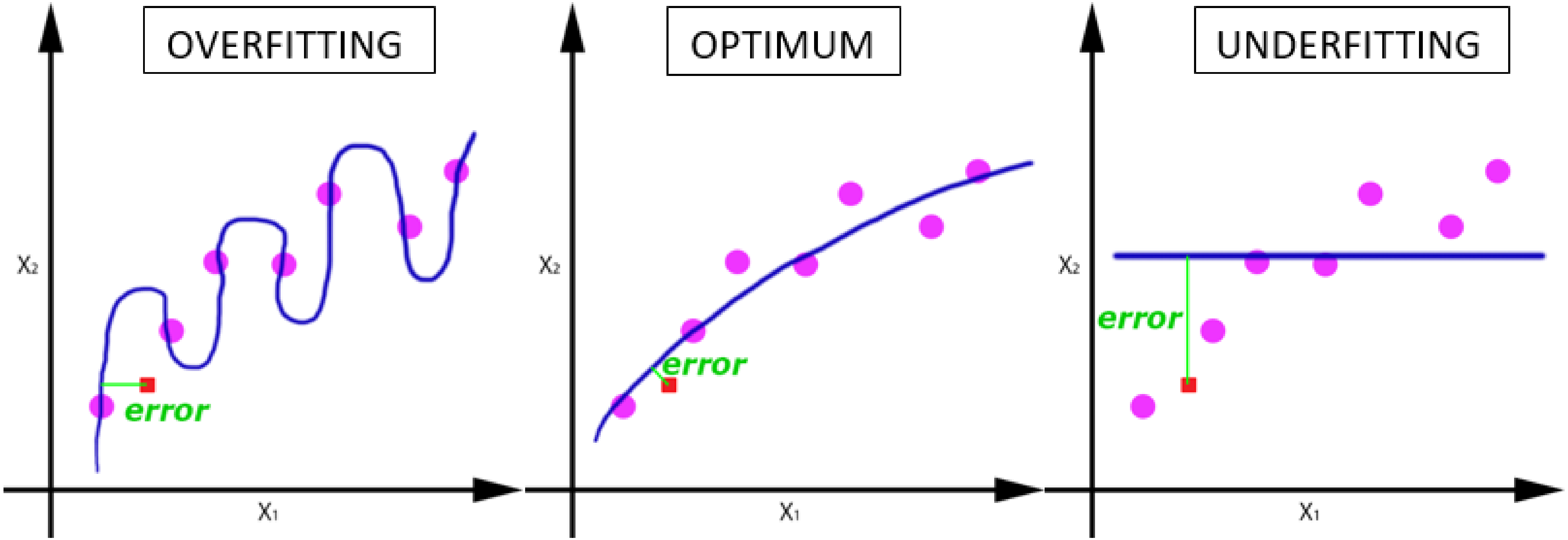
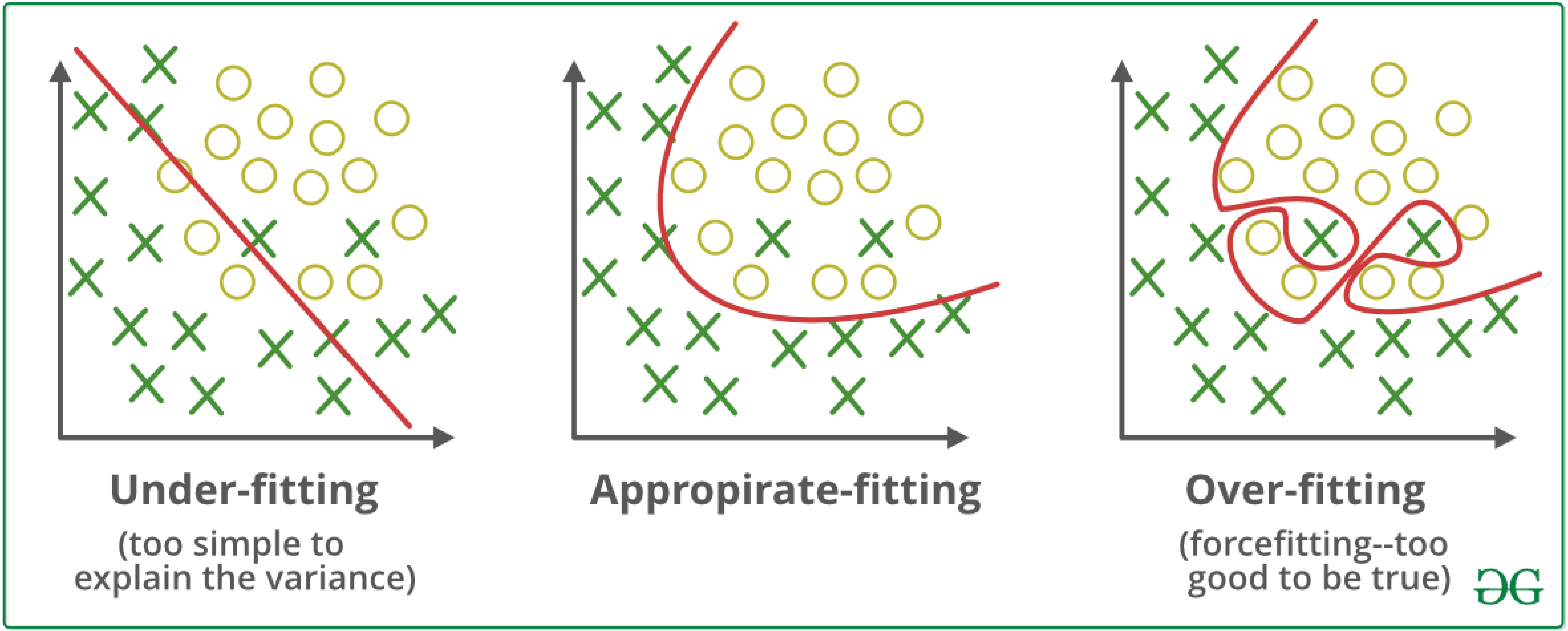
Underfitting & Overfitting
Regularization

