SimCenter NHERI
Center for Computational Modeling and Simulation

2020 Programming Bootcamp

Machine Learning in Natural Hazard Engineering

Chaofeng Wang
University of California, Berkeley

NSF award: CMMI 1612843

Outline

Part 1 Conventional Machine Learning

Introduction to machine learning

Applications in Natural Hazard Engineering

Algorithms in Conventional Machine Learning

Software and Platforms

Demos

Part 2 Deep Learning

Deep neural networks

Image Classification

Demo

Part 1
Conventional Machine Learning

Supervised Learning
- Classification
- Regression

Unsupervised Learning
- Clustering
- Dimension reduction

Reinforcement Learning
- Decisions
- Robotics
- …

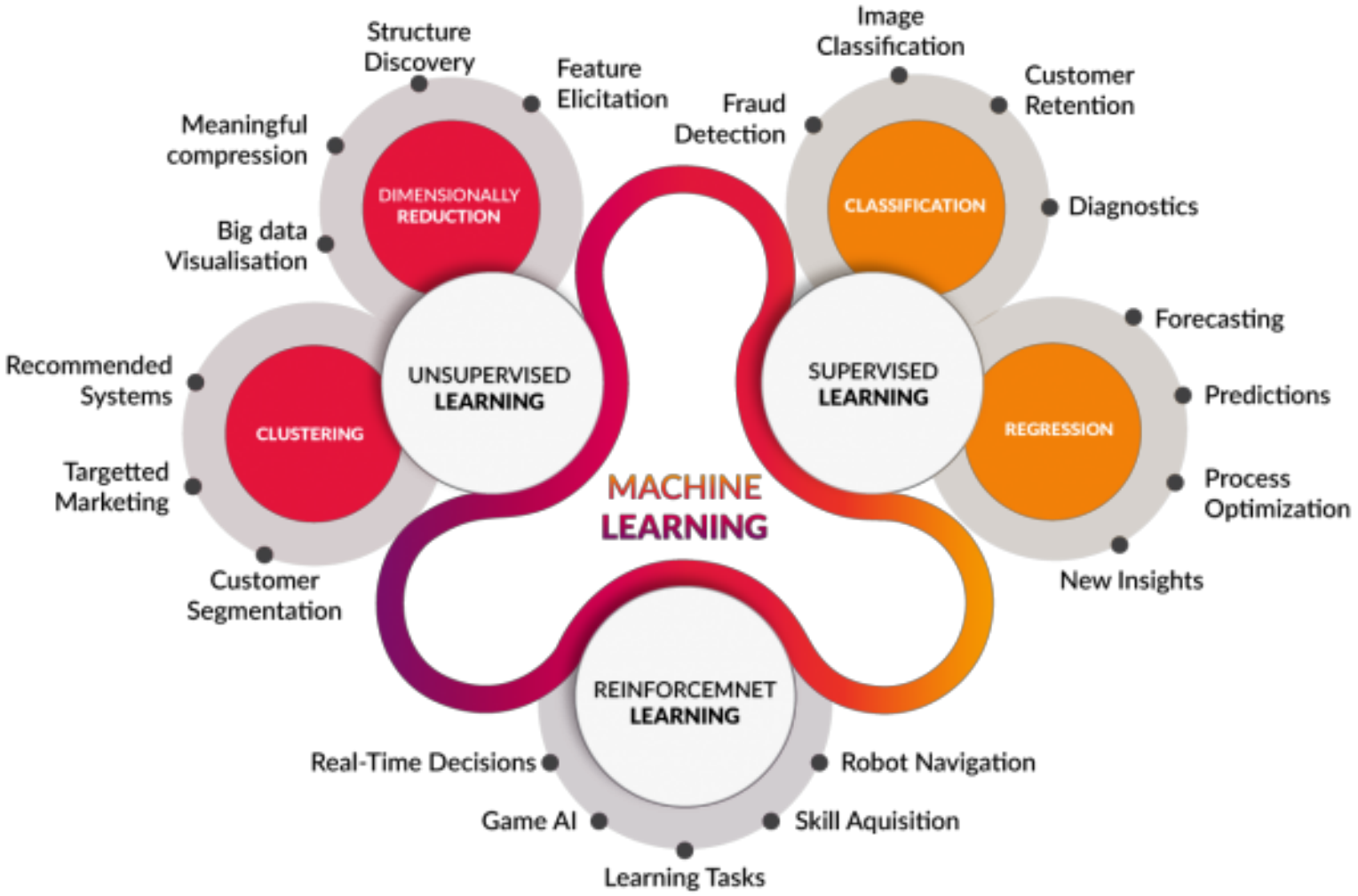


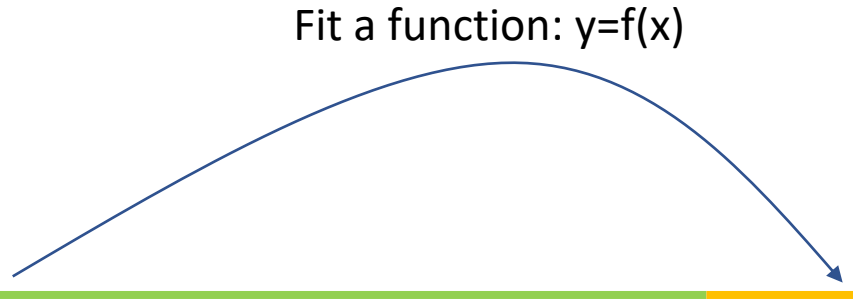
Image from ironhack

Supervised Learning:

- Classification
- Regression

Algorithms:

Regressions
Decision trees
Support Vector Machines
Linear discriminant analysis
K-nearest neighbor algorithm
Multilayer perceptron
…



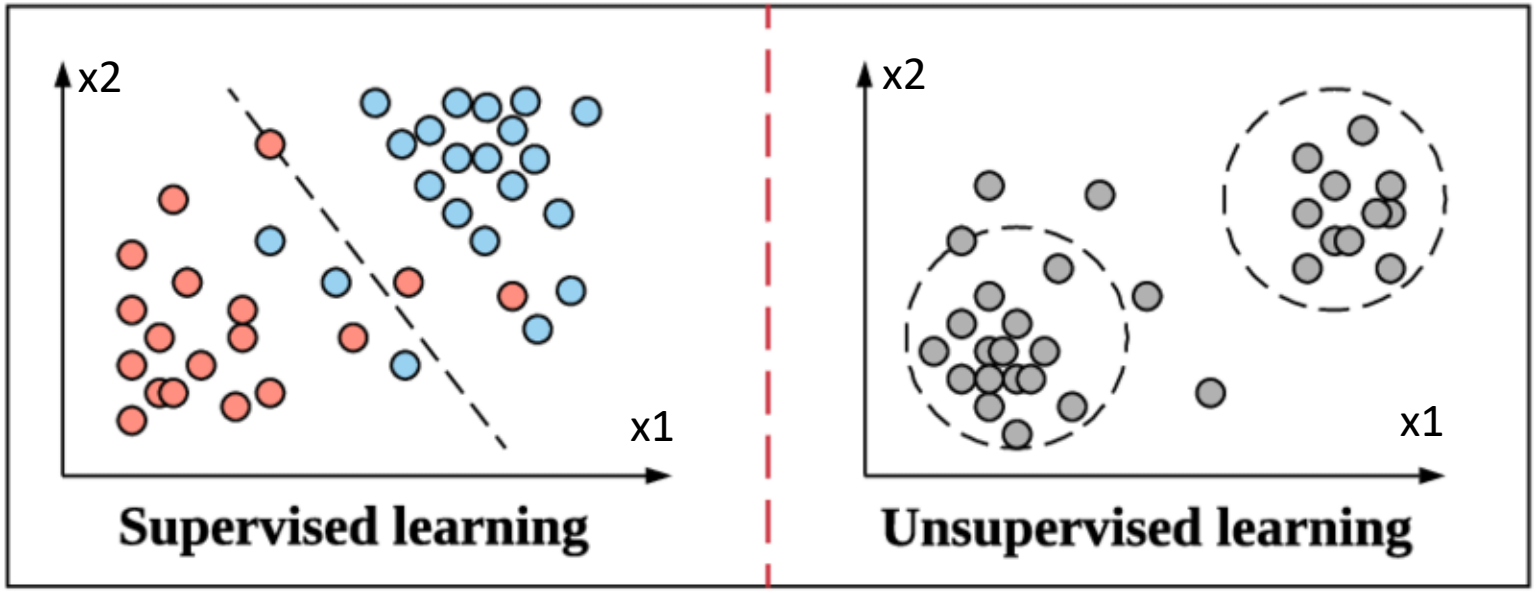
x1	x2	x3	x4	x5	y
0.21	0.20	0.65	0.87	0.29	0.22
0.83	0.47	0.14	0.77	0.43	0.63
0.42	0.31	0.41	0.43	0.11	0.92
0.83	0.49	0.52	0.01	0.94	0.17
0.99	0.05	0.47	0.72	0.01	0.60
0.31	0.31	0.74	0.41	0.93	0.13
0.29	0.03	0.32	0.16	0.24	0.35
0.91	0.91	0.24	0.23	0.51	0.23
0.47	0.04	0.17	0.77	0.34	0.08
0.10	0.10	0.73	0.82	0.32	0.23
0.09	0.66	0.10	0.98	0.21	0.66
0.00	0.35	0.38	0.18	0.89	0.02

Unsupervised Learning

- Clustering
- Dimension reduction

Algorithms:

K-means
Principal component analysis
Autoencoder
Generative adversarial networks



Classification

Trained with labeled data

Can predict the class name


Sky


Vegetation

Clustering

Trained with unlabeled data

Similar data points are grouped together


Group A


Group B

Reinforcement Learning

- Decisions
- Robotics
- …

Algorithms:

Q-learning
SARSA
DQN
…

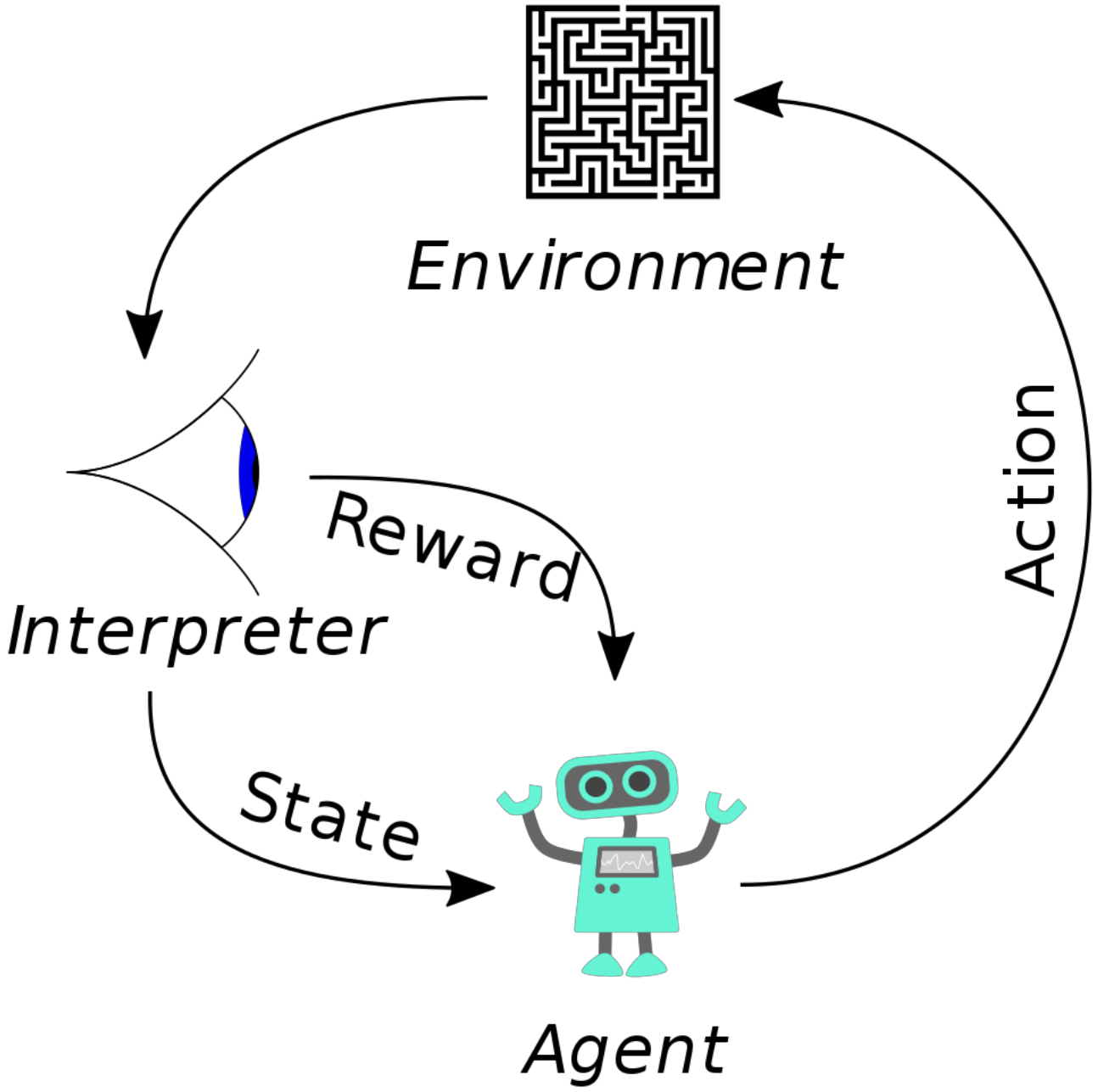
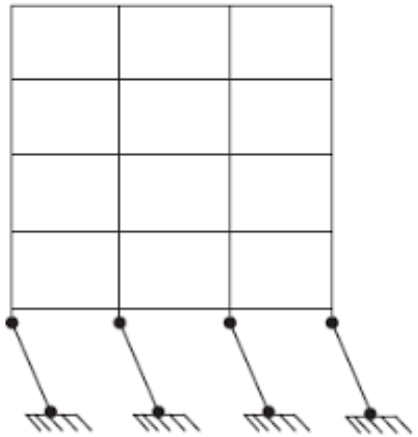
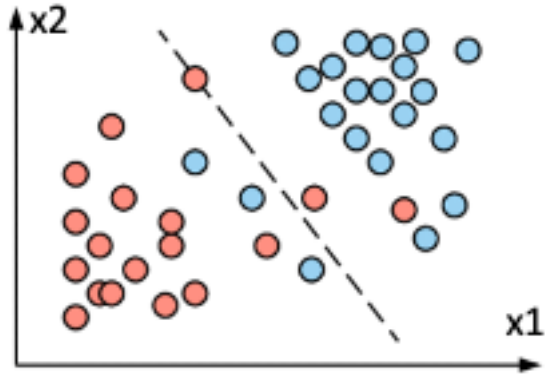


Image from wikipedia

Applications in Natural Hazard Engineering



Physics-based simulation



Statistical method (ML)

