Machine Learning: The Basics

Dripta Mj
Department of Mathematics
Ramakrishna Mission Vivekananda Educational and Research Institute
Belur Math, India

Machine Learning is everywhere . . . .

Astronomy

Social Networks

Healthcare

Banking

Genomics

Weather predictions

● ● ● ●

Dogs and Cats …..





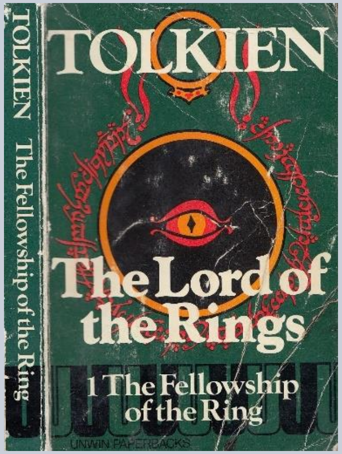




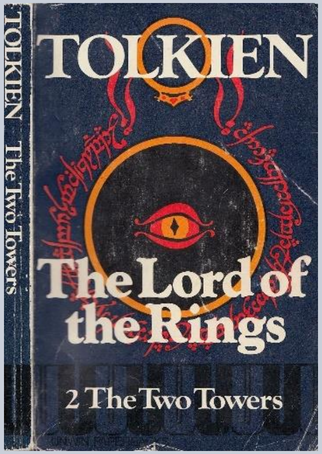

?

?

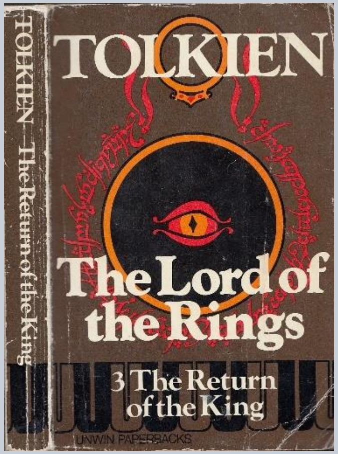
Product recommendation

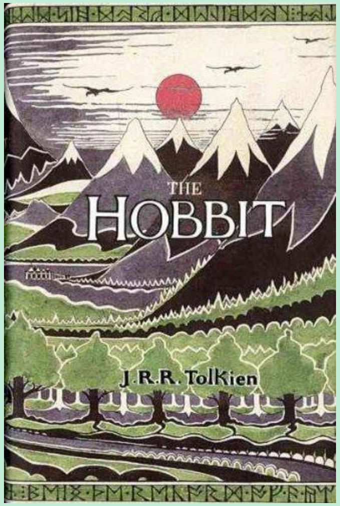


+



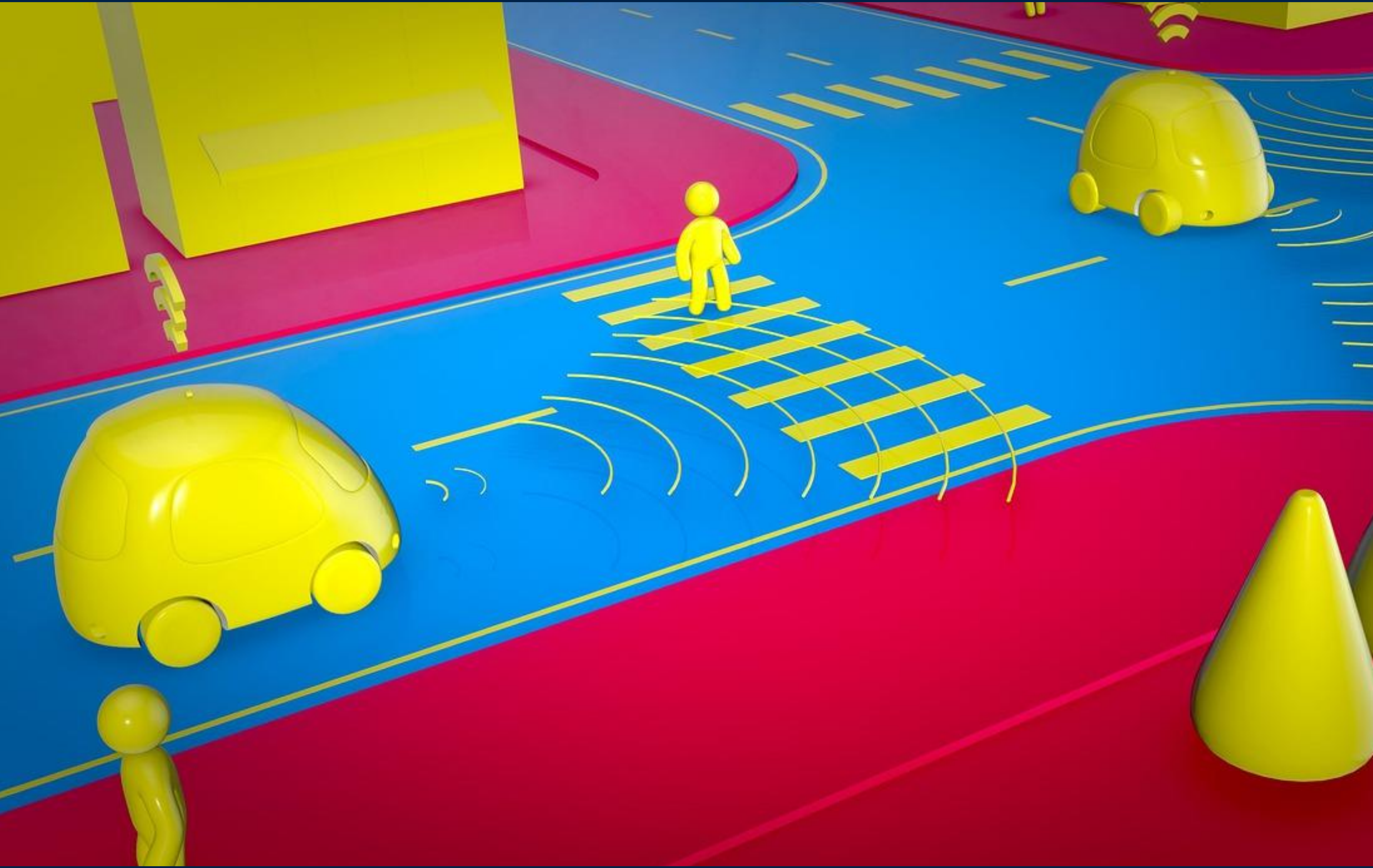
+





Images from *amazon.com*

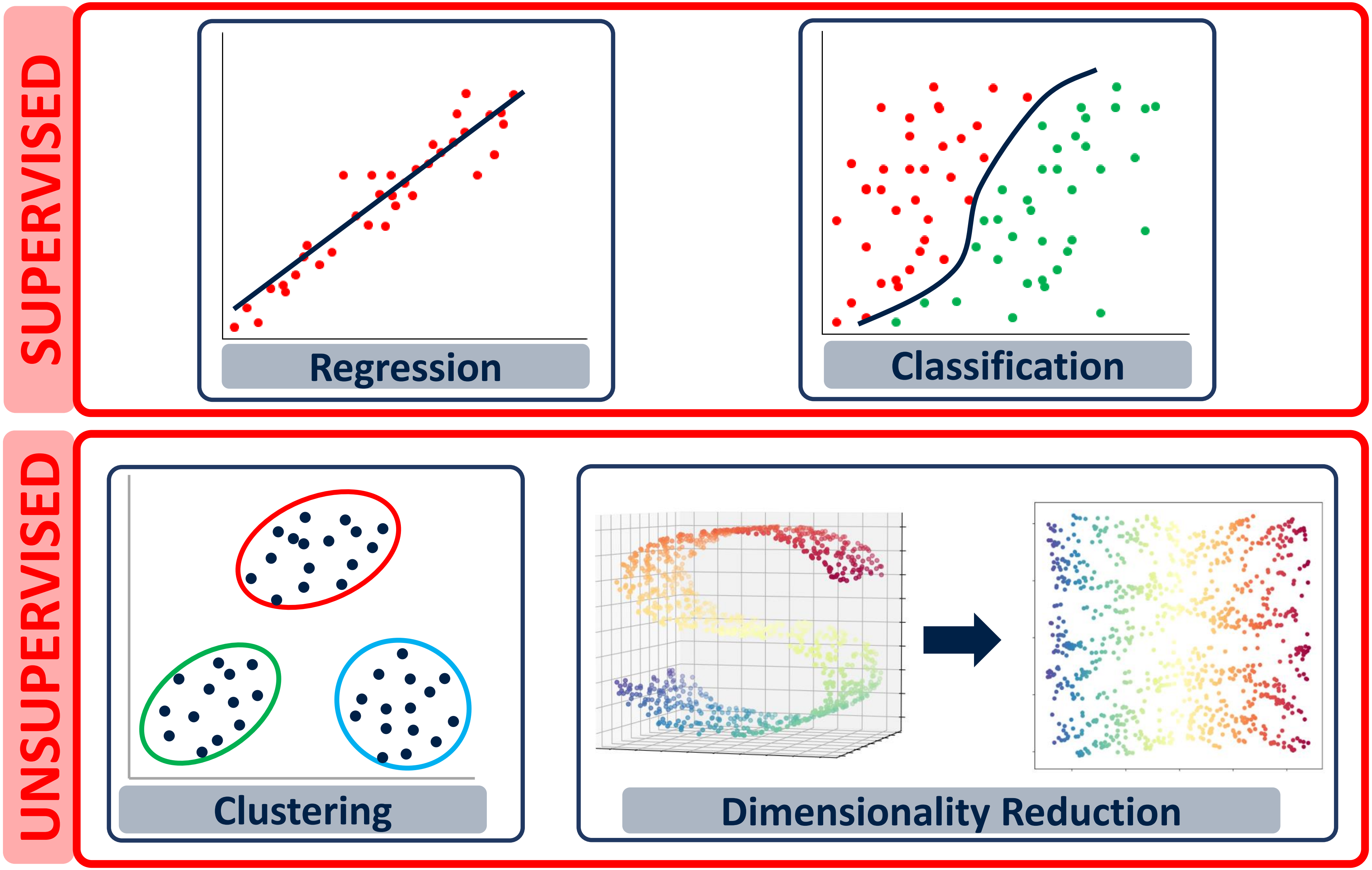
Autonomous vehicles



Creativity

Figure source: Gatys, Ecker and Bethge, Image style transfer using convolutional neural networks, CVPR 2016.

