

Statistical methods for big data in life sciences and health with R

Philippe Jacquet

Data Scientist at CADMOS

June 2018

CADMOS

- CADMOS: Center for Advanced Modeling Of Science

- CADMOS: Center for Advanced Modeling Of Science
- Free service for all UNIL, EPFL and UNIGE researchers

- CADMOS: Center for Advanced Modeling Of Science
- Free service for all UNIL, EPFL and UNIGE researchers
- Help with HPC parallel computing (code optimization) or Big Data analysis (pipeline development), including some computing power at SCITAS (the EPFL cluster)

- CADMOS: Center for Advanced Modeling Of Science
- Free service for all UNIL, EPFL and UNIGE researchers
- Help with HPC parallel computing (code optimization) or Big Data analysis (pipeline development), including some computing power at SCITAS (the EPFL cluster)
- Contact:
 - UNIL: philippe.jacquet@unil.ch, etienne.orliac@unil.ch
 - EPFL: vincent.keller@epfl.ch
 - UNIGE: jean-luc.falcone@unige.ch

Outline

TODAY

13:30 – 15:00 Lecture on Neural Network

15:00 – 15:30 Coffee break

15:30 – 17:00 Practicals on Neural Network

Outline

TODAY

13:30 – 15:00 Lecture on Neural Network

15:00 – 15:30 Coffee break

15:30 – 17:00 Practicals on Neural Network

TOMORROW (Room 321 - Amphipôle)

09:00 – 10:30 Lecture on Decision Tree and Random Forest

10:30 – 11:00 Coffee break

11:00 – 12:30 Practicals on Decision Tree and Random Forest

Outline

If you have any question during the lecture
you are welcome to ask

Outline

We will analyse the same dataset (iris data)
with all the methods

Outline

And present some applications
in health science

The iris dataset



Iris Versicolor



Iris Setosa



Iris Virginica

The iris dataset

4 measurements on 50 flowers in 3 species

The iris dataset

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.3	3.3	6.0	2.5	virginica

The iris dataset

input				output
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.3	3.3	6.0	2.5	virginica

The iris dataset

GOAL

Given an input (4 measurements)
predict its output (species)

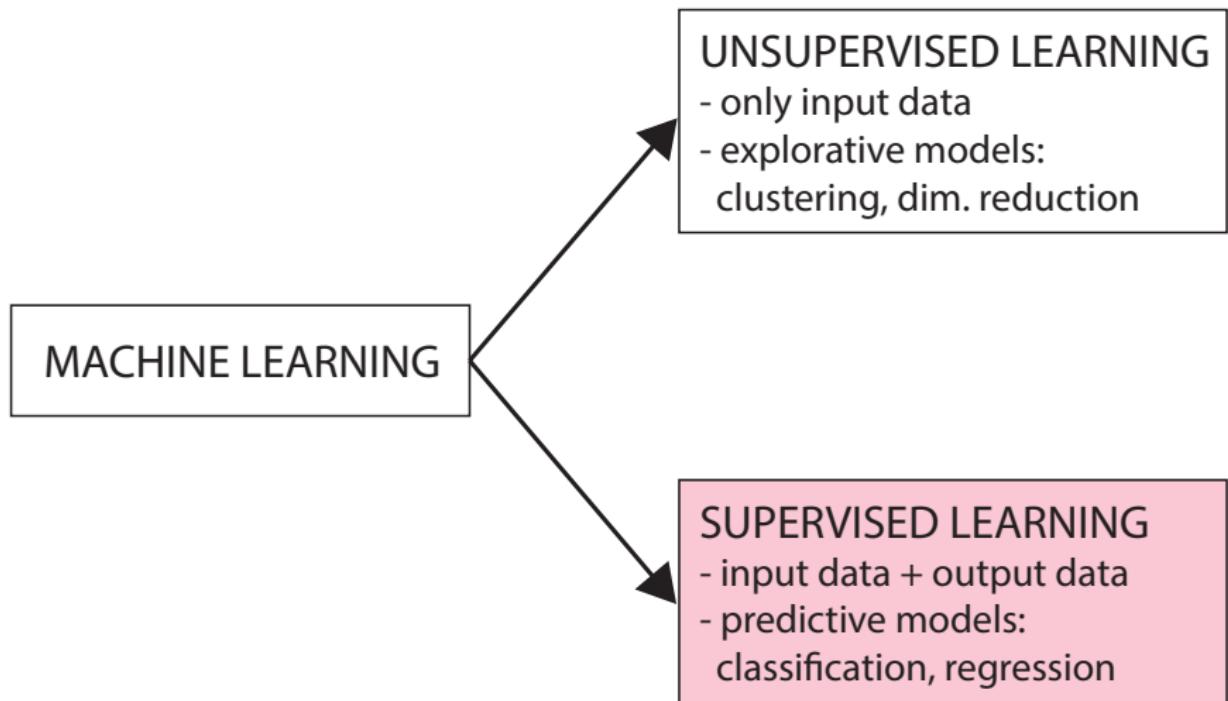
Machine learning



UNSUPERVISED LEARNING

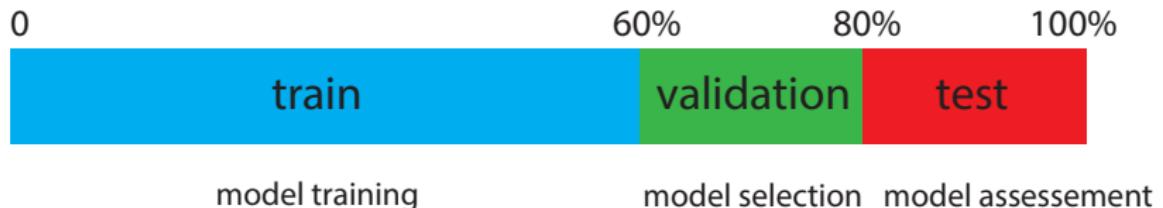
- only input data
- explorative models:
clustering, dim. reduction

Machine learning



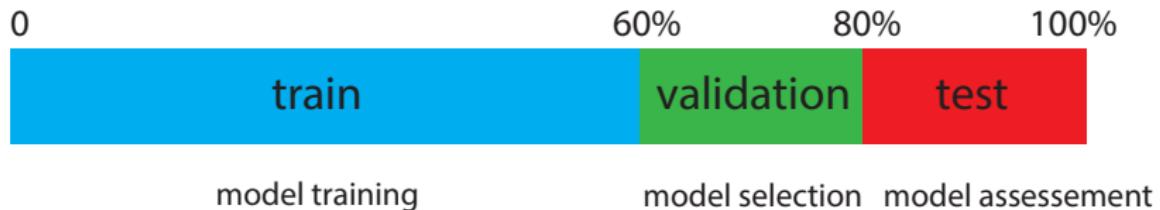
Supervised learning

Data splitting:



Supervised learning

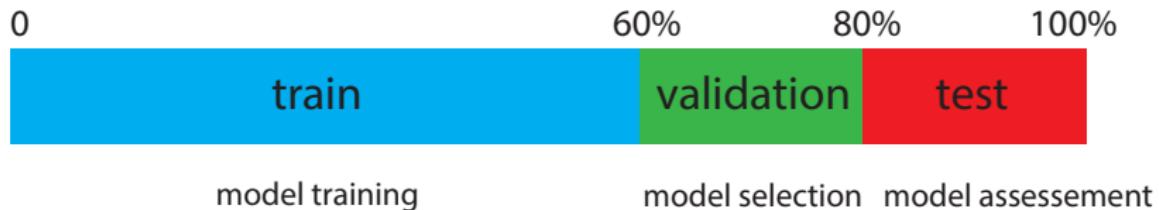
Data splitting:



- Model training: estimate the model parameters

Supervised learning

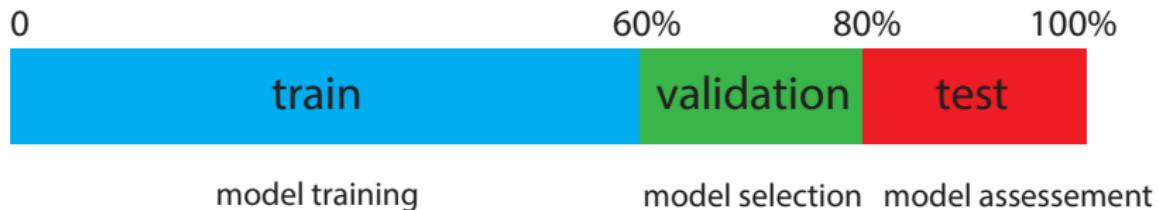
Data splitting:



- **Model training:** estimate the model parameters
- **Model selection:** estimate the performance of different models in order to choose the best one

Supervised learning

Data splitting:



- **Model training:** estimate the model parameters
- **Model selection:** estimate the performance of different models in order to choose the best one
- **Model assessment:** having chosen a final model, estimate its prediction accuracy on new data

Supervised learning

EXAMPLE

Model 1= Neural Network

Model 1= Neural Network

- Estimate the model parameters using the **training set**

Model 1= Neural Network

- Estimate the model parameters using the **training set**
- Measure the model accuracy based on the **validation set**

Model 2 = Decision Tree

Model 2 = Decision Tree

- Estimate the model parameters using the **training set**
- Measure the model accuracy based on the **validation set**

Model 3 = Random Forest

Model 3 = Random Forest

- Estimate the model parameters using the **training set**
- Measure the model accuracy based on the **validation set**

Model Selection

Model Selection

Select the best model based on its accuracy

Model Selection

Select the best model based on its accuracy

Neural Network = Decision Tree

Model Assessment

Model Assessment

Measure best model accuracy on the **test set**

Model Assessment

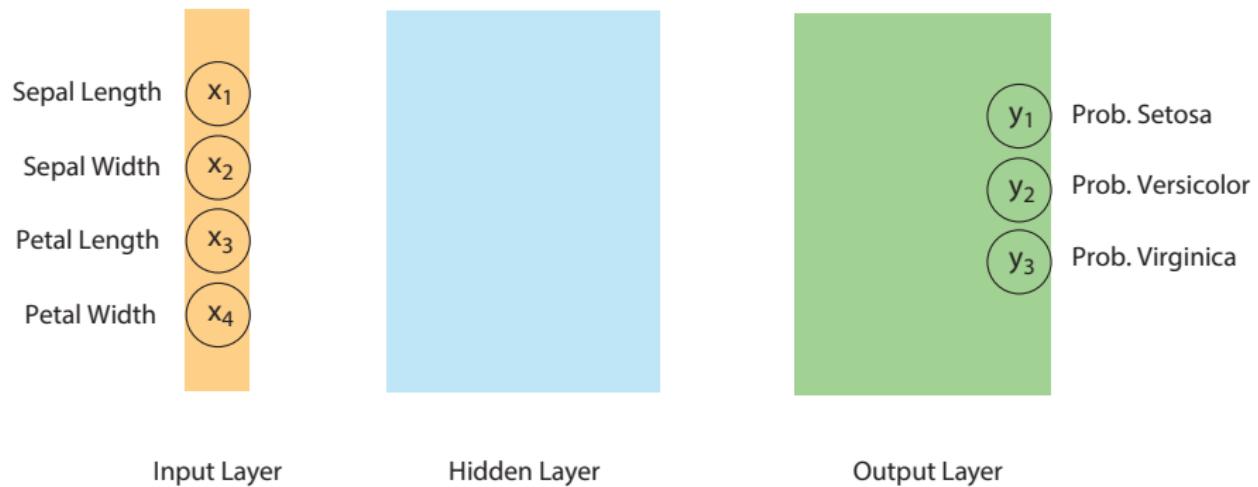
Measure best model accuracy on the **test set**

THE END

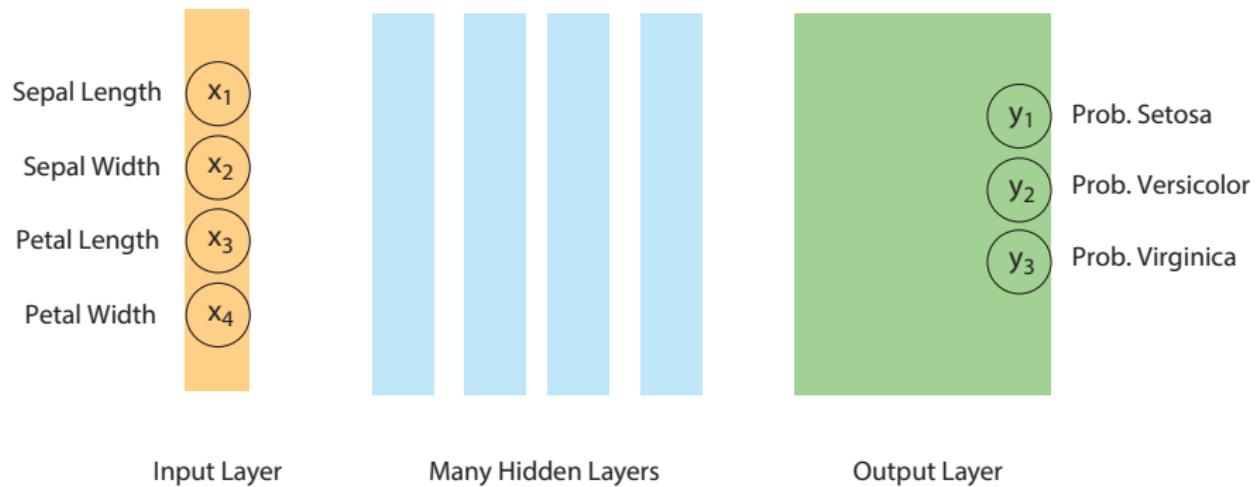
Neural network for the iris data

How to build a neural network
for the iris data ?

