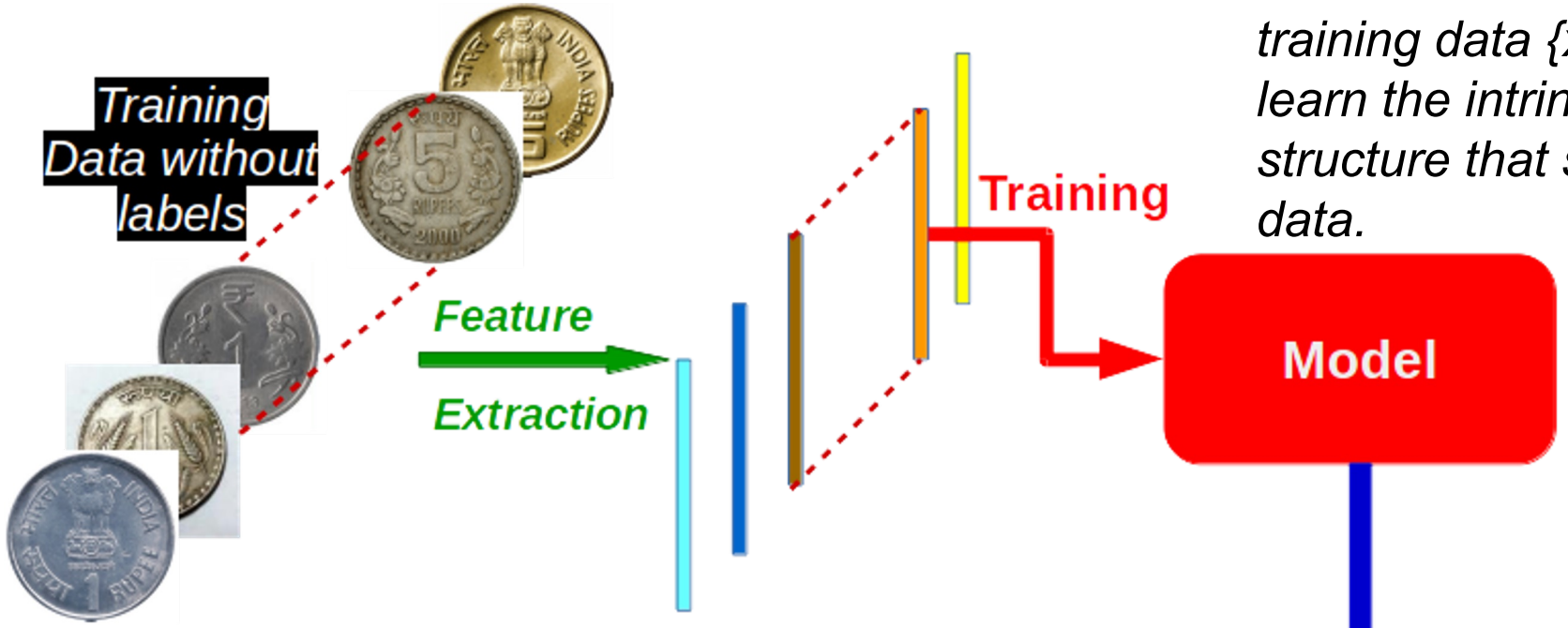


Problem: *Using training data {(x_1 , y_1), . . . , (x_N, y_N)}, learn a model (or rules) to predict output y on new test input x*

- Feature extraction can also be part of model learning. It is known as “feature learning” or “representation learning”. Modern “deep learning” algorithms follow this.
- Supervised learning is useful in: classification, regression and ranking

Unsupervised Learning

Problem: *Using unlabelled training data $\{x_1, \ldots, x_N\}$, learn the intrinsic hidden structure that summarizes data.*