Machine Learning



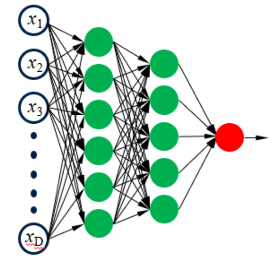
Some key components

Data pre-processing

x_1	x_2	x_3	y
2.2	0.8	2.7	1
4.9	3.1	1.6	-1

- Data cleaning
- Training-test data splitting
- Feature engineering ……

ML Model



Linear Regression

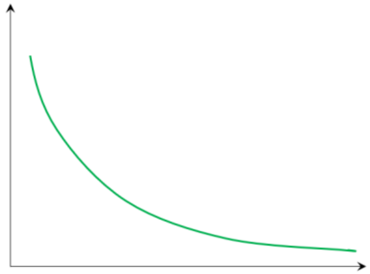
*k*NN | SVM | Decision Tree

Neural Network | ……..

Training

- Loss function
- Optimization algorithm
- Regularization ……

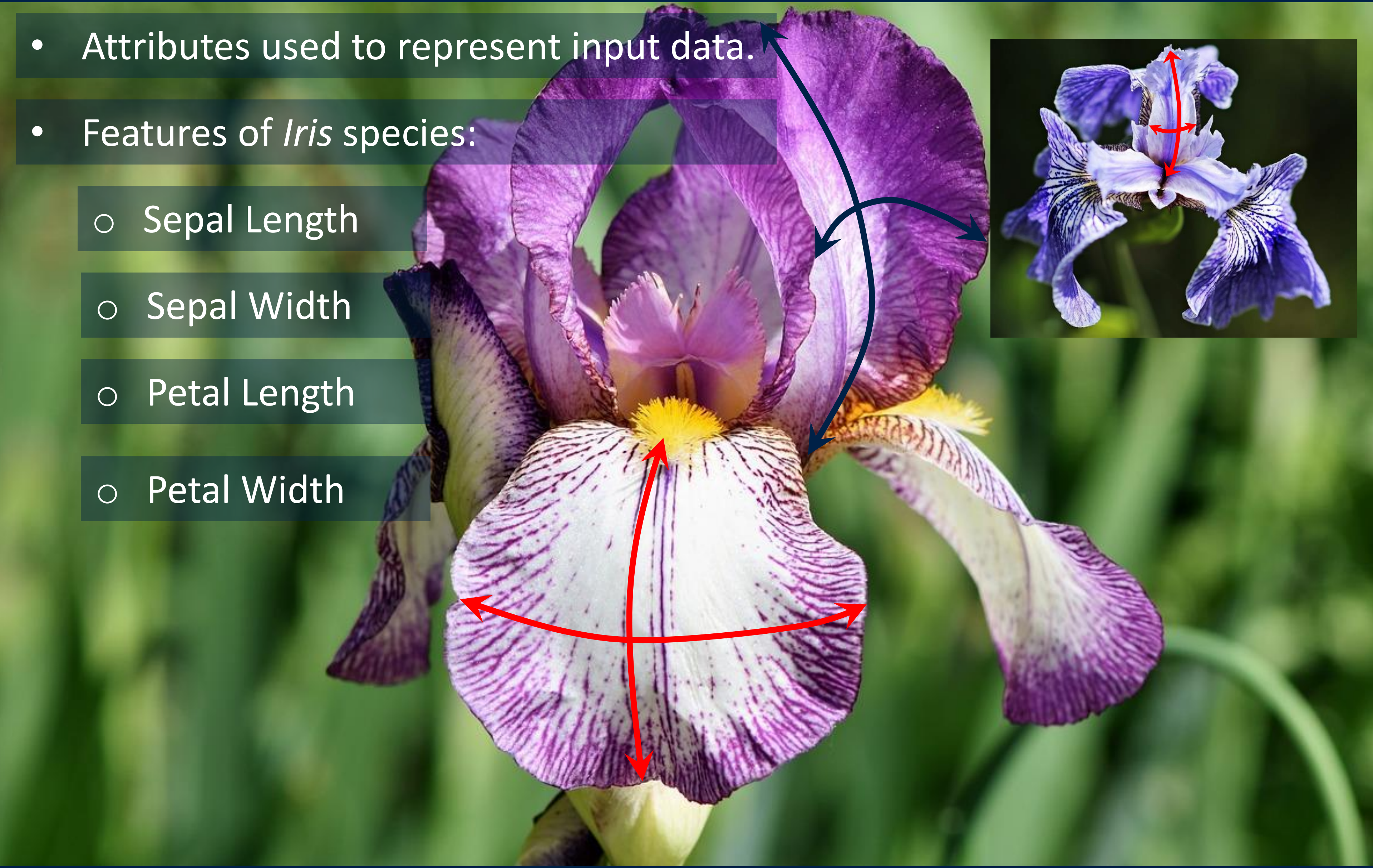
Evaluation



- Generalization error
- Cross-validation
- Metric ……….

Features

- Attributes used to represent input data.
- Features of *Iris* species:
  - Sepal Length
  - Sepal Width
  - Petal Length
  - Petal Width



Iris dataset

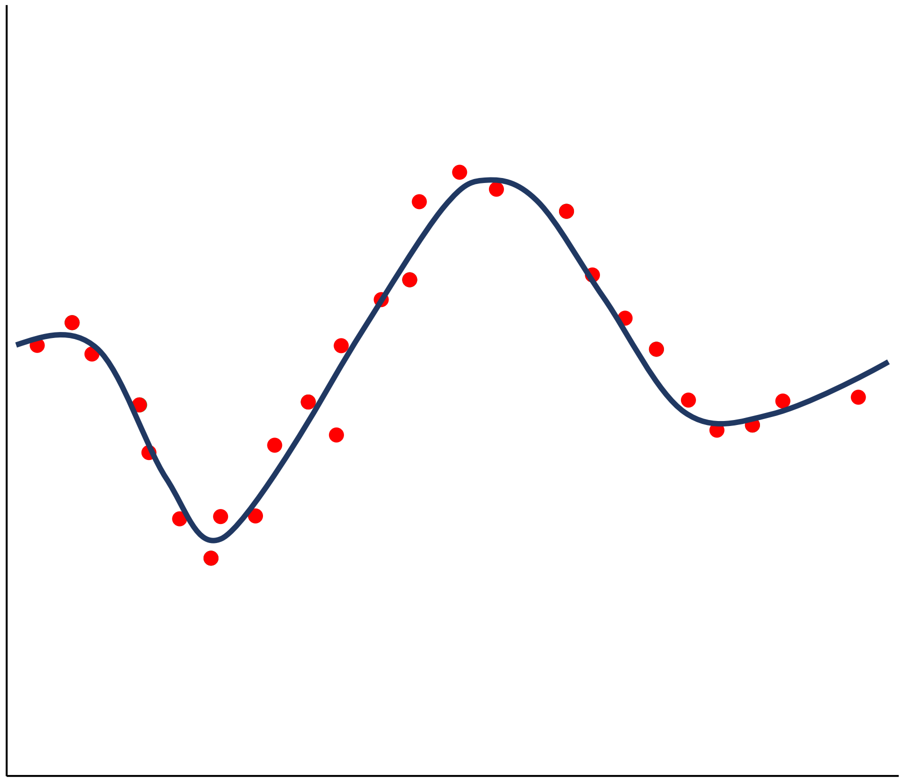
INPUTS

Sepal Length (cm)	Sepal Width (cm)	Petal Length (cm)	Petal Width (cm)
5.1	3.5	1.4	0.2
4.9	3	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5	3.6	1.4	0.2
5.4	3.9	1.7	0.4
4.6	3.4	1.4	0.3
5	3.4	1.5	0.2
4.4	2.9	1.4	0.2
.	.	.	.
.	.	.	.

OUTPUTS

Species	
Iris Setosa	0
Iris Virginica	1
Iris Versicolor	2

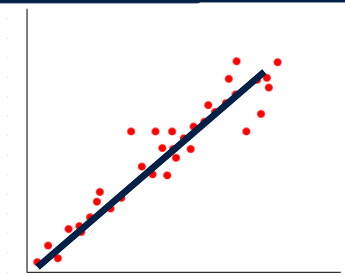
Training and Test data



- Training data: Used for training the ML algorithm.
- Test data: Used for assessing the performance of the ML algorithm.

Loss function

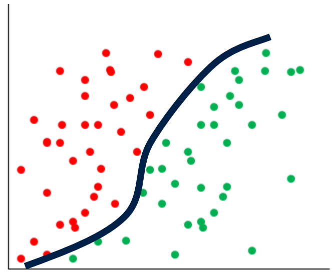
REGRESSION



Squared loss:

$$\mathcal{L}(\mathbf{y}^{(n)}, \mathbf{y}^{*(n)}) = \frac{1}{2}\sum_{j=1}^{J}\left(y_j^{(n)} - y_j^{*(n)}\right)^2$$

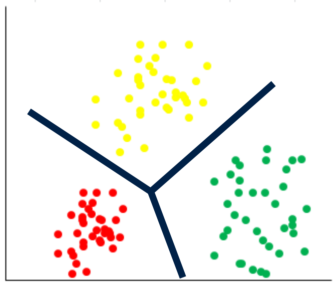
BINARY CLASSIFICATION



Binary cross-entropy loss:

$$\mathcal{L}(y^{(n)}, y^{*(n)}) = -y^{(n)}\log(y^{*(n)}) - (1 - y^{(n)})\log(1 - y^{*(n)})$$

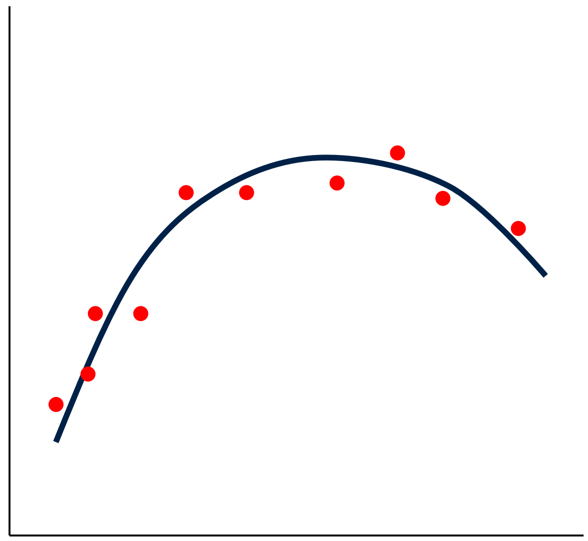
MULTI-CLASS CLASSIFICATION



Cross-entropy loss:

$$\mathcal{L}(\mathbf{y}^{(n)}, \mathbf{y}^{*(n)}) = -\sum_{j=1}^{J} y_j^{(n)}\log y_j^{*(n)}$$