Neural network for the iris data

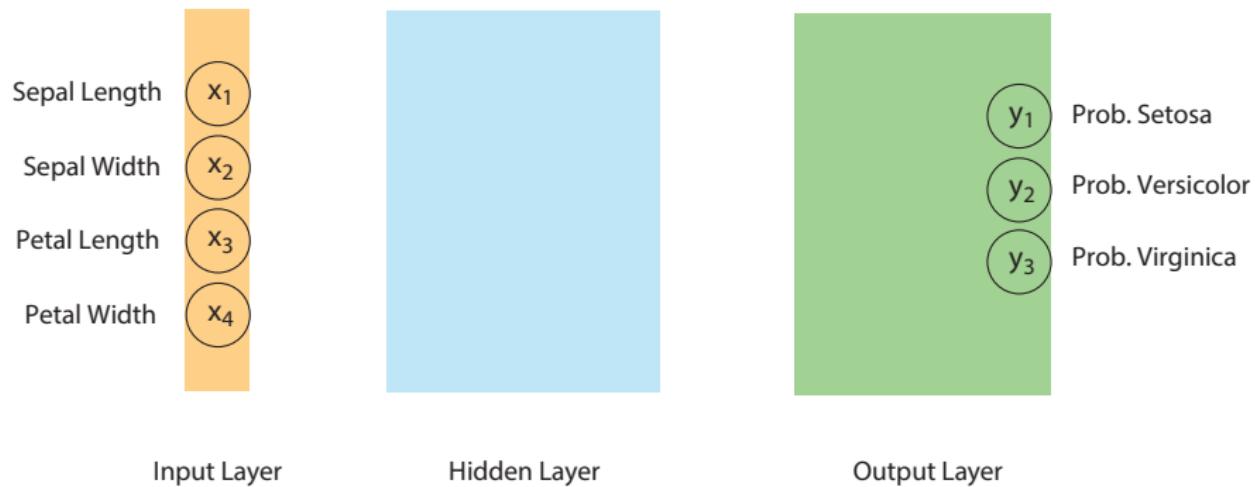


Neural network for the iris data

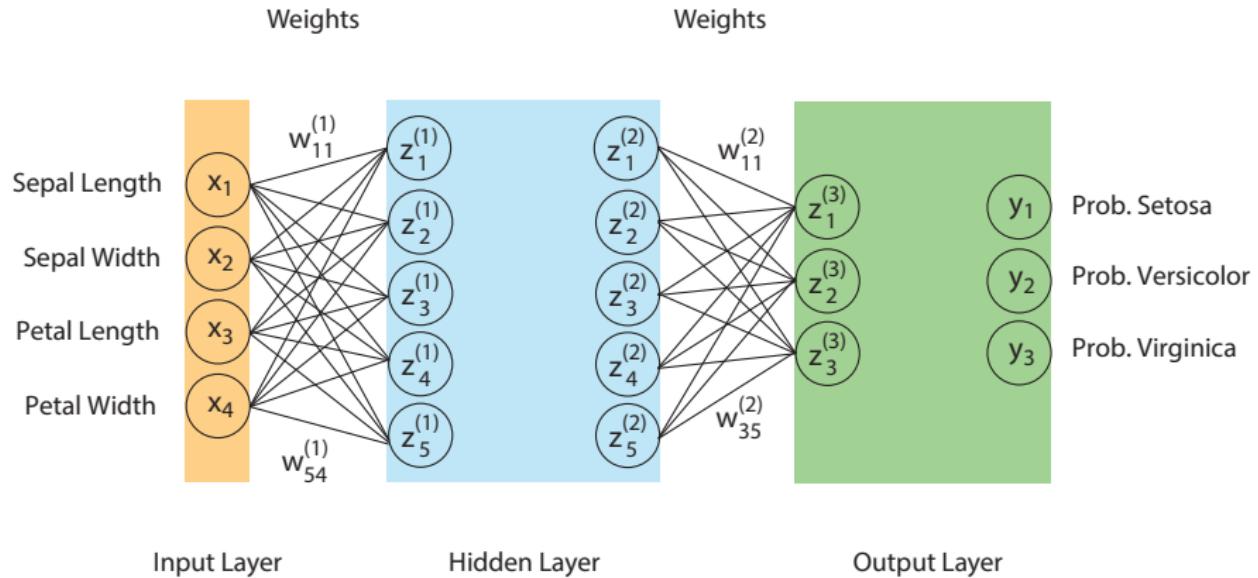


Deep Learning = More than 1 Hidden Layer

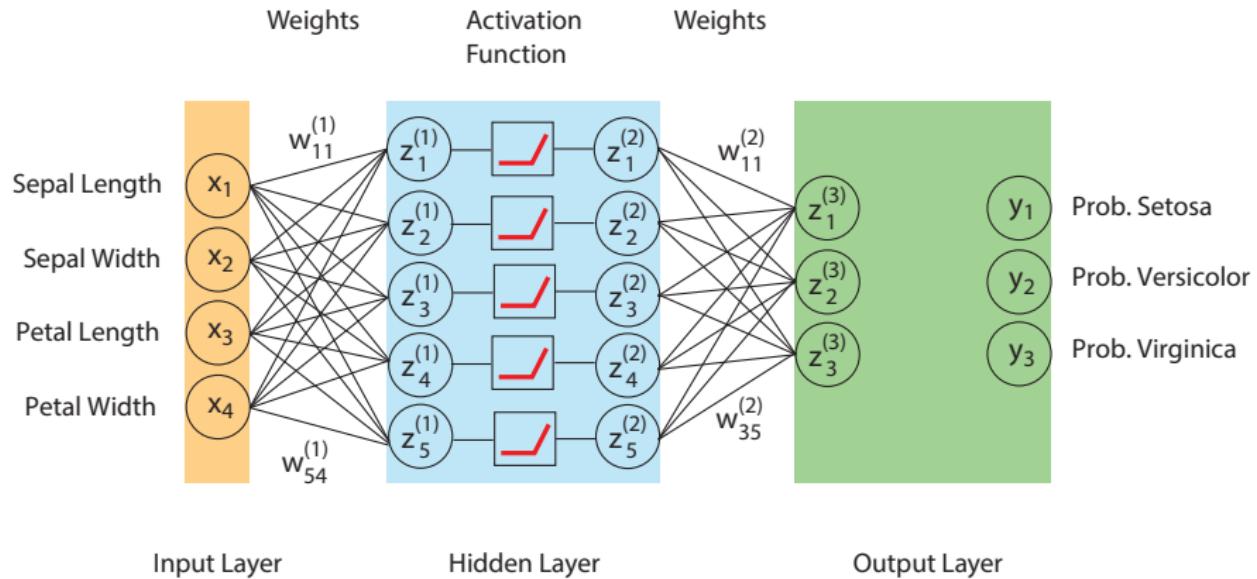
Neural network for the iris data



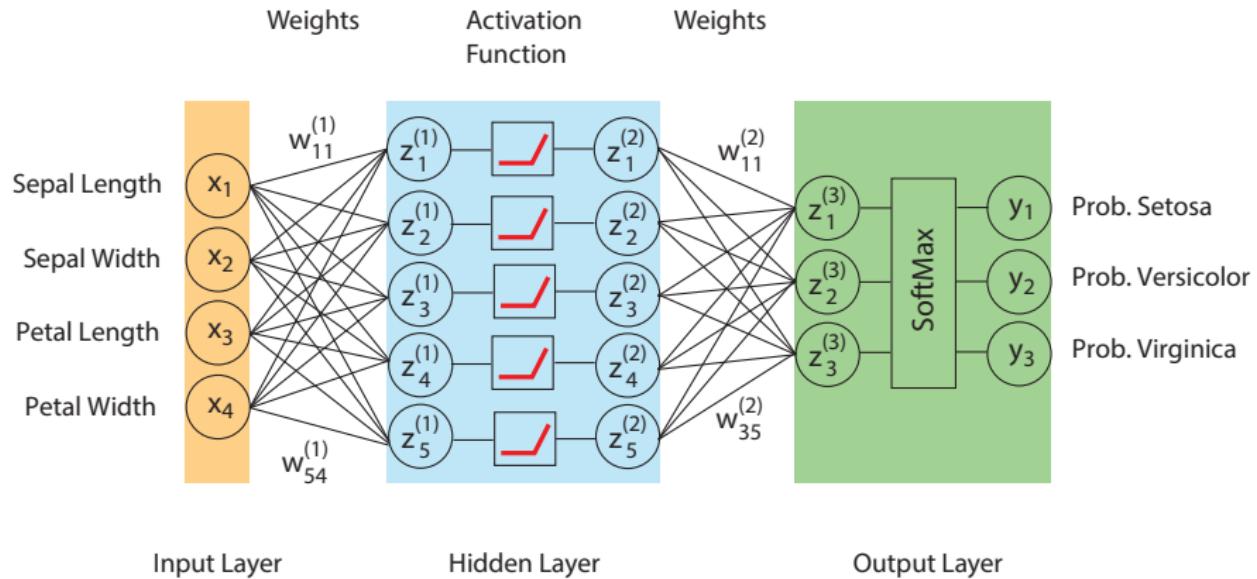
Neural network for the iris data



Neural network for the iris data

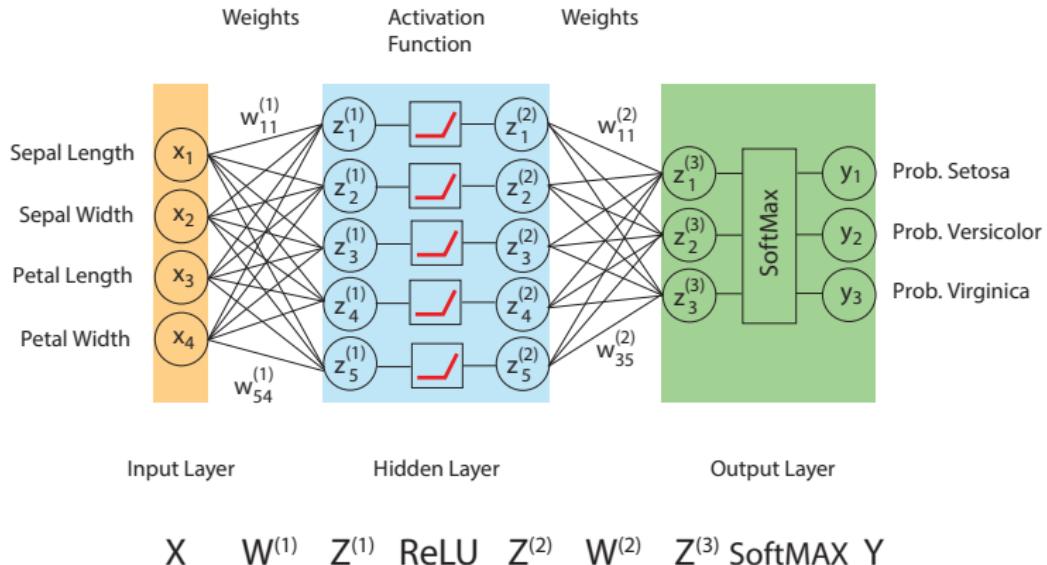


Neural network for the iris data

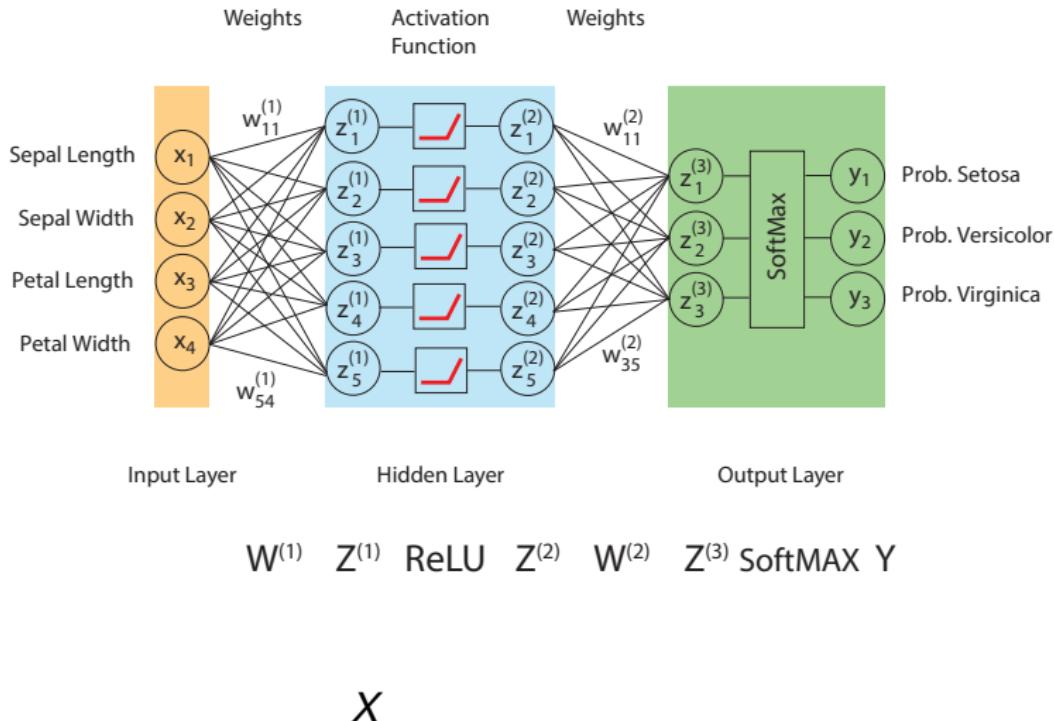


What is
the mathematical formula
for this neural network ?