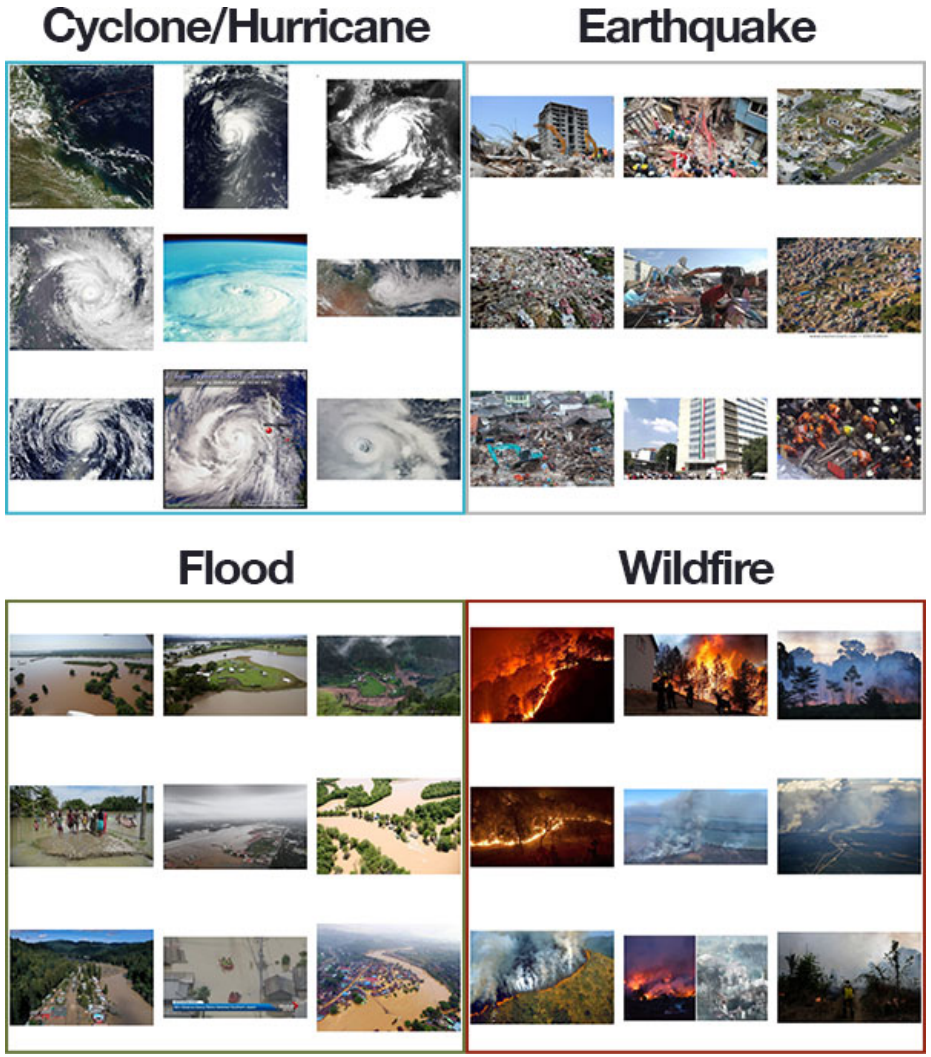


Image from Gautam Kumar

Popular Algorithms in **Supervised Machine Learning**
that are used in natural hazard engineering

Regression
Decision tree / Random forest
K-nearest neighbor
Support vector machines
Multilayer perceptron (Neural network)
…

Keywords:
Data Model Training

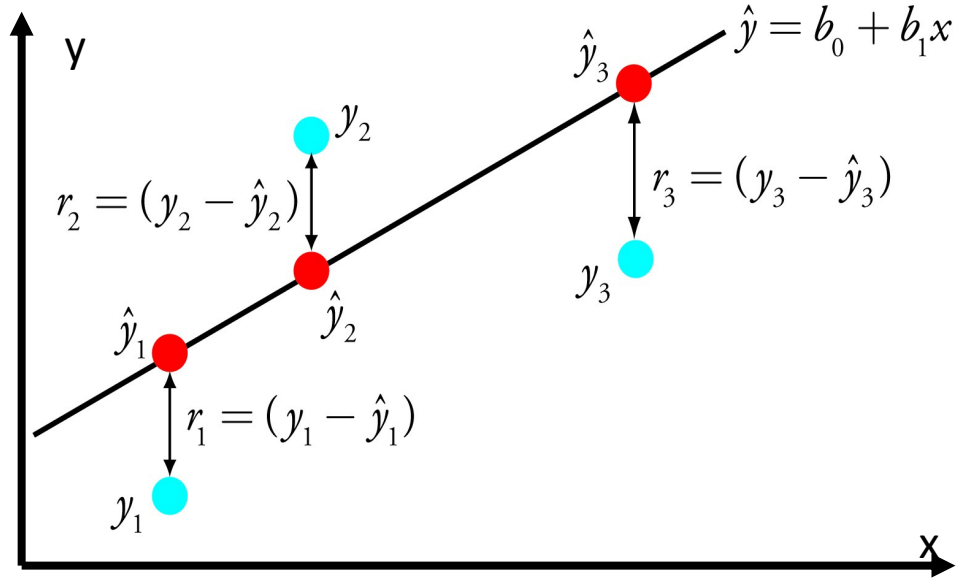
Basic frame:
Use data to set the parameters of a model to fit the labels.

Linear Regression

x	y
0.1	0.05
0.2	0.3
0.4	0.2
x_i	?

Fit the relation by a **linear function**:

$$y = \boldsymbol{X\beta} + \epsilon$$



Math:

Find coefficient $\boldsymbol{\beta}$ *and error* ϵ for

$y = \boldsymbol{X\beta} + \epsilon$

that minimize the residual:

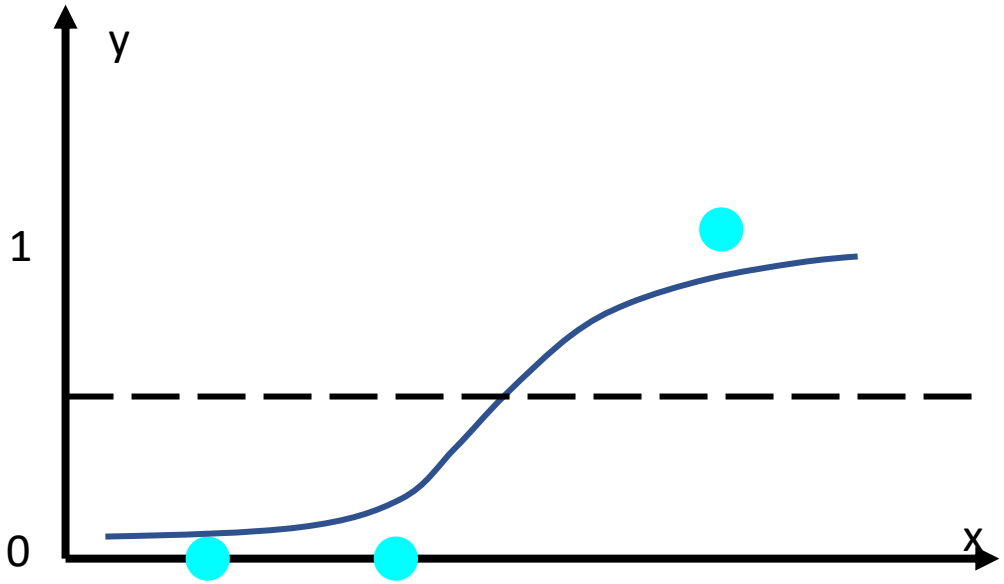
$$\mathcal{R} = \sum_1^n r_i^2$$

Logistic Regression

x	y
0.1	0
0.2	0
0.4	1
x_i	?

Fit the relation by a **logistic function**:

$$P(y = 1|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$$



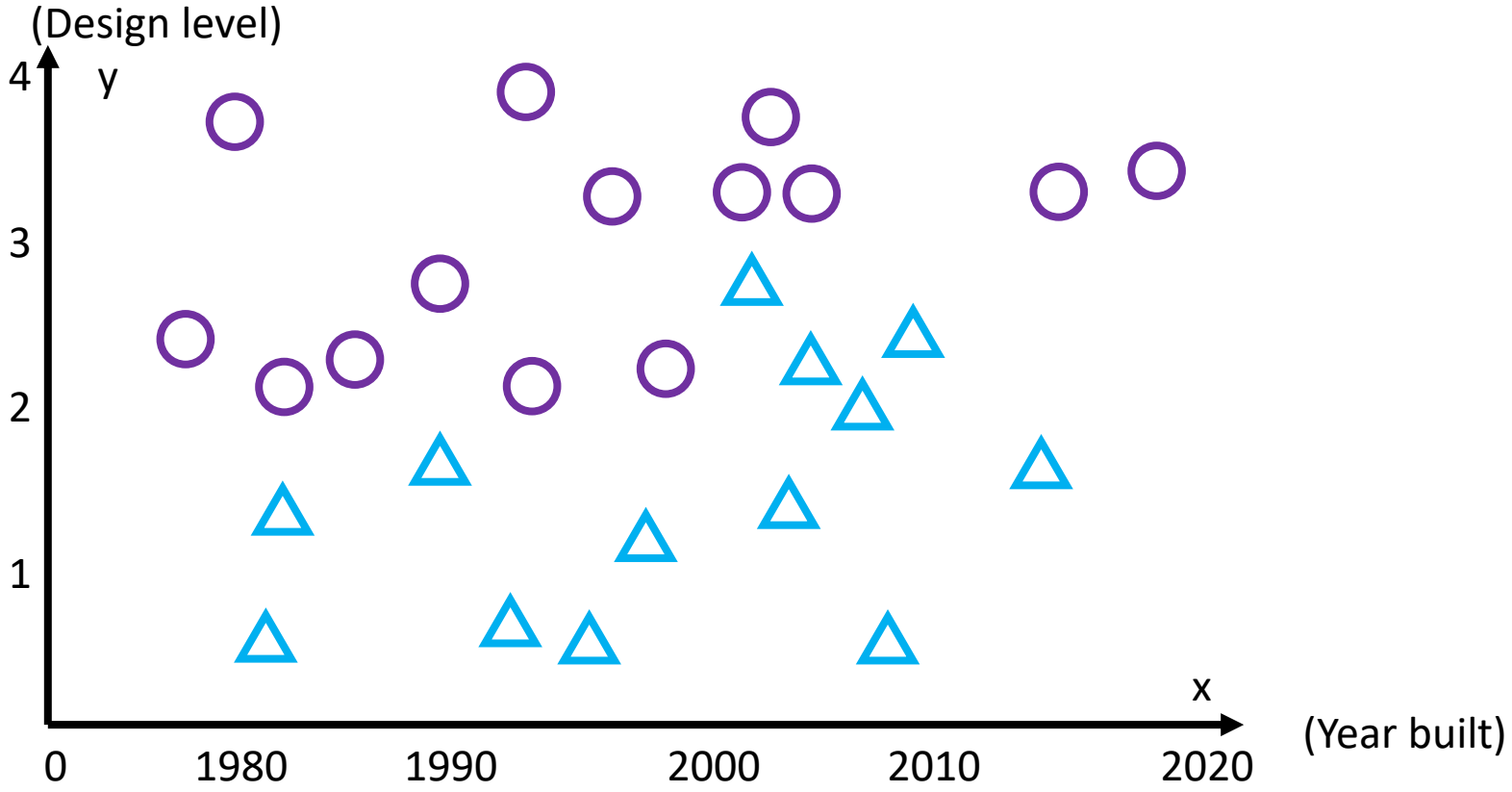
Math:

Find coefficient $\boldsymbol{\beta}$ for

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0+\beta_1 x)}}$$

that minimize the residual between the prediction and the ground truth

Decision Tree

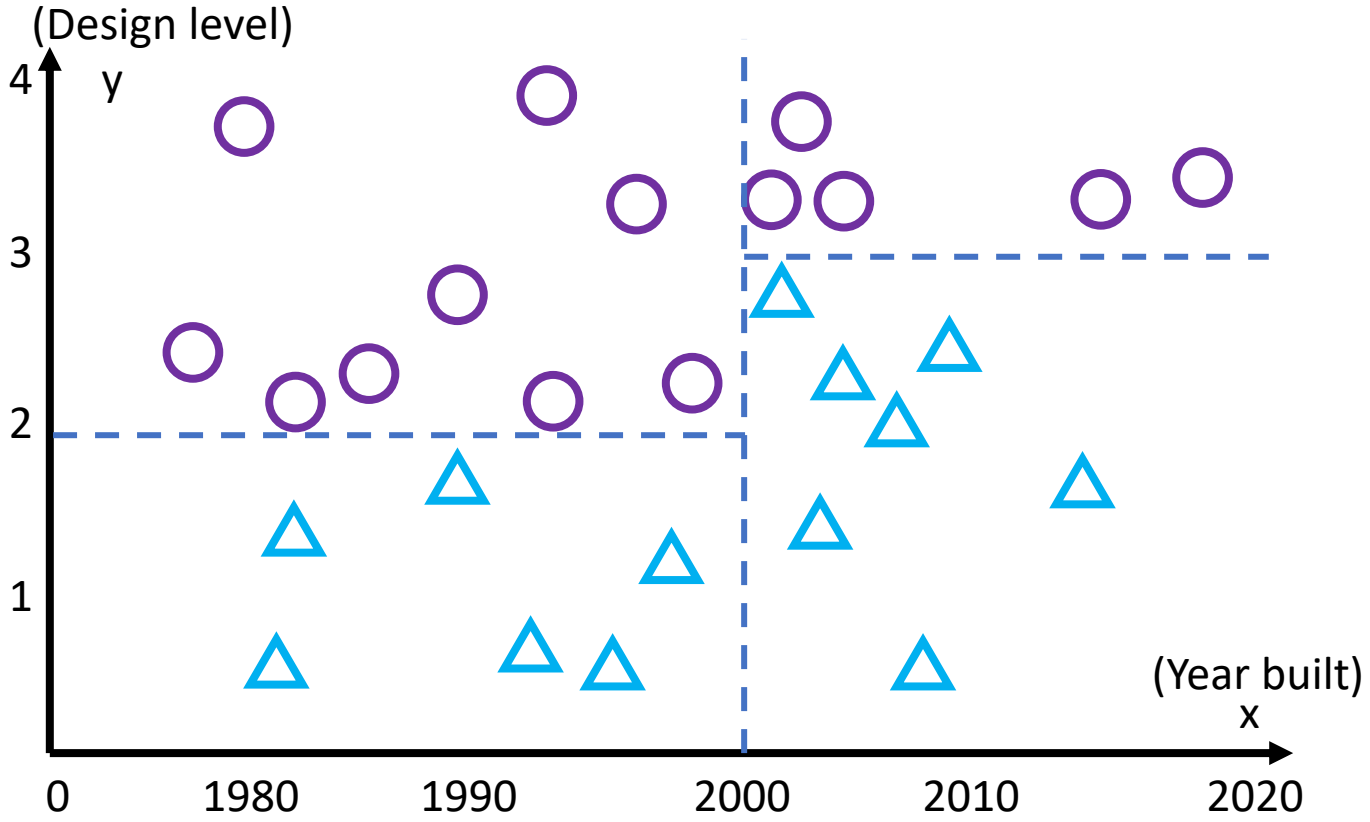


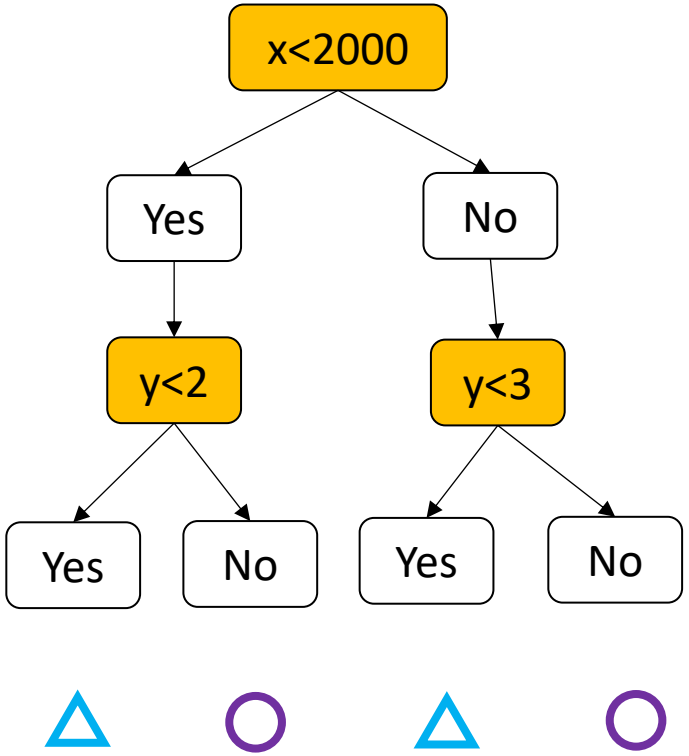
x and y are called **features**, each data point can be represented by features:

○ (x, y)

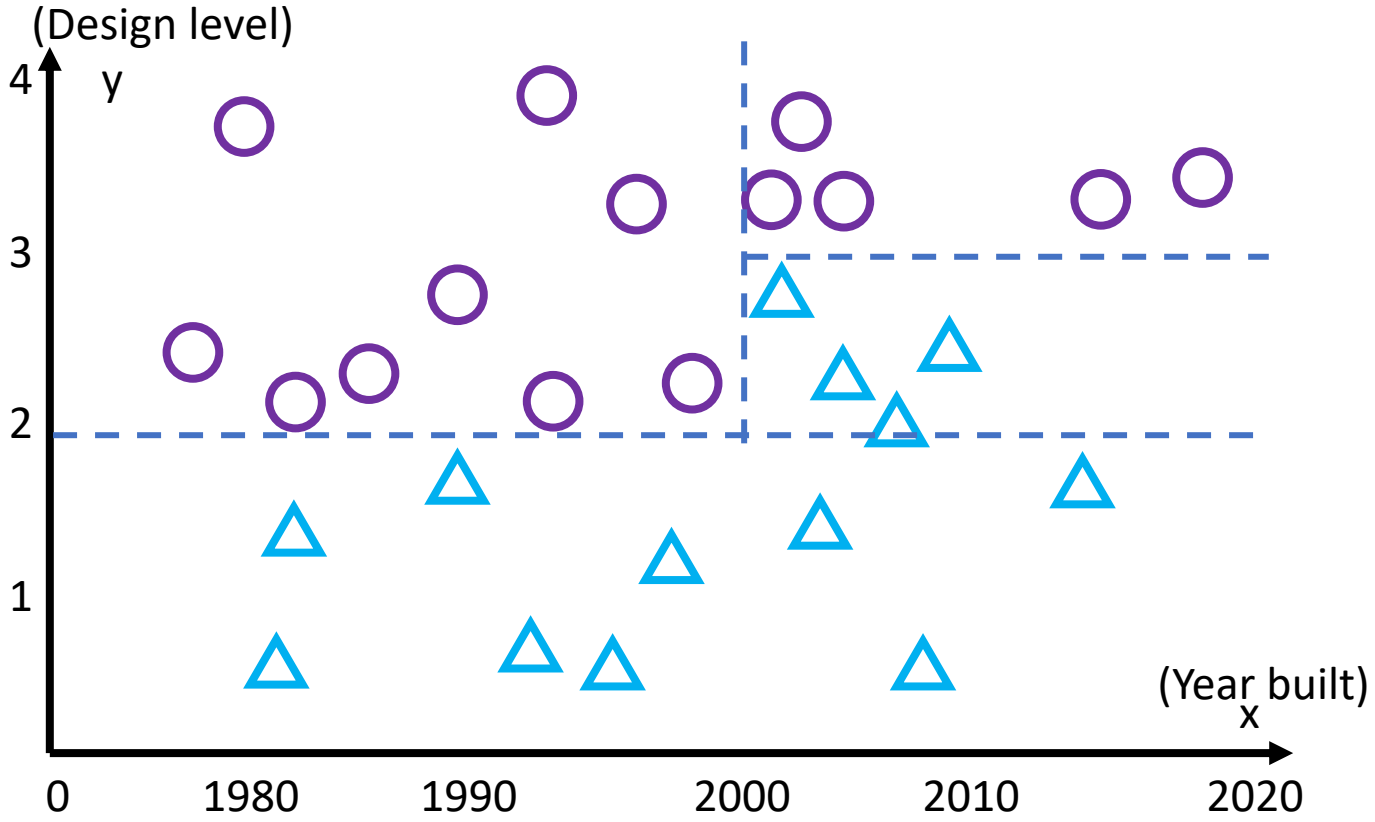
△ (x, y)