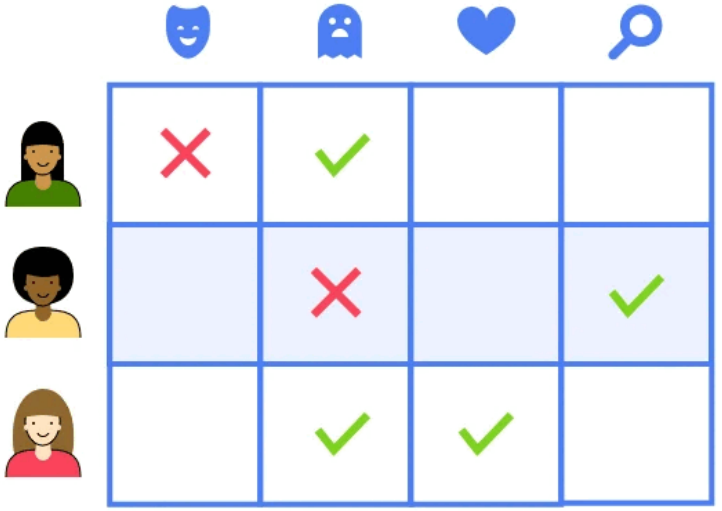
- It can have “test” phase where it predict the cluster.
- Used as a preprocessing in supervised learning to learn good features or to make them computationally efficient.
- It is usually harder than supervised learning.
- It is useful in clustering and dimensionality reduction.

Unsupervised Learning

Examples:

- Targeted marketing: Segment customers according to their similarities
- Targeted production: Group people of similar sizes and taste for cloth production
- Youtube: Video indexing
- And so on….

Collaborative Filtering
https://en.wikipedia.org/wiki/Recommender_system



Netflix Recommendation System
https://www.codecademy.com/article/how-netflix-recommendation-works-data-science

Unsupervised Learning + some supervision

https://thispersondoesnotexist.com/

A dog wearing a Superhero outfit with red cape flying through the sky.

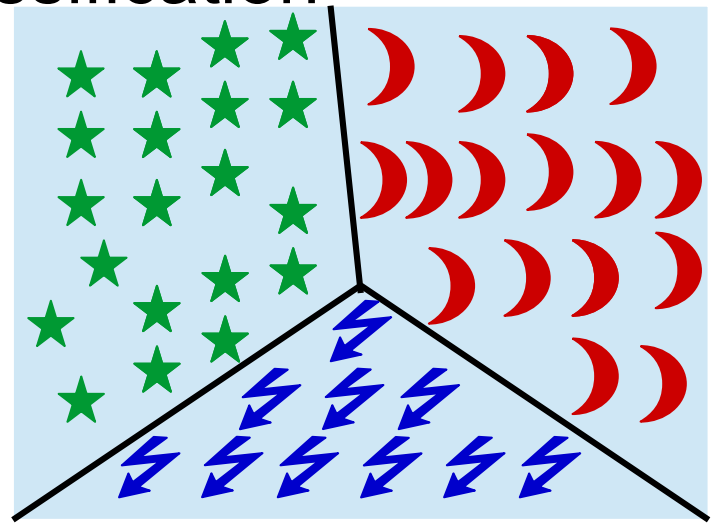
https://makeavideo.studio/

PR: Some Predictive Tasks

Classification

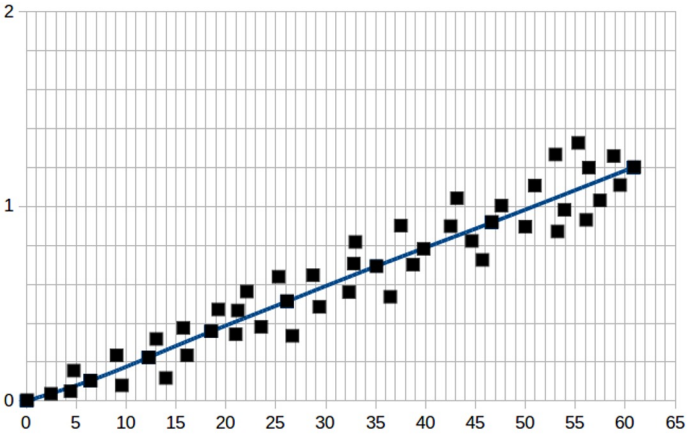
It maps data into predefined labels.