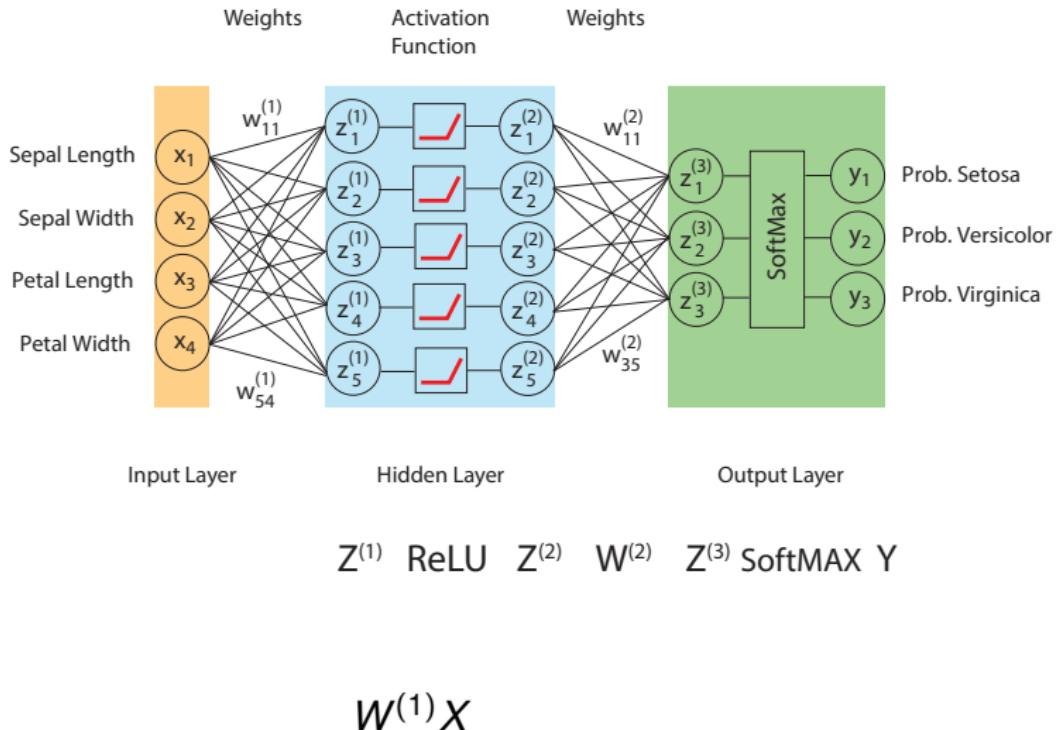
Neural network for the iris data



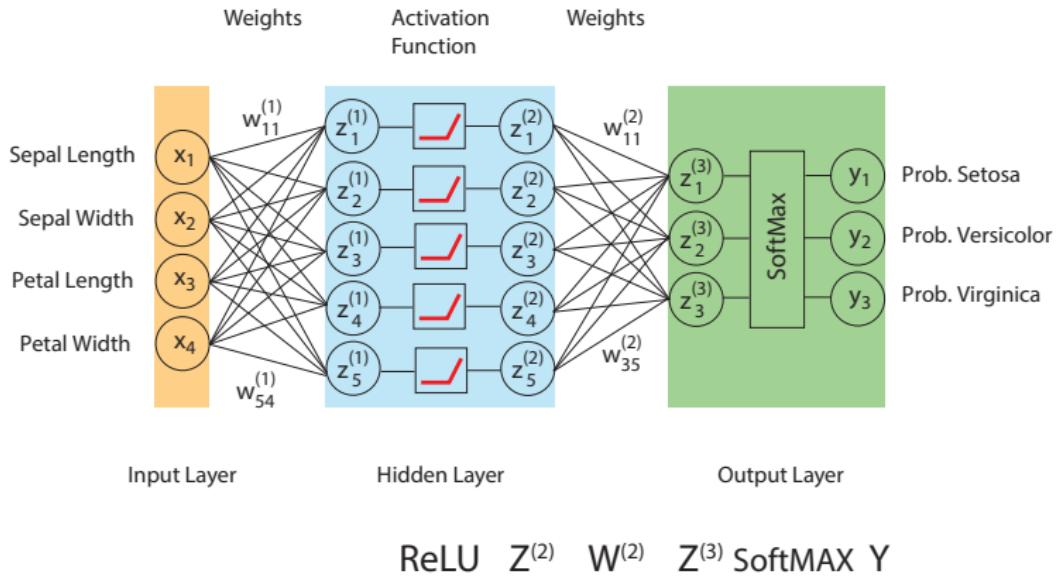
Neural network for the iris data



Neural network for the iris data

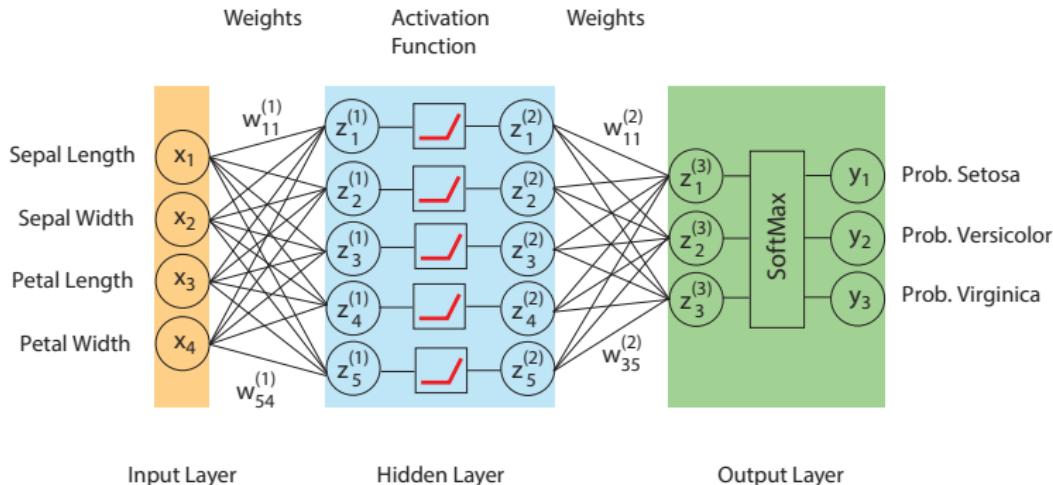


Neural network for the iris data



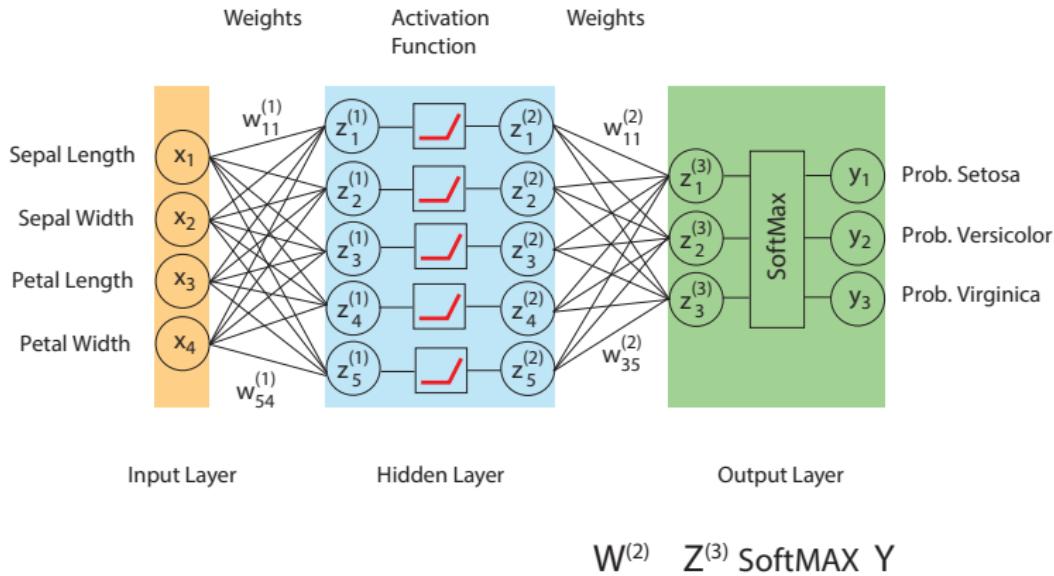
$$\underbrace{W^{(1)} X}_{Z^{(1)}}$$

Neural network for the iris data



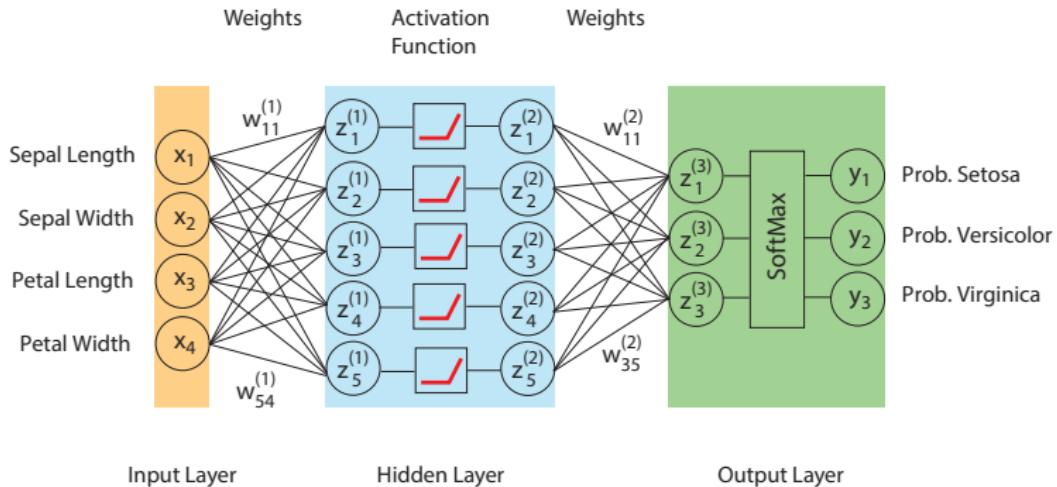
$$\overbrace{\text{ReLU}(W^{(1)}X)}^{Z^{(1)}} \quad Z^{(2)} \quad W^{(2)} \quad Z^{(3)} \quad \text{SoftMAX} \quad Y$$