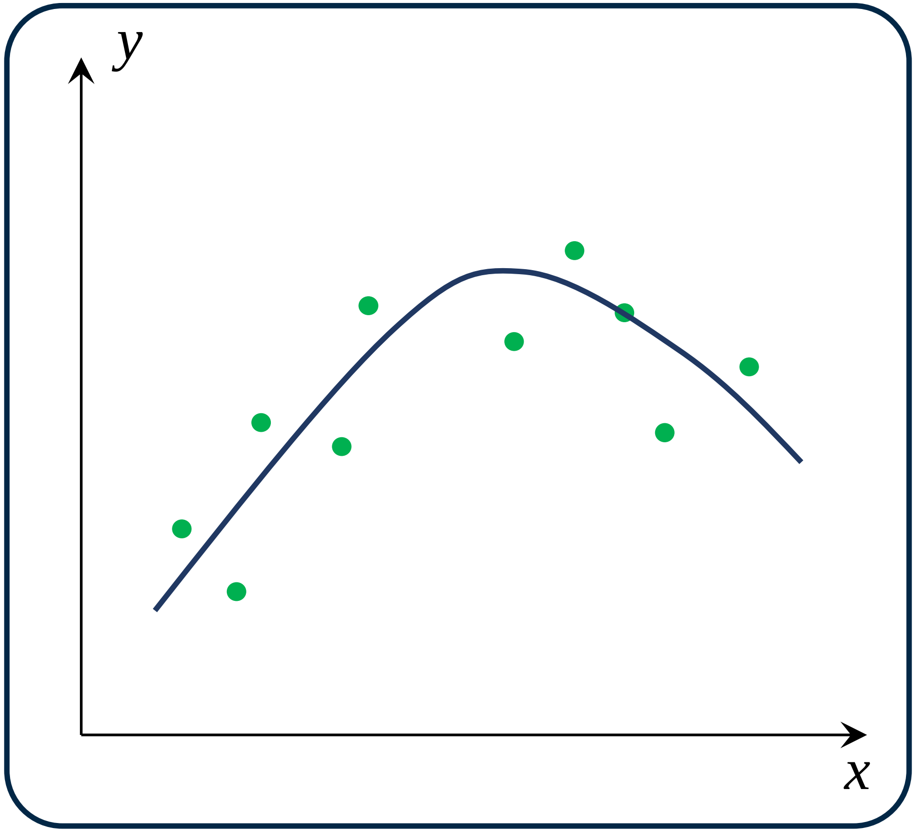
Generalization



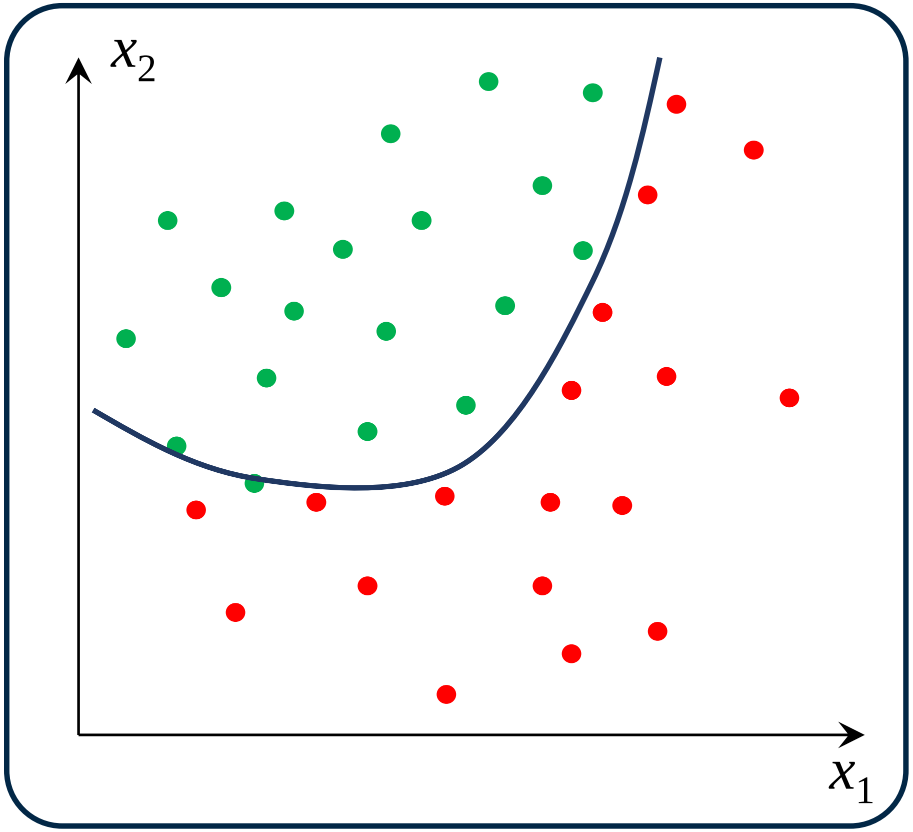
- Larger class of functions $\rightarrow$ more complexity of the hypothesis class $\mathcal{C}(\mathbb{H})$.
- Objective: Good prediction at unobserved locations $\rightarrow$ good **generalization**.

Generalization

REGRESSION

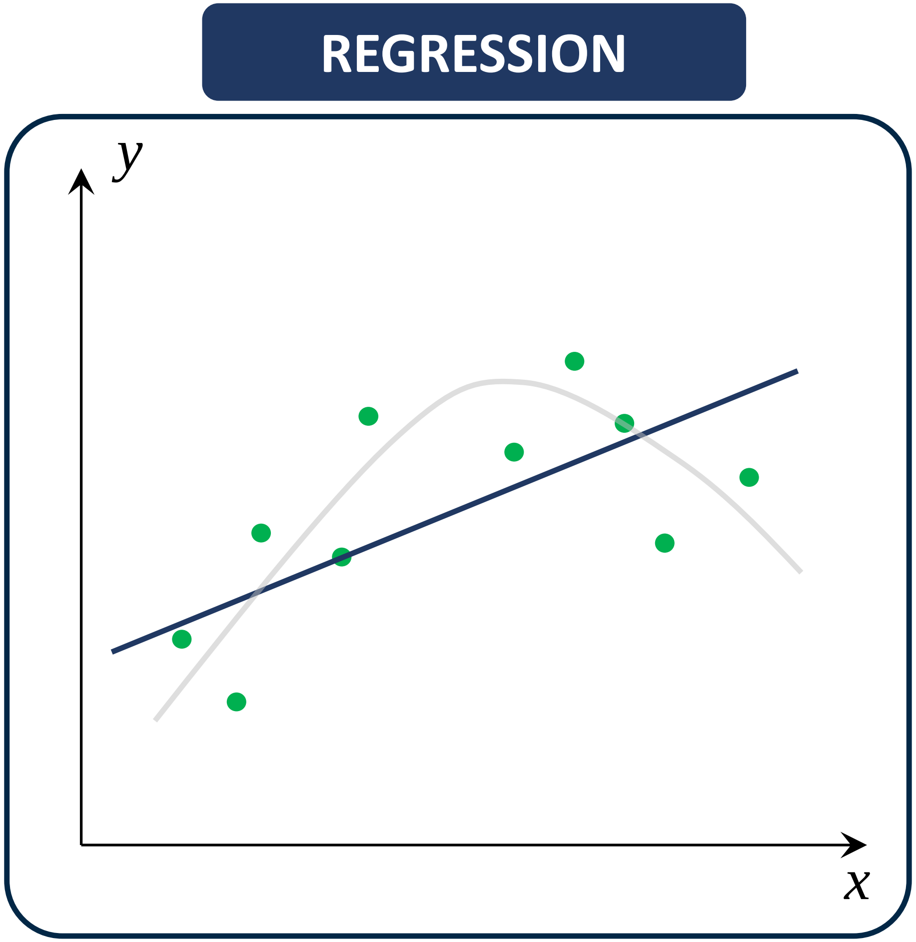


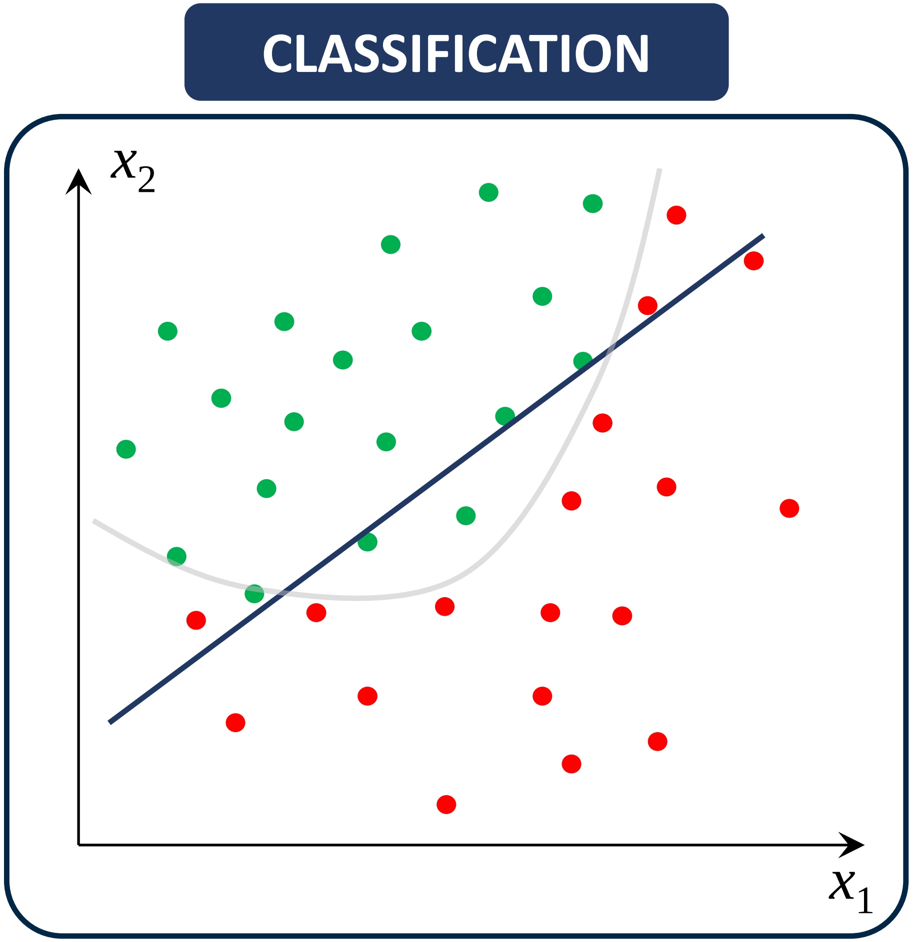
CLASSIFICATION



Figures for illustration only.