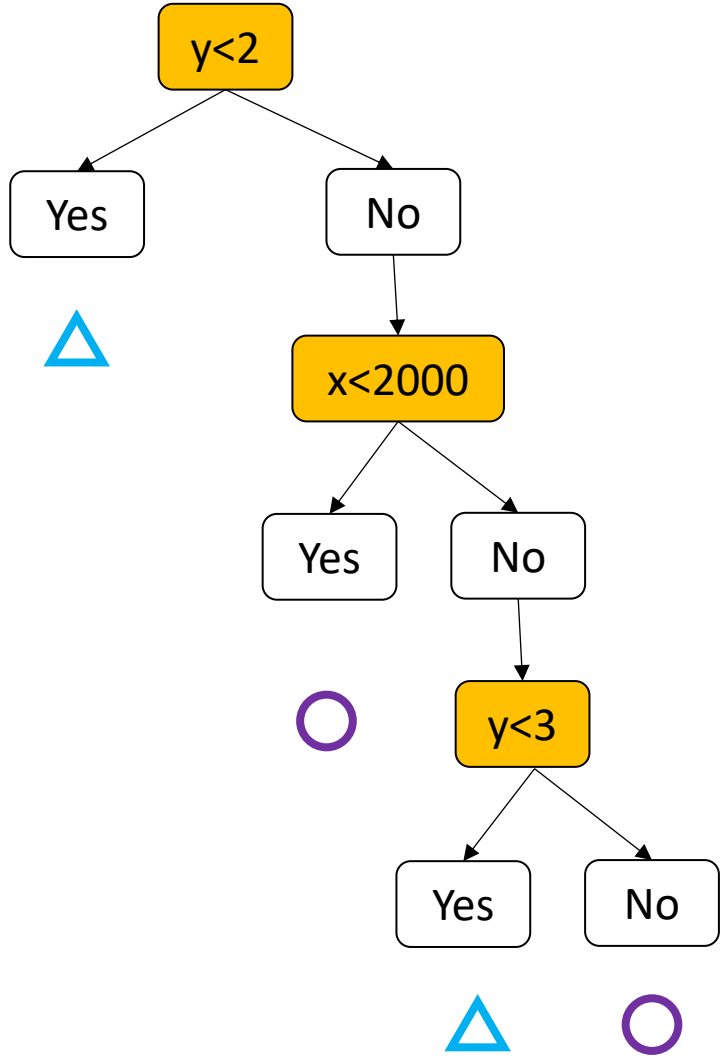
Decision Tree





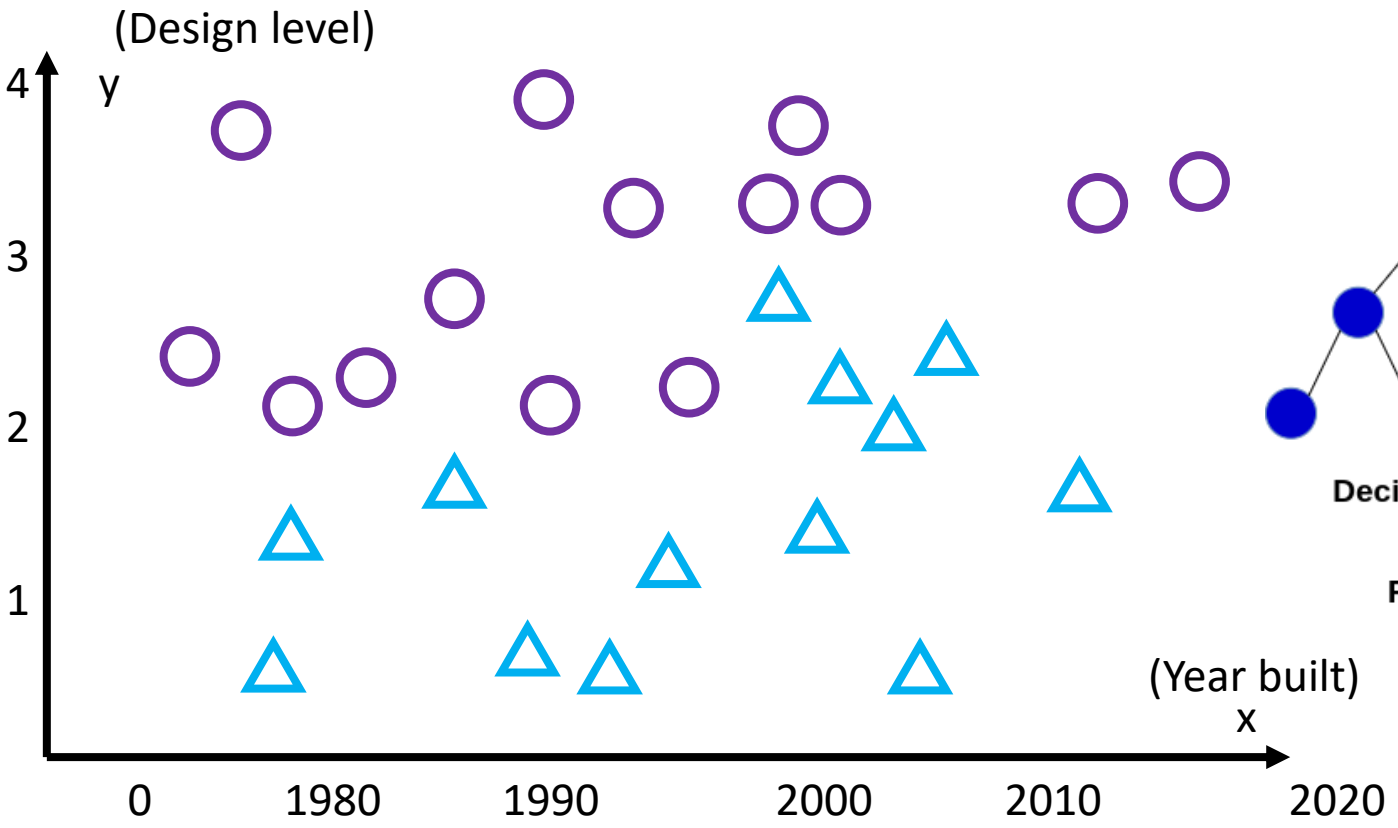
Decision Tree

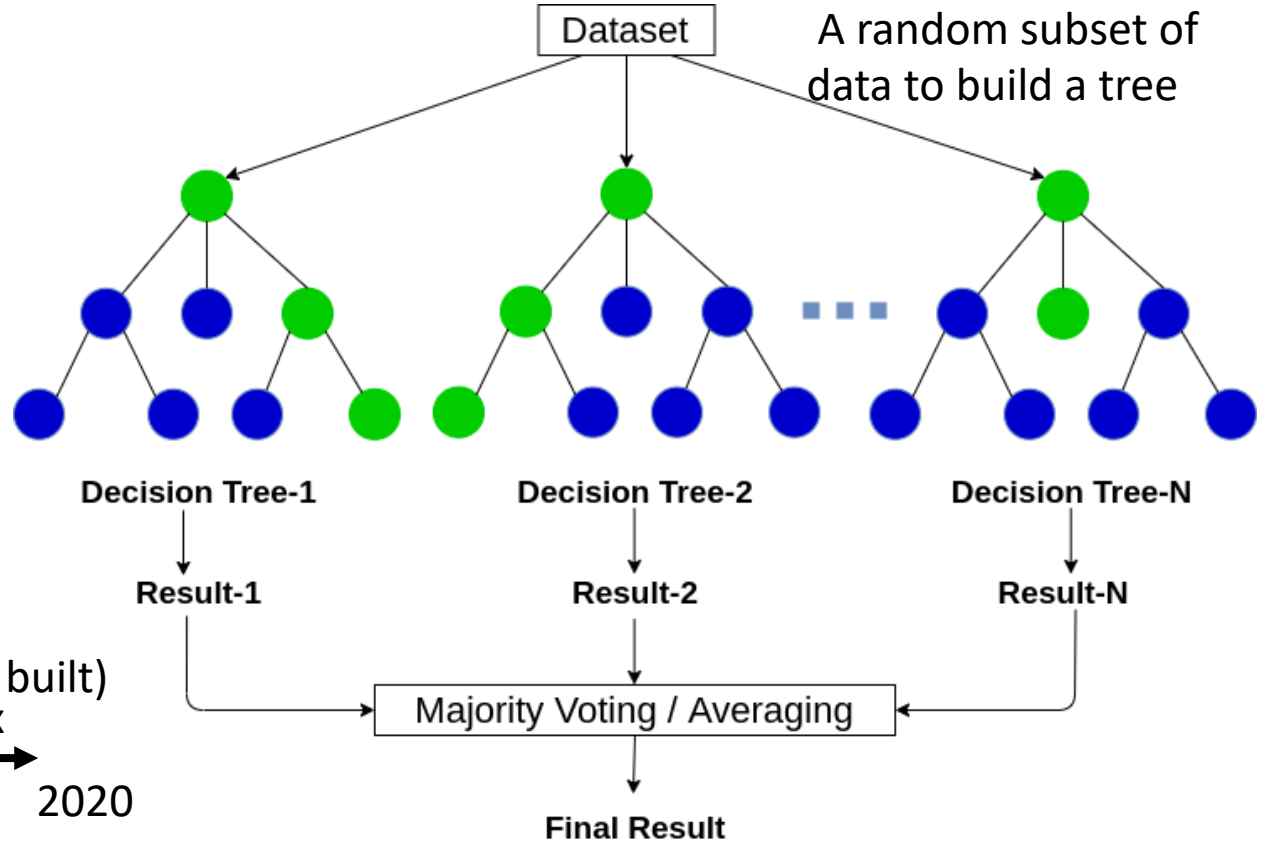




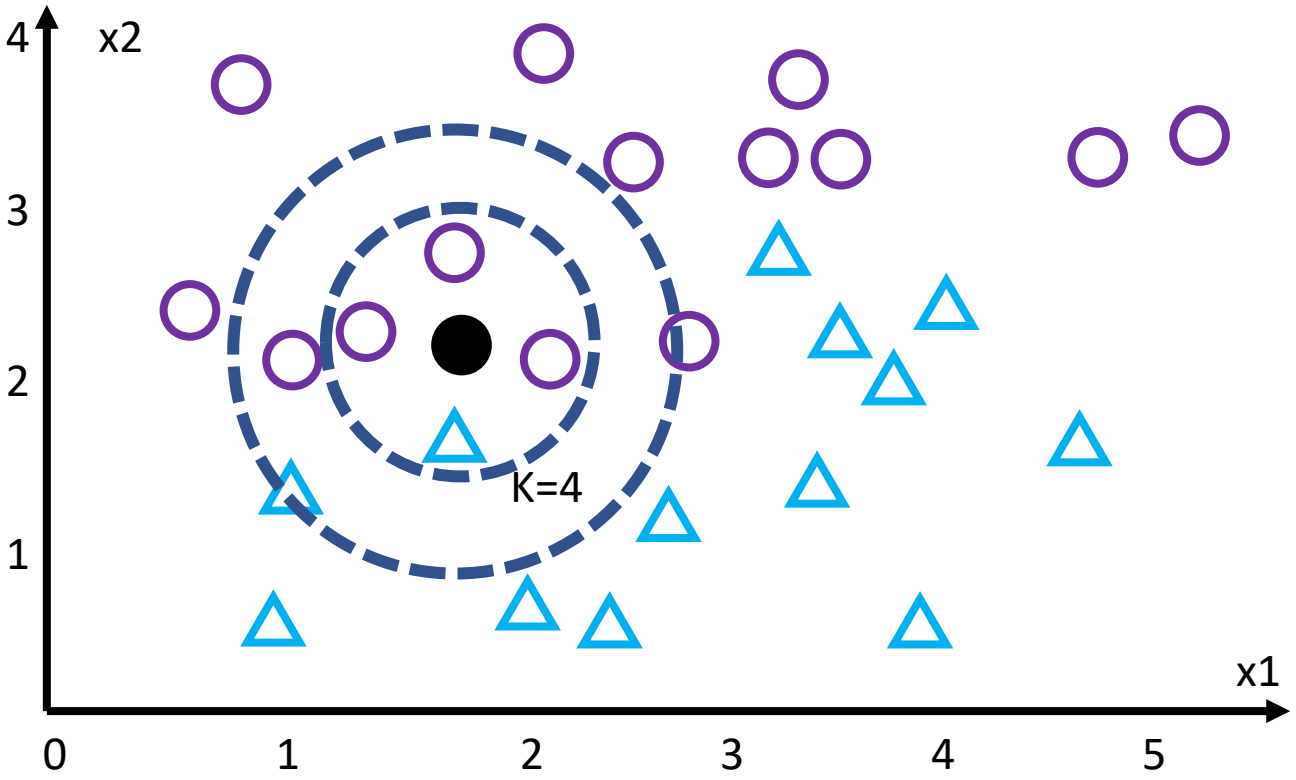
Another tree

Random Forest

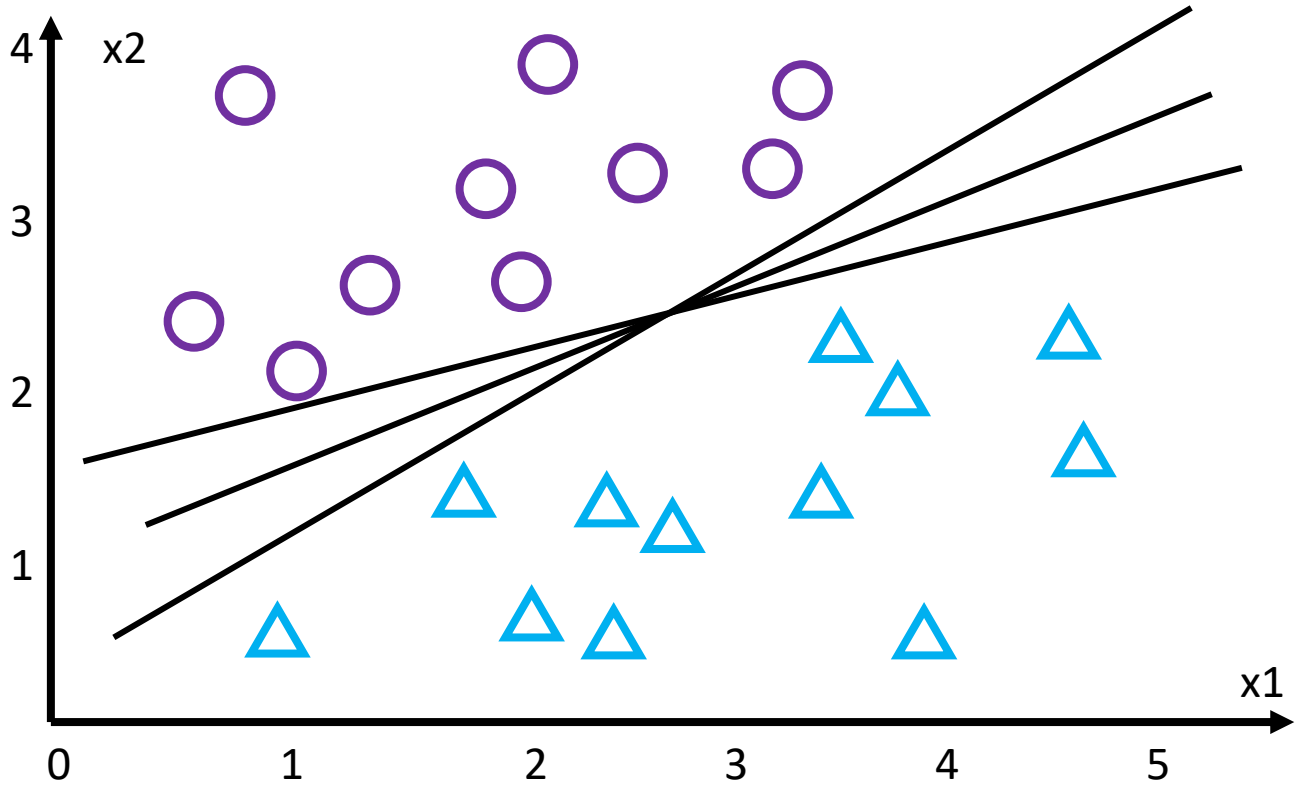




K-nearest neighbor

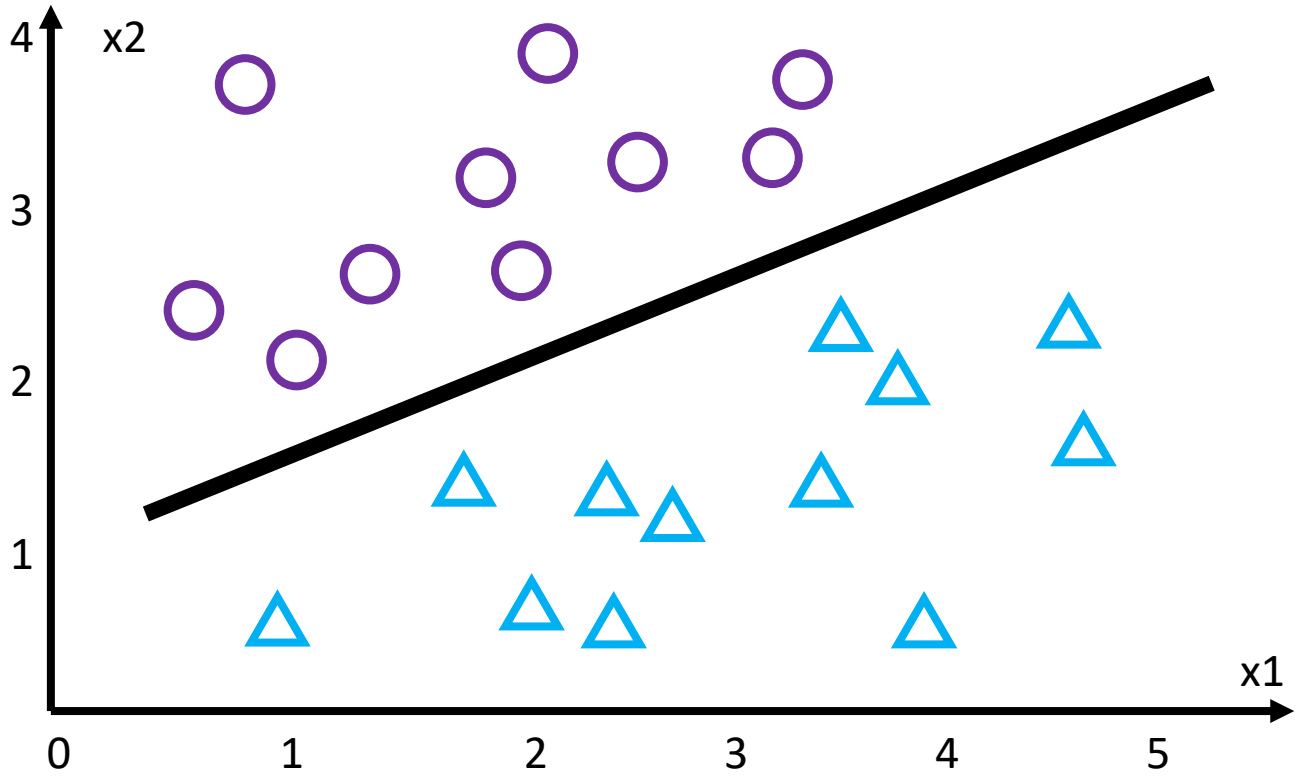


Support vector machines

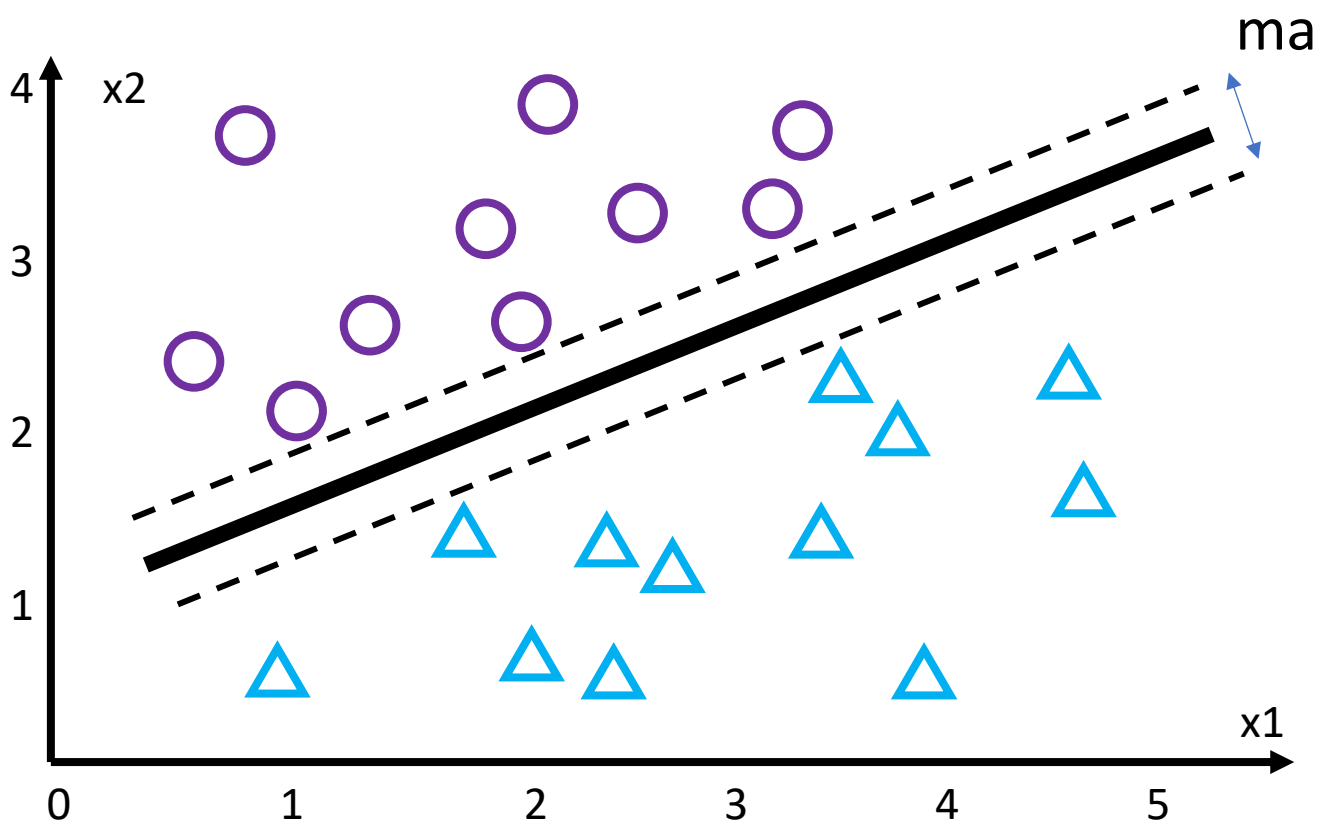


Which is the best?