Neural network for the iris data



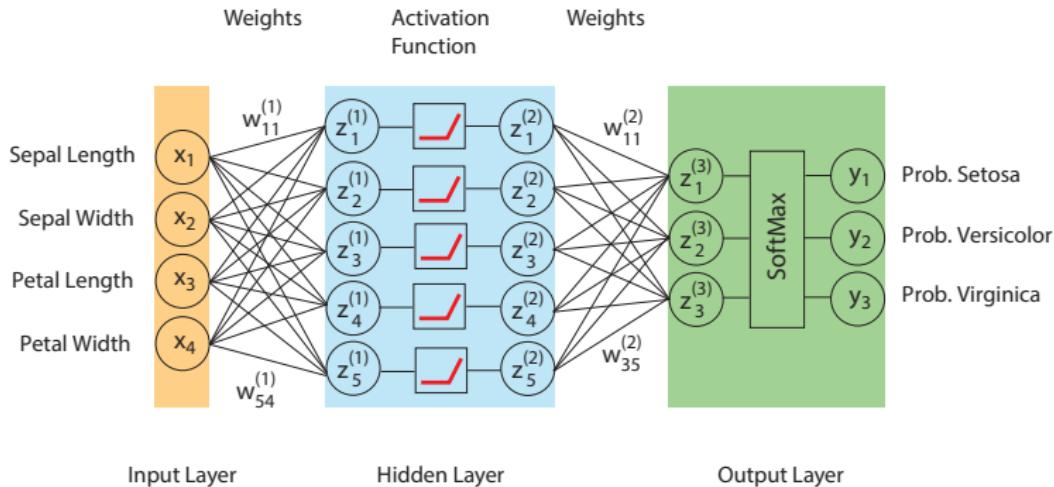
$$\underbrace{\text{ReLU}(W^{(1)}X)}_{Z^{(2)}} \quad Z^{(1)}$$

Neural network for the iris data



$$W^{(2)} \underbrace{\text{ReLU}(W^{(1)} X)}_{Z^{(2)}} \overbrace{Z^{(1)}}^{W^{(2)}}$$

Neural network for the iris data

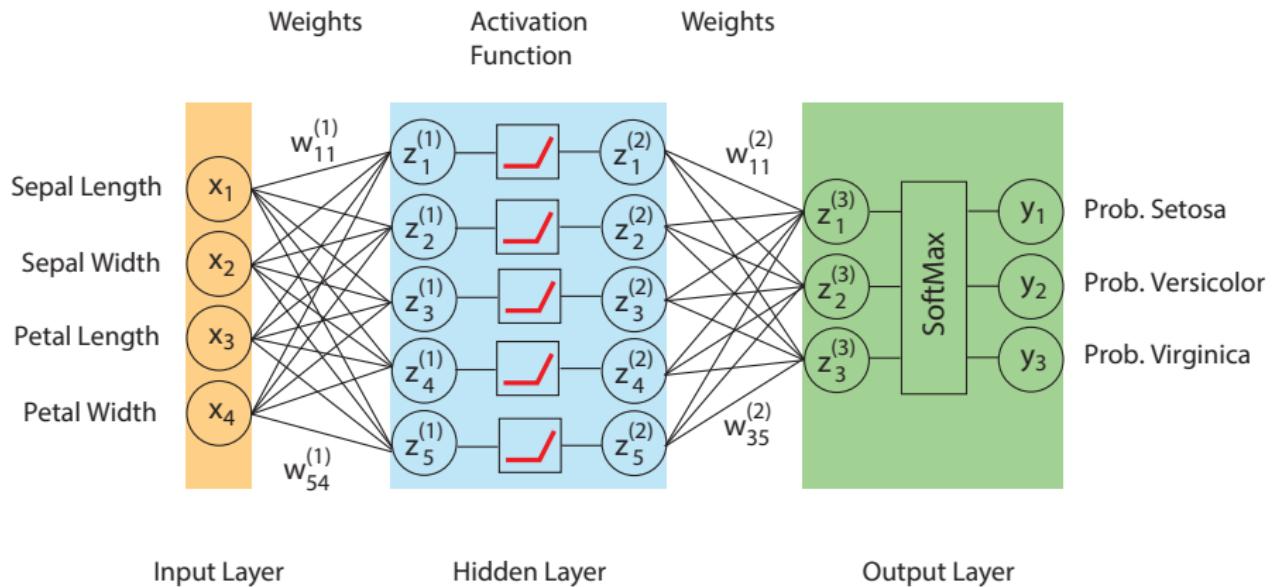


$$Y = \text{SoftMax}(\underbrace{W^{(2)} \text{ReLU}(W^{(1)} X)}_{Z^{(2)}})$$
$$Z^{(1)}$$

Neural network for the iris data

In summary

Neural network for the iris data



$$Y = f(X) = \text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)}X))$$

Forward propagation

In forward propagation, we go from X to Y by applying a series of functional transformations :

$$Y = \text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)}X))$$

Forward propagation

In forward propagation, we go from X to Y by applying a series of functional transformations :

$$Y = \text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)}X))$$

Questions:

- How do we determine the weights $W^{(1)}$ and $W^{(2)}$?

Forward propagation

In forward propagation, we go from X to Y by applying a series of functional transformations :

$$Y = \text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)}X))$$

Questions:

- How do we determine the weights $W^{(1)}$ and $W^{(2)}$?
- Why do we need the activation function ReLU ?

Forward propagation

In forward propagation, we go from X to Y by applying a series of functional transformations :

$$Y = \text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)}X))$$

Questions:

- How do we determine the weights $W^{(1)}$ and $W^{(2)}$?
- Why do we need the activation function ReLU ?
- Why do we use the SoftMax function at the end ?

Weight matrices

How do we determine
the weight values ?

Weight matrices

We want to choose the weights to make
the best predictions possible

Weight matrices

We will use an algorithm
to find these weights

Weight matrices

This algorithm needs
some initial weight values

Weight matrices

How do we choose
the initial weight values ?

Weight matrices

The naive choice

Weight matrices

We set all the weights to zero

Weight matrices

All the predictions are the same:

$P=1/3$ for the three species

Weight matrices

The weights are updated by the same amount
and thus cannot get unequal values

Weight matrices

The network cannot learn

Weight matrices

A good choice

Weight matrices

Random initialisation around 0

Weight matrices

We choose the weights according to
a normal distribution $\mathcal{N}(0, \sigma^2)$

Weight matrices

We choose the weights according to

a normal distribution $\mathcal{N}(0, \sigma^2)$

The value of σ^2 affects the performance
(speed and accuracy)

Weight matrices

Two common choices

$$\sigma^2 = \frac{1}{N_{in}} \text{ (Xavier)}$$

$$\sigma^2 = \frac{2}{N_{in} + N_{out}} \text{ (Glorot & Bengio)}$$

N_{in} = the number of nodes going in the weight matrix W

N_{out} = the number of nodes going out the weight matrix W

Probabilistic framework

Linear model:

$$Y = W X$$

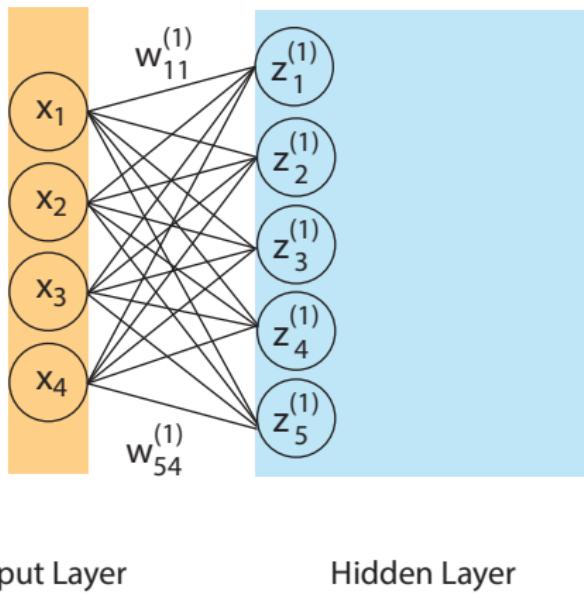
$$Y_i = \sum_{j=1}^{N_{\text{in}}} W_{ij} X_j, \quad i = 1, \dots, N_{\text{out}}$$

Assumptions: $X_j \sim \mathcal{N}(0, \sigma_x^2)$ and $W_{ij} \sim \mathcal{N}(0, \sigma^2)$

The condition $V(Y_i) = V(X_j)$ leads to the values for σ^2 :
Xavier and Glorot (when including backpropagation)

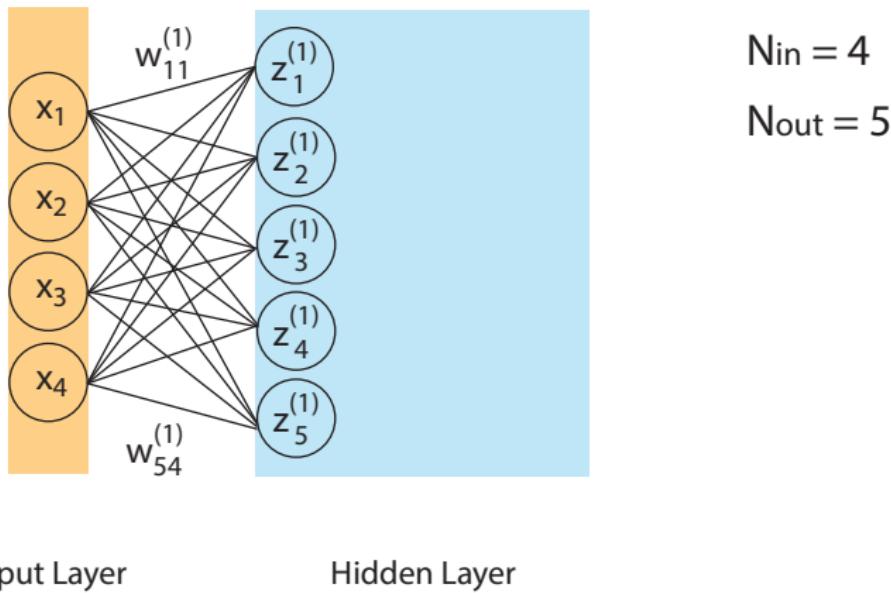
Weight matrices

EXAMPLE



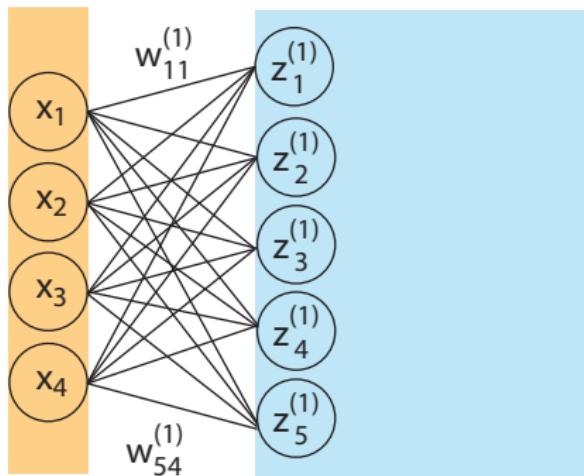
Weight matrices

EXAMPLE



Weight matrices

EXAMPLE



$N_{in} = 4$

$N_{out} = 5$

$$\begin{pmatrix} 0.5 & 0.1 & -0.2 & -0.4 \\ -0.4 & 1.0 & 0.5 & 1.0 \\ -0.2 & -0.2 & -0.5 & -0.1 \\ 0.2 & 0.7 & 0.3 & 0.2 \\ 0.6 & 0.6 & 0.1 & -0.4 \end{pmatrix}$$

Input Layer

Hidden Layer

STEP 1

$$Y = \text{SoftMax}(\mathcal{W}^{(2)} \text{ReLU}(\mathcal{W}^{(1)} X))$$

STEP 1

$$Y = \text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)}X))$$

input

output

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.3	3.3	6.0	2.5	virginica

STEP 1

$$Y = \text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)} X))$$

Matrix multiplication ($\sigma = 0.5$):

$$W^{(1)} X = \begin{pmatrix} 0.5 & 0.1 & -0.2 & -0.4 \\ -0.4 & 1.0 & 0.5 & 1.0 \\ -0.2 & -0.2 & -0.5 & -0.1 \\ 0.2 & 0.7 & 0.3 & 0.2 \\ 0.6 & 0.6 & 0.1 & -0.4 \end{pmatrix} \begin{pmatrix} 5.1 \\ 3.5 \\ 1.4 \\ 0.2 \end{pmatrix} = \begin{pmatrix} 5.4 \\ 2.2 \\ 2.9 \\ -1.7 \\ -2.4 \end{pmatrix}$$

Activation function

Why do we need
the activation function ReLU ?