Eg: Spam vs non-spam by analysing frequency of words



Regression

It map data into real value variable.
Eg: Estimate cost of a building by analysing location, plot area, construction year, etc...



Time Series Analysis

Values of attribute are examined over time.

Eg: Comparing stocks or weather prediction by analysing past and current estimates.

Eg: Weather forecasting by (20jul, no-rain), (22jul, rain), (24jul, rain), (26jul, ...?...)

Classification or Regression?
Supervised or Unsupervised?

Some descriptive tasks involve summarization, captioning, association rules mining, sequence discovery and so on...

Basic Maths refresher

$$\frac{\partial \mathbf{x}}{\partial y} = \begin{bmatrix} \frac{\partial x_1}{\partial y} \\ \frac{\partial x_2}{\partial y} \\ \vdots \\ \frac{\partial x_d}{\partial y} \end{bmatrix}$$

$$\frac{\partial \mathbf{x}}{\partial \mathbf{y}} = \begin{bmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} & \cdots & \frac{\partial x_1}{\partial y_n} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} & \cdots & \frac{\partial x_2}{\partial y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_d}{\partial y_1} & \frac{\partial x_d}{\partial y_2} & \cdots & \frac{\partial x_d}{\partial y_n} \end{bmatrix}$$

Jacobian

For two matrix, A and B, we have :

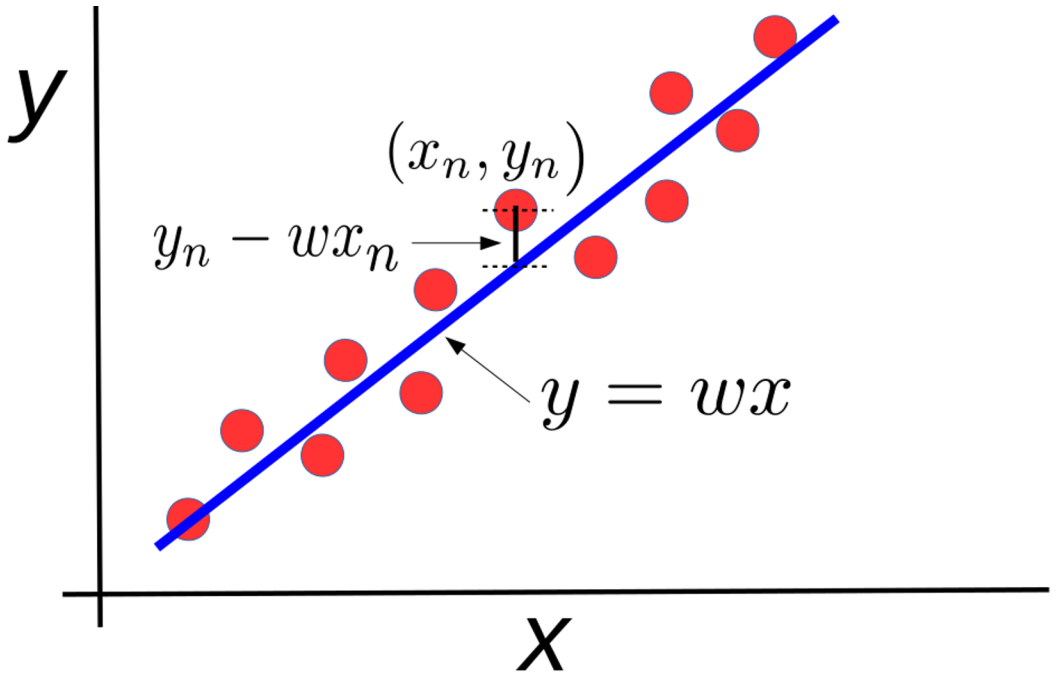
a) $(AB)^T = B^T A^T$

b) $(A+B)^T = A^T + B^T$

Linear Regression

- Linear Models are used in SVM, Deep learning, and etc...
- Defining a rule is not always feasible.
- How to learn their weights (or unknowns) from data?
- Formulate learning as optimization problem w.r.t. weights

Parameteric ML



1-D pictorial example of linear regression

Equation of line:
y = mx +c

c can be considered as bias then, m is the weight.