Support vector machines

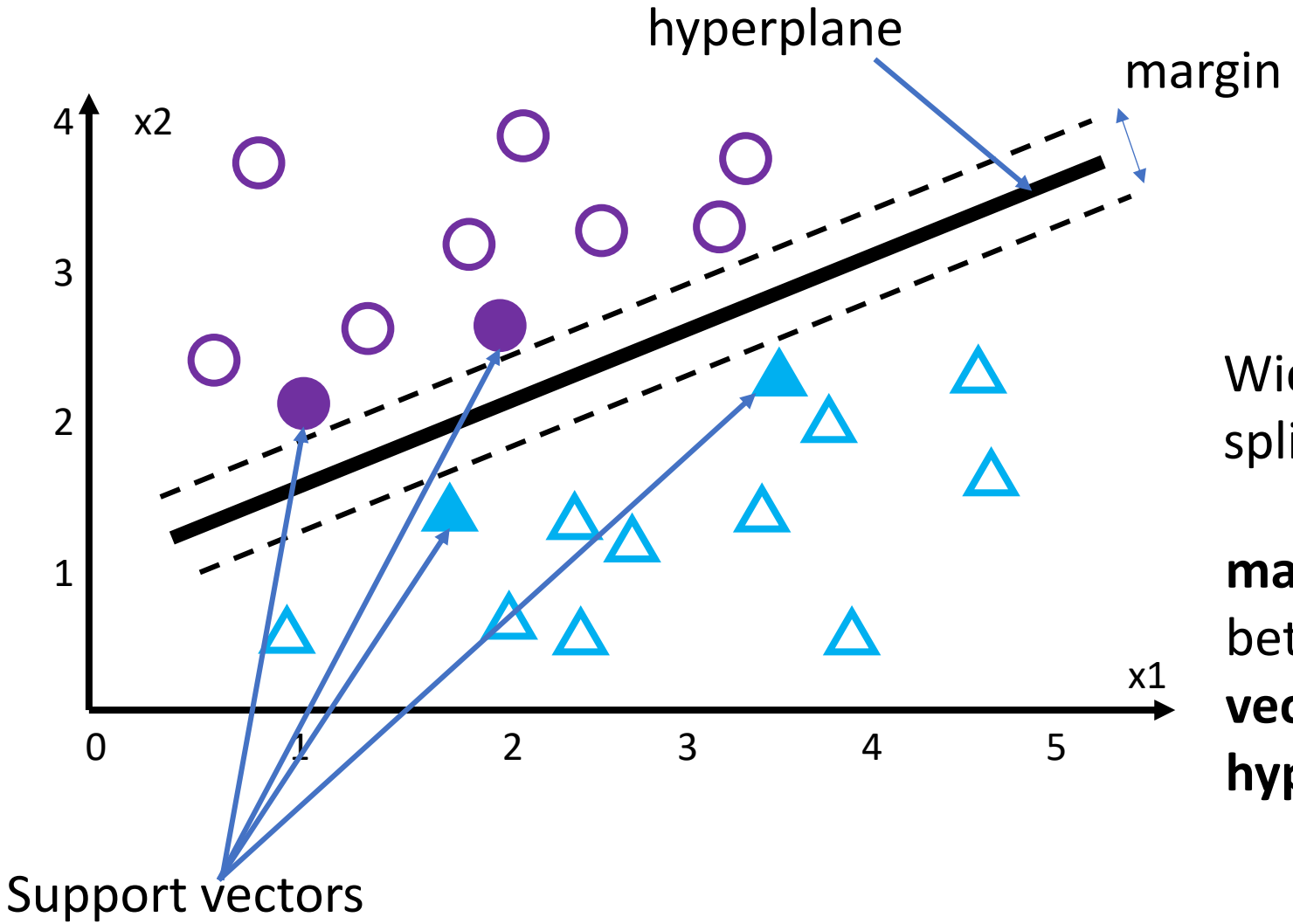


Support vector machines



Widest **margin** to split the groups.

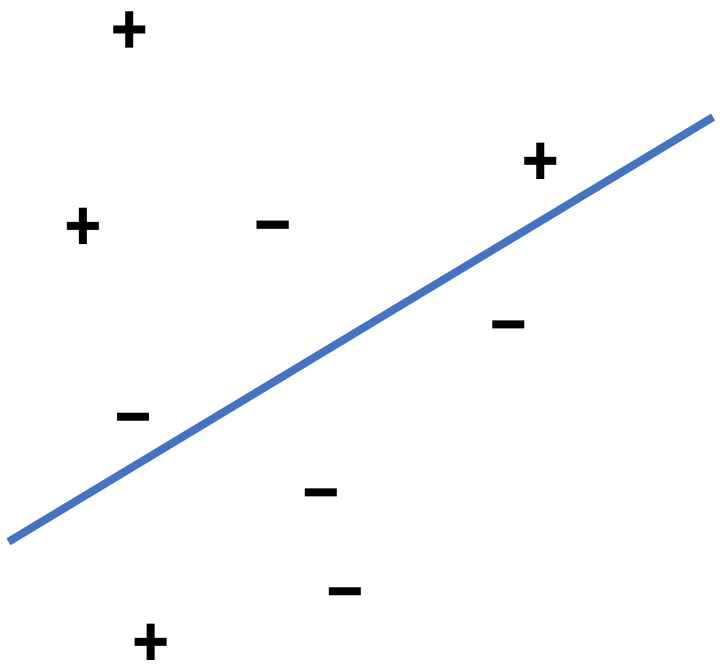
Support vector machines



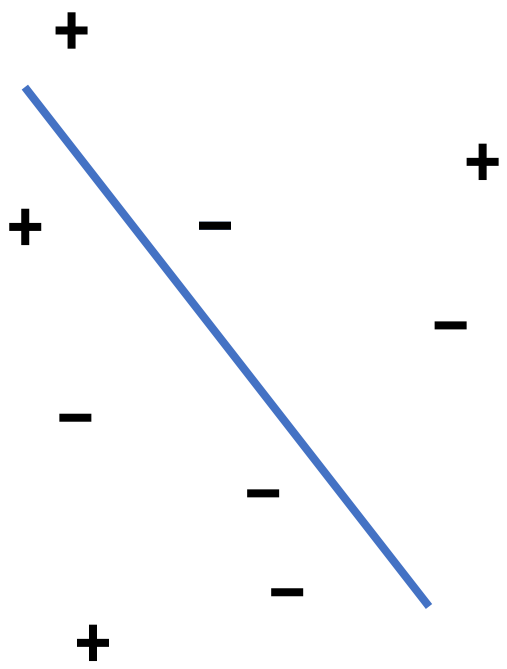
Widest **margin** to split the groups.

margin = distance between the **support vectors** and the **hyperplane**.

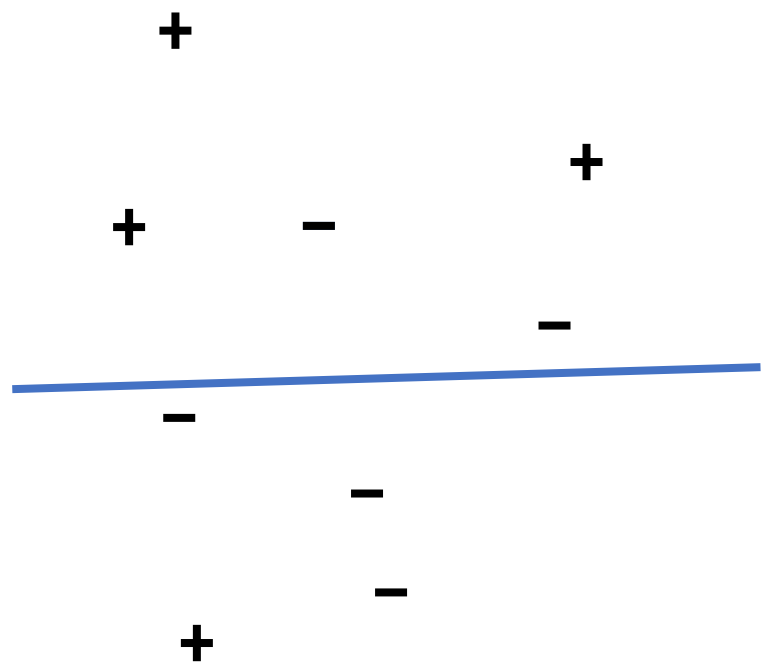
Support vector machines: Kernel Trick



Not a good separator

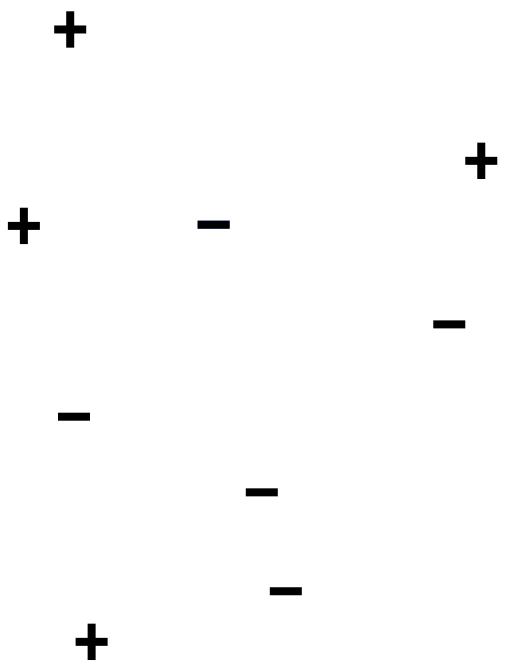


Not a good separator



Not a good separator

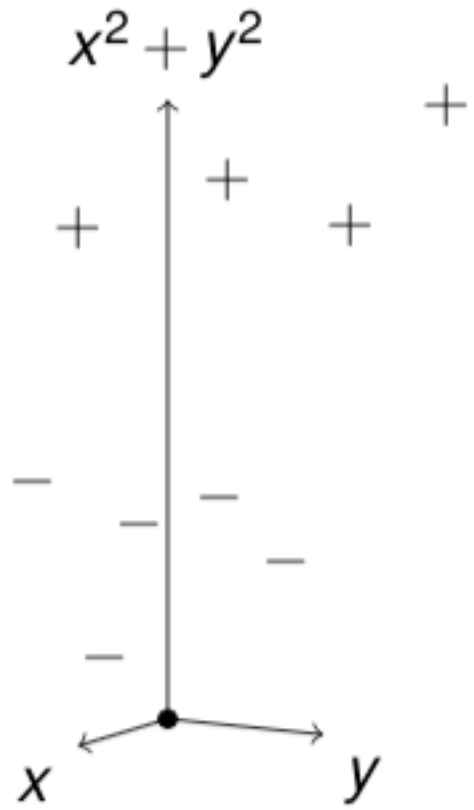
Support vector machines: Kernel Trick



Kernelization

To map point (x,y) to point $(x,y,x2 + y2)$

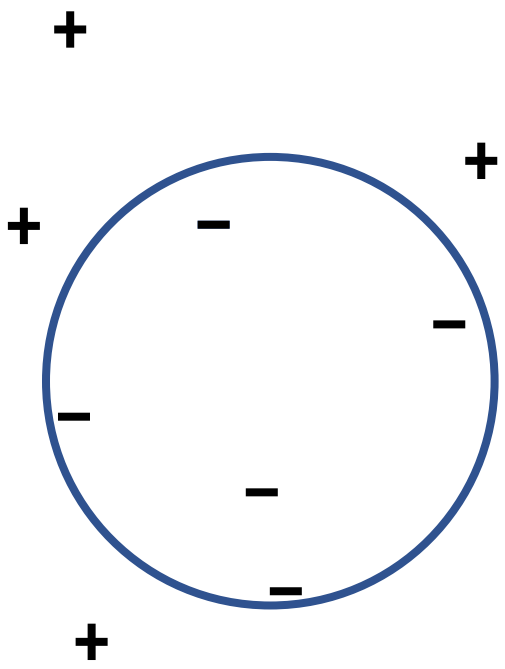
Kernel functions:
Linear
Radial Basis Function (RBF)
Polynomial
Sigmoid



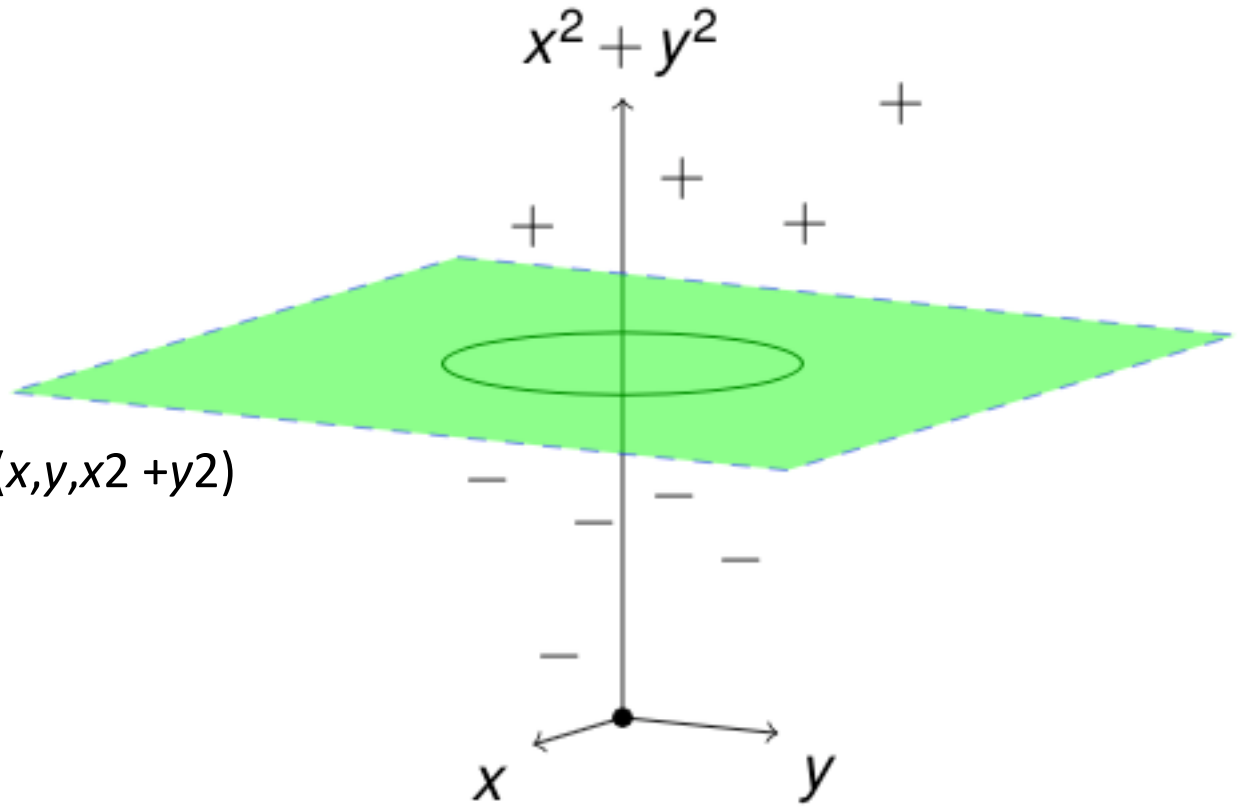
Adapted from Satish Rao, CS270

Support vector machines: Kernel Trick

A visualization: https://www.youtube.com/watch?v=3liCbRZPrZA



Kernelization



To map point (x,y) to point $(x,y,x2 + y2)$

Kernel functions:
Linear
Radial Basis Function (RBF)
Polynomial
Sigmoid

Adapted from Satish Rao, CS270

What is a good model?