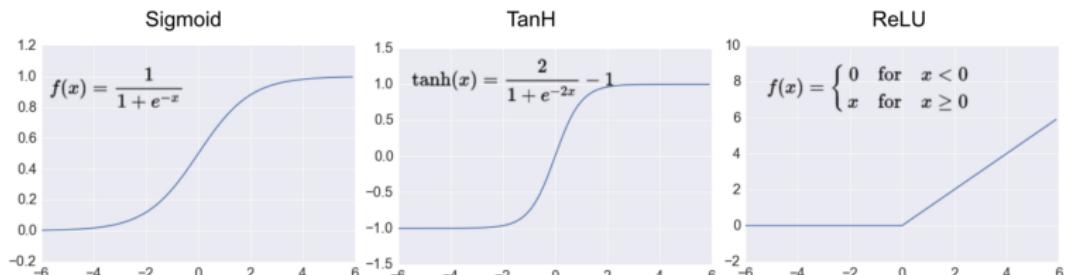
Activation function

A neural network without any non-linear activation function would simply be a linear regression model

Activation function

A neural network without any non-linear activation function would simply be a linear regression model

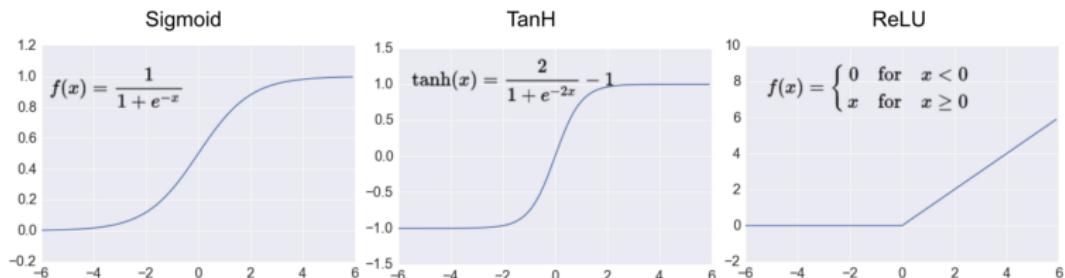
There are several possible non-linear activation functions:



Activation function

A neural network without any non-linear activation function would simply be a linear regression model

There are several possible non-linear activation functions:



ReLU is very simple and very popular in deep neural networks

STEP 2

$$Y = \text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)} X))$$

STEP 2

$$Y = \text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)} X))$$

Activation function (ReLU):

$$\text{ReLU}(W^{(1)} X) = \text{ReLU}\left(\begin{pmatrix} 5.4 \\ 2.2 \\ 2.9 \\ -1.7 \\ -2.4 \end{pmatrix}\right) = \begin{pmatrix} 5.4 \\ 2.2 \\ 2.9 \\ 0 \\ 0 \end{pmatrix}$$

STEP 3

$$Y = \text{SoftMax}(\textcolor{red}{W^{(2)} \text{ReLU}(W^{(1)}X)})$$

STEP 3

$$Y = \text{SoftMax}(\textcolor{red}{W^{(2)} \text{ReLU}(W^{(1)}X)})$$

Matrix multiplication ($\sigma = 0.5$):

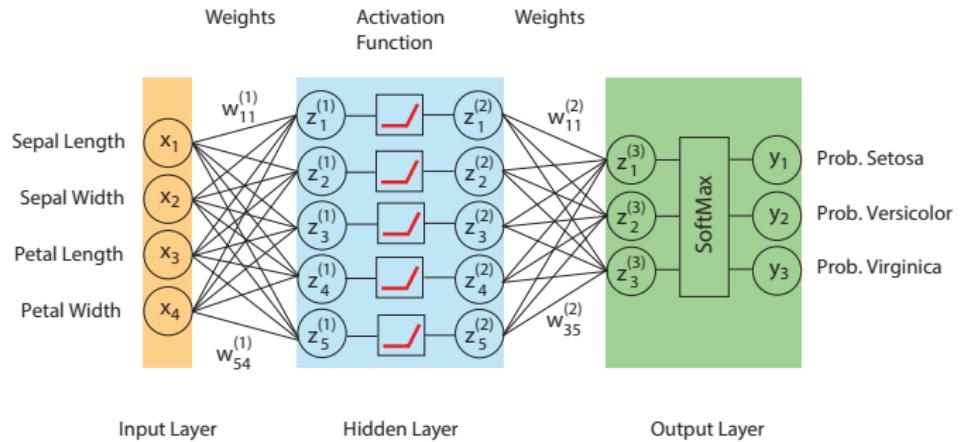
$$\textcolor{red}{W^{(2)} \text{ReLU}(W^{(1)}X)} =$$

$$\begin{pmatrix} 0.6 & 0.1 & 0.9 & -0.2 & -0.5 \\ 0.3 & -0.3 & 0.3 & -0.9 & -0.9 \\ 0.3 & 0.2 & 0.4 & -1.0 & 0.6 \end{pmatrix} \begin{pmatrix} 5.4 \\ 2.2 \\ 2.9 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1.9 \\ 0.3 \\ 2.9 \end{pmatrix}$$

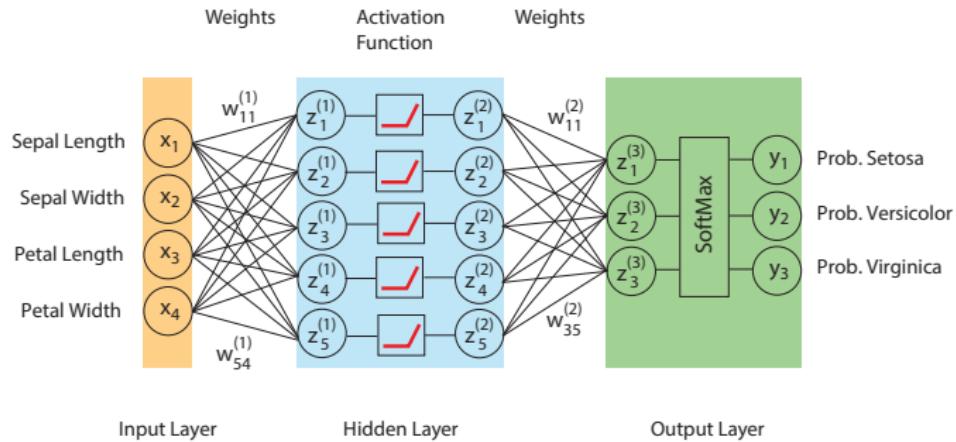
SoftMax function

Why do we use the
SoftMax function ?

SoftMax function



SoftMax function



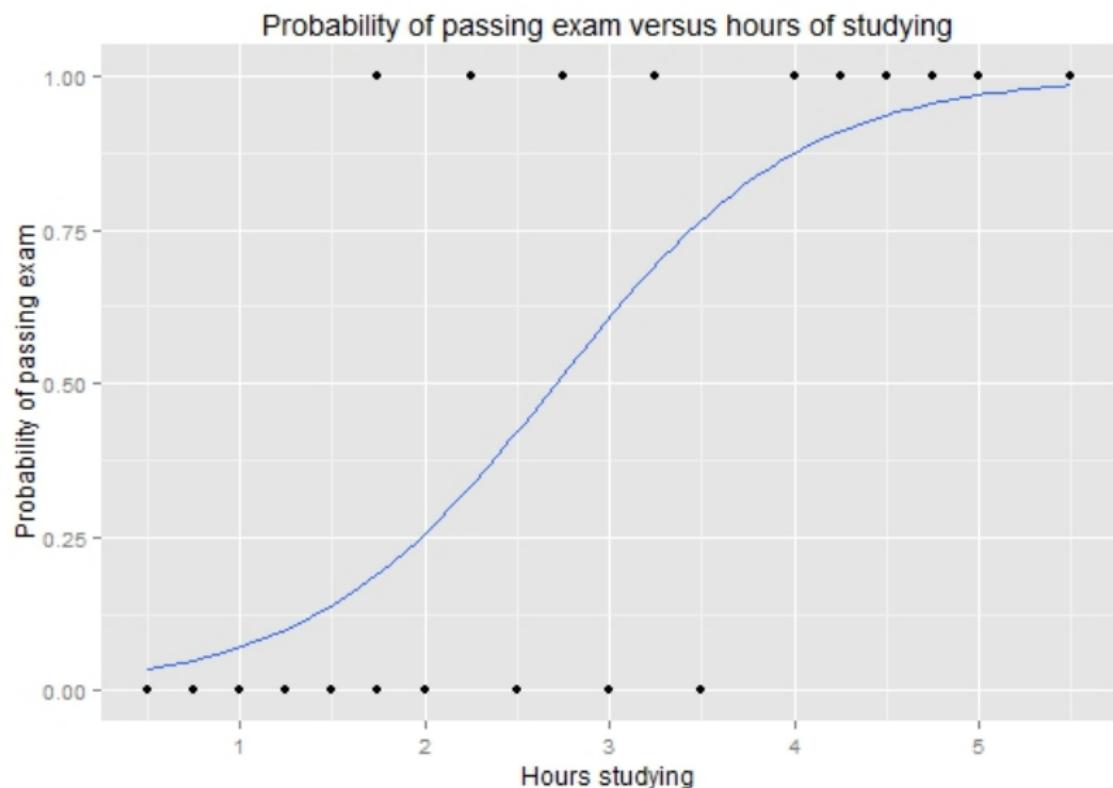
The output $Z^{(3)}$ are arbitrary real values and we want to convert them into probabilities:

$$\text{SoftMax}(Z^{(3)})_i = \frac{e^{z_i^{(3)}}}{e^{z_1^{(3)}} + e^{z_2^{(3)}} + e^{z_3^{(3)}}} = \text{Prob. Class } i \in (0, 1)$$

SoftMax function

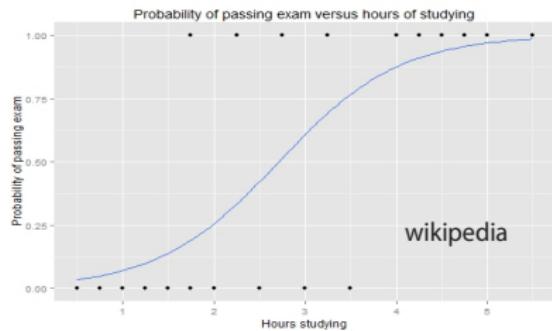
Binary classification

SoftMax function



SoftMax function

Binary classification:



Empirical observation - logistic regression:

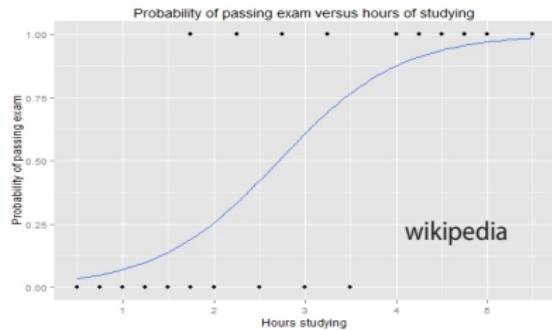
$$p(x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$

Motivate - sigmoid function (1 output neuron):

$$S(x) = \frac{e^x}{1 + e^x}$$

SoftMax function

Binary classification:



Empirical observation - logistic regression:

$$p(x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$

Motivate - sigmoid function (1 output neuron):

$$S(x) = \frac{e^x}{1 + e^x}$$

Multinomial logistic regression:

$$P(X)_\ell = \frac{e^{\beta_\ell X}}{\sum_{k=1}^K e^{\beta_k X}}$$

SoftMax function (K classes):

$$\text{SoftMax}(X)_\ell = \frac{e^{X_\ell}}{\sum_{k=1}^K e^{X_k}}$$

STEP 4

$$Y = \text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)} X))$$

STEP 4

$$Y = \text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)} X))$$

SoftMax function:

$$\text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)} X)) = \text{SoftMax}(\begin{pmatrix} 1.9 \\ 0.3 \\ 2.9 \end{pmatrix}) = \begin{pmatrix} 0.25 \\ 0.05 \\ 0.70 \end{pmatrix}$$

STEP 4

$$Y = \text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)} X))$$

SoftMax function:

$$\text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)} X)) = \text{SoftMax}(\begin{pmatrix} 1.9 \\ 0.3 \\ 2.9 \end{pmatrix}) = \begin{pmatrix} 0.25 \\ 0.05 \\ 0.70 \end{pmatrix}$$

STEP 4

$$Y = \text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)} X))$$

SoftMax function:

$$\text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)} X)) = \text{SoftMax}(\begin{pmatrix} 1.9 \\ 0.3 \\ 2.9 \end{pmatrix}) = \begin{pmatrix} 0.25 \\ 0.05 \\ 0.70 \end{pmatrix}$$

Prediction: Virginica

STEP 4

$$Y = \text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)} X))$$

SoftMax function:

$$\text{SoftMax}(W^{(2)} \text{ReLU}(W^{(1)} X)) = \text{SoftMax}(\begin{pmatrix} 1.9 \\ 0.3 \\ 2.9 \end{pmatrix}) = \begin{pmatrix} 0.25 \\ 0.05 \\ 0.70 \end{pmatrix}$$

Prediction: **Virginica** This is wrong! It is Setosa

Network training using gradient descent

What to do when
the prediction is wrong ?