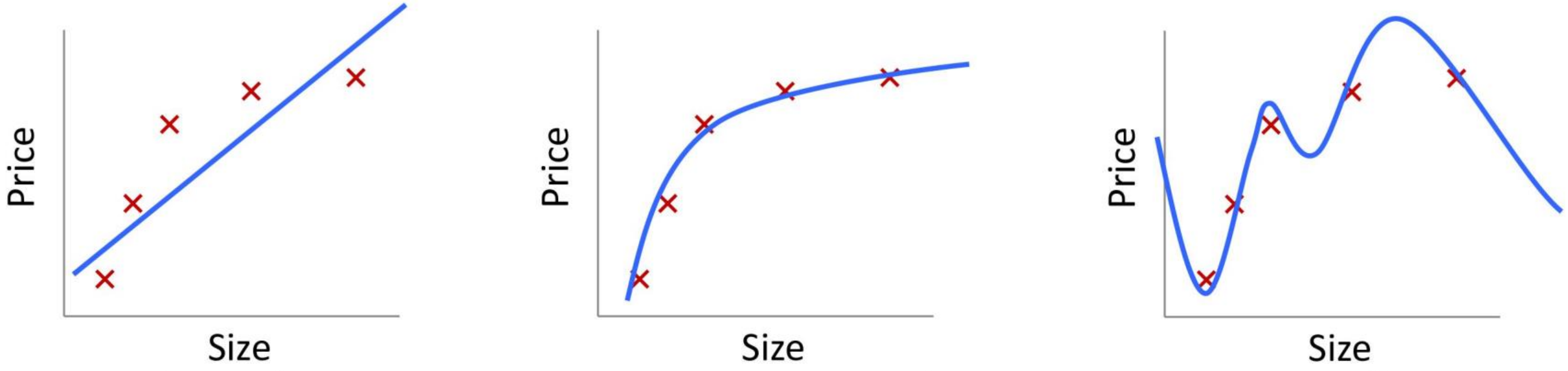
Regression



Classification



Quality of Fit



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

Underfitting (high bias) | Correct fit | Overfitting (high variance)

Overfitting:

- The learned hypothesis may fit the training set very well ($J(\boldsymbol{\theta}) \approx 0$)
- ...but fails to generalize to new examples

Large θ-ல என்ன பிரச்சனை?

Regularization

- A method for automatically controlling the complexity of the learned hypothesis

- **Idea**: penalize for large values of θ_j
  - Can incorporate into the cost function
  - Works well when we have a lot of features, each that contributes a bit to predicting the label

- Can also address overfitting by eliminating features (either manually or via model selection)

Regularization

- Linear regression objective function

$$
\underbrace{J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2}_{\text{model fit to data}} + \underbrace{\frac{\lambda}{2} \sum_{j=1}^{d} \theta_j^2}_{\text{regularization}}
$$

– λ is the regularization parameter ($\lambda \geq 0$)

– No regularization on θ_0!

Understanding Regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2$$

- Note that $\sum_{j=1}^{d}\theta_j^2 = \|\boldsymbol{\theta}_{1:d}\|_2^2$
  - This is the magnitude of the feature coefficient vector!

- We can also think of this as:

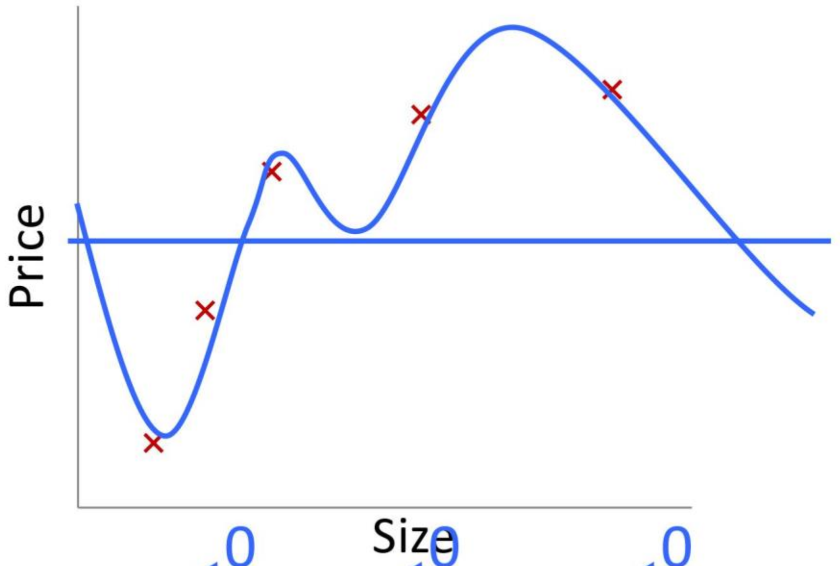
$$\sum_{j=1}^{d}(\theta_j - 0)^2 = \|\boldsymbol{\theta}_{1:d} - \vec{\mathbf{0}}\|_2^2$$

  - L_2 regularization pulls coefficients toward 0

Understanding Regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} \theta_j^2$$

- What happens if we set λ to be huge (e.g., 10^{10})?



$$\theta_0 + \cancelto{0}{\theta_1} x + \cancelto{0}{\theta_2} x^2 + \cancelto{0}{\theta_3} x^3 + \cancelto{0}{\theta_4} x^4$$

Regularized Linear Regression

- Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} \theta_j^2$$

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

- Gradient update:

$$\frac{\partial}{\partial \theta_0} J(\theta) \qquad \theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)$$

$$\frac{\partial}{\partial \theta_j} J(\theta) \qquad \theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right) x_j^{(i)} \underbrace{- \lambda \theta_j}_{\text{regularization}}$$

Regularized Linear Regression

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} \theta_j^2$$

$$\theta_0 \leftarrow \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)$$

$$\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right) x_j^{(i)} - \lambda \theta_j$$

- We can rewrite the gradient step as:

$$\theta_j \leftarrow \theta_j \left(1 - \alpha \lambda \right) - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right) x_j^{(i)}$$

Understanding Regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2$$

- Note that $\sum_{j=1}^{d}\theta_j^2 = \|\boldsymbol{\theta}_{1:d}\|_2^2$
  - This is the magnitude of the feature coefficient vector!

- We can also think of this as:

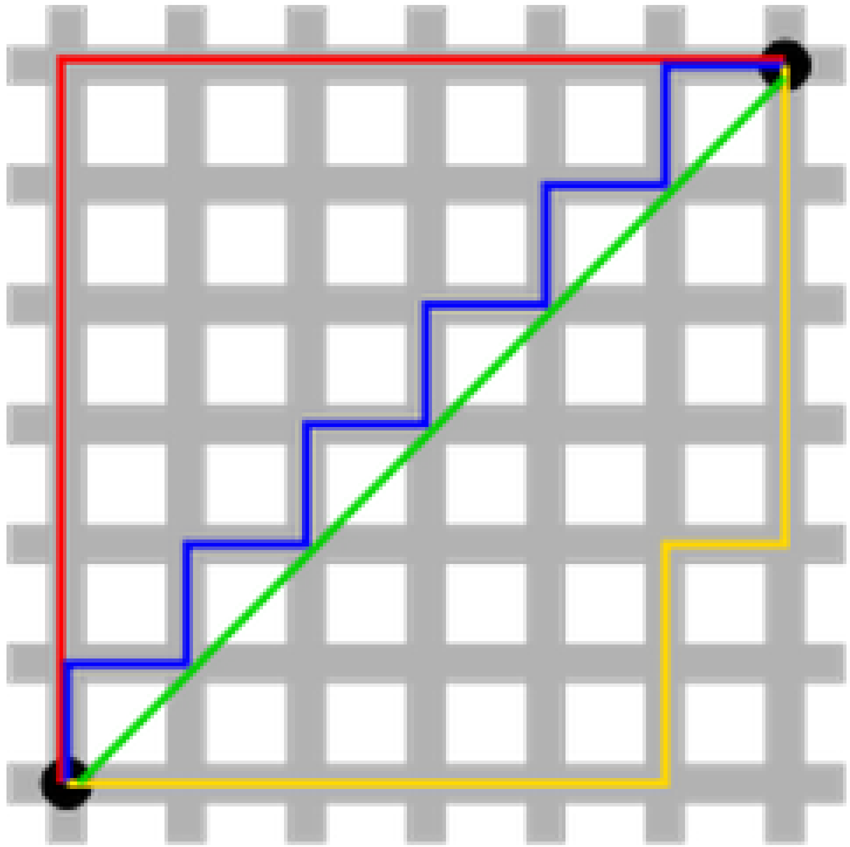
$$\sum_{j=1}^{d}(\theta_j - 0)^2 = \|\boldsymbol{\theta}_{1:d} - \vec{\mathbf{0}}\|_2^2$$