Simple models

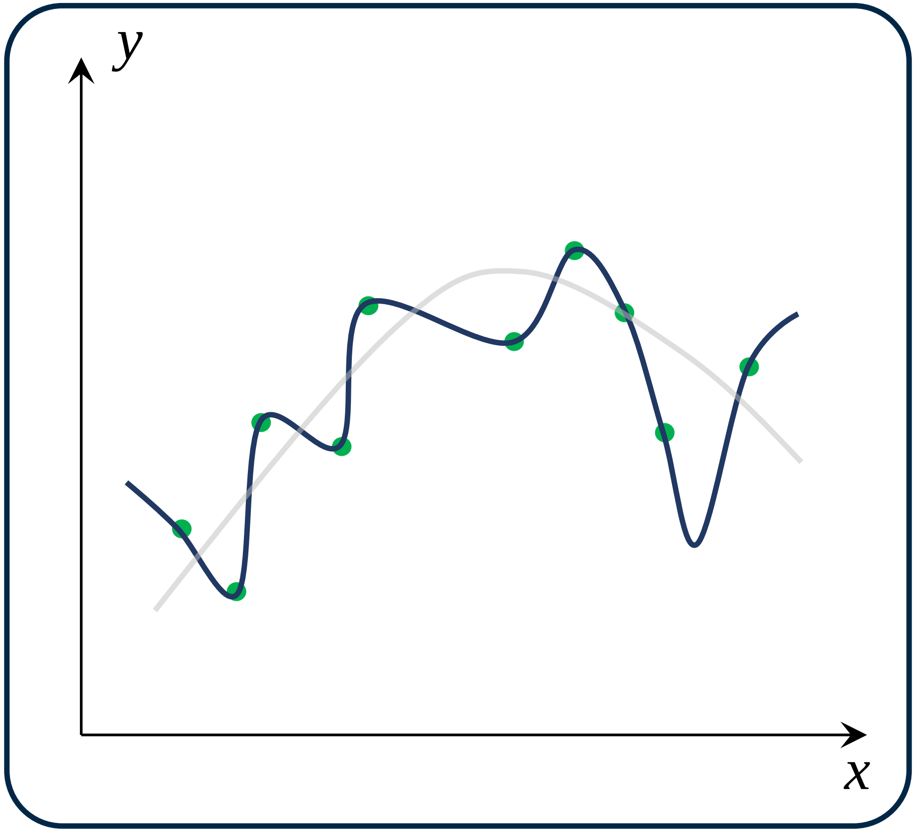




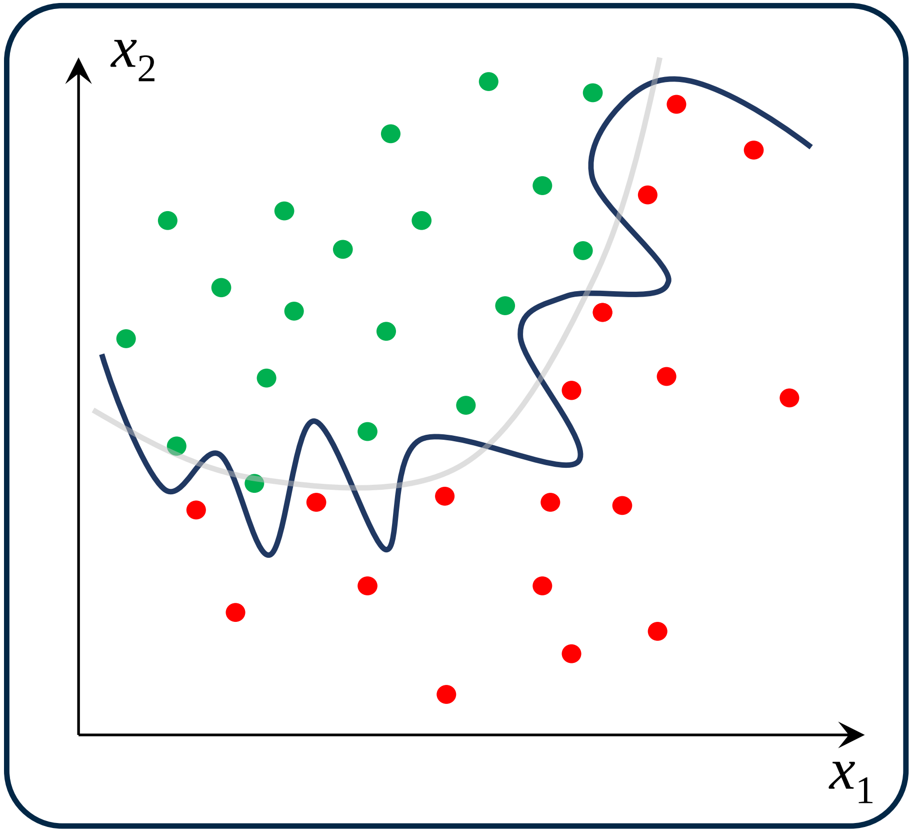
Figures for illustration only.

Complex models

REGRESSION

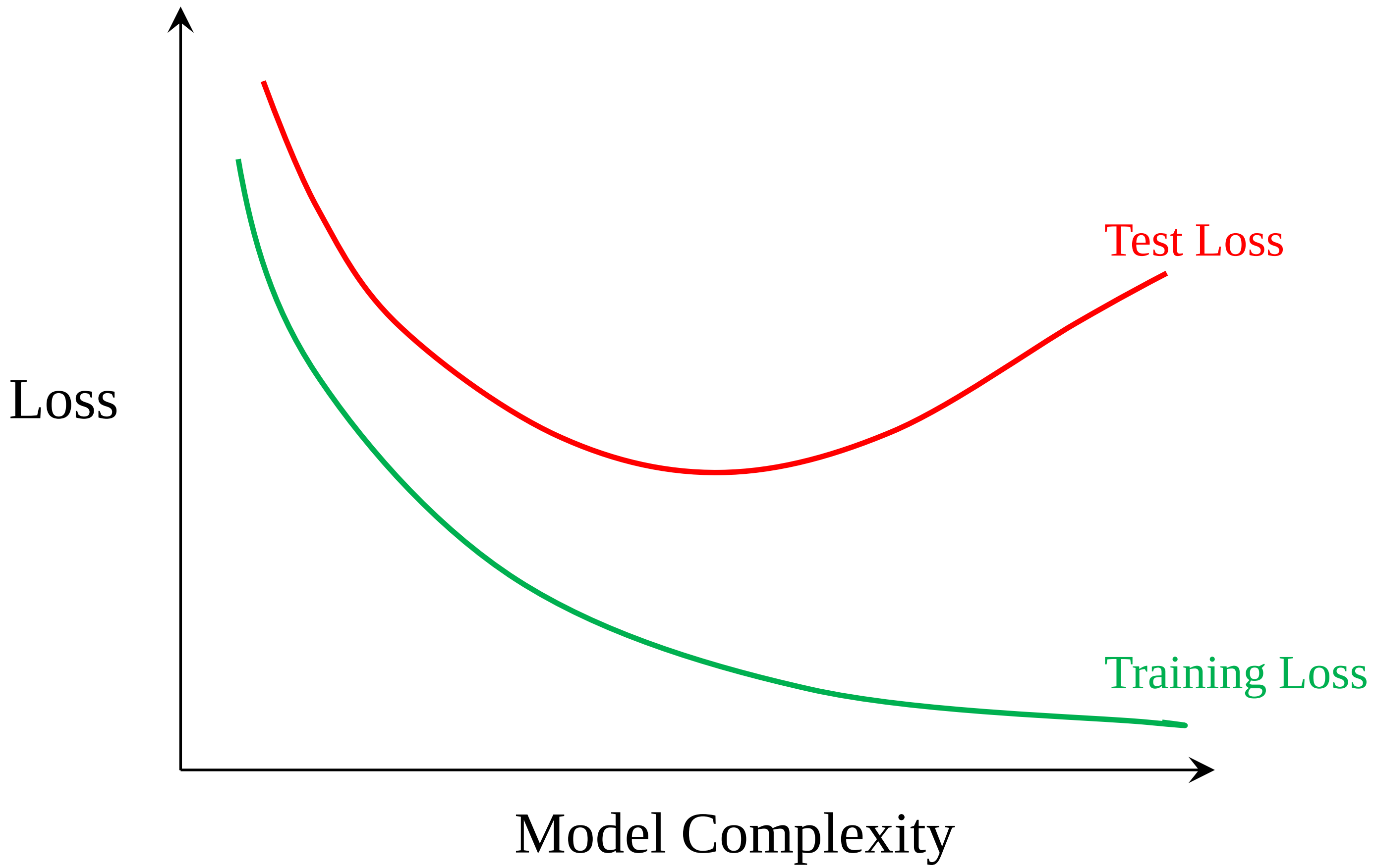


CLASSIFICATION



Figures for illustration only.

Loss vs complexity



Figures for illustration only.

Bias-variance decomposition

- Dataset: $\mathcal{D} = \left\{ \left(\mathbf{x}^{(1)}, y^{(1)}\right), \left(\mathbf{x}^{(2)}, y^{(2)}\right), ..., \left(\mathbf{x}^{(N)}, y^{(N)}\right) \right\}$
- Let $g_{\mathcal{D}}$ be the hypothesis which is fit to a particular training dataset $\mathcal{D}$
- Want to compute the expected prediction error at an arbitrary test point with input $\mathbf{x}$ and output y: $\mathbb{E}_{\mathbf{x},y,\mathcal{D}}\left[(g_{\mathcal{D}}(\mathbf{x}) - y)^2\right]$.
- Mean prediction of the machine learning algorithm:

$$\overline{g}(\mathbf{x}) = \mathbb{E}_{\mathcal{D}}\left[g_{\mathcal{D}}(\mathbf{x})\right]$$

- So determining the value of $\overline{g}(\mathbf{x})$ involve
  - generating different training datasets $(\mathcal{D})$,
  - training separate functions $(g_{\mathcal{D}})$ for every generated dataset,
  - making predictions at an arbitrary test point $\mathbf{x}$ with all trained functions,
  - and finally, averaging over all the predictions.
- Let $\overline{y}(\mathbf{x})$ be the expected value of the output at $\mathbf{x}$, i.e. $\overline{y}(\mathbf{x}) = \mathbb{E}_{y|\mathbf{x}}[y]$.