Network training using gradient descent

Modify the weight values
to obtain better predictions

Network training using gradient descent

We need a way to measure the difference
between the predictions and the true species
(the cross-entropy error)

Network training using gradient descent

And our goal is to find the weights
that minimizes that error function
(using gradient descent)

Network training using gradient descent

Notation

The matrices $W^{(1)}$ and $W^{(2)}$
are combined as W

Network training using gradient descent

row	features				true species			predictions		
n	S.L.	S.W.	P.L.	P.W.	t_{n1}	t_{n2}	t_{n3}	y_{n1}	y_{n2}	y_{n3}
4	4.6	3.1	1.5	0.2	1	0	0	1	0	0
5	5.0	3.6	1.4	0.2	1	0	0	0.94	0.05	0.01

The cross-entropy error/loss function for the n th row:

$$E_n(W) = - \sum_{k=1}^3 \underbrace{t_{nk}}_{\text{true species}} \cdot \log[\underbrace{y_{nk}(W)}_{\text{prediction}}]$$

Network training using gradient descent

row	features				true species			predictions		
n	S.L.	S.W.	P.L.	P.W.	t_{n1}	t_{n2}	t_{n3}	y_{n1}	y_{n2}	y_{n3}
4	4.6	3.1	1.5	0.2	1	0	0	1	0	0
5	5.0	3.6	1.4	0.2	1	0	0	0.94	0.05	0.01

The cross-entropy error/loss function for the n th row:

$$E_n(W) = - \sum_{k=1}^3 \underbrace{t_{nk}}_{\text{true species}} \cdot \log[\underbrace{y_{nk}(W)}_{\text{prediction}}]$$

Example:

$$E_4(W) = -t_{n1} \cdot \log[y_{n1}(W)] = -\log(1) = 0$$

Network training using gradient descent

row	features				true species			predictions		
n	S.L.	S.W.	P.L.	P.W.	t_{n1}	t_{n2}	t_{n3}	y_{n1}	y_{n2}	y_{n3}
4	4.6	3.1	1.5	0.2	1	0	0	1	0	0
5	5.0	3.6	1.4	0.2	1	0	0	0.94	0.05	0.01

The cross-entropy error/loss function for the n th row:

$$E_n(W) = - \sum_{k=1}^3 \underbrace{t_{nk}}_{\text{true species}} \cdot \log[\underbrace{y_{nk}(W)}_{\text{prediction}}]$$

Example:

$$E_5(W) = -t_{n1} \cdot \log[y_{n1}(W)] = -\log(0.94) = 0.06$$

Network training using gradient descent

Find the weights W that minimize
the total cross-entropy error:

$$E(W) = \sum_{n=1}^N E_n(W)$$

How to derive
the cross-entropy formula ?

Network training using gradient descent

The maximum likelihood method

Data:

$$X_1, \dots, X_N \sim \mathcal{N}(\mu, \sigma^2)$$

Point estimate of μ :

$$\hat{\mu} = \operatorname{argmax}_{\mu} \mathcal{L}(\mu | X) = \operatorname{argmax}_{\mu} \prod_{i=1}^N \mathcal{N}_i(\mu, \sigma^2) = \frac{1}{N} \sum_{i=1}^N X_i$$

Equivalently:

$$\hat{\mu} = \operatorname{argmin}_{\mu} [-\log \mathcal{L}(\mu | X)]$$

Network training using gradient descent

The total cross-entropy error is defined
as the negative log-likelihood

$$E(W) = -\log \mathcal{L}(W|T) ; \quad \hat{W} = \operatorname{argmin}_W E(W)$$

where

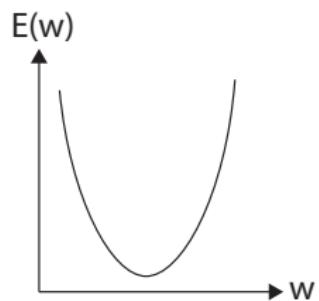
$$\mathcal{L}(W|T) = \prod_{n=1}^N \prod_{k=1}^K y_{nk}^{t_{nk}}(W)$$

In RED: the probability for the correct class

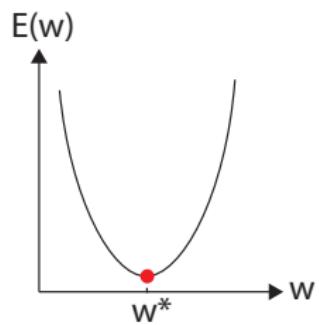
Network training using gradient descent

How does
gradient descent work ?

Network training using gradient descent

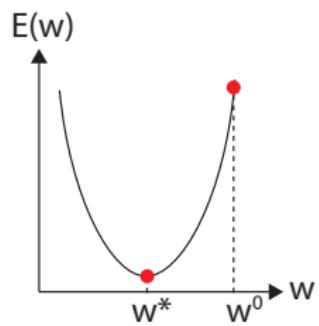


Network training using gradient descent



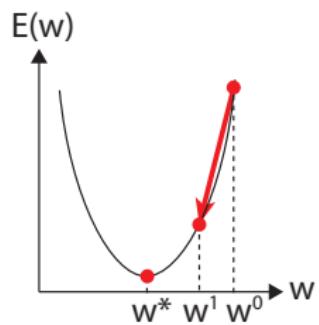
$$E(w^*) = \min(E)$$

Network training using gradient descent



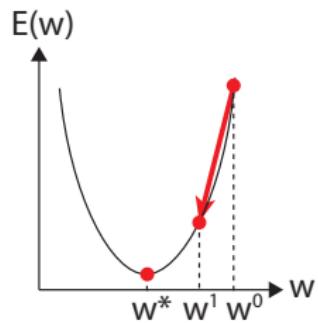
$$E(w^*) = \min(E)$$

Network training using gradient descent



$$E(w^*) = \min(E)$$

Network training using gradient descent

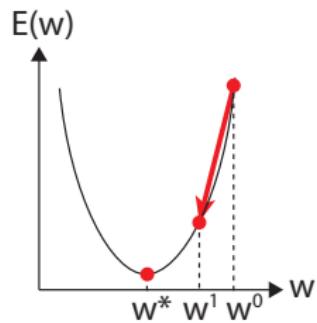


Several iterations:

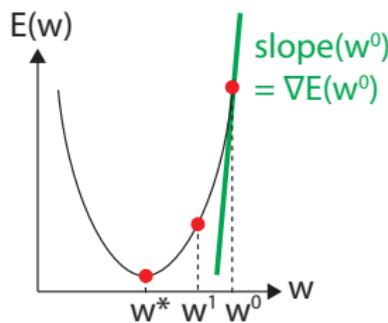
$w^0 \rightarrow w^1 \rightarrow w^2 \rightarrow w^3 \cdots \rightarrow w^*$

$$E(w^*) = \min(E)$$

Network training using gradient descent

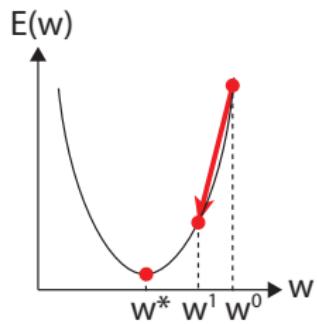


$$E(w^*) = \min(E)$$

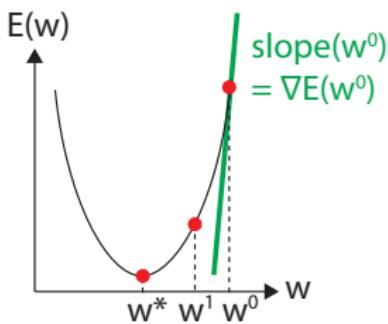


$$E(w^*) = \min(E)$$

Network training using gradient descent



$$E(w^*) = \min(E)$$



$$E(w^*) = \min(E)$$

Gradient Descent:

$$w^1 = w^0 - \eta * \nabla E(w^0)$$

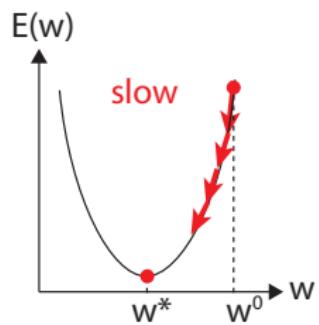
$$\nabla E(w^*) = 0; \quad \eta > 0$$

η = learning parameter

Network training using gradient descent

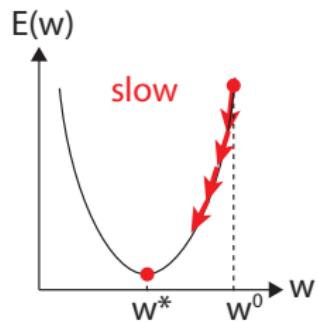
Possible problems

Network training using gradient descent

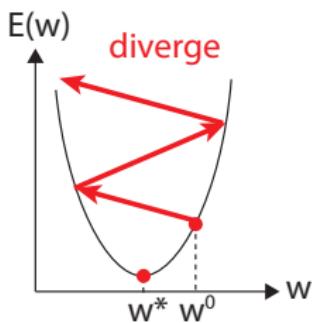


η too small

Network training using gradient descent

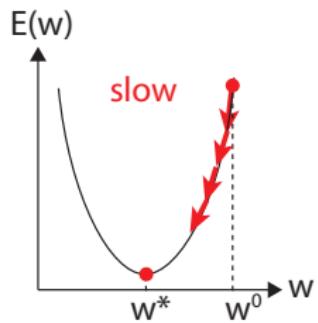


η too small

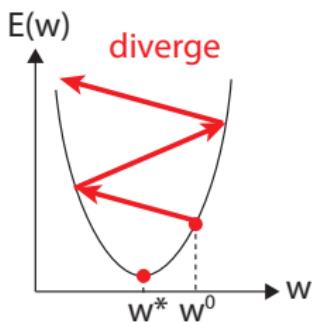


η too large

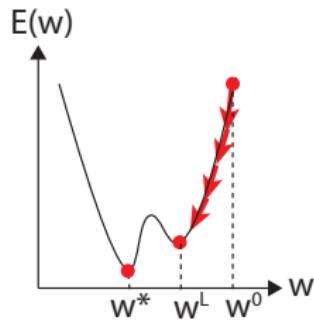
Network training using gradient descent



η too small



η too large



local minimum at w^L

Network training using gradient descent

Initialization $\tau = 0$: Choose the initial weights W^0 with $\mathcal{N}(0, \sigma^2)$

Network training using gradient descent

Initialization $\tau = 0$: Choose the initial weights W^0 with $\mathcal{N}(0, \sigma^2)$

$$W^{(1)} = \begin{pmatrix} 0.5 & 0.1 & -0.2 & -0.4 \\ -0.4 & 1.0 & 0.5 & 1.0 \\ -0.2 & -0.2 & -0.5 & -0.1 \\ 0.2 & 0.7 & 0.3 & 0.2 \\ 0.6 & 0.6 & 0.1 & -0.4 \end{pmatrix}$$

$$W^{(2)} = \begin{pmatrix} 0.6 & 0.1 & 0.9 & -0.2 & -0.5 \\ 0.3 & -0.3 & 0.3 & -0.9 & -0.9 \\ 0.3 & 0.2 & 0.4 & -1.0 & 0.6 \end{pmatrix}$$

Network training using gradient descent

3 possibilities for the next steps

3 possibilities for the next steps

- ① Batch Gradient Descent
- ② Mini-batch Gradient Descent
- ③ Stochastic Gradient Descent

Batch Gradient Descent

- ① Apply the NN to **all** the train set
- ② Record **all** the errors
- ③ Update the weights:

$$W^\tau = W^{\tau-1} - \eta \cdot \nabla E(W^{\tau-1})$$

Mini-batch Gradient Descent

- ① Apply the NN to **a batch** of the train set
- ② Record the corresponding errors
- ③ Update the weights:

$$W^\tau = W^{\tau-1} - \eta \cdot \nabla \sum_{n \in \text{batch}} E_n(W^{\tau-1})$$

Stochastic Gradient Descent

- ① Apply the NN to **one sample** of the train set
- ② Record the one sample error
- ③ Update the weights:

$$W^\tau = W^{\tau-1} - \eta \cdot \nabla E_n(W^{\tau-1})$$

Network training using gradient descent

Iteration

1 iteration (or pass) is one weight update

Epoch

1 epoch is reached

when the NN has passed through
all the training data

Network training using gradient descent

EXAMPLE

If you have 100 training samples,
and your batch size is 50,
then it will take 2 iterations to complete 1 epoch

Gradient Descent

- ① Batch Gradient Descent:
1 epoch = 1 iteration

- ② Mini-batch Gradient Descent:
1 epoch = (N/batch) iterations

- ③ Stochastic Gradient Descent:
1 epoch = N iterations

Performance metrics

What are
the performance metrics ?