  - L_2 regularization pulls coefficients toward 0

L1 norm vs L2 norm

Lp-Norm

- Consider an n-dimensional vector:
  $x = [x_1, x_2, \ldots, x_n]$
- Define the p-Norm: $|x|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$
- L1-Norm is $|x|_1 = \sum_{i=1}^{n} |x_i|$
- L2-Norm is $|x|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$

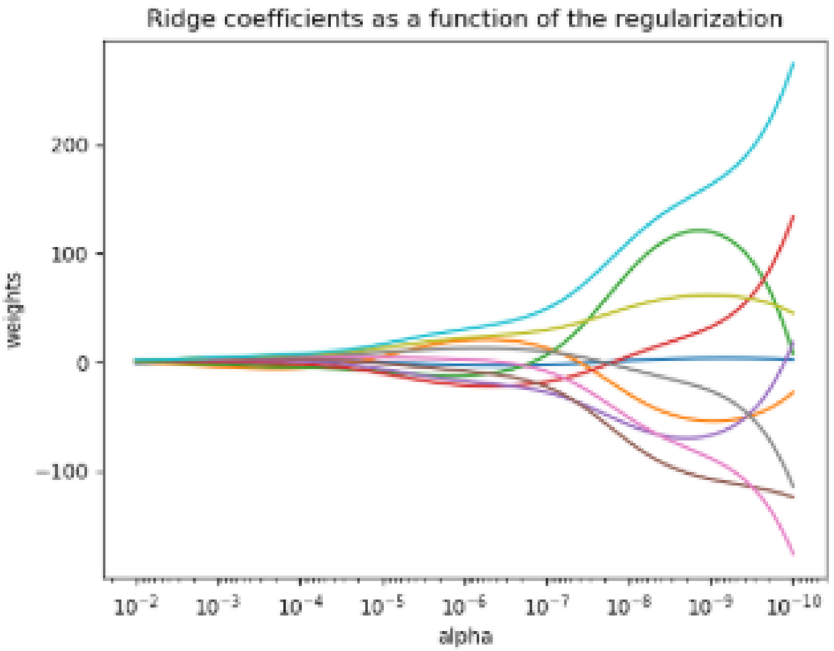


1.1.2. Ridge Regression

`Ridge` regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of the coefficients. The ridge coefficients minimize a penalized residual sum of squares:

$$\min_{w} ||Xw - y||_2^2 + \alpha ||w||_2^2$$

The complexity parameter $\alpha \geq 0$ controls the amount of shrinkage: the larger the value of α, the greater the amount of shrinkage and thus the coefficients become more robust to collinearity.



1.1.3. Lasso

The `Lasso` is a linear model that estimates sparse coefficients. It is useful in some contexts due to its tendency to prefer solutions with fewer non-zero coefficients, effectively reducing the number of features upon which the given solution is dependent. For this reason Lasso and its variants are fundamental to the field of compressed sensing. Under certain conditions, it can recover the exact set of non-zero coefficients (see Compressive sensing: tomography reconstruction with L1 prior (Lasso)).

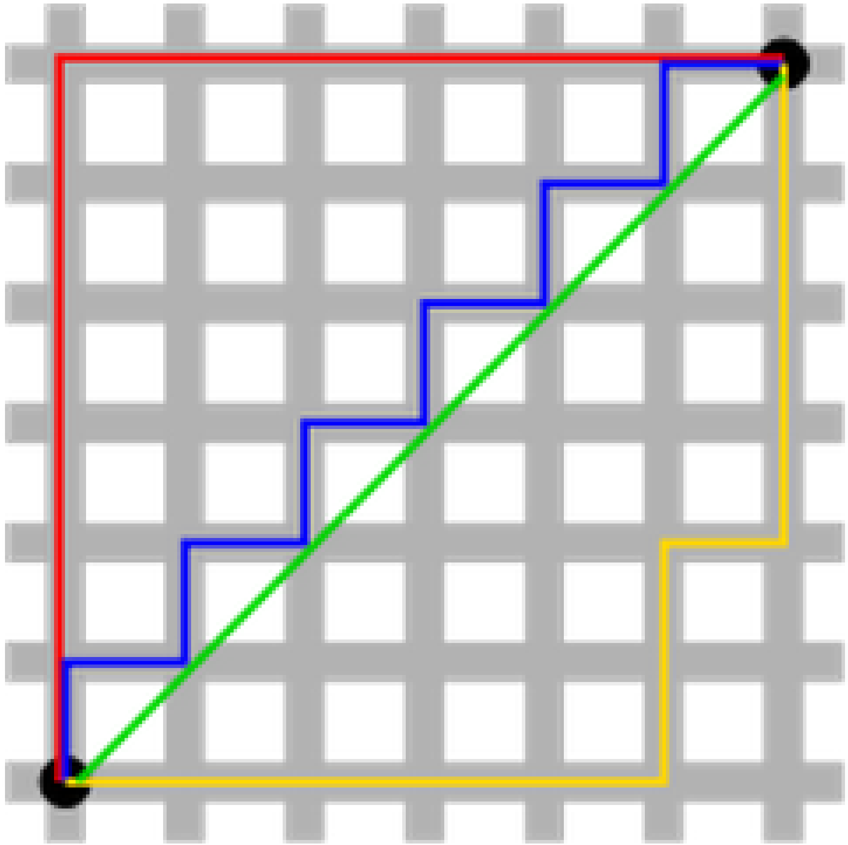
Mathematically, it consists of a linear model with an added regularization term. The objective function to minimize is:

$$\min_{w} \frac{1}{2n_{\text{samples}}} ||Xw - y||_2^2 + \alpha ||w||_1$$

The lasso estimate thus solves the minimization of the least-squares penalty with $\alpha ||w||_1$ added, where α is a constant and $||w||_1$ is the ℓ_1-norm of the coefficient vector.

The implementation in the class `Lasso` uses coordinate descent as the algorithm to fit the coefficients. See Least Angle Regression for another implementation:

L1 vs L2 norm



1.1.5. Elastic-Net

`ElasticNet` is a linear regression model trained with both ℓ_1 and ℓ_2-norm regularization of the coefficients. This combination allows for learning a sparse model where few of the weights are non-zero like `Lasso`, while still maintaining the regularization properties of `Ridge`. We control the convex combination of ℓ_1 and ℓ_2 using the `l1_ratio` parameter.