For simplicity, we consider the following:

- **w** = $[m\ c]^T$
- **x** = $[x\ 1]^T$

Linear Regression

Total error on training data, $\mathcal{L}(\mathbf{w}) = \sum_{n=1}^{N} \left(y_n - \mathbf{w}^T \mathbf{x_n}\right)^2$

- Squared loss chosen for simplicity.
- The best **w** minimizes training error w.r.t. **w**

Modelling weight, $\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}}\ \mathcal{L}(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} \left(y_n - \mathbf{w}^T \mathbf{x_n}\right)^2$

Least squares linear regression : Set derivative of $\mathcal{L}(\mathbf{w})$ w.r.t. $\mathbf{w}$ to zero

$$\sum_{n=1}^{N} 2\left(y_n - \mathbf{w}^T \mathbf{x_n}\right) \frac{\partial}{\partial \mathbf{w}} \left(y_n - \mathbf{x_n}^T \mathbf{w}\right) = 0$$

$$\Rightarrow \sum_{n=1}^{N} \mathbf{x_n}^T \left(y_n - \mathbf{x_n}^T \mathbf{w}\right) = 0$$

$$\Rightarrow \mathbf{w} = \sum_{n=1}^{N} \left(\mathbf{x_n}^T \mathbf{x_n}^T\right)^{-1} \sum_{n=1}^{N} \mathbf{x_n}^T y_n = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y}$$

Closed form solution for **w**

Linear Regression

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_D \end{bmatrix} = \mathbf{y} \approx \mathbf{X}\mathbf{w} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1D} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2D} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & x_{N3} & \dots & x_{ND} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} = \begin{bmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \\ \vdots \\ \mathbf{x_N}^T \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}$$

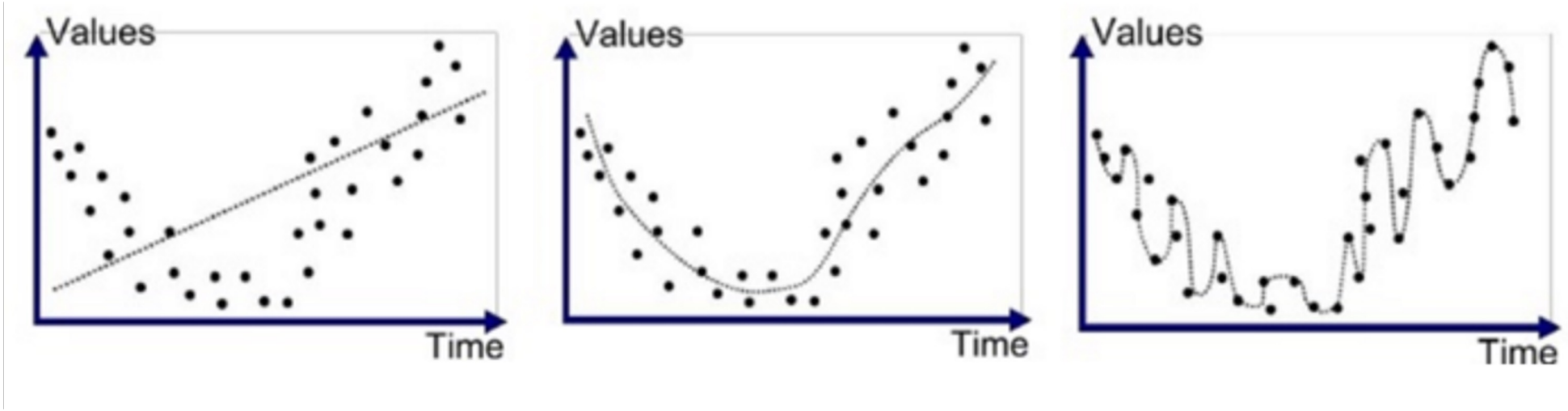
$$\mathbf{w} = \sum_{n=1}^{N} \left(\mathbf{x_n}\mathbf{x_n}^T\right)^{-1} \sum_{n=1}^{N} y_n \mathbf{x_n} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}^T\mathbf{y}$$