Performance metrics

They may be used on
the training, validation and test sets

Cross-entropy error/loss function

$$E = - \sum_{n=1}^N \sum_{k=1}^3 t_{nk} \cdot \log y_{nk}$$

Performance metrics

Confusion matrix

		predictions		
		setosa	versicolor	virginica
actuals	setosa	14	0	0
	versicolor	0	9	0
virginica	virginica	0	2	10

Performance metrics

Accuracy rate = 1 - Error rate

$$\text{Accuracy rate} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{33}{35} = 94\%$$

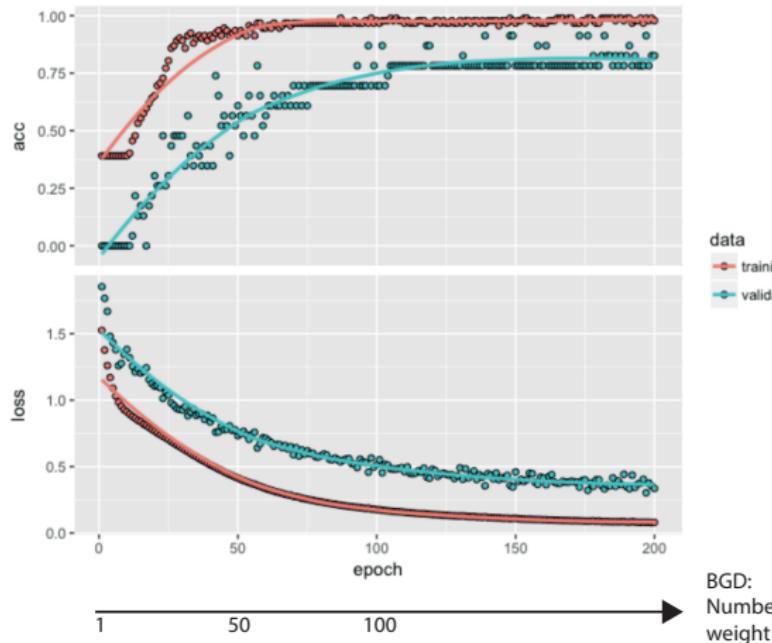
$$\text{Error rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}} = \frac{2}{35} = 6\%$$

Performance metrics

EXAMPLES

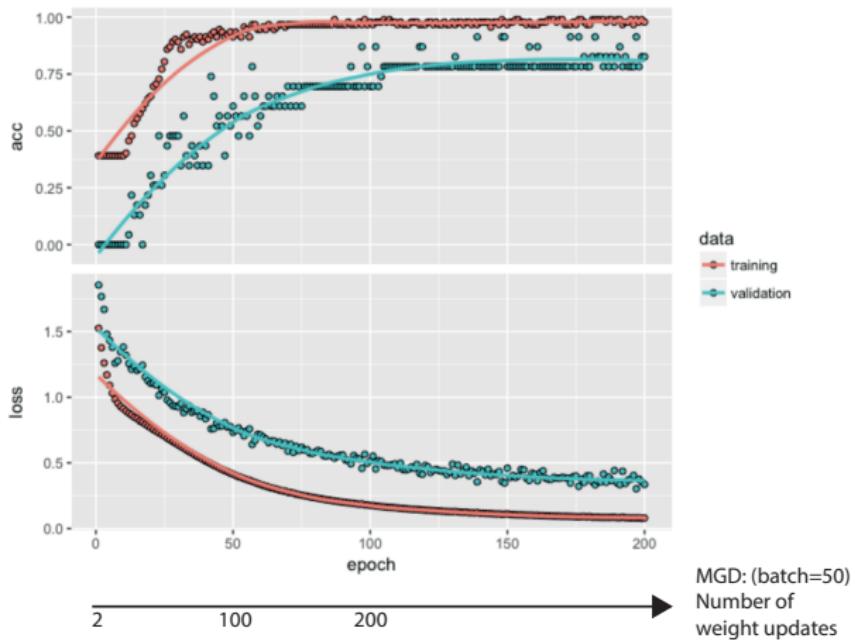
Performance metrics

Training and validation datasets:



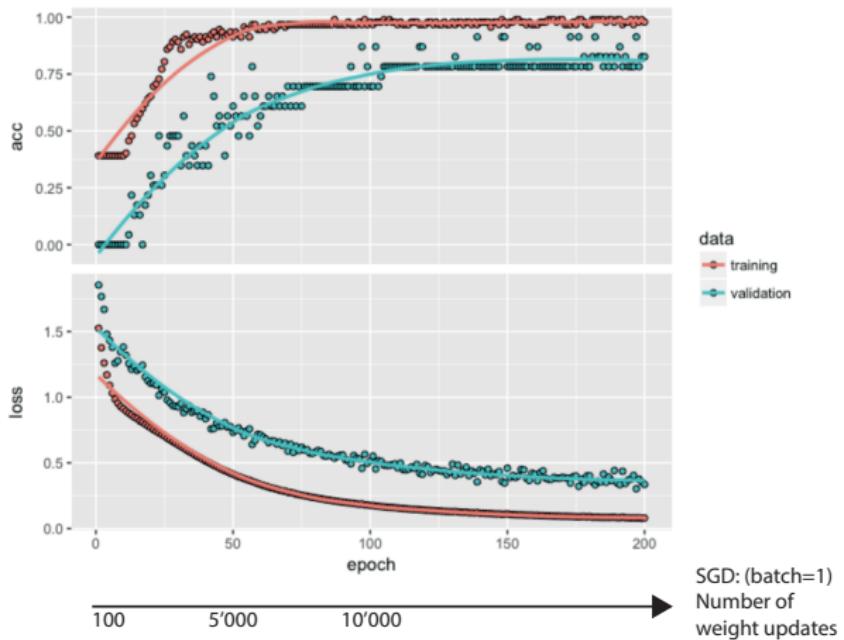
Performance metrics

Training and validation datasets:



Performance metrics

Training and validation datasets:



Performance metrics

Test set

Test set

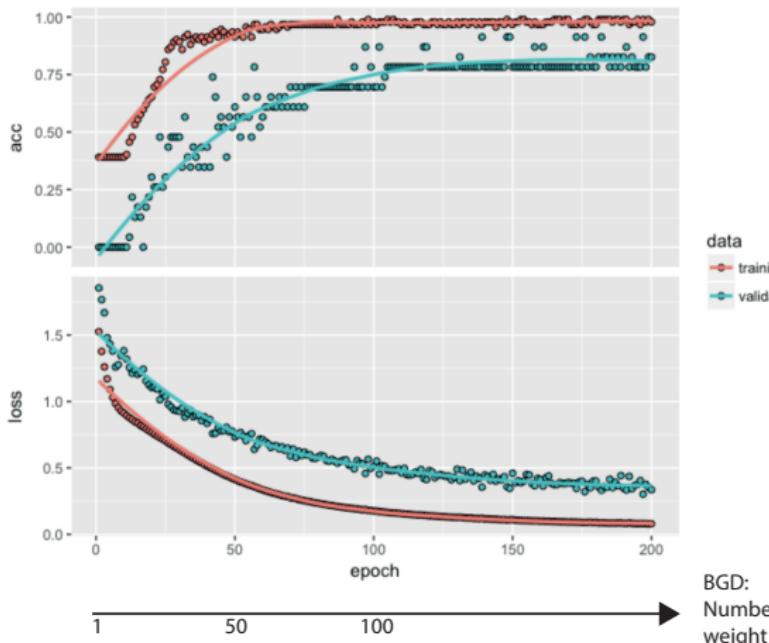
Accuracy=0.91 and cross-entropy loss=0.22

Test set

Accuracy=0.91 and cross-entropy loss=0.22

THE END

Optimum number of epochs



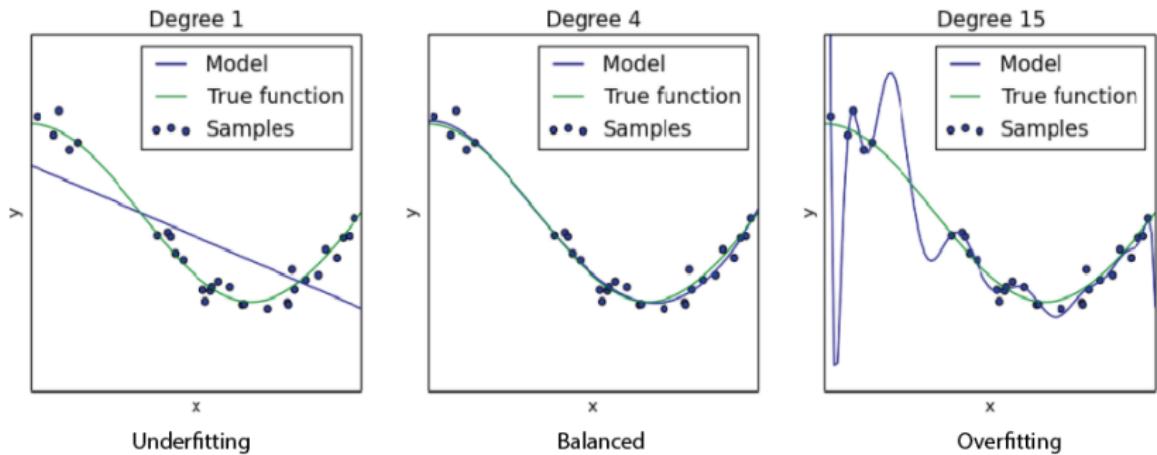
Optimum number of epochs

What is the optimum
number of epochs ?

Optimum number of epochs

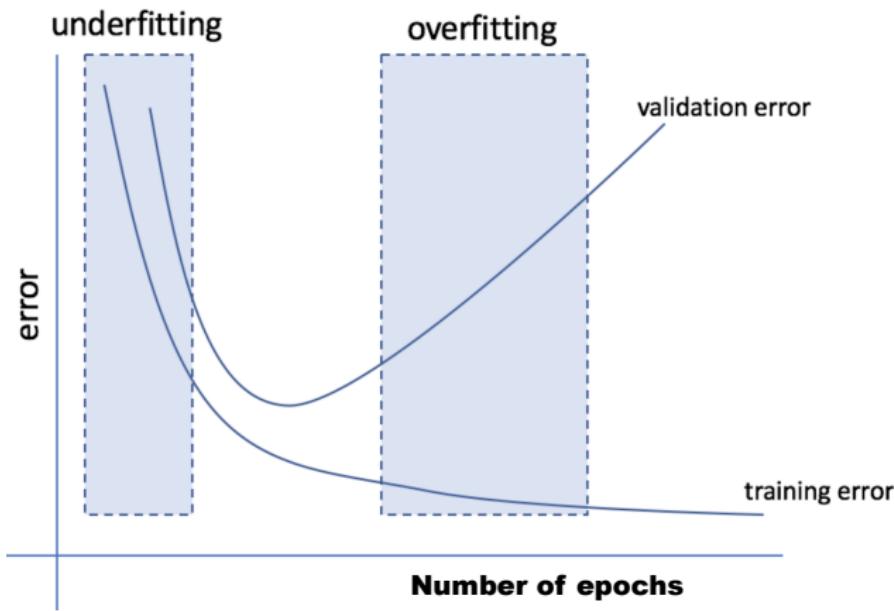
The answer is related to the problem of
under-fitting and over-fitting

Optimum number of epochs

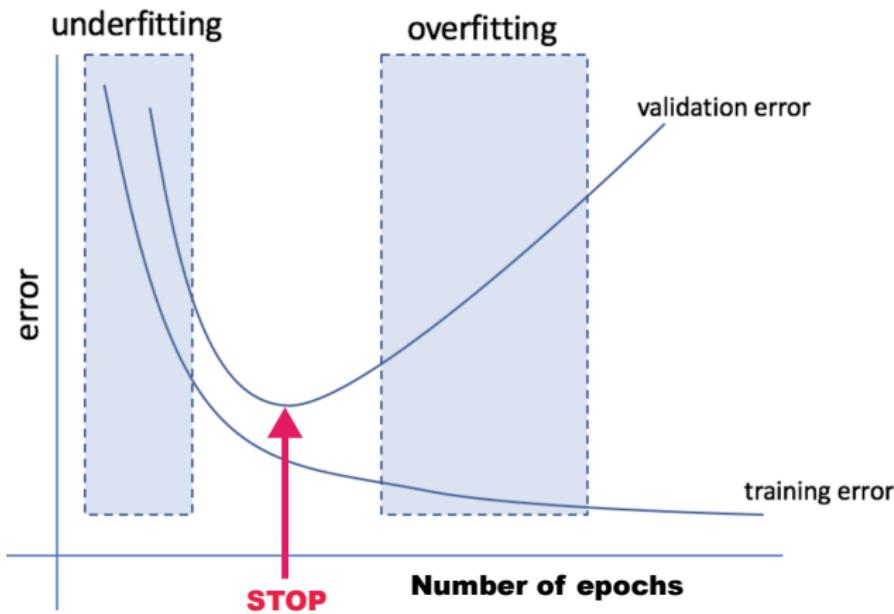


Ref: scikit-learn 0.18 documentation

Optimum number of epochs



Optimum number of epochs



Conclusion on neural network

Can one reach 100% accuracy ?