Elastic-net is useful when there are multiple features which are correlated with one another. Lasso is likely to pick one of these at random, while elastic-net is likely to pick both.

A practical advantage of trading-off between Lasso and Ridge is that it allows Elastic-Net to inherit some of Ridge's stability under rotation.

The objective function to minimize is in this case

$$\min_{w} \frac{1}{2n_{\text{samples}}} ||Xw - y||_2^2 + \alpha\rho||w||_1 + \frac{\alpha(1-\rho)}{2}||w||_2^2$$

The class `ElasticNetCV` can be used to set the parameters `alpha` (α) and `l1_ratio` (ρ) by cross-validation.

Regularization in Neural Network

Regularization for a Neural Network

$$J\left(w^{[1]}, b^{[1]}, \dots, w^{[l]}, b^{[l]}\right) = \frac{1}{m}\Sigma_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) + \frac{\lambda}{2m}\Sigma_{l=1}^{L} \left|\left|w^{[l]}\right|\right|_F^2$$

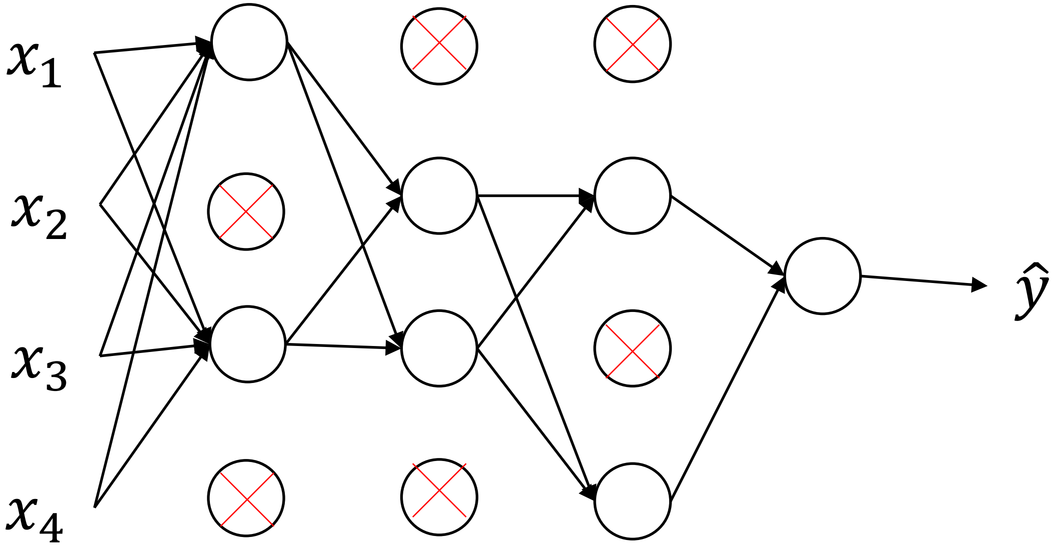
$$\left|\left|w^{[l]}\right|\right|_F^2 = \Sigma_{i=1}^{n^{[l-1]}} \Sigma_{j=1}^{n^{l}} \left(w_{ij}^{[l]}\right)^2$$

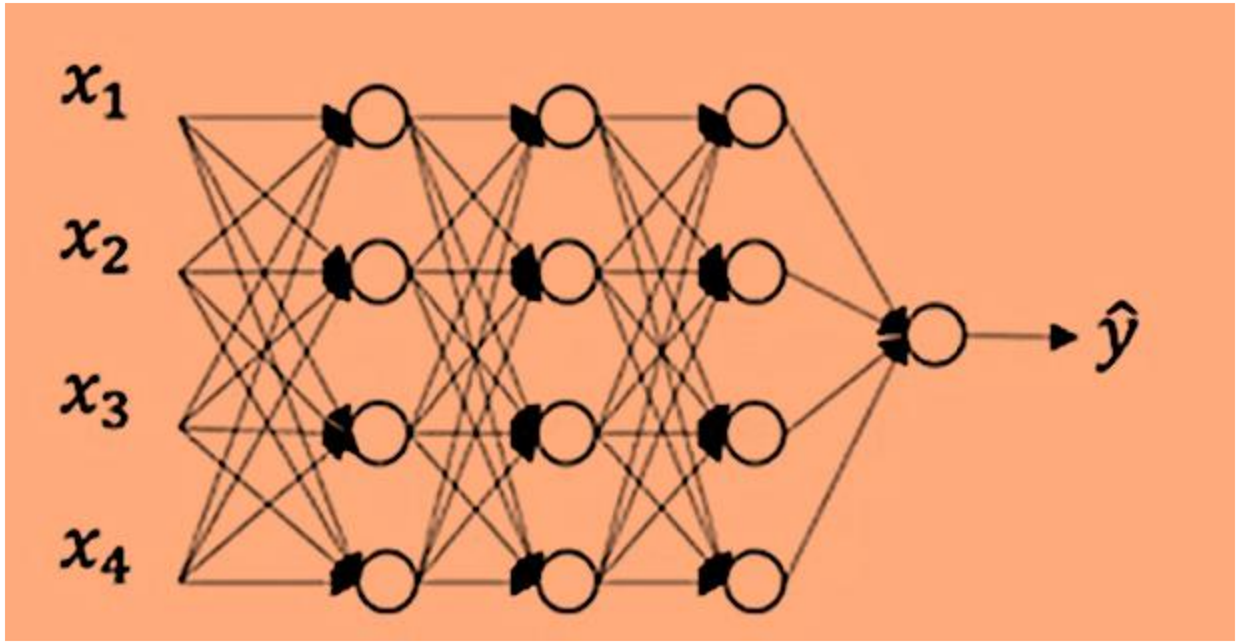
$w: (n^{[l-1]}, n^{[l]})$

Frobenius Norm

Weight decay $w^{[l]} = \left(1 - \frac{\alpha\lambda}{m}\right) w^{[l]} - \alpha dw^{[l]}$