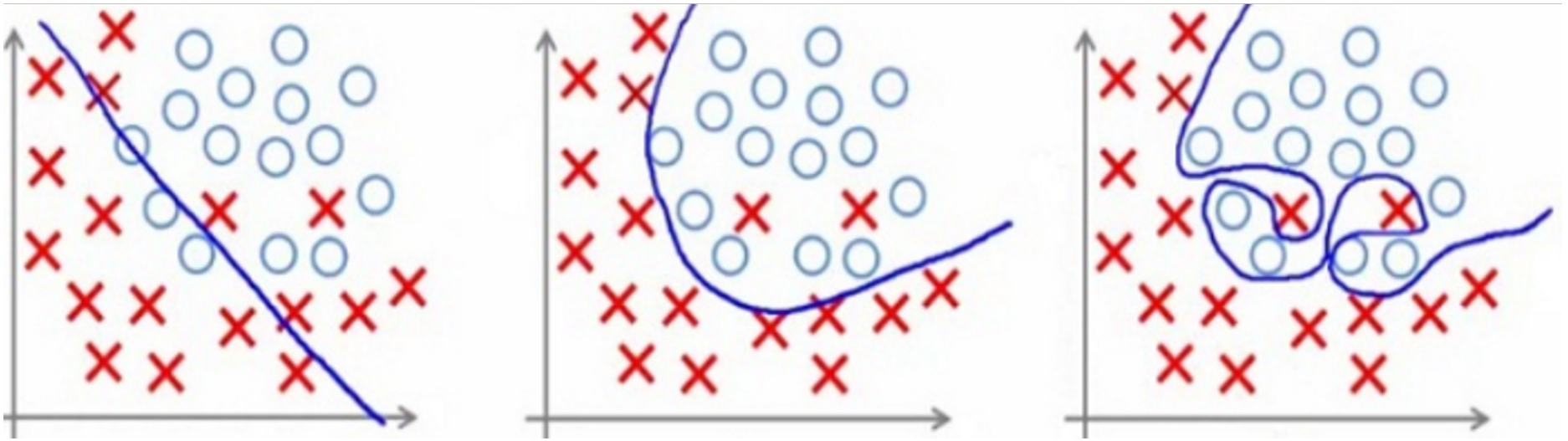
In **w**, w_d denotes the importance of d^{th} input feature for predicting **y**

Problems with the closed form solution:

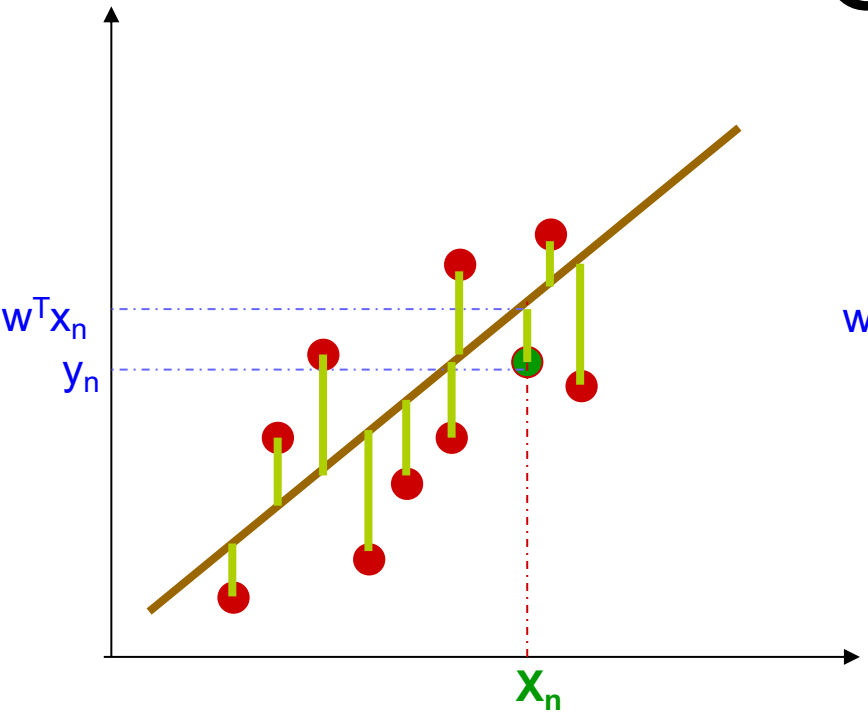
- Outliers or noise
- ($\mathbf{X}^T\mathbf{X}$) may not be invertible
- ***Overfitting***: Based solely on minimizing the training error ***What is this?***
- Expensive inversion for large D

Overfitting vs underfitting

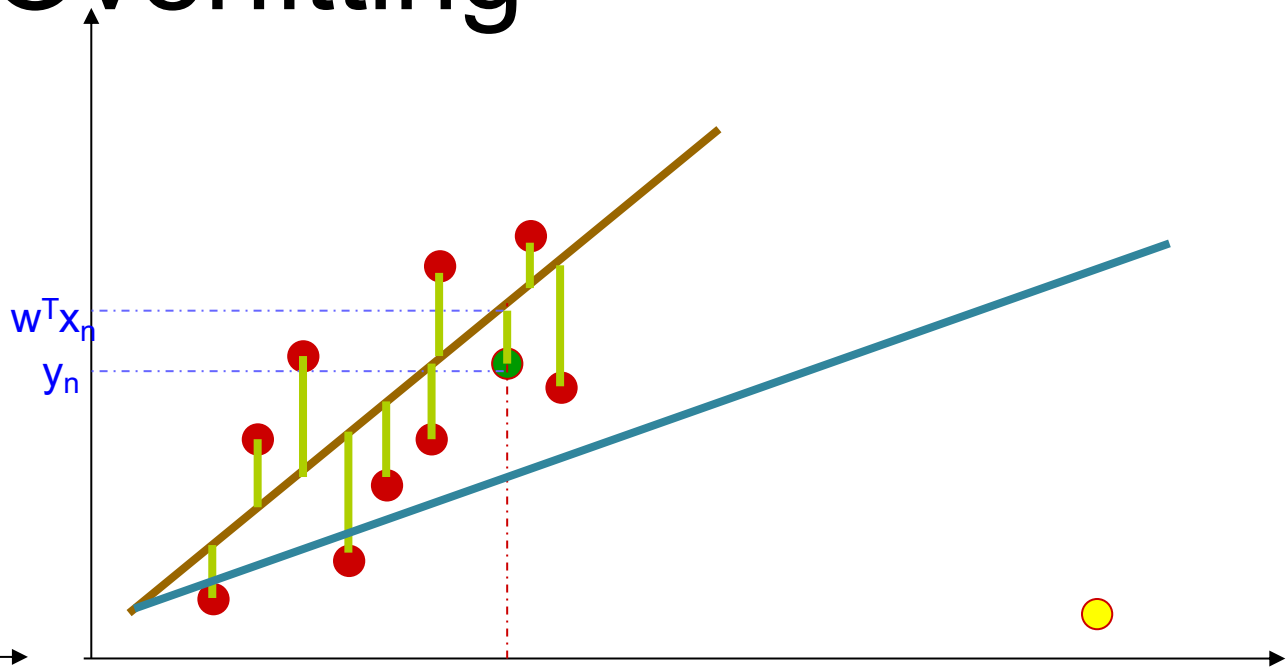




Overfitting



Linear regression tries to minimize the sum of actual squared **error**.



