Reviewing some learning models

ML: Tree-based models

- Hypothesis space: Tree-based models

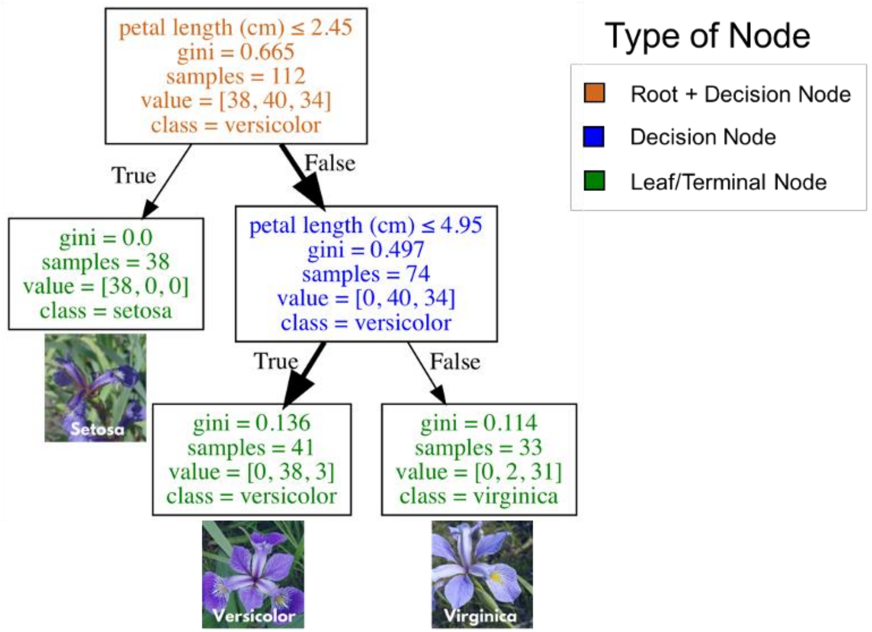
- Representation = Tree or Forest
- Learning algorithm = Greedy algorithms
  - Split the data based on an attribute that optimizes certain criterion
  - Examples:
    - CHAID, CART
    - ID3, C4.5, SLIQ, SPRINT
  - The learning algorithms different on the:
    - The considered attributes
    - Theirs types
    - How they are used (one or several time)
    - The form of the tree/forest
    - The criteria to optimize

ML: Tree-based models

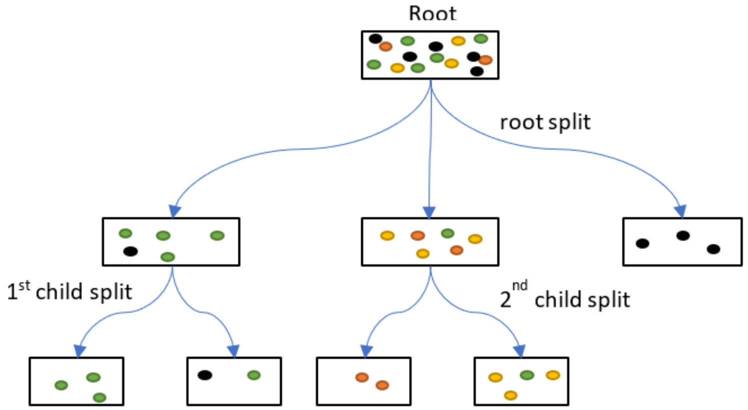
- The most important criteria in Trees is the notion of impurity
  - Used to judge the splits and to mesure the order/desorder

What class (species) is a flower with the following feature?

petal length (cm): 4.5



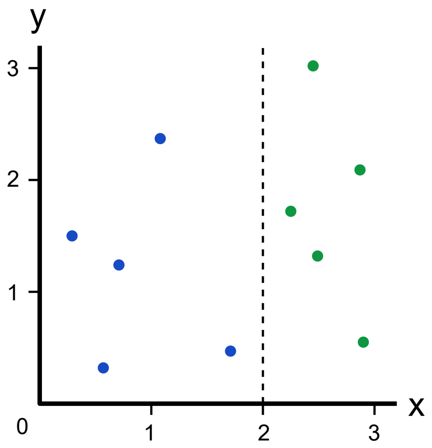
Species counts are: setosa=0, versicolor=38, virginica=3
Prediction is **versicolor** as it is the majority class