Confusion matrix

Actual Values	Positive (1)	Negative (0)
Positive (1)	**TP**	FN
Negative (0)	FP	**TN**

Predicted

TP: Your prediction is 1 and it's true
TN: Your prediction is 0 and it's true
FP: Your prediction is 1 but it's false
FN: Your prediction is 0 but it's false

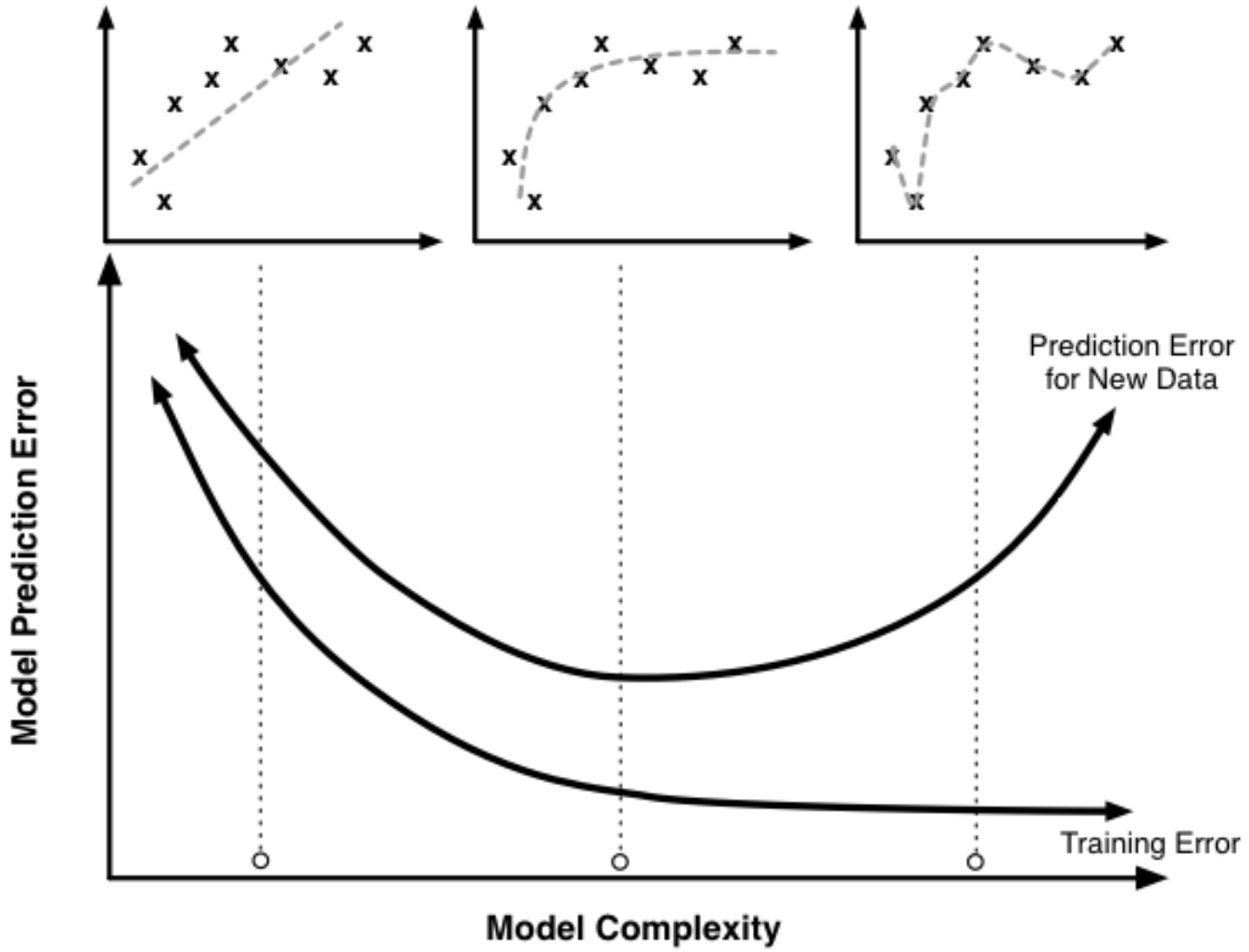
Recall=$\frac{TP}{TP+FN}$ How much of the positive class are correctly predicted?

Precision= $\frac{TP}{TP+FP}$ In all the positive cases we predicted, how much are actually positive?

Accuracy= $\frac{TP+TN}{ALL}$ Out of all the cases, how much we predicted correctly

F-1 = $\frac{2*Recall*Precision}{Recall+Precision}$ Harmonic Mean

What is a good model?



underfit optimal overfit

Demos

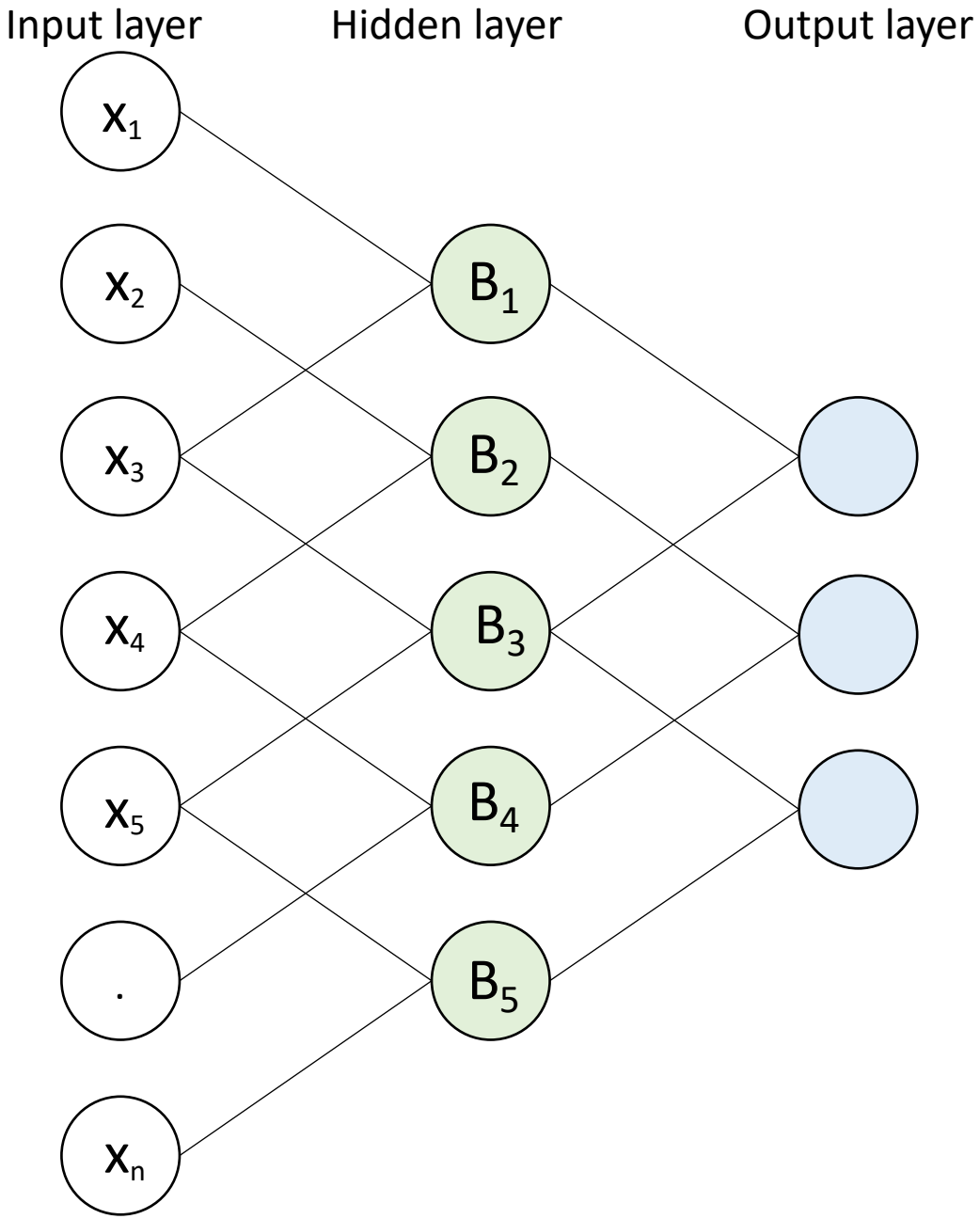
Building performance during hurricane

overall_building_condition	max_mph	age_yrs	number_of_stories	roof_shape	roof_cover	wall_cladding	structural_framing_system
0	80	14.0	1.0	Complex	Tile (clay)	Stucco	Wood-frame
1	80	29.0	2.0	Gable	Tile (clay)	Stucco	Wood-frame
0	85	27.0	2.0	Complex	Tile (clay)	Stucco	Wood-frame
1	85	24.0	1.0	Hip	Asphalt shingles (laminated)	Stucco	Wood-frame
1	85	30.0	2.0	Gable	Asphalt shingles (laminated)	Wood Siding	Wood-frame
...	...	...	...	...	...	...	...
1	95	40.0	1.0	Gambrel	Asphalt shingles (3-tab)	Wood Siding	Wood-frame
1	90	38.0	1.0	Gable	Asphalt shingles (3-tab)	Wood Siding	Wood-frame
1	105	22.0	2.0	Gable	Asphalt shingles (3-tab)	Vinyl Siding	Wood-frame
0	100	16.0	1.0	Hip	Asphalt shingles (3-tab)	Stucco	Wood-frame
2	105	50.0	2.0	Gable	Asphalt shingles (3-tab)	Wood Siding	Wood-frame

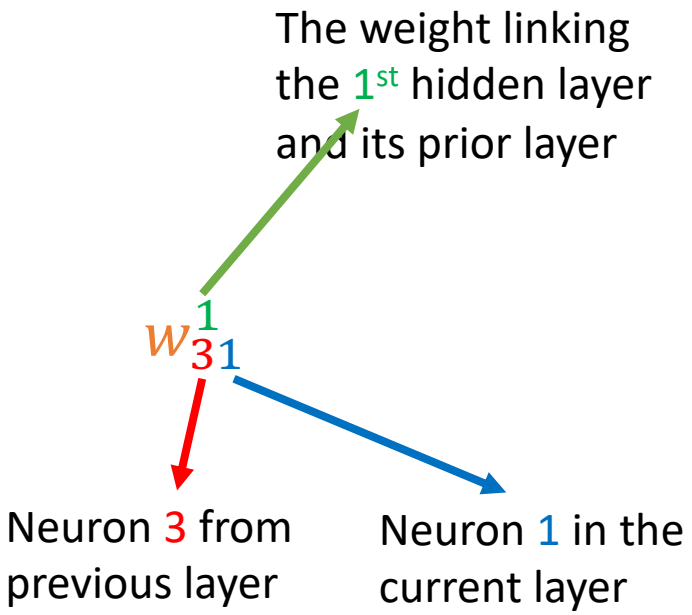
Will do the demo in Jupyter notebooks

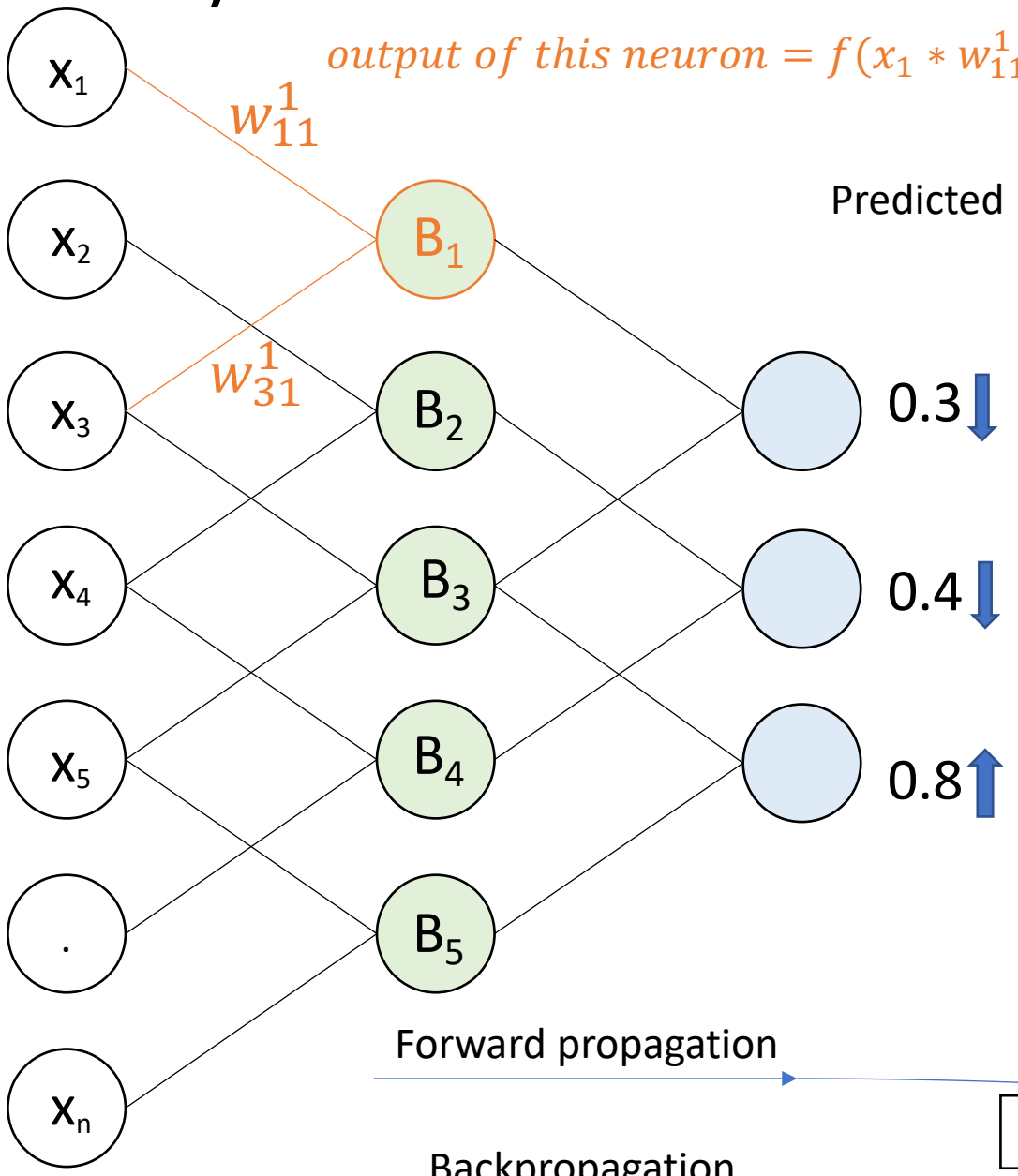
Part 2
Deep Learning

Multilayer perceptron (Neural network)



Multilayer perceptron (Neural network)