Bias-variance decomposition

- Dataset: $\mathcal{D} = \left\{ \left(\mathbf{x}^{(1)}, y^{(1)}\right), \left(\mathbf{x}^{(2)}, y^{(2)}\right), ..., \left(\mathbf{x}^{(N)}, y^{(N)}\right) \right\}$
- Let $g_{\mathcal{D}}$ be the hypothesis which is fit to a particular training dataset $\mathcal{D}$
- Want to compute the expected prediction error at an arbitrary test point with input $\mathbf{x}$ and output y: $\mathbb{E}_{\mathbf{x},y,\mathcal{D}}\left[(g_{\mathcal{D}}(\mathbf{x}) - y)^2\right]$.
- Mean prediction of the machine learning algorithm:

$$\overline{g}(\mathbf{x}) = \mathbb{E}_{\mathcal{D}}\left[g_{\mathcal{D}}(\mathbf{x})\right]$$

- So determining the value of $\overline{g}(\mathbf{x})$ involve
  - generating different training datasets ($\mathcal{D}$),
  - training separate functions ($g_{\mathcal{D}}$) for every generated dataset,
  - making predictions at an arbitrary test point $\mathbf{x}$ with all trained functions,
  - and finally, averaging over all the predictions.
- Let $\overline{y}(\mathbf{x})$ be the expected value of the output at $\mathbf{x}$, i.e. $\overline{y}(\mathbf{x}) = \mathbb{E}_{y|\mathbf{x}}[y]$.

Bias-variance decomposition

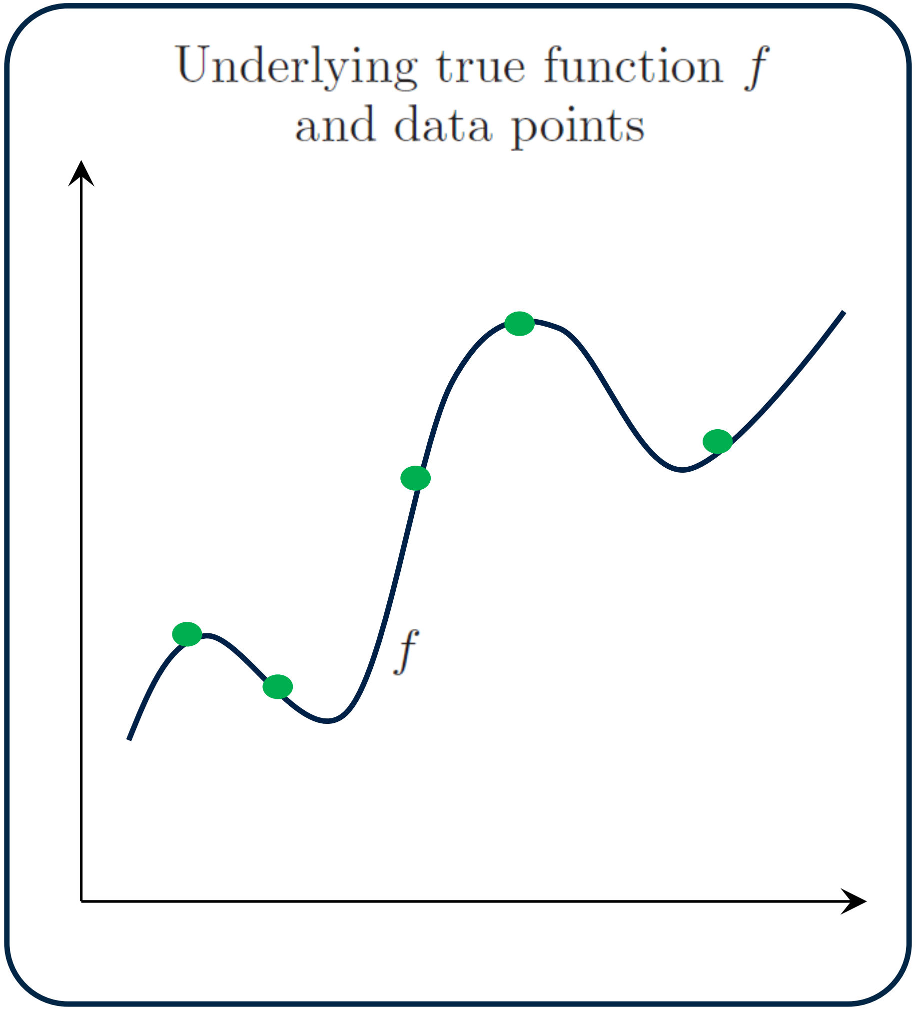
$$\mathbb{E}_{\mathbf{x},y,\mathcal{D}}\left[\left(g_{\mathcal{D}}(\mathbf{x})-y\right)^2\right]=\underbrace{\mathbb{E}_{\mathbf{x},\mathcal{D}}\left[\left(g_{\mathcal{D}}(\mathbf{x})-\overline{g}(\mathbf{x})\right)^2\right]}_{\text{Variance}}+\underbrace{\mathbb{E}_{\mathbf{x}}\left[\left(\overline{g}(\mathbf{x})-\overline{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2}+\underbrace{\mathbb{E}_{\mathbf{x},y}\left[\left(\overline{y}(\mathbf{x})-y\right)^2\right]}_{\text{Noise}}$$

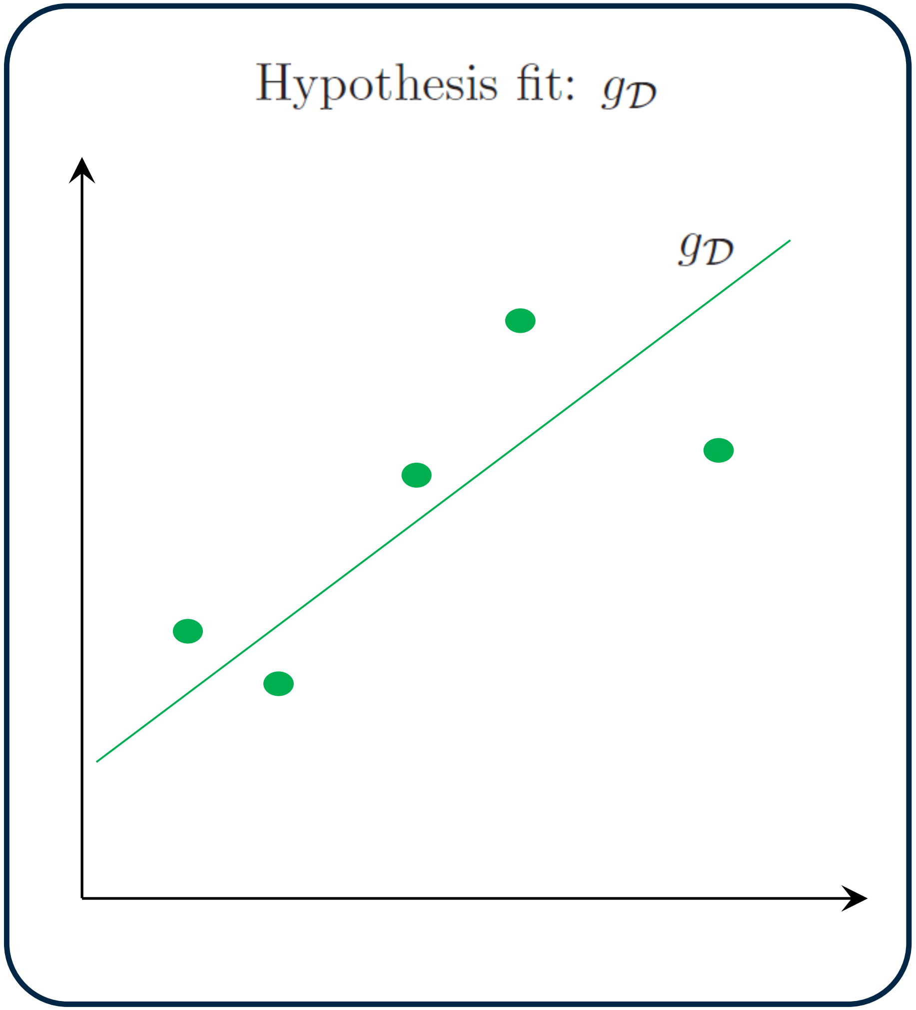
- Variance: It expresses the sensitivity of the solution on the particular choice of dataset $\mathcal{D}$.

- Bias: Difference between the expected prediction (averaged over different datasets) and the expected output value. This is the inherent error arising from the choice of model.

- Noise: Expresses the noise in the data.