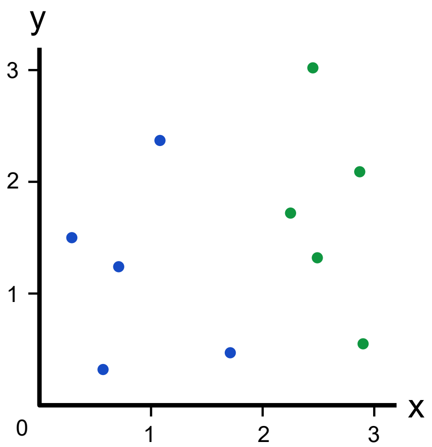


Based on attributes and their modalities

ML: Tree-based models

- Gini impurity
  - If we have *C* total classes and *p(i)* is the probability of picking a datapoint with class i, then the Gini Impurity is calculated as: $\mathrm{G} = \sum_{i=1}^{C} p(i) * (1 - p(i))$





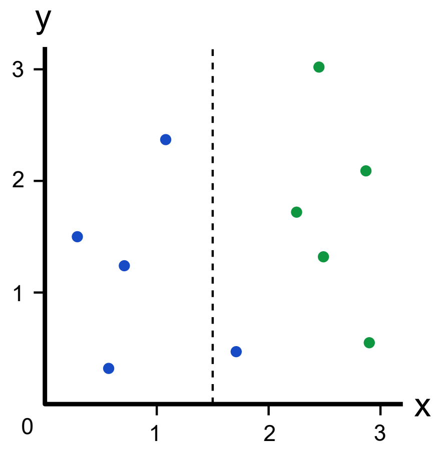
$$G_{left} = 1 * (1 - 1) + 0 * (1 - 0) = \boxed{0}$$

$$G_{right} = 0 * (1 - 0) + 1 * (1 - 1) = \boxed{0}$$

$$\begin{aligned} G &= p(1) * (1 - p(1)) + p(2) * (1 - p(2)) \\ &= 0.5 * (1 - 0.5) + 0.5 * (1 - 0.5) \\ &= \boxed{0.5} \end{aligned}$$

ML: Tree-based models

- Gini impurity
  - If we have *C* total classes and *p(i)* is the probability of picking a datapoint with class i, then the Gini Impurity is calculated as: $\mathrm{G} = \sum_{i=1}^{C} p(i) * (1 - p(i))$



$$G_{right} = \frac{1}{6} * (1 - \frac{1}{6}) + \frac{5}{6} * (1 - \frac{5}{6})$$
$$= \frac{5}{18}$$
$$= \boxed{0.278}$$

$$G_{left} = \boxed{0}$$

The information gain: how much Gini we removed

$$0.5 - 0.167 = \boxed{0.333}$$

Since Left Branch has 4 elements and Right Branch has 6, we get (Mean):