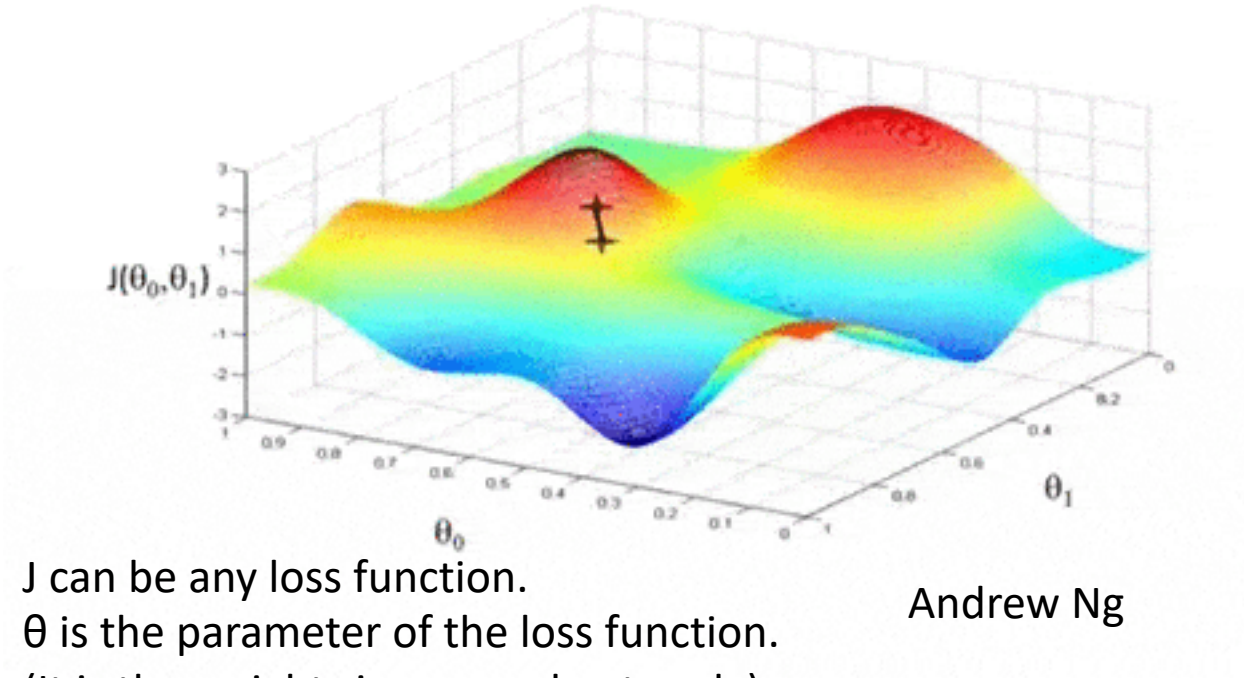




True Value (target)	Error
0	-0.3
0	-0.4
1	0.2

Multilayer perceptron (Neural network)

Gradient Descent

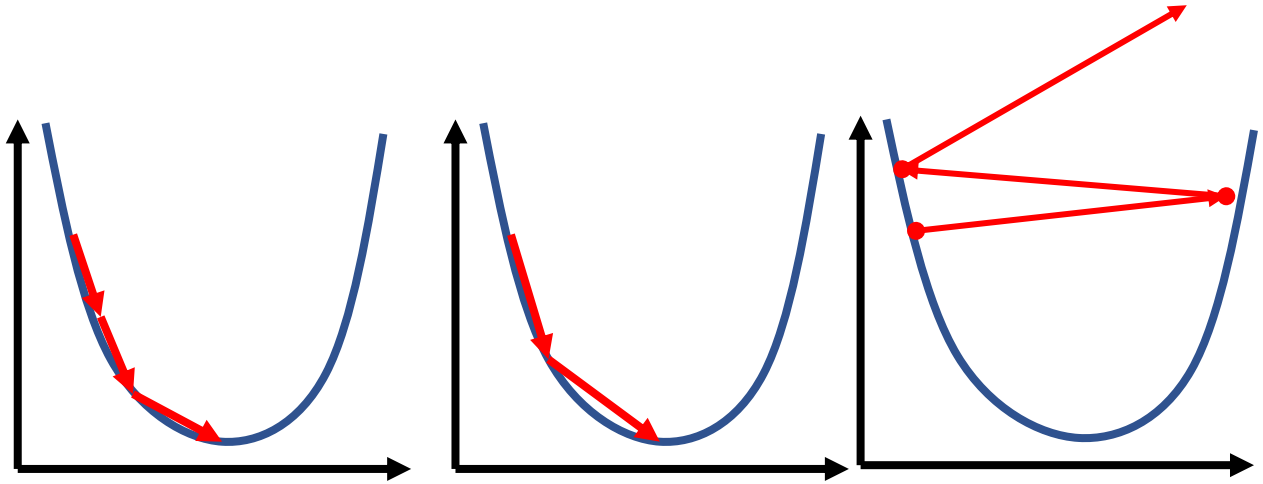


J can be any loss function.
θ is the parameter of the loss function.
(It is the weights in a neural network.)

Andrew Ng

1. **Compute the slope (gradient) at the current step**
2. **Make a move in the direction opposite to the slope**

Learning rate

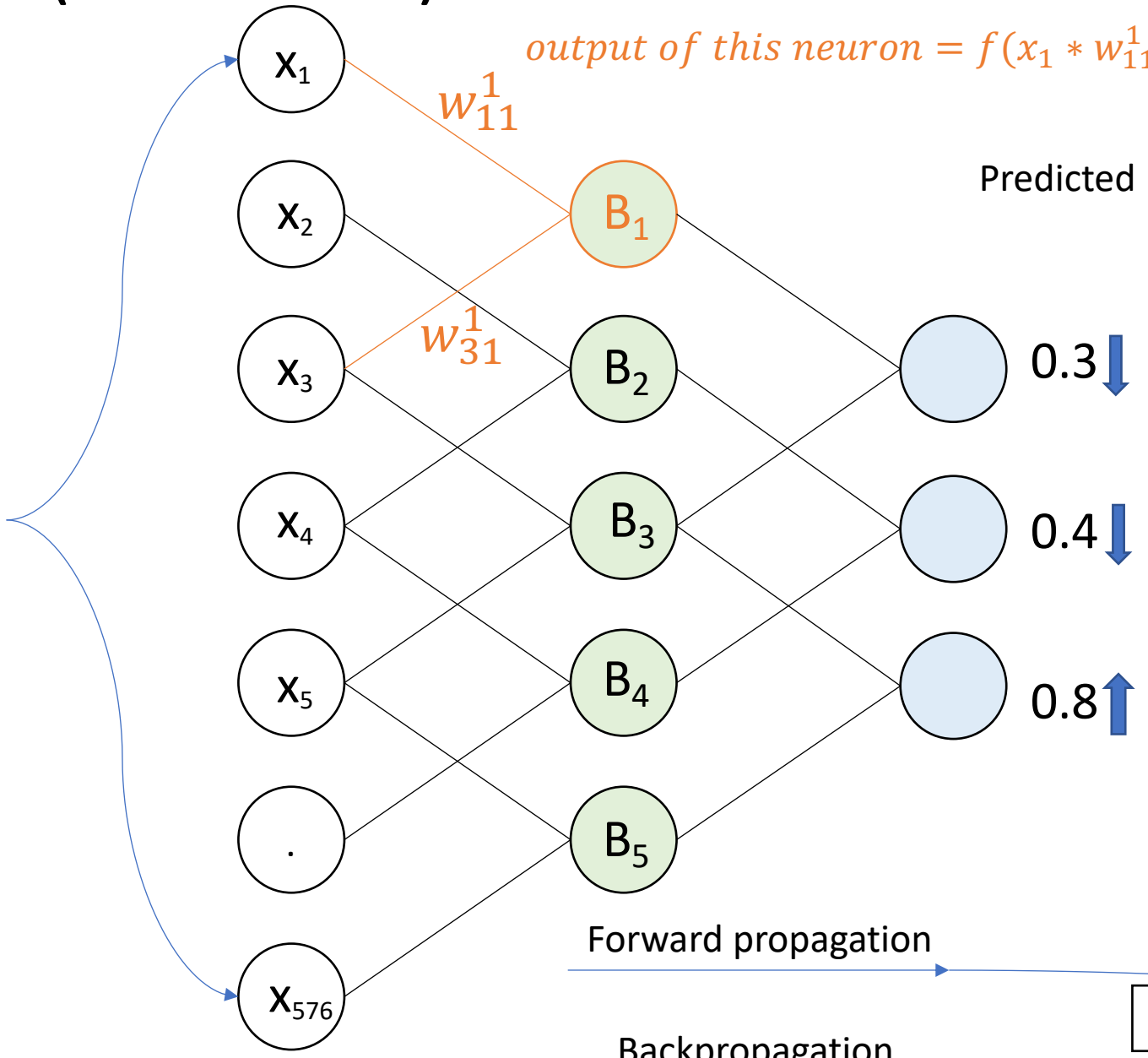


Too small **Good** **Overshoot**

Multilayer perceptron (Neural network)

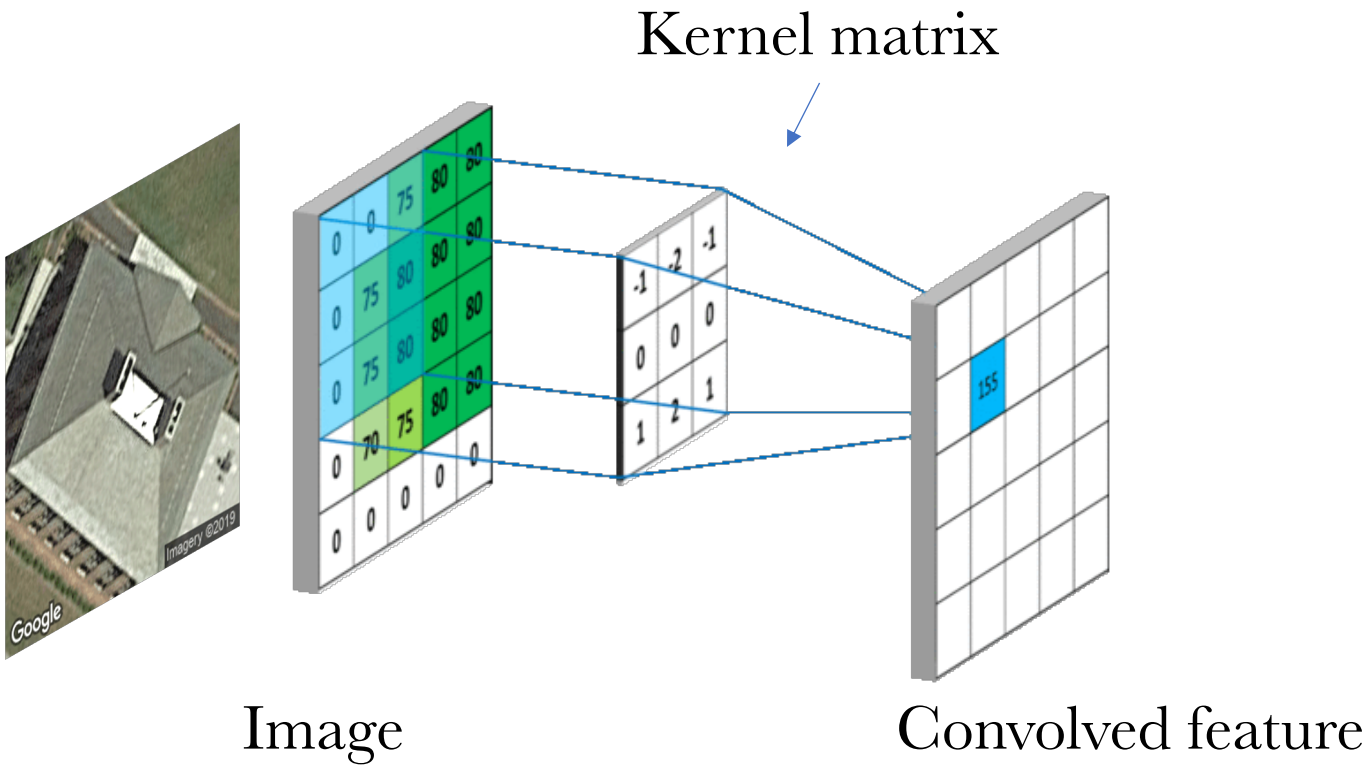
A 24x24 image can be expanded as a vector $[x_1, x_2, \ldots x_{576}]$