Dropout regularization





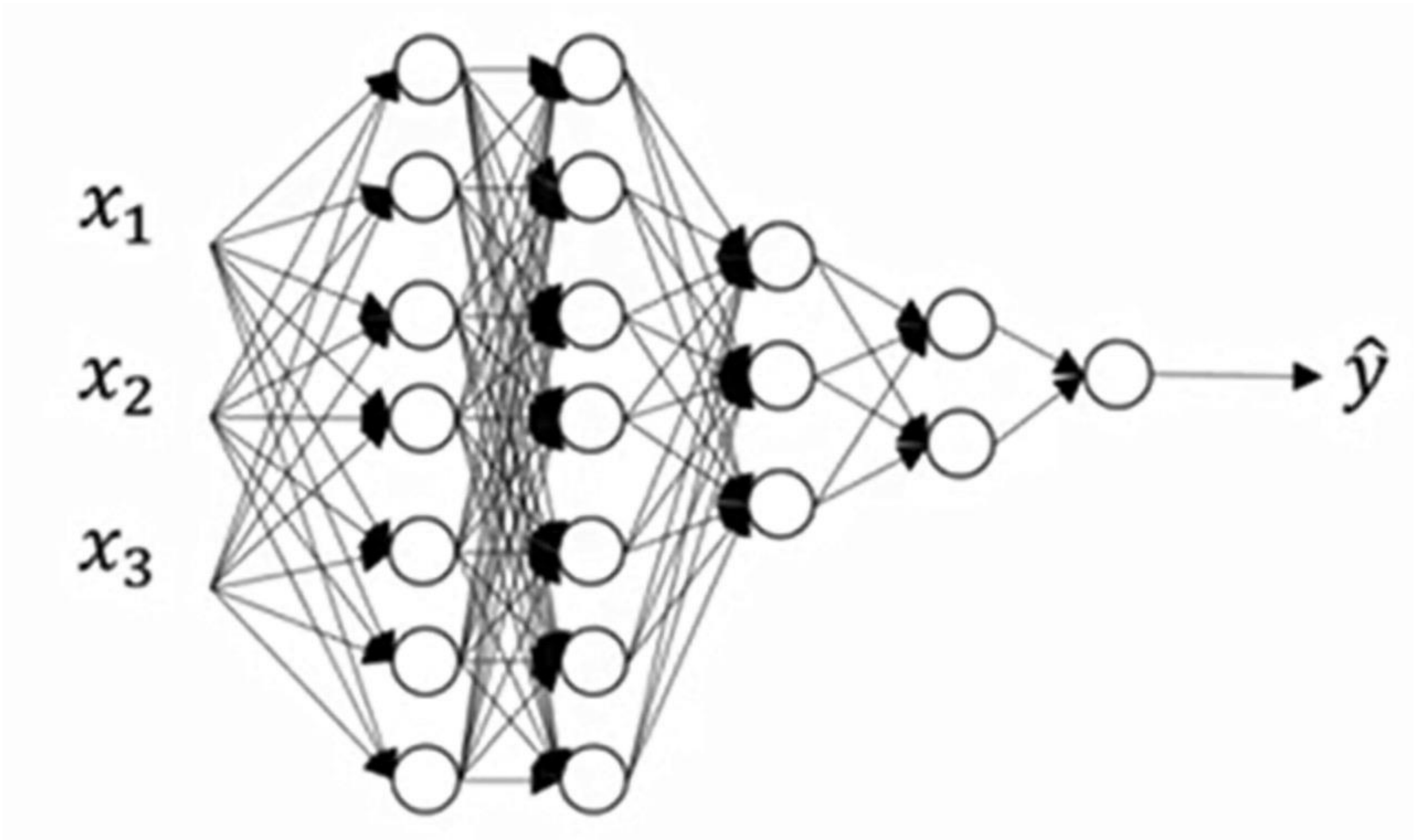
Drop out regularization: Prevents Overfitting

This technique has also become popular recently. We drop out some of the hidden units for specific training examples. Different hidden units may go off for different examples. In different iterations of the optimization the different units may be dropped randomly.

The drop outs can also be different for different layers. So, we can select specific layers which have higher number of units and may be contributing more towards overfitting; thus suitable for higher dropout rates.

For some of the layers drop-out can be 0, that means no dropout

Layer wise drop out



0 0.2 0.2 0 0 0

Drop out

- Drop out also help in spreading out the weights at all layers as the system will be reluctant to put more weight on some specific node. So it help in shrinking weights and has an adaptive effect on the weights.
- Dropout has a similar effect as L2 regularization for overfitting.
- We don’t use dropout for test examples
- We also need to bump up the values at the output of each layer corresponding to the dropout

Data Augmentation

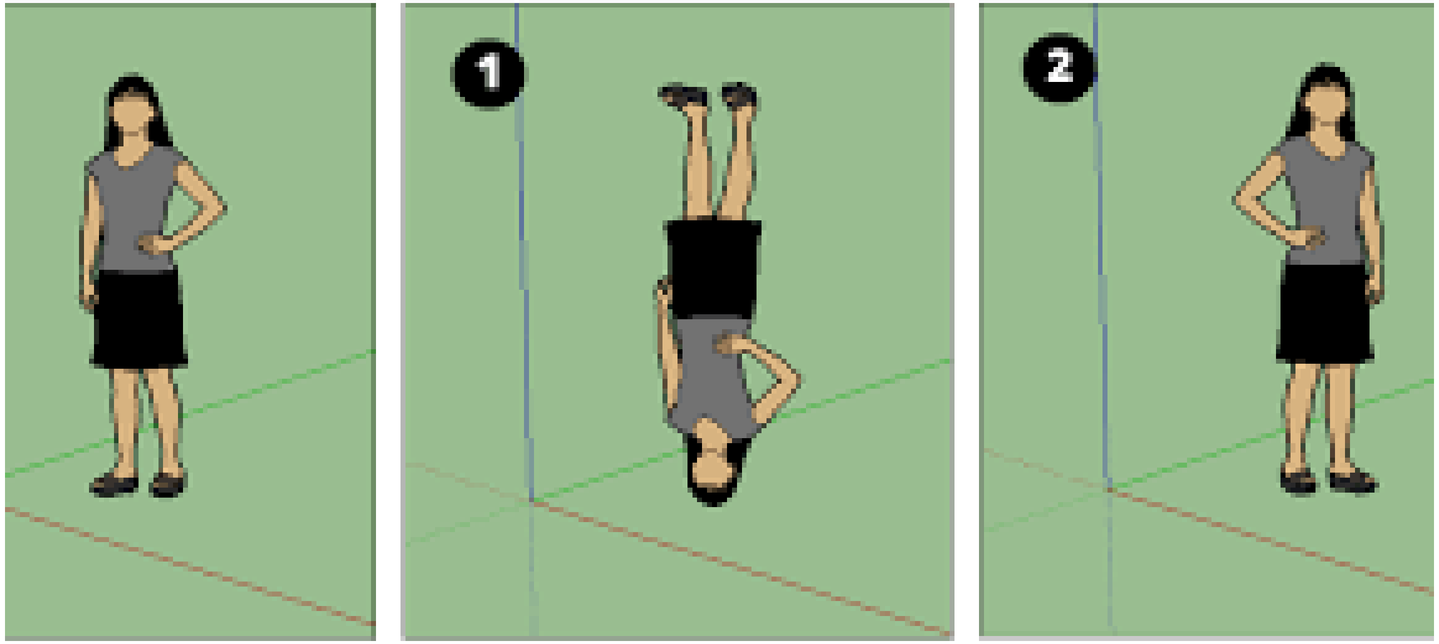
More training data is one more solution for overfitting.

As getting additional data may be expensive and may not be possible

Flipping of all the images can be one of the ways to increase your data.