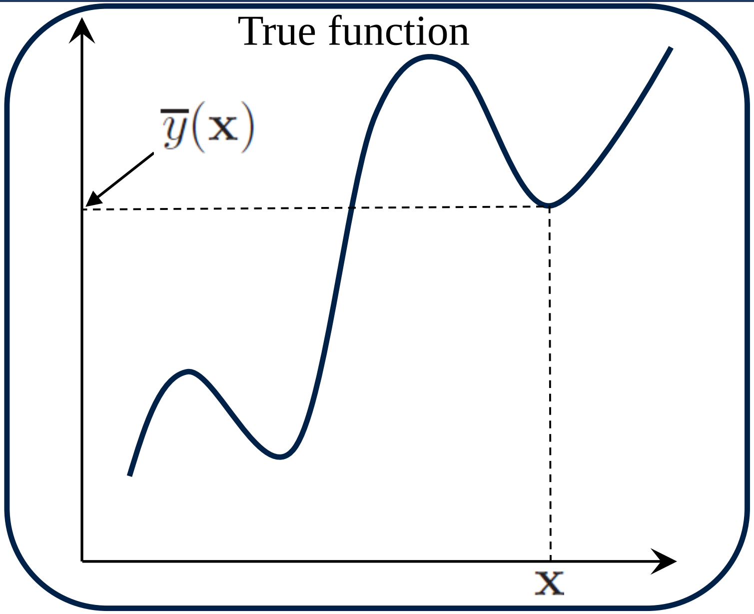
Example

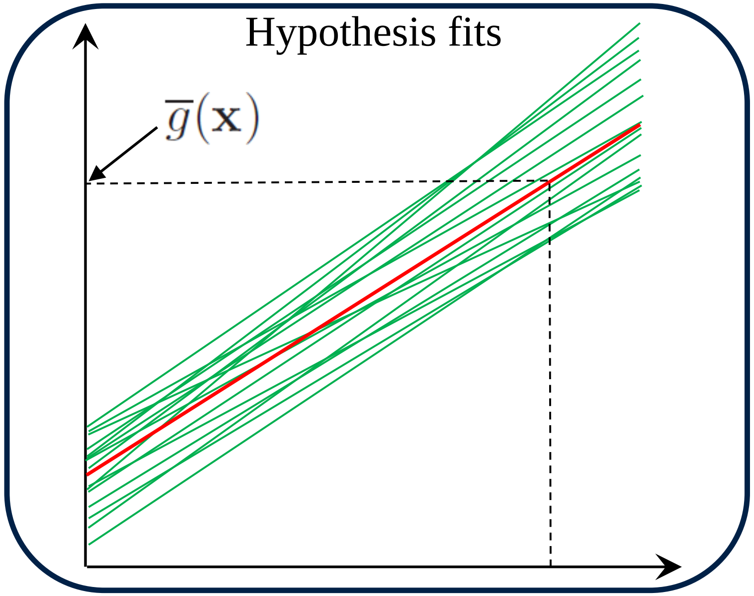


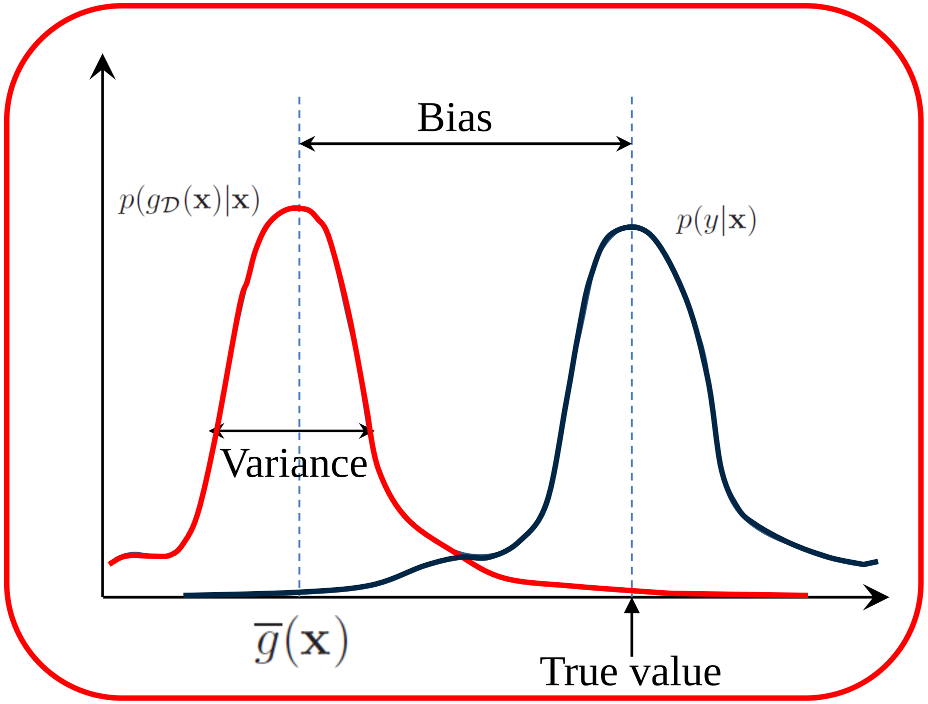


Figures for illustration only.

Visualization

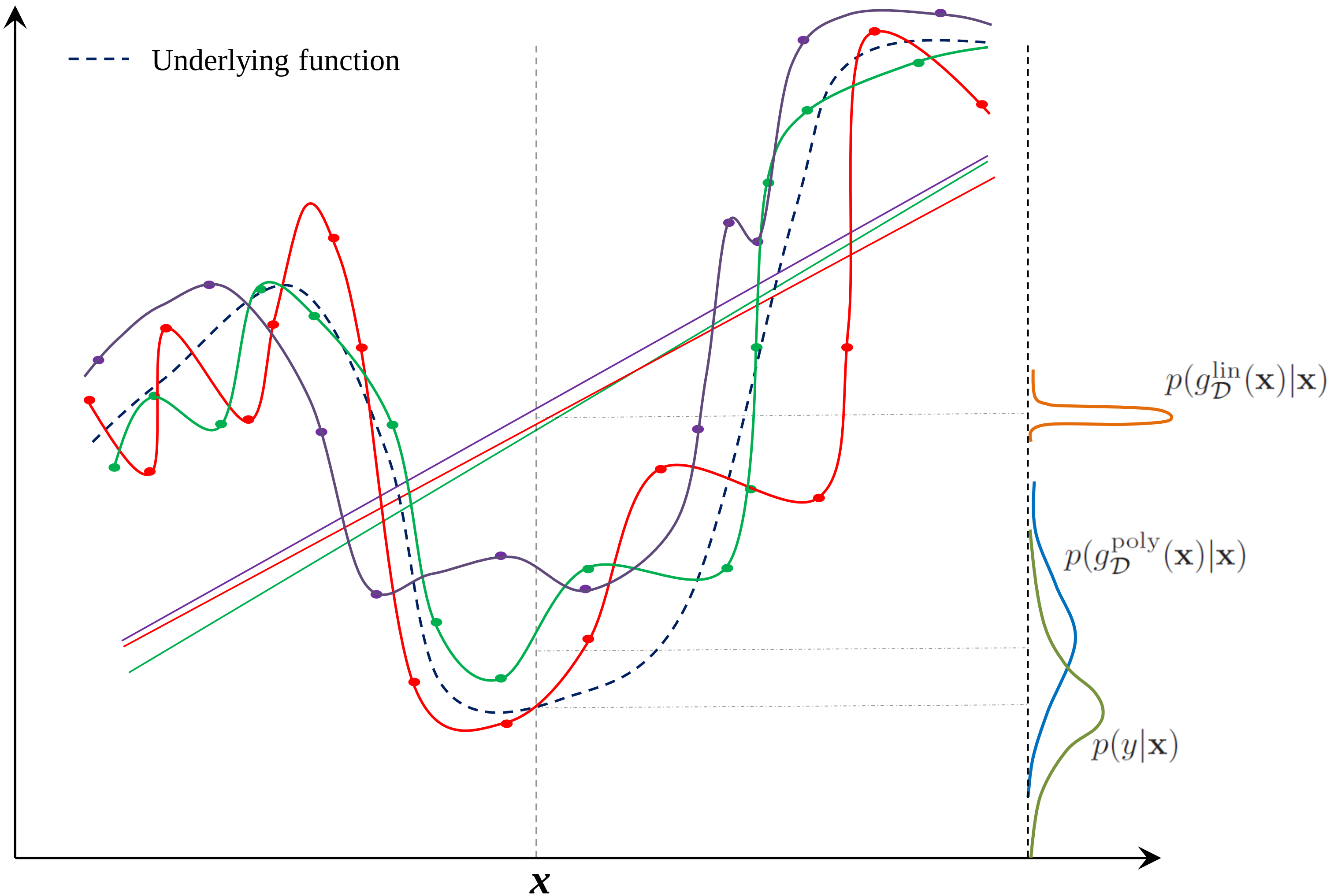






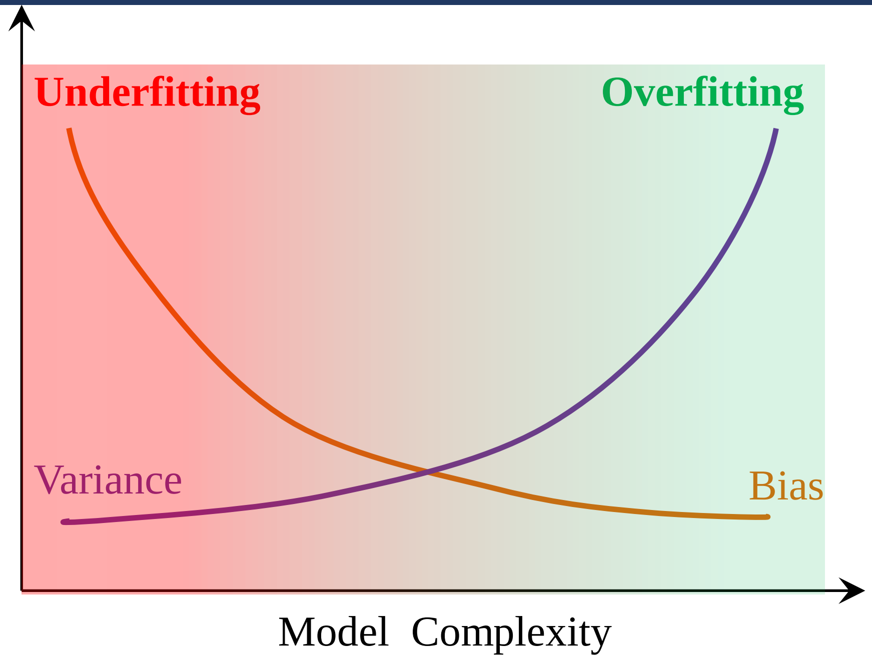
Figures for illustration only.

Another example



Figures for illustration only.

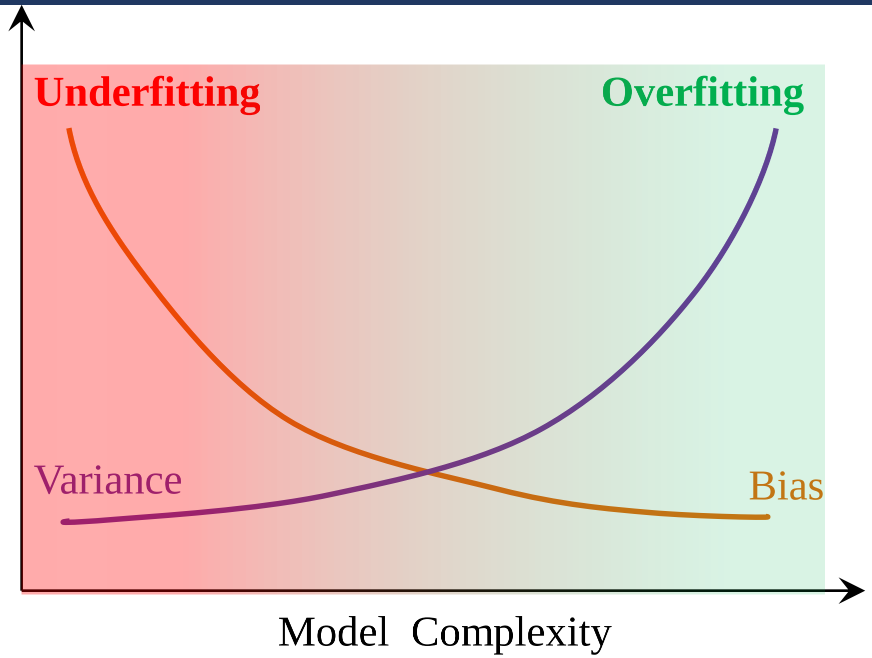
Bias, variance vs model complexity



- **High Bias**: Model is too simple, and so unable to fit the data properly.
  - Results in underfitting.
  - Training and test errors are both large.
- **High Variance**: Model is too complex, and so small changes in the data produce significant changes in the solution.
  - Results in overfitting.
  - Test Error $\gg$ Training Error

Figures for illustration only.

Underfitting & Overfitting



- **Underfitting** can be addressed by
  - Increasing the complexity of the model.
  - Minimizing the cost function properly in the training stage.
- **Overfitting** can be addressed by
  - Reducing the complexity of the model.
  - Incorporating some form of regularization inside the cost function.