$$(0.4 * 0) + (0.6 * 0.278) = 0.167$$

ML: Tree-based models

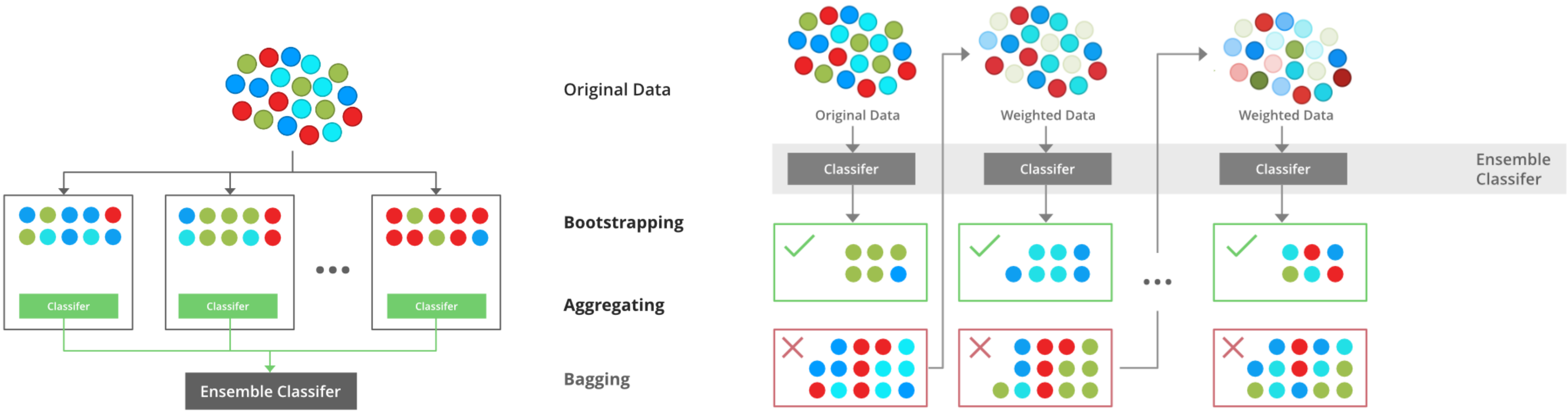
- Entropy impurity
  - If we have *C* total classes and *p(i)* is the probability of picking a datapoint with class i, then the Gini Impurity is calculated as: $\mathrm{E} = -\sum_{i=1}^{C} p(i) * log2(p(i))$
  - Maximum Gini or Entropy gain is used as splitting criterion

ML: Tree-based models

- Learning algorithms
  - Node impurity
    - Gini, Entropy, Gain, Khi2, ...
  - Bagging = Bootstrap + aggregating
    - **Subsamples (random forest) vs all samples (DT, ET, ...)**
  - Attribute selection: (random) greedy heuristics
    - **optimum split (random forest)**
    - **randomly split (extra trees)**
  - Nodes with homogenous class distribution are preferred
  - Boosting (more general)
  - Bias-variance Tradeoff
    - Multiple objectives: Misclassification error; Tree size (Prunning), depth, ..

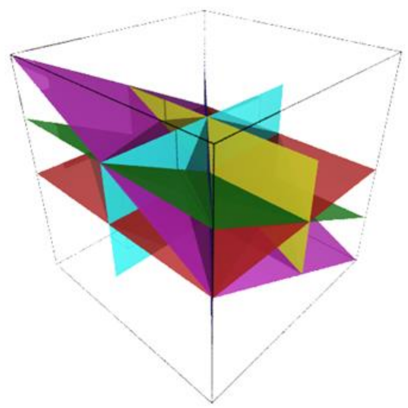
ML: Tree-based models

- Bagging vs Boosting



ML: Hyperplane models

- Hypothesis space: SVM models



- Representation = Hyperplane
- Learning algorithm = quadratic optimization algorithm with linear constraints