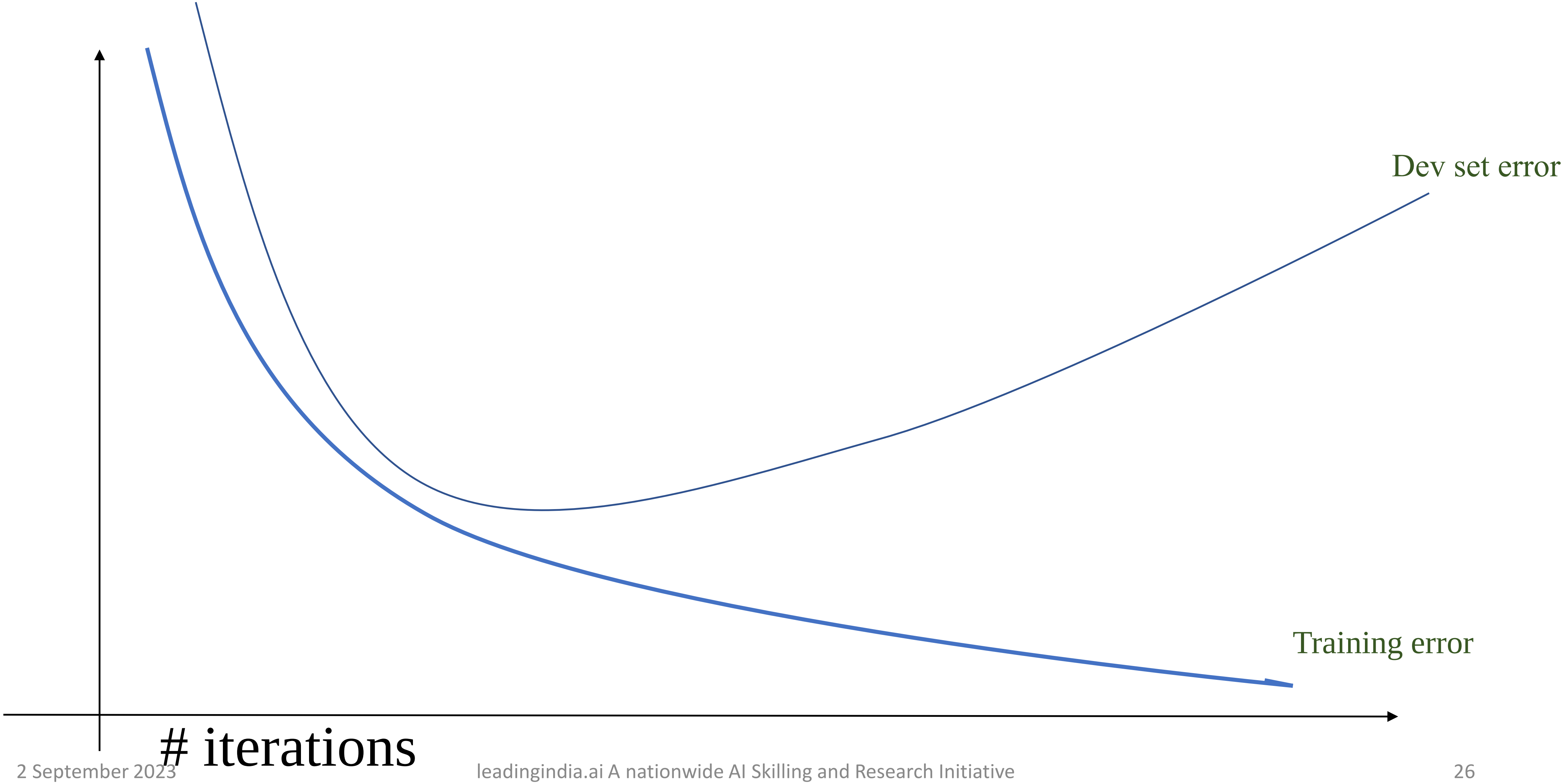
Randomly zooming in and zooming out can be another way

Distorting some of the images based on your application may be another way to increase your data.



Data Augmentation

Early stopping



Early Stopping

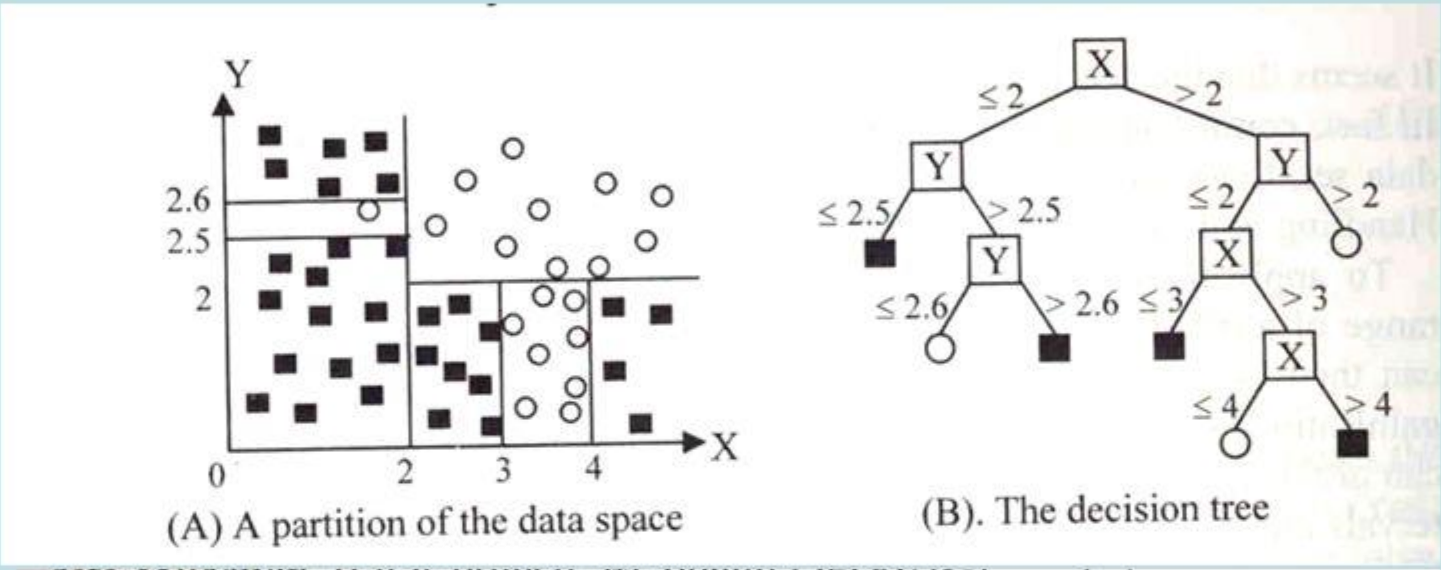
Sometime dev set error goes down and then it start going up. So you may decide to stop where the curve has started taking a different turn.

By stopping halfway we also reduce number of iterations to train and the computation time.