What will happen when a noisy sample is introduced?

Reasons of noise:

- Some values of attributes are incorrect because of errors in the data acquisition process or the preprocessing phase
- The classification is wrong because of some error

Overfitting reason 1: When a model gets trained with so much of data, it starts learning from the noise and inaccurate data entries.

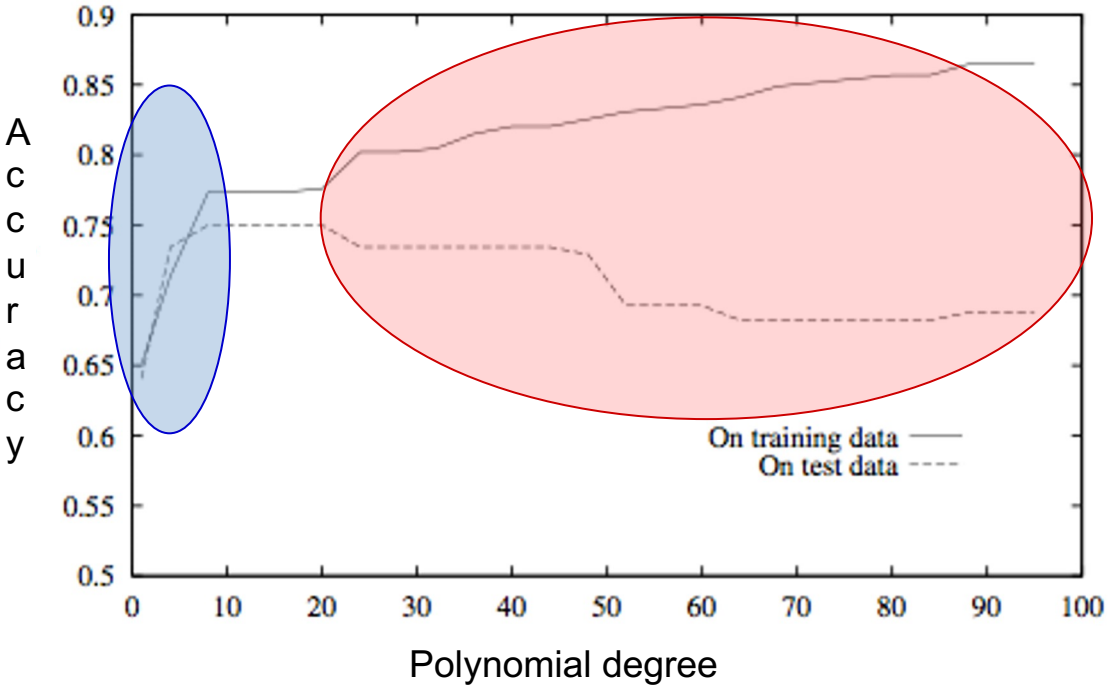
Overfitting

Overfitting reason 2: If there are large number of attributes, DL/ML/PR algorithms may find meaningless regularity in the data that is irrelevant to the true, important, distinguishing features. It is due to lack of data points. e.g.,

1) predicting rain whether you go out or not is irrelevant.
2) predict the roll of a die using day of the week and color of the die

Overfitting means fitting the training set "too well" on the performance on the test set degrades.

Underfitting refers to a model that can neither model the training data nor generalize to new data.



- Model will keep on learning and thus the error for the model on the training and testing data will keep on decreasing.
- If learning goes too long, overfitting starts due to noise and less relevant attributes. Hence the performance of the model on test set decreases.
- For good model, we will stop at a point just before where the error starts increasing, i.e., the point where the modelperforms well on training and unseen testing dataset.