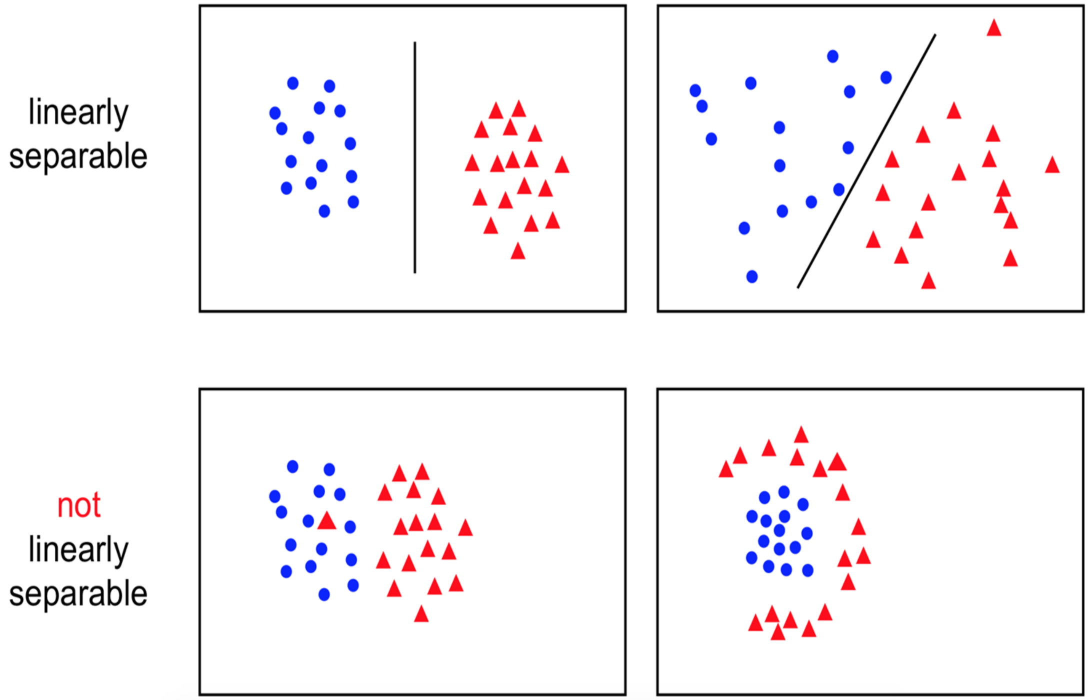
- A method used for regression and classification (mostly used in classification problems)
- SVM is fundamentally a binary classifier, but can be extended for multiclass problems
- Classification performed by learning a linear separator of the data

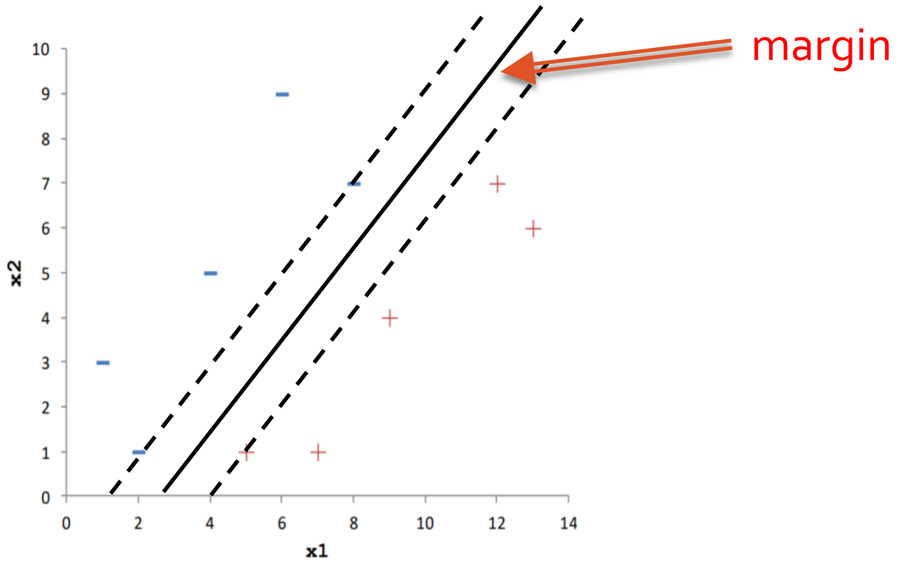
ML: SVM models

- Linear SVM

There exists an infinite number of linear separators which one is optimal?