Figures for illustration only.

Training and Test datasets

- Dataset is split into two groups:
  - Training dataset is used to train the ML algorithm.
  - Test dataset is used to estimate the error rate of the trained model.

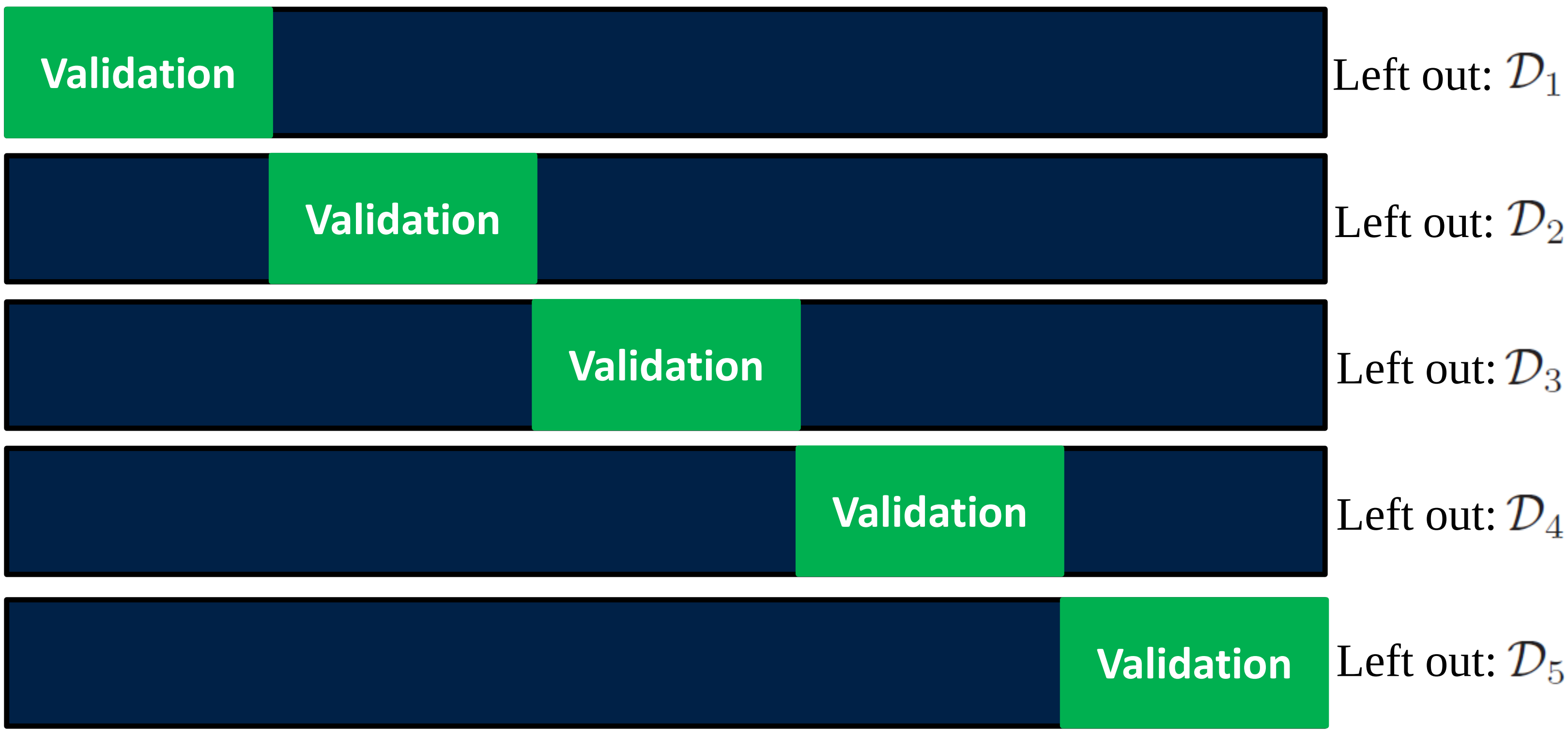
- Shortcomings:
  - If the size of the dataset is small, then keeping aside a separate test dataset can lead to loss of some vital information in the model training stage.
  - "Unfortunate" data split can result in misleading error estimates.
- Solution:
  - K-fold cross-validation
  - Leave-one-out cross-validation

K-fold cross validation

- Training data is subdivided into K separate subsets – $\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_K$ of equal size (say n_K). Let's take $K = 5$.

$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$

- Can generate K training-test datasets using the K subsets



K-fold cross validation

- For $k = 1, 2, .., K$
  - Leave out the kth fold data $\mathcal{D}_k$ and train the model on the remaining $k-1$ folds.

Left out: $\mathcal{D}_k$

  - Use the trained model to make prediction on the kth fold data $\mathcal{D}_k$ and compute the (cross validation) error for this fold

$$E_k = \frac{1}{n_K} \sum_{i=1}^{n_K} \left(y_{k,i} - f_{-k}(\mathbf{x}_i) \right)^2$$

where f_{-k} is the model trained excluding the kth fold data $\mathcal{D}_k$.

K-fold cross validation

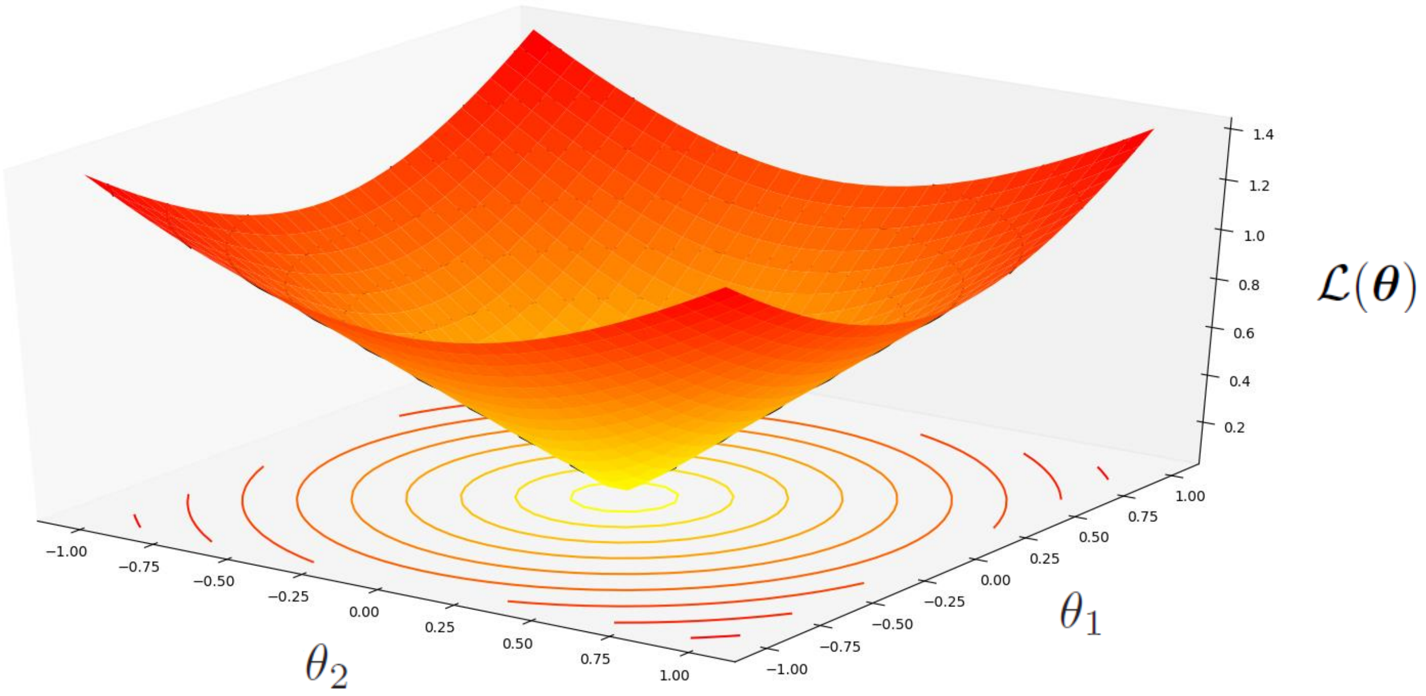
- Estimated generalization error:

$$\mathbf{E} = \frac{1}{K}\sum_{k=1}^{K} E_k$$

- When $K = N$ (size of the training dataset), the approach is known as leave-one-out cross-validation.

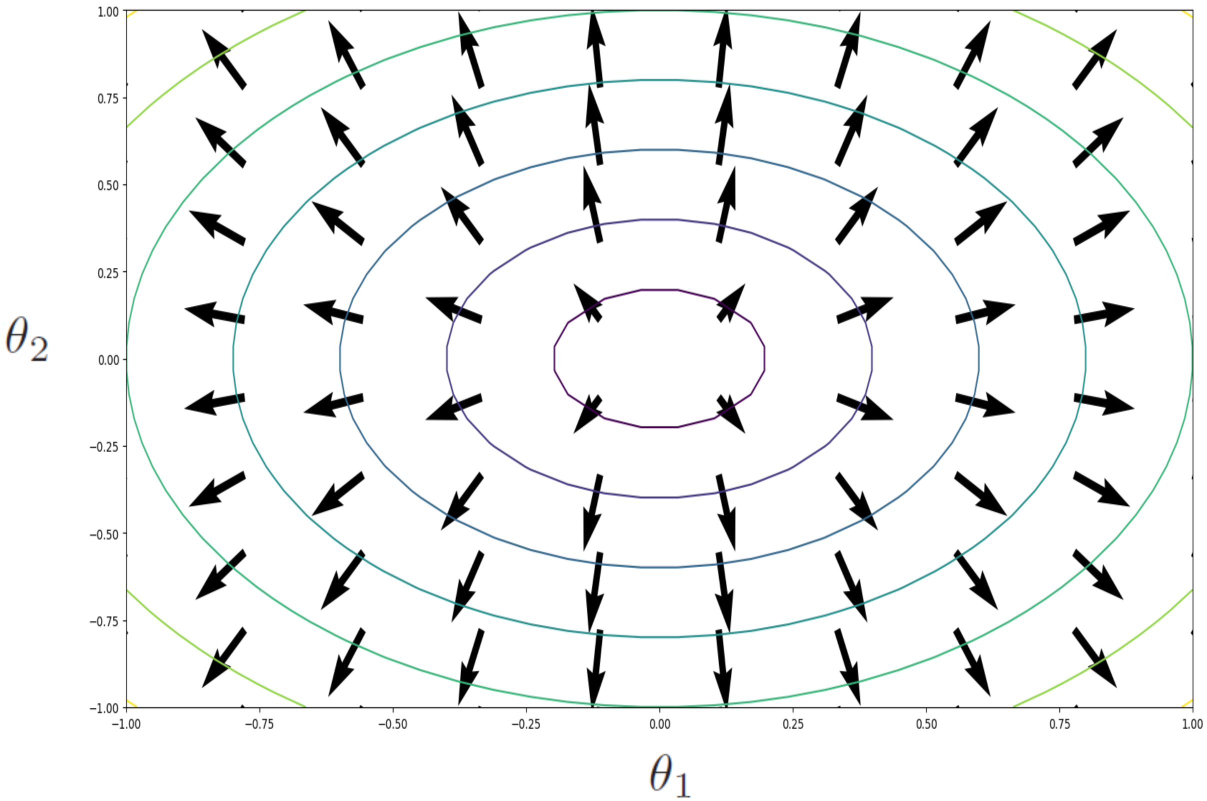
- Note: Cross-validation is also used to tune the hyperparameters of a model.
  - The optimal value of a hyperparameter is the one yielding the least value of $\mathbf{E}$.