Ridge Regression or Regularized Linear Regression

Aim : Minimize regularized loss or

Aim : Find $\mathbf{w}$ *that minimizes both training error and Regularization*

Regularization : l_2*(or squared) norm of* $\mathbf{w}$*, i.e.,* $\|\mathbf{w}\|_2$

$$\mathcal{L}_{reg}(\mathbf{w}) = \left[\sum_{n=1}^{N}\left(y_n - \mathbf{w}^T\mathbf{x_n}\right)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}\right]$$

Tradeoff between error and norm is managed by hyperparameter λ :

- $\lambda > 0$
- λ *small means less regularization (overfitting)*
- λ *high means high regularization (underfitting)*
- *Choose* λ *using crossvalidation*

Optimal $\mathbf{w}$ *for* $\mathcal{L}_{reg}$ *is given by :* $\mathbf{w} = \left(\mathbf{X}^T\mathbf{X} + \lambda I_D\right)^{-1}\mathbf{X}^T\mathbf{y}$

Requires matrix inversion.

Why l_2 regularization?

L_2 *regularization enforces each* $\mathbf{w}$ *to be small*
It also enforces $y = f(x)$ *to be smooth*
(i.e., similar $\mathbf{x}$ *will provide similar* y*).*

How?
Consider two data points, $\mathbf{x_n}$ *and* $\mathbf{x_m}$ *which are same in all dimensions except* d^{th} *dimension where they differ by* δ*.*
Hence, their outputs y_n *and* y_m*should be close if* $f(x)$ *is smooth.*
Mathematically, they differ by $\delta\mathbf{w_d}$
If w_d *is large, the difference will be large (hence, smoothness violates)*

One can use l_0 *or* l_1 *norm instead of* l_2 *norm which provide sparsity and therby perform feature selection. (Revisit later)*

Linear and Ridge Regression

Regularized least squares regression (Ridge regression) minimizes training error and provide regularization, i.e.,

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}}\ \mathcal{L}_{reg}(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} \left[\sum_{n=1}^{N} \left(y_n - \mathbf{w}^T \mathbf{x_n} \right)^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \right]$$

Supervised learning aims to learn a function $\mathbf{f}$*, which can be solved similarly using optimation in the following way :*

$$\hat{\mathbf{f}} = \underset{\mathbf{f}}{\operatorname{argmin}}\ \mathcal{L}_{reg}(\mathbf{w}) = \underset{\mathbf{f}}{\operatorname{argmin}} \sum_{n=1}^{N} l\left(y_n, f(x_n)\right) + \lambda R(\mathbf{f})$$

In least squares regression,

- $f(\mathbf{x_n}) = \mathbf{w}^T \mathbf{x_n}$
- *Regularizer* $R(\mathbf{f}) = \mathbf{w}^T \mathbf{w}$
- $l\left(y_n, f(x_n)\right) = \left(y_n - \mathbf{w}^T \mathbf{x_n}\right)^2$

Different supervised learning algorithms differ in the choice of $f, R(.),$ *and* l*.*

Linear and Ridge Regression

Linear and ridge regression suffer from the problem of matrix inversion which is computationally expensive for large of D
Batch or stochastic gradient descent *can be used to handle it. (Revisit later)*

Linear regression can be considered as **solving system of linear equations**
Thus, solving over/underdetermined systems can also solve linear regression and most of them does not require matrix inversion.

The regression approaches can be extended for :
•**Nonlinear regression** *by replacing* $\mathbf{x_n}$ *with a nonlinear transformation* $\Phi(\mathbf{x_n})$ *i.e,* $y_n \approx \mathbf{w}^T\Phi(\mathbf{x_n})$, *where* Φ *is learned or predefined.*
•**Generalized Linear Model** *where* y_n *is not real but binary/categorical, etc. Here, g known as function is used as :* $y_n \approx g(\mathbf{w}^T\mathbf{x_n})$