Larger margin is preferred for generalization



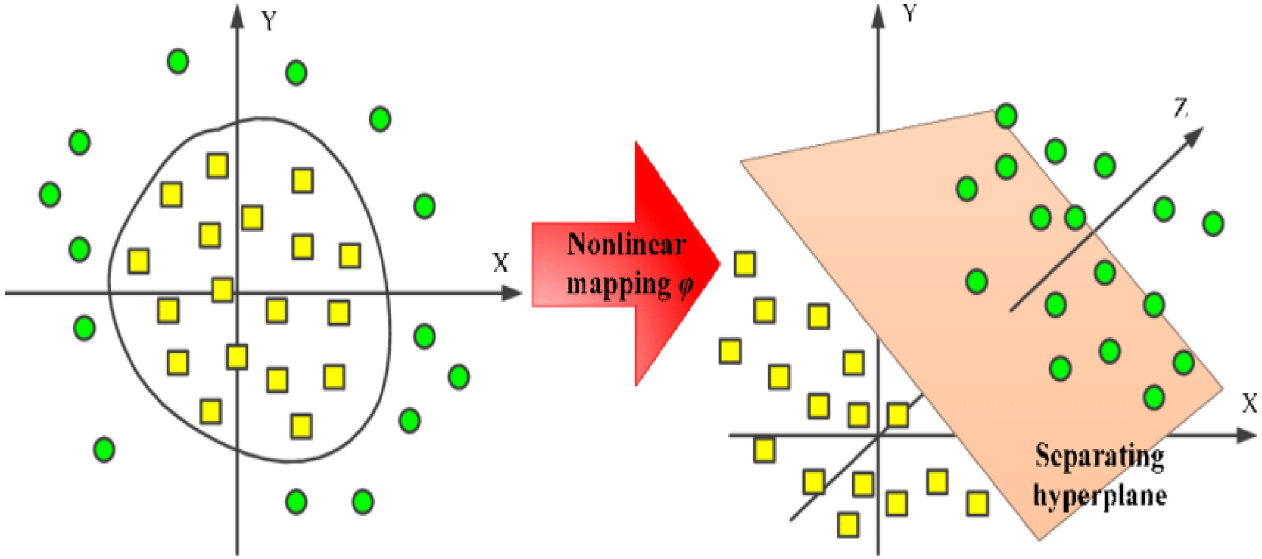


ML: SVM models

- Non-Linear SVM

  Kernel methods: sigmoid, polynomial, ...

  More difficult problem, we use metaheuristics



- Transform the data to a higher dimensional space where it can be separated by a linear hyperplane