Optimization



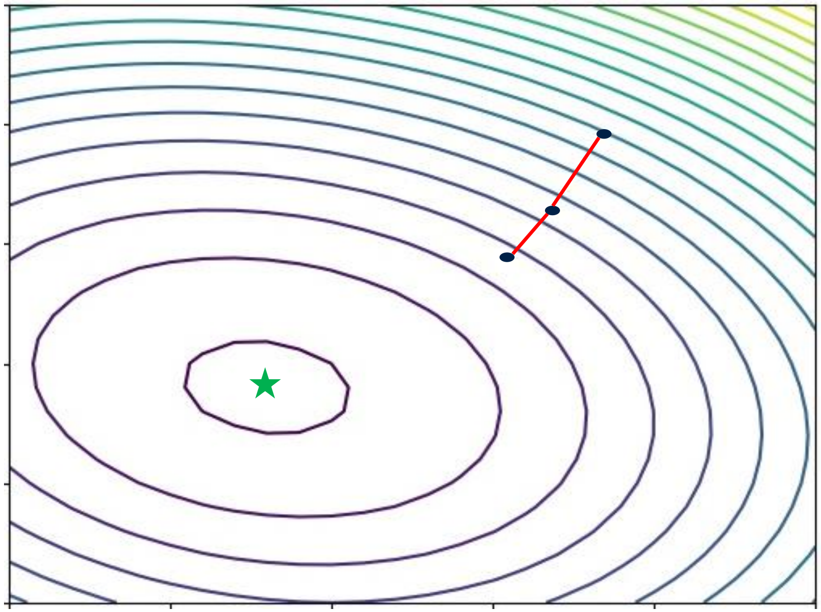
- Objective: $\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$

Contours and gradients



- Contour line is a level curve which is the set of all real-valued solutions for a fixed value of the objective function.
- Gradients are perpendicular to contour lines.

Gradient Descent algorithm



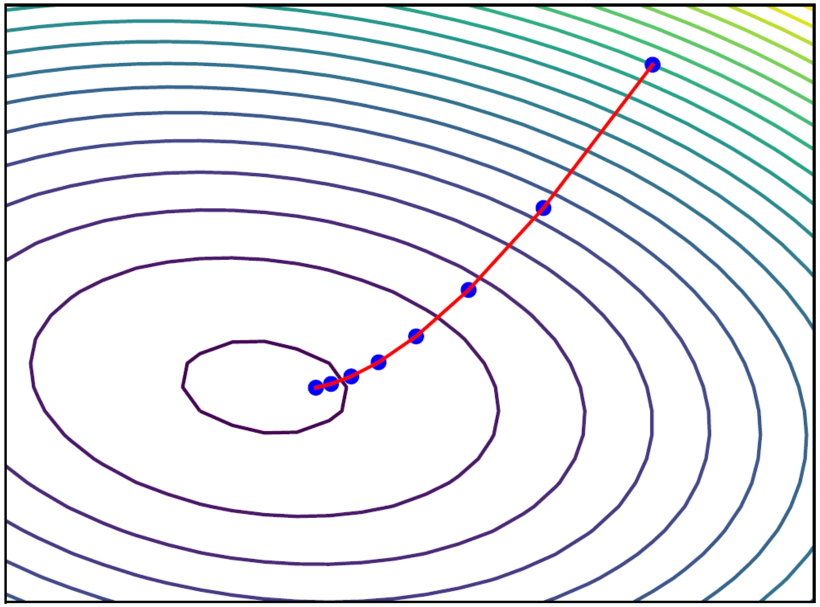
- Gradient descent algorithm (1st order method)

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \xi_k \nabla g(\boldsymbol{\theta}^{(k)})$$

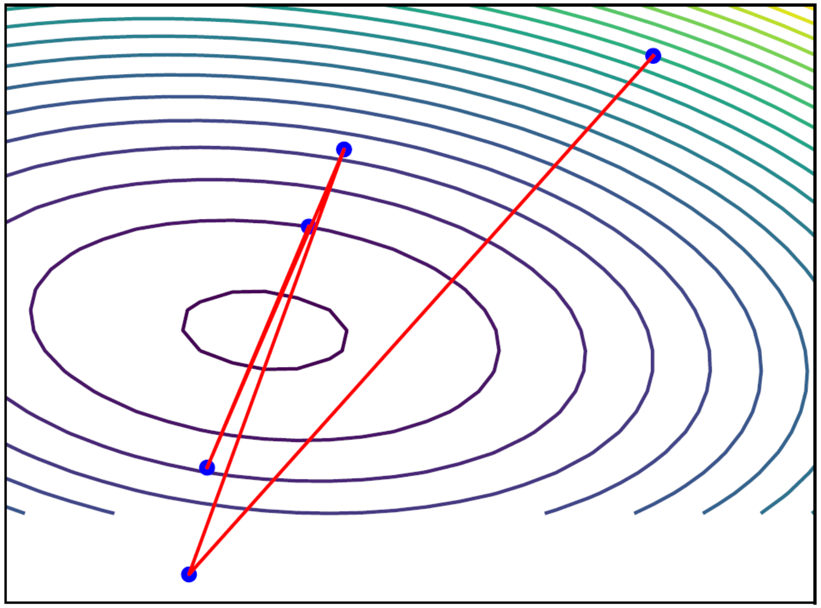
where k is the iteration no. and $\xi_k > 0$ is the learning rate or step size.

- Note that the gradients point towards the maximum, and therefore a negative sign in introduced in front of ξ_k since we are interested in the minimum.

Step size



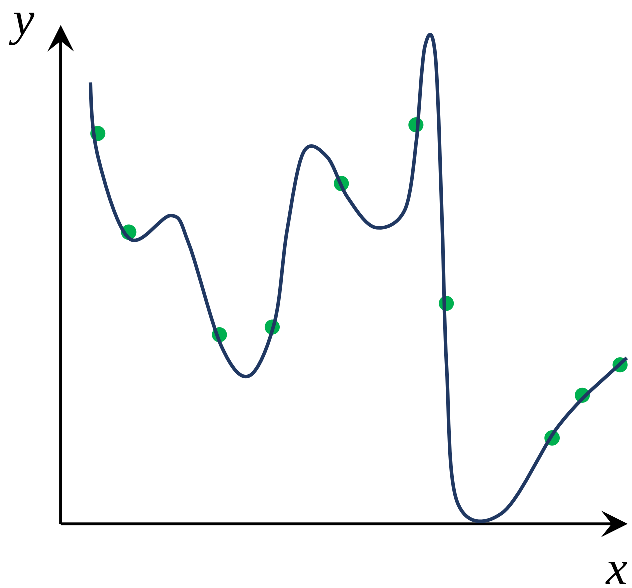
Fixed step-size: ξ_a

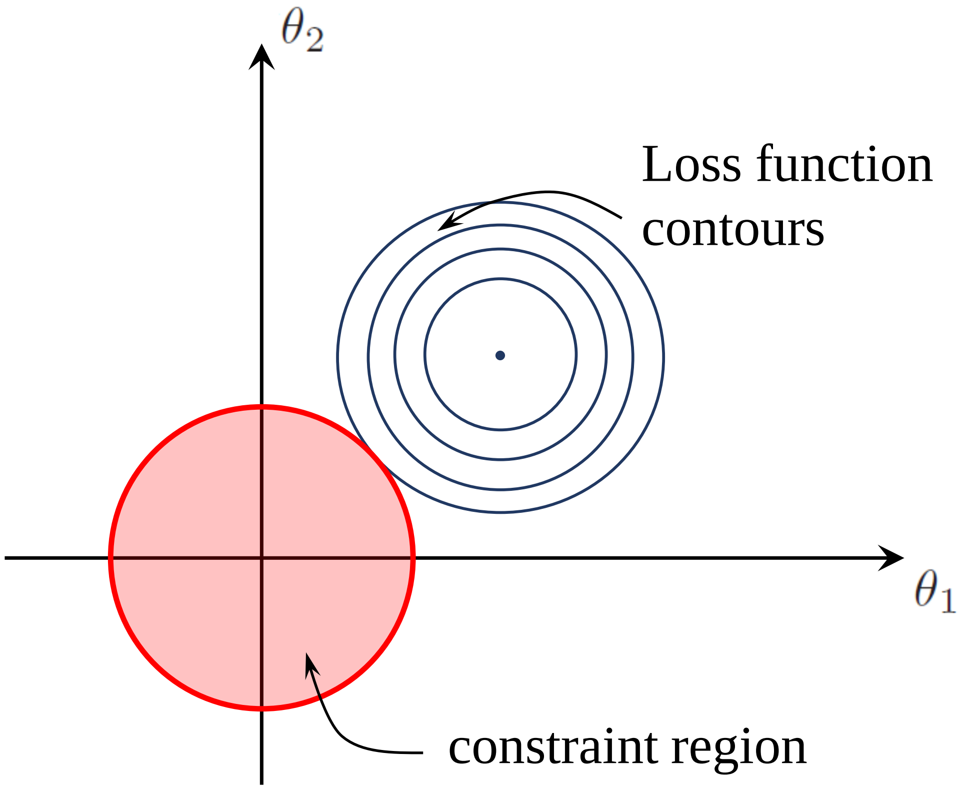


Fixed step-size: ξ_b

$\xi_a < \xi_b$

Regularization





Figures for illustration only.

Classification: Confusion matrix

- A table used to describe/visualize the performance of a classification algorithm.
- Confusion matrix for a binary classification problem:

		Prediction	
		Negative	Positive
Actual	Negative	980	6
	Positive	4	10

- Standard metric: Accuracy
  - Ratio of number of correct predictions to all predictions.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{False Positive} + \text{True Negative} + \text{False Negative}}$$