$$output\ of\ this\ neuron = f(x_1 * w_{11}^1 + x_3 * w_{31}^1 + B_1)$$



True Value (target)	Error
0	-0.3
0	-0.4
1	0.2

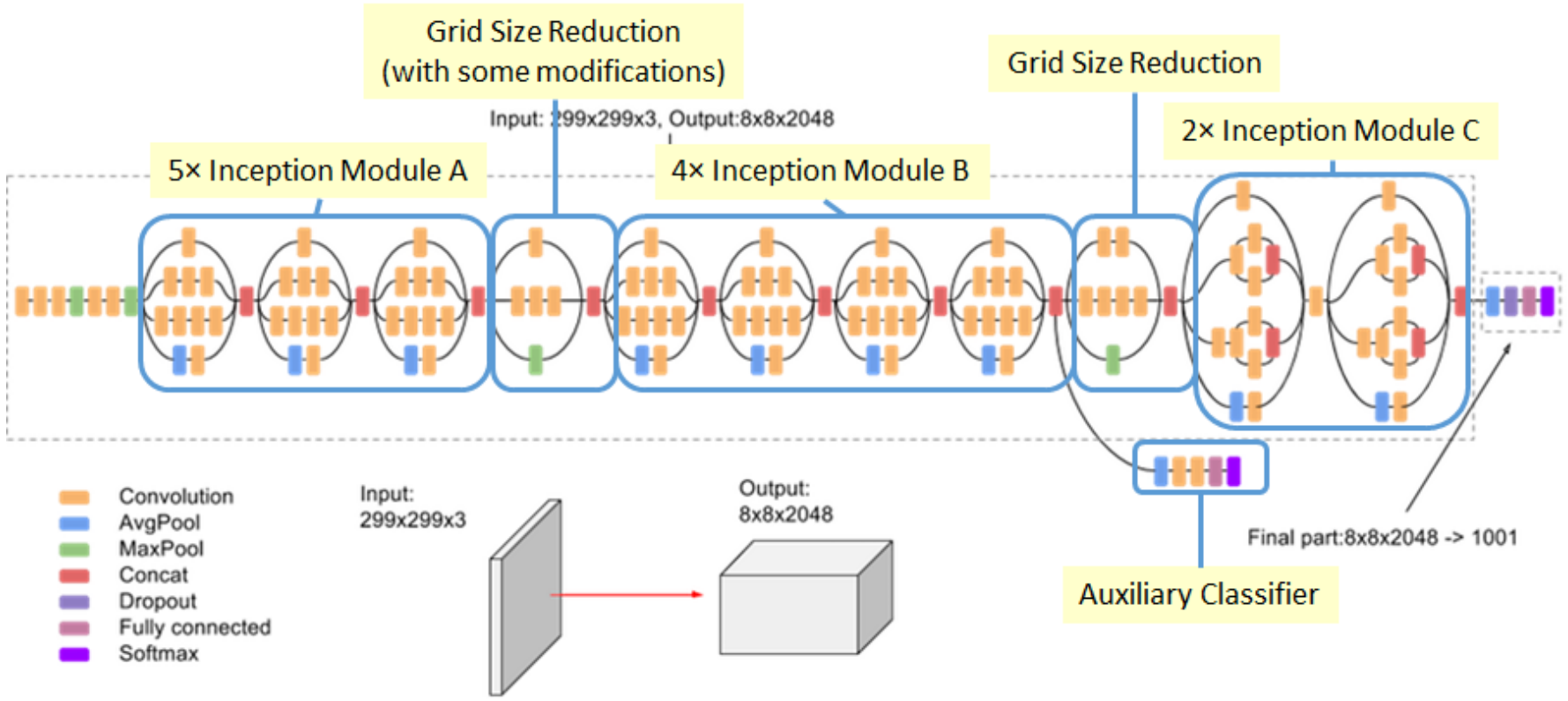
Convolutional neural network



A 2D convolution operation

Popular deep CNN architectures

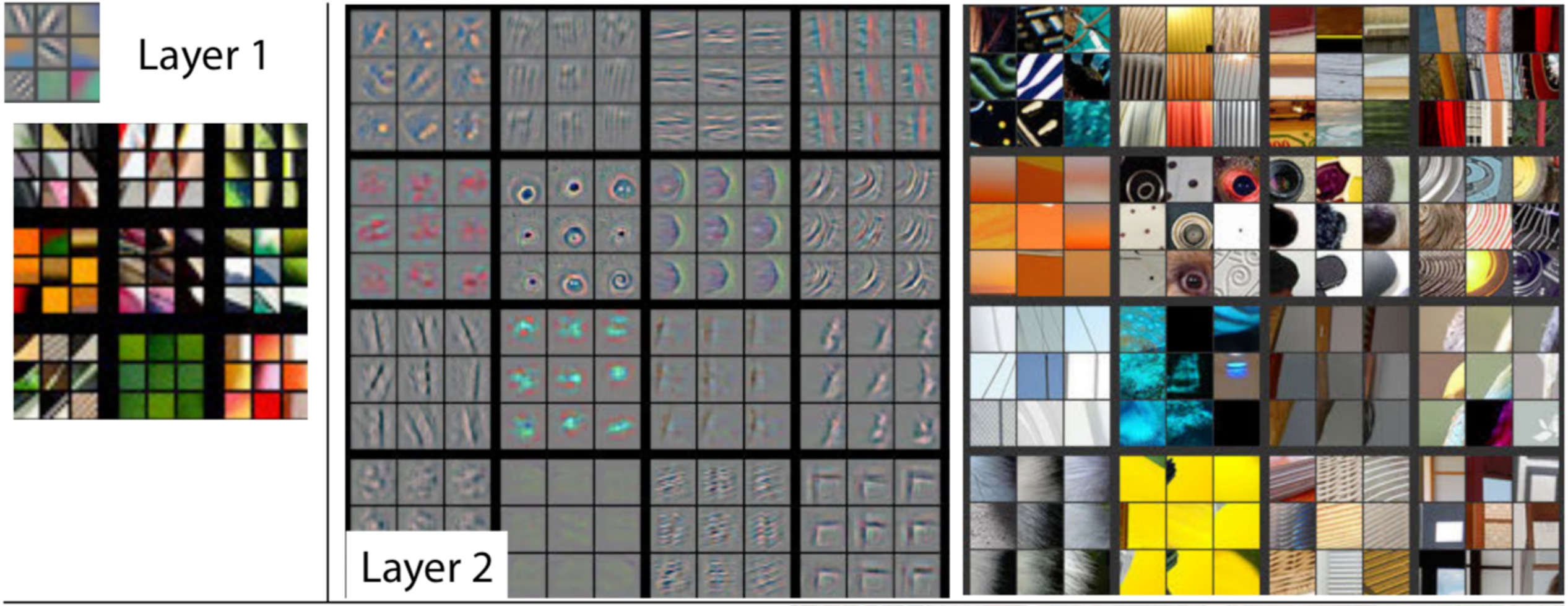
- AlexNet
- VGGNet
- GoogLeNet
- Microsfot ResNet
- Google Inception
- …



Example: Inception v3

Transfer Learning

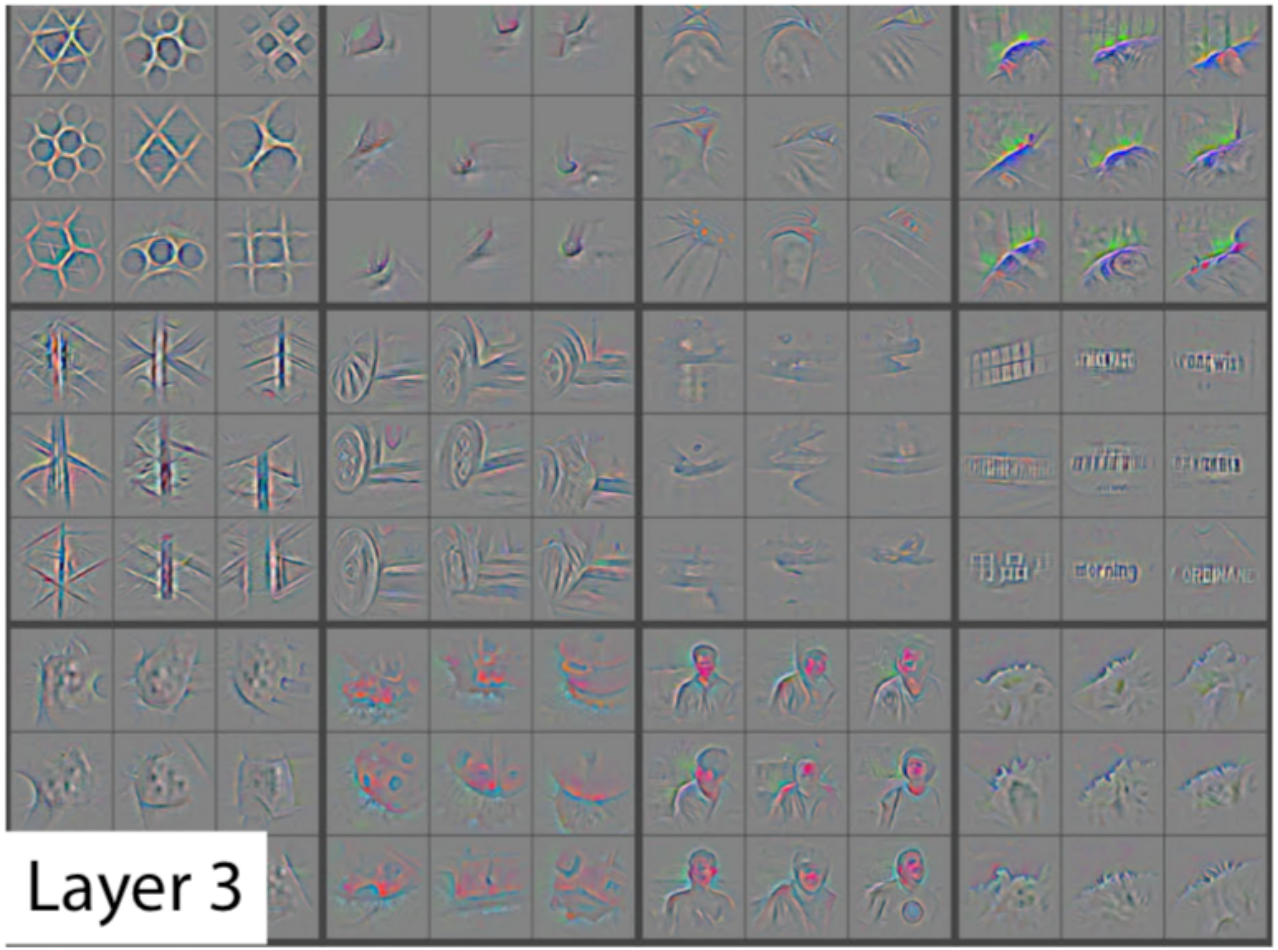
The first few layers of CNN detect general features: Edges, Corners, Circles, Blobs colors, …



Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." European conference on computer vision. Springer, Cham, 2014.

Transfer Learning

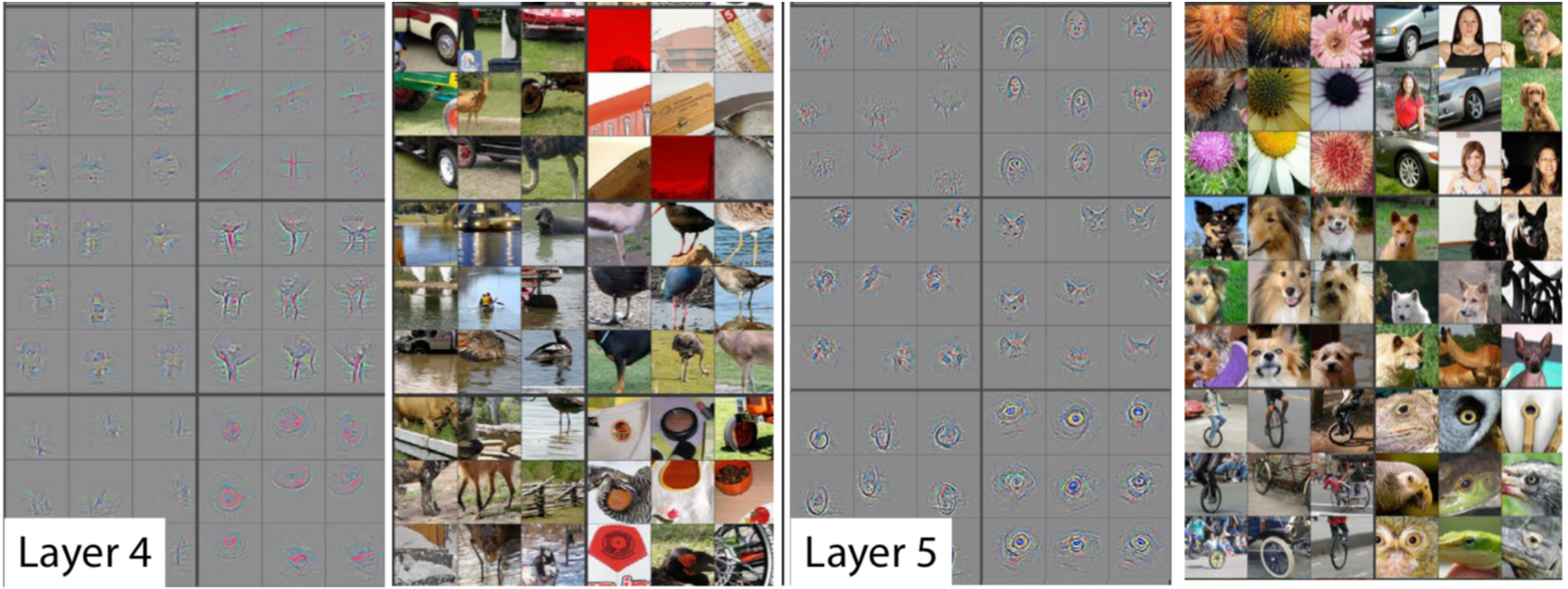
As go deeper into the CNN, it start to detect more concrete things such as eyes, faces, and full objects.



Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." European conference on computer vision. Springer, Cham, 2014.

Transfer Learning

More concrete things …



Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." European conference on computer vision. Springer, Cham, 2014.

Transfer Learning

The weights in a pretrained neural network is the leaned knowledge.

So a deep CNN trained on a large dataset contains knowledge (weights) that can be used to understand basic features in any given new image. This is the concept of transfer learning.

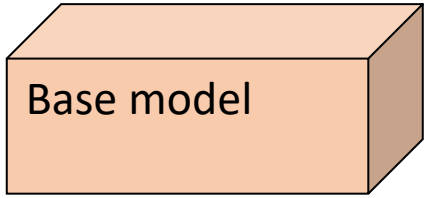
To do transfer learning, we

- Freeze the first layers of the pretrained neural network. These are the layers that detect general features that are common across all domains.

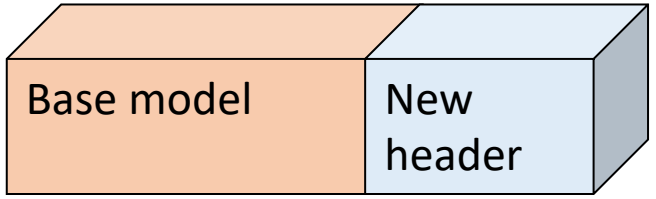
- Then we finetune the deeper layers with our own training data and add new layers to classify new categories included in our training dataset.

Transfer Learning: Fine tuning

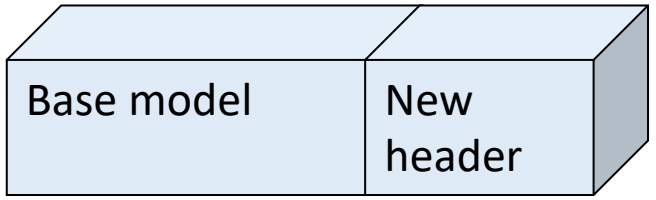
Take a pre-trained model (with learned weights) as base model



Add a header and train with the base model's weights frozen



Unfreeze the base model and train



Software and Platforms

scikit-learn (conventional ML)

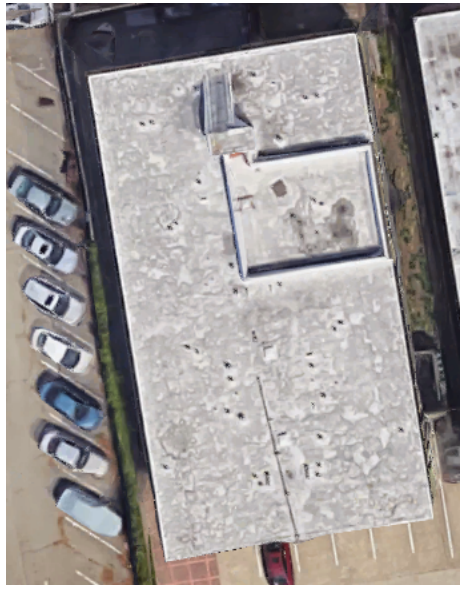
Tensorflow (deep neural networks)

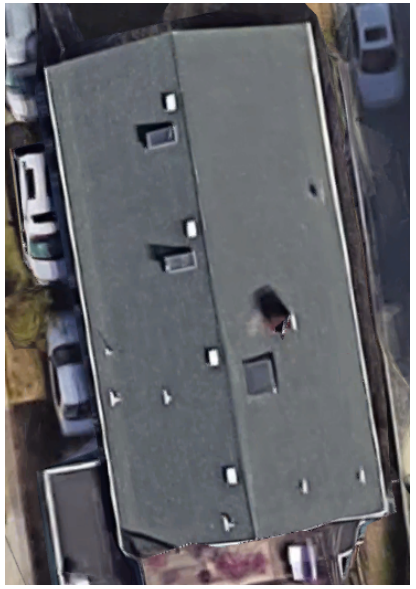
PyTorch (deep neural networks)

…

Demo: Roof shape classification

Will do this demo in a Jupyter notebook.





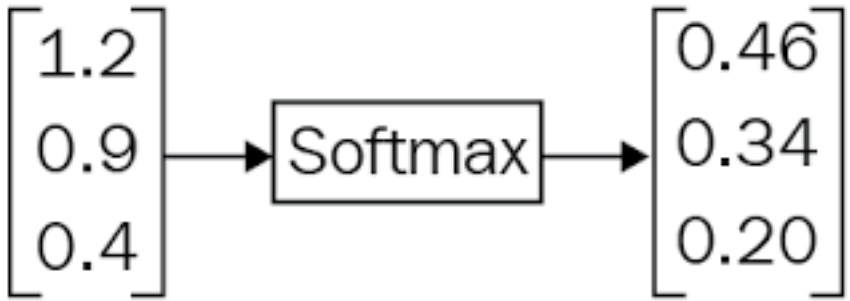
Flat

Gabled

Hipped

Softmax

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}}$$



P (class 1)

P (class 2)

P (class 3)

Probabilities, sum is 1.0