- Learn a linear classifier for the new space : $f(x) = w^T \varphi(x) + b$

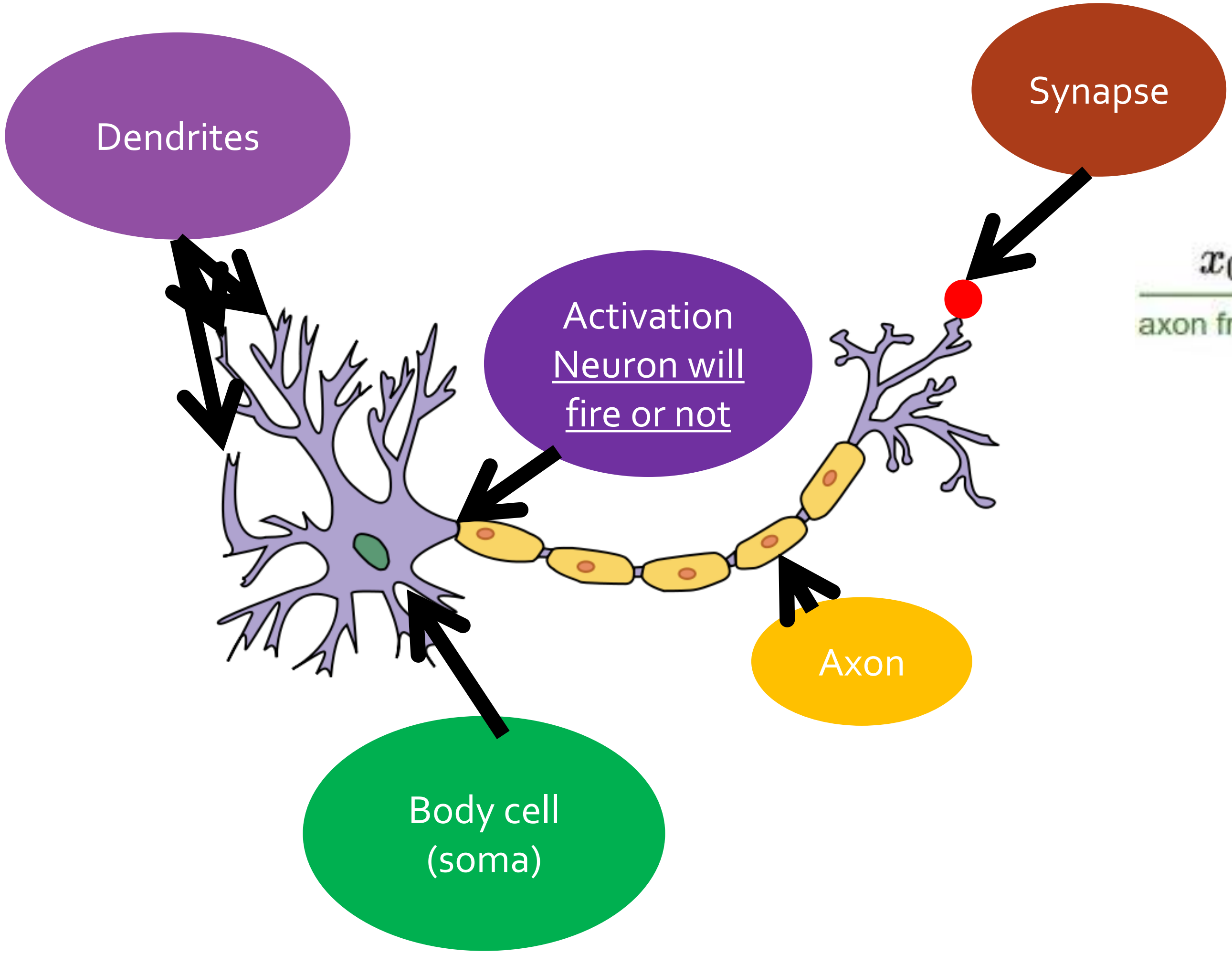
ML: Neural Network models

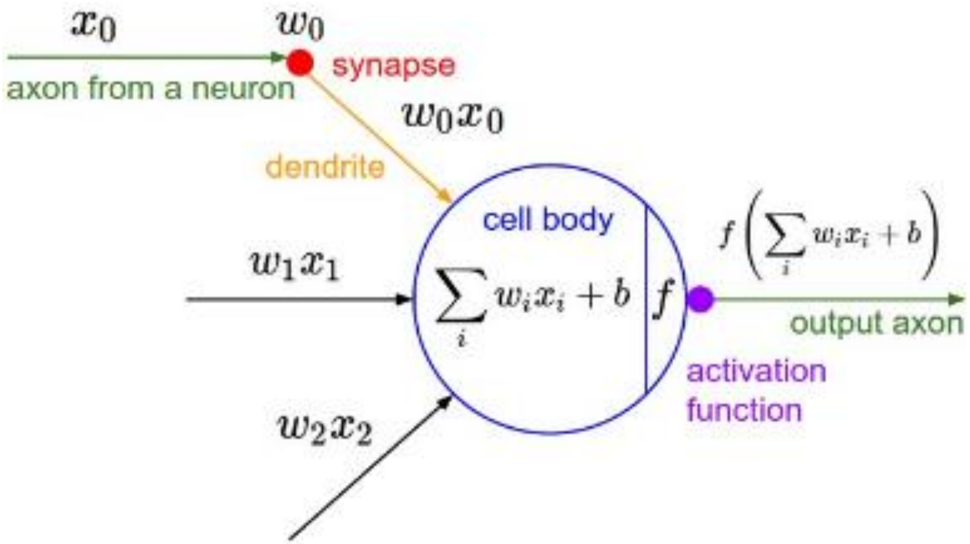
- Hypothesis space: NN models

- NN = Direct graph
- Objective function: classification error
- Learning algorithm = optimization algorithm (backpropagation using gradient, ...)