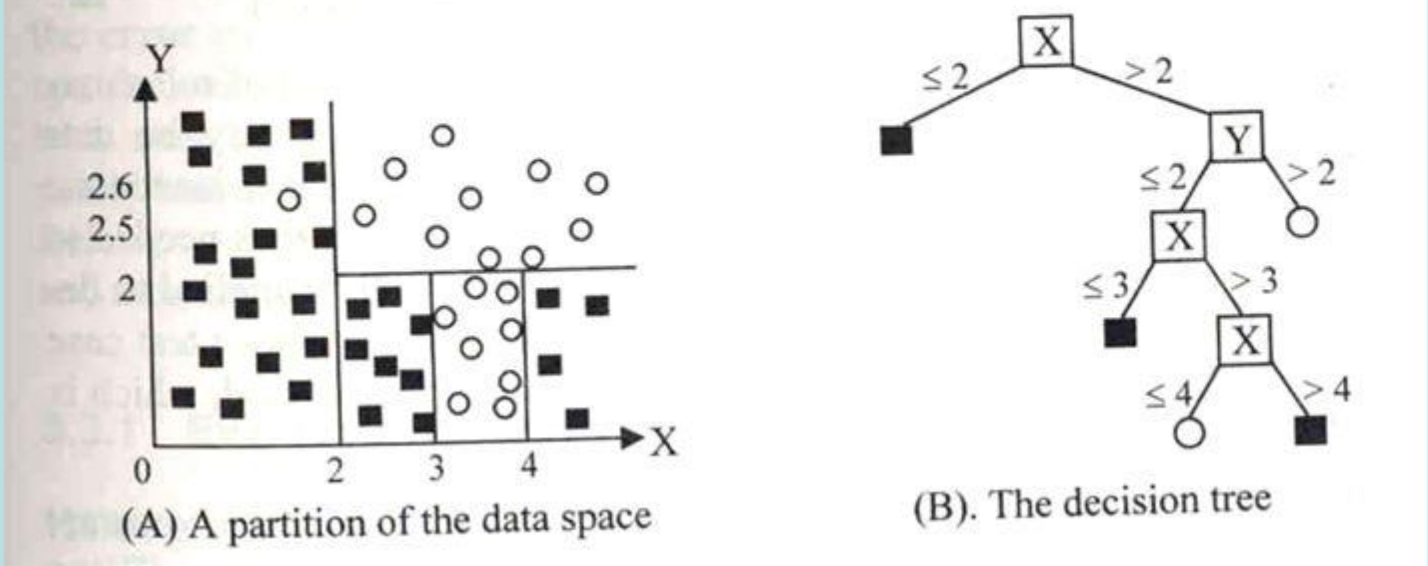
Early stopping does not go fine with orthogonalization because it contradicts with our original objective of optimizing(w,b) to the minimum possible cost function.

We are stopping the process of optimization in between to take care of the overfitting which is a different objective then optimization.

Overfitting (Example)



(A) A partition of the data space (B). The decision tree



(A) A partition of the data space (B). The decision tree

Overfitting in Decision Trees

- Many kinds of "noise" can occur in the examples:
  - Two examples have same attribute/value pairs, but different classifications
  - Some values of attributes are incorrect because of errors in the data acquisition process or the preprocessing phase
  - The instance was labeled incorrectly (+ instead of -)

- Also, some attributes are irrelevant to the decision-making process
  - e.g., color of a die is irrelevant to its outcome

Based on Slide from M. desJardins & T. Finin

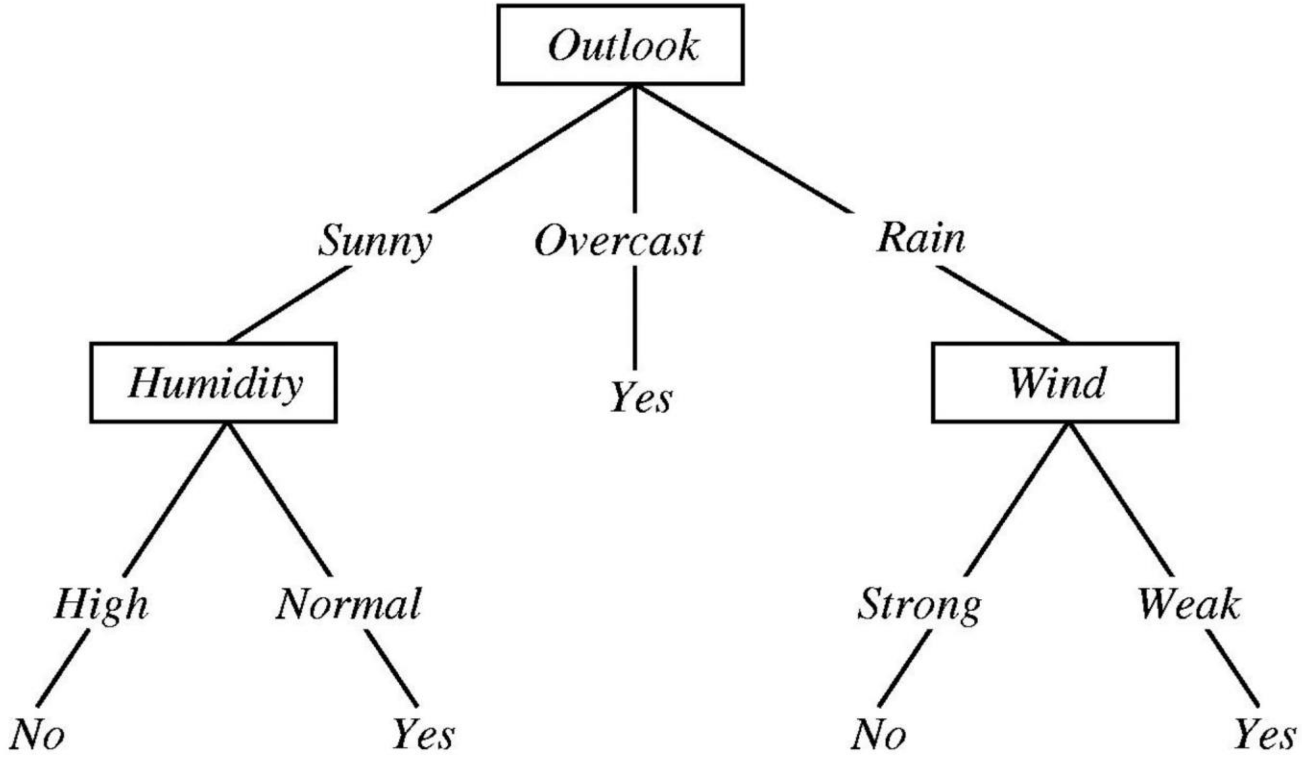
Overfitting in Decision Trees

- Irrelevant attributes can result in *overfitting* the training example data
  - If hypothesis space has many dimensions (large number of attributes), we may find **meaningless regularity** in the data that is irrelevant to the true, important, distinguishing features

- If we have too little training data, even a reasonable hypothesis space will 'overfit'

Overfitting in Decision Trees

Consider adding a **noisy** training example to the following tree:

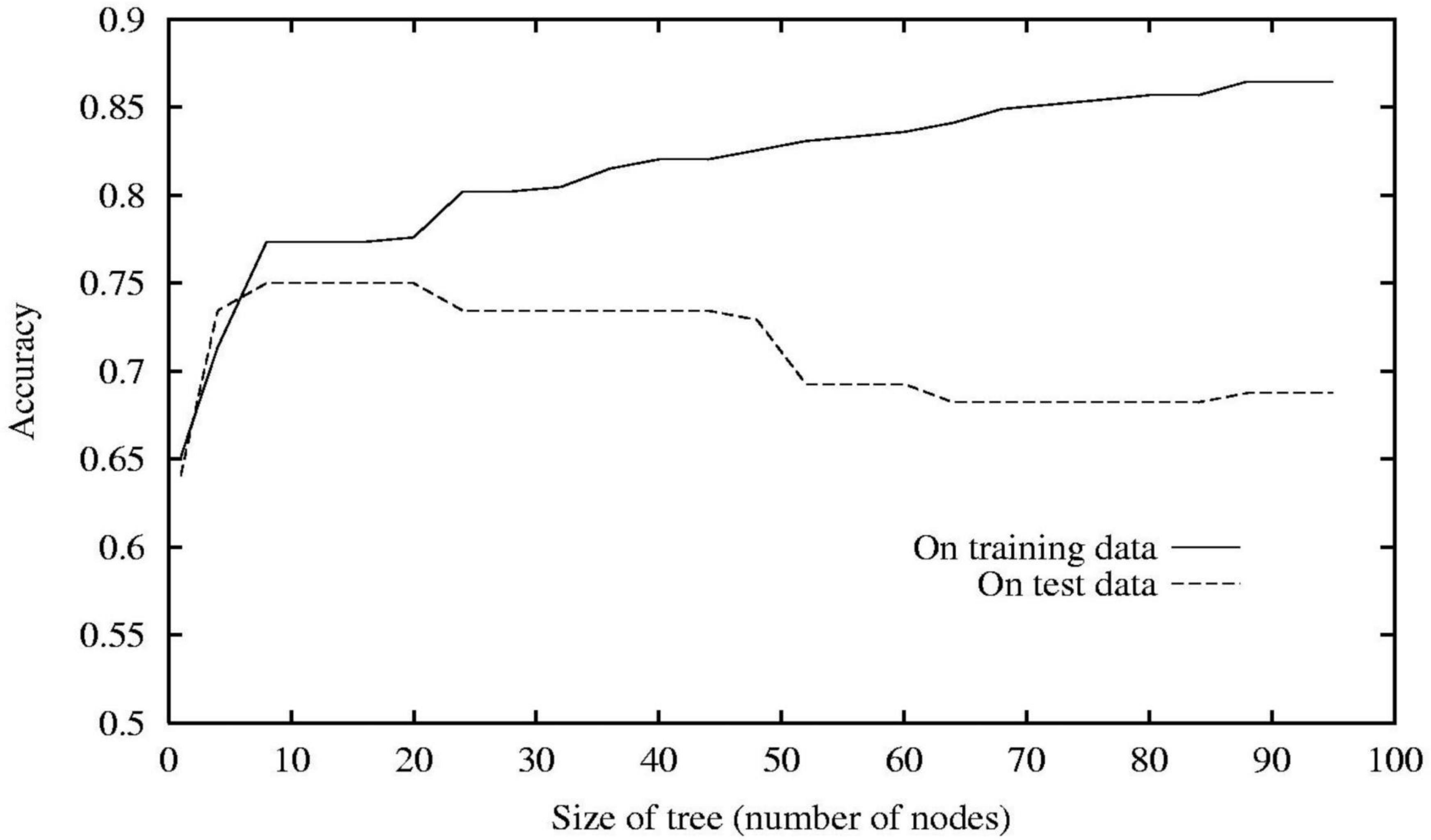


What would be the effect of adding:

<outlook=sunny, temperature=hot, humidity=normal, wind=strong, playTennis=No> ?

Based on Slide by Pedro Domingos

Overfitting in Decision Trees



Slide by Pedro Domingos

Avoiding Overfitting in Decision Trees

How can we avoid overfitting?

- Stop growing when data split is not statistically significant
- Acquire more training data
- Remove irrelevant attributes (manual process – not always possible)
- **Grow full tree, then post-prune**

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation data set
- Add complexity penalty to performance measure
  (heuristic: simpler is better)

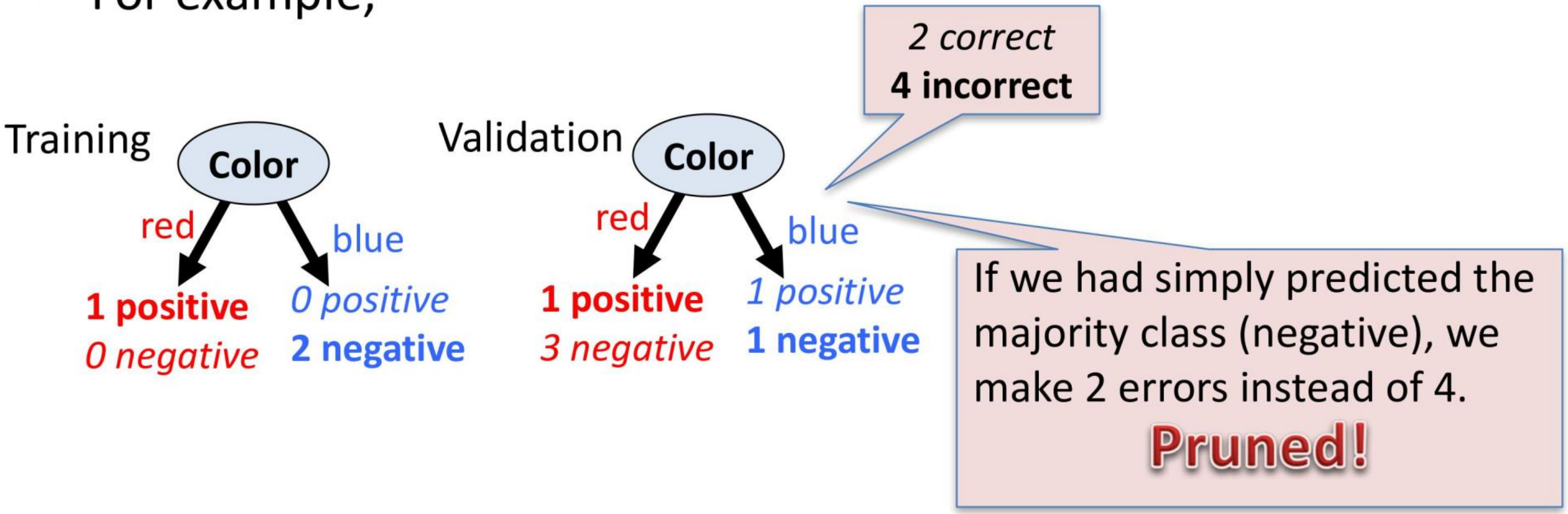
Based on Slide by Pedro Domingos

DT in Sklearn

class sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, ccp_alpha=0.0)

Pruning Decision Trees

- Pruning of the decision tree is done by replacing a whole subtree by a leaf node.
- The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf.
- For example,



Based on Example from M. desJardins & T. Finin


MEMEES.IN
தமிழ்
MEMES CREATOR
memees.in