- NP-Hard and complex problem
- Metaheuristics have been widely used

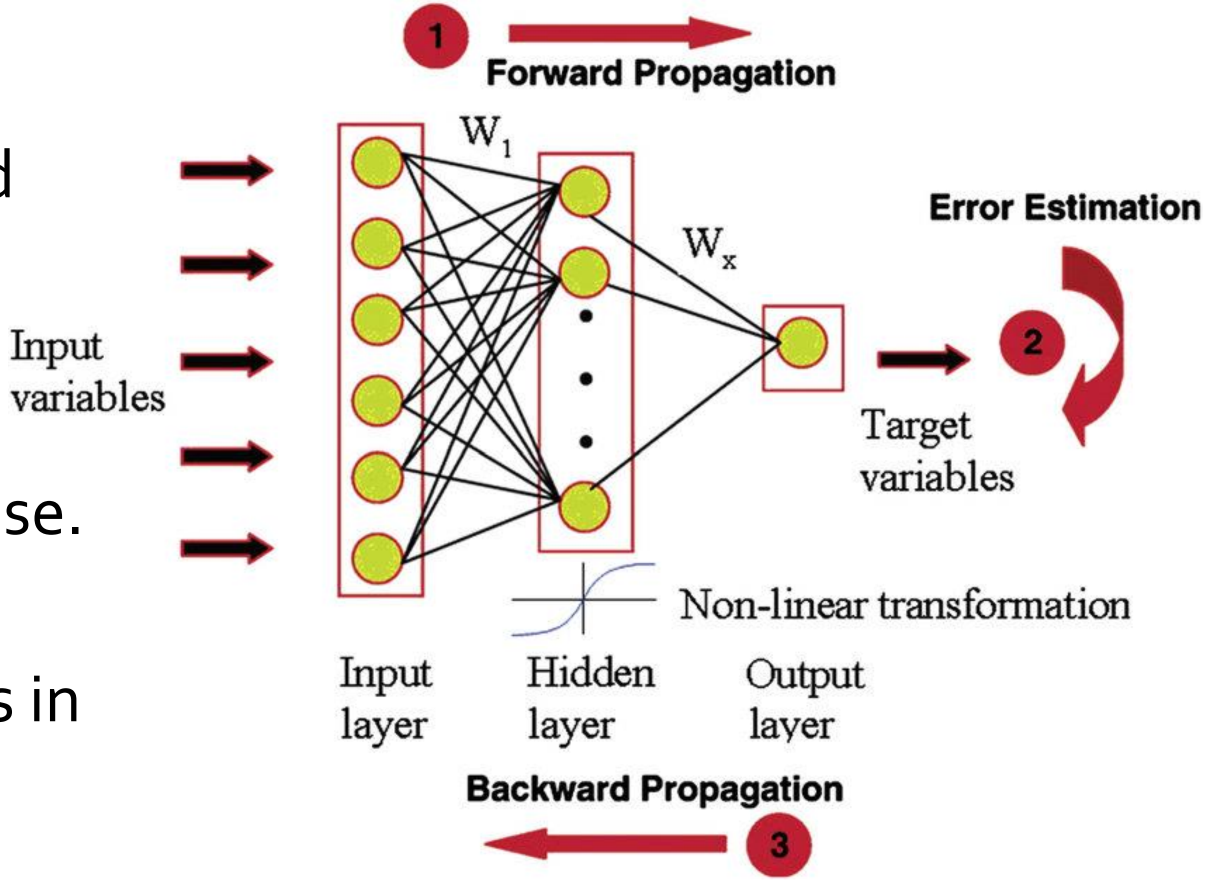
ML: Neural Network models





ML: Neural Network models

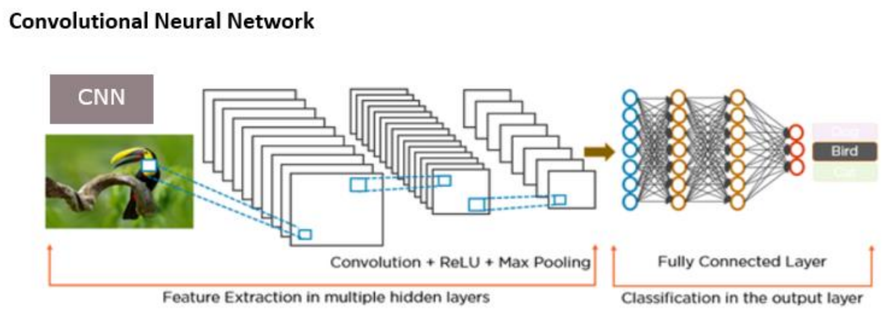
- Forward propagation: the weights are fixed and the input signal is propagated through the network layers, until it reaches the output.
- Error is estimated by comparing the network output and the desired response.
- Backward propagation: the error is propagated through the network layers in the backward direction

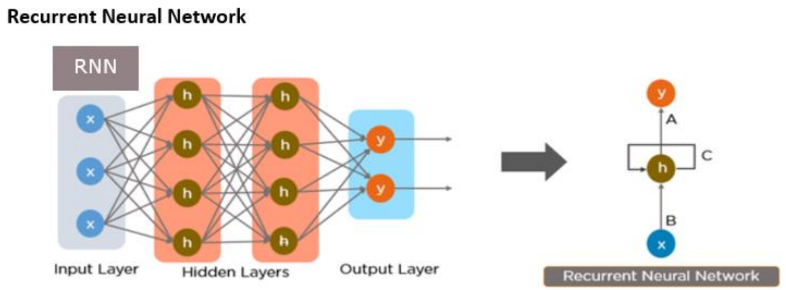


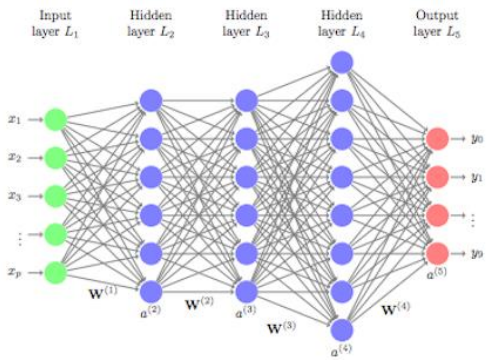
ML: Neural Network models

- Forward
  - Several architectures: FFNN, CNN, RNN

- Backpropagation
  - Several optimizer to deal with vanishing and exploding gradient:
    - Gradient Descent (GD), Stochastic GD, mini-batch GD, Momentum, Adam, AdaGrad, RMSPROP,







ML: Neural Network models

- Overfitting

You should always consider using regularization, unless you have a very large dataset,

- Reduce overfitting by training the network on more examples.

- Reduce overfitting by changing the complexity of the network.
    - Change network complexity by changing the network structure (number of weights).
    - Change network complexity by changing the network parameters (values of weights)

ML: Neural Network models

- Regularization Methods for Neural Networks
    - Weight Regularization (weight decay): Penalize the model during training based on the magnitude of the weights.
    - Activity Regularization: Penalize the model during training base on the magnitude of the activations.
    - Weight Constraint: Constrain the magnitude of weights to be within a range or below a limit.
    - Dropout: Probabilistically remove inputs during training.
    - Noise: Add statistical noise to inputs during training.
    - Early Stopping: Monitor model performance on a validation set and stop training when performance degrades.

ML: Neural Network models

- Some more specific recommendations for Multilayer Perceptrons and Convolutional Neural Networks.
  - Classical: use early stopping and weight decay (L2 weight regularization).
  - Alternate: use early stopping and added noise with a weight constraint.
  - Modern: use early stopping and dropout, in addition to a weight constraint.

- Some more specific recommendations for recurrent neural nets
  - Classical: use early stopping with added weight noise and a weight constraint such as maximum norm.
  - Modern: use early stopping with a backpropagation-through-time-aware version of dropout and a weight constraint.

Backup slides