Short answer

It is possible only if

there is enough information in the input X
to predict Y uniquely

Conclusion on neural network

EXAMPLE

If two plants have the same four attributes

$$(X_1 = X_2)$$

but belong to two different species

$$(Y_1 \neq Y_2),$$

then we need additional features

to characterize uniquely the three iris species

Conclusion on neural network

If two people have the same gender and age

$$(X_1 = X_2)$$

but only one has a specific disease

$$(Y_1 \neq Y_2),$$

then we need additional features
(physical activity, smoking, genetics)

to characterize uniquely the risk of this disease

Conclusion on neural network

IN GENERAL

$$Y = f(X) + \text{error}$$

Hyperparameters

Number of layers, number of nodes,
initial weight values, activation function,
error/loss function, number of epochs,
learning rate, batch size, bias node

Conclusion on neural network

How to choose them ?

Trial and Error

Select the combination that performs best
(highest validation accuracy)

Trial and Error

The goal is to predict
(and not really to explain)

Conclusion on neural network

Main advantage:

- Works well on a whole range of problems including image and signal recognitions.

Conclusion on neural network

Main advantage:

- Works well on a whole range of problems including image and signal recognitions.

Main disadvantage:

- Black box: difficult to understand what are the main features that the neural network uses to make prediction.
Decision trees are better suited for interpretation.

QUESTIONS ?

Applications

Applications

Application 1

Predict the 1-year mortality rate
of elderly patients
with intertrochanteric fractures

Ref: Artificial neural network models for predicting 1-year mortality in elderly patients with intertrochanteric fractures in China, L. Shi, X.C. Wang and Y.S. Wang, Brazilian Journal of Medical and Biological Research (2013)

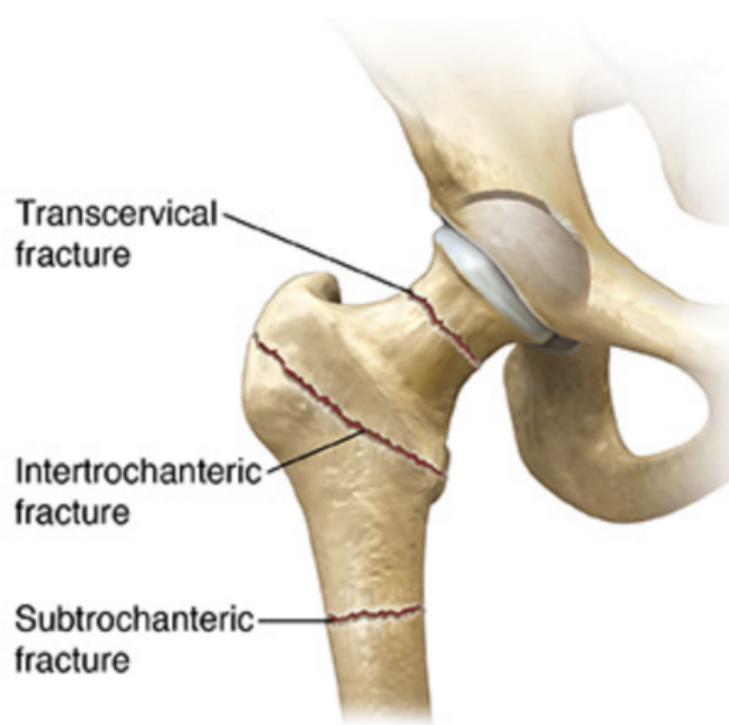
Application 1

Some older people fall
and break one of their hips

Application 1

50% of hip fractures
are intertrochanteric fractures

Application 1



Application 1

There is an increase of death
after intertrochanteric fractures
(because of reduced mobility)

Application 1

1-year mortality rate = D/N

D = number of deaths occurring within 1 year

N = the size of the population
(all patients with intertrochanteric fractures)

Data

2150 patients with intertrochanteric fractures:

70% in the training group

30% patients in the testing group

Application 1

After some trial and error
with different hyperparameters
(number of layers and nodes)

they end up with the following neural network

Application 1

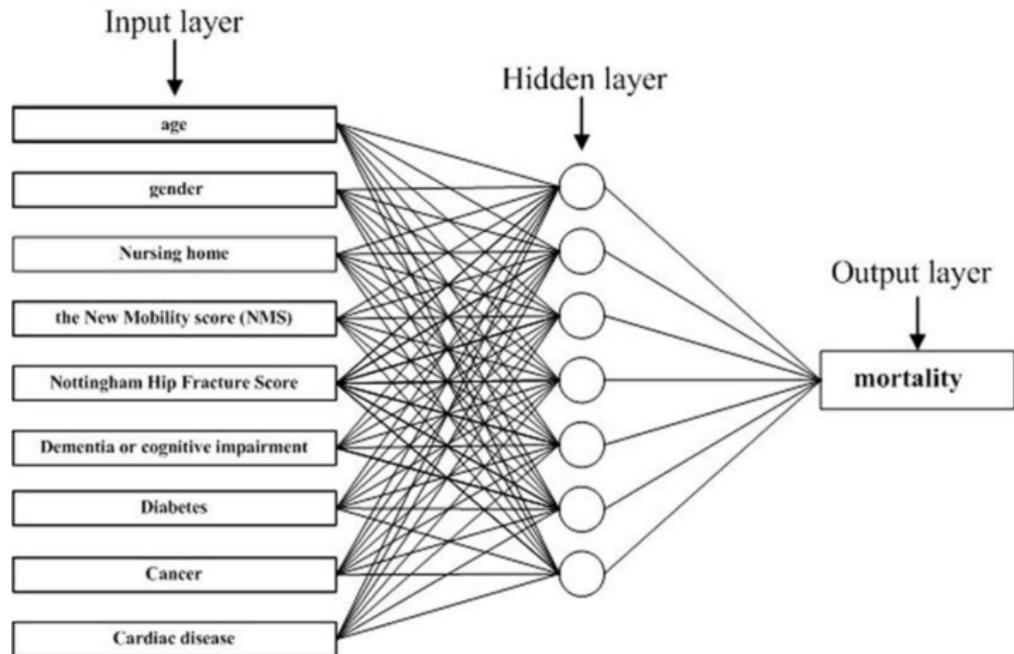


Figure 2 Schematic representation showing the structure of the artificial neural network models, which have 8 input nodes, 6 nodes in hidden layer, and 1 output node, which represents 1-year mortality in elderly patients with intertrochanteric fracture.

Application 1

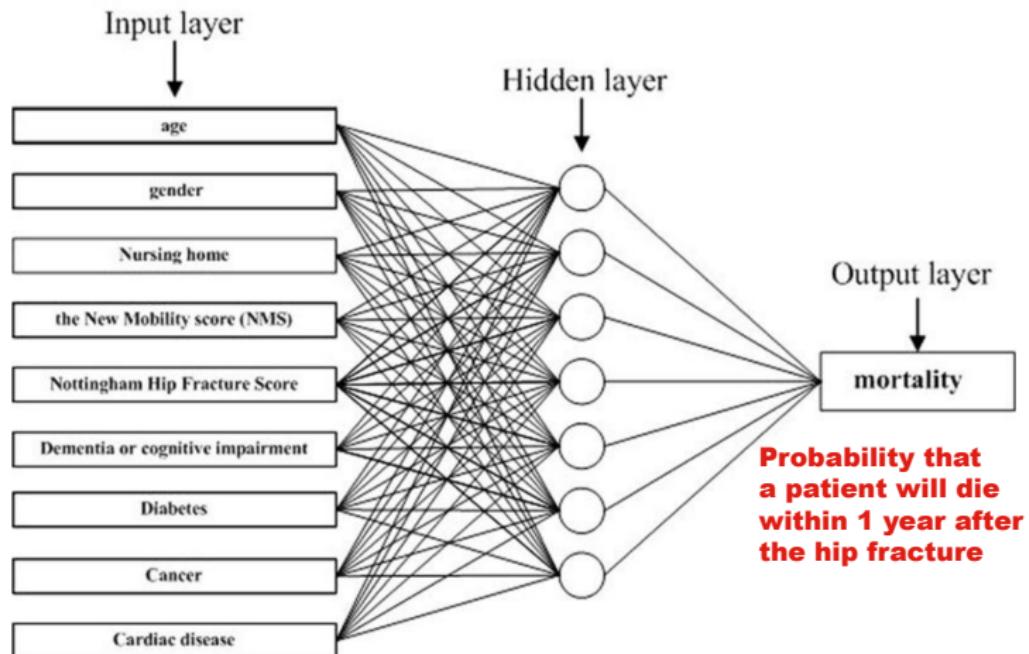


Figure 2 Schematic representation showing the structure of the artificial neural network models, which have 8 input nodes, 6 nodes in hidden layer, and 1 output node, which represents 1-year mortality in elderly patients with intertrochanteric fracture.

Accuracy

92% for the training group

86% for the testing group

Application 2

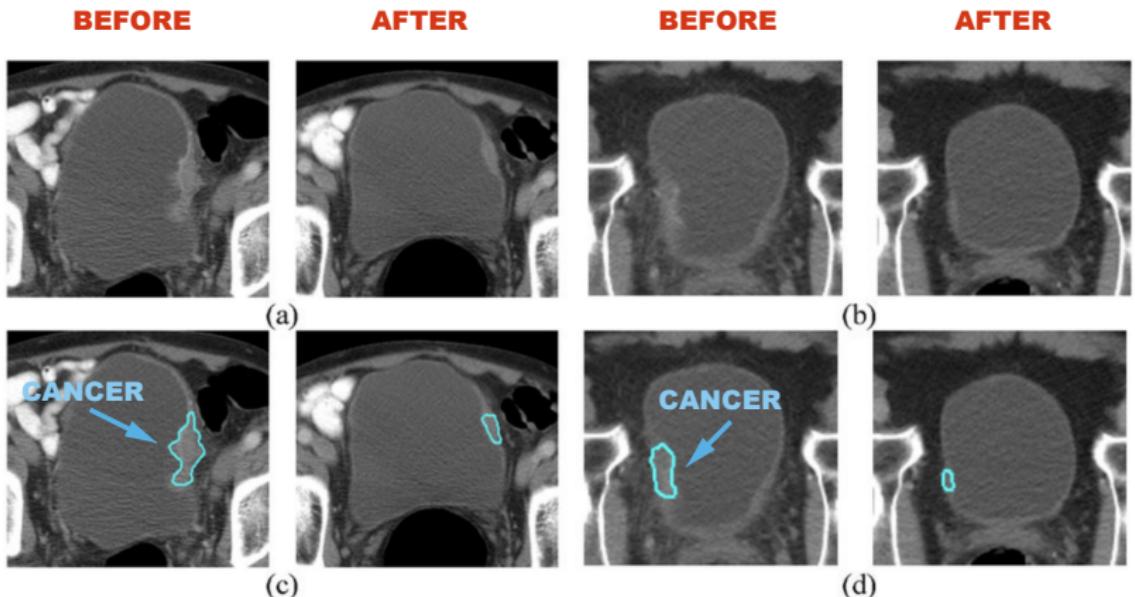
Predict if there is a residual tumor
after bladder cancer treatment

Ref: Bladder Cancer Treatment Response Assessment in CT using Radiomics with Deep-Learning, Kenny H. Cha,
Lubomir Hadjiiski, Heang-Ping Chan, Alon Z. Weizer, Ajjai Alva, Richard H. Cohan, Elaine M. Caoili, Chintana
Paramagul and Ravi K. Samala, Scientific Reports volume 7, Article number: 8738 (2017)

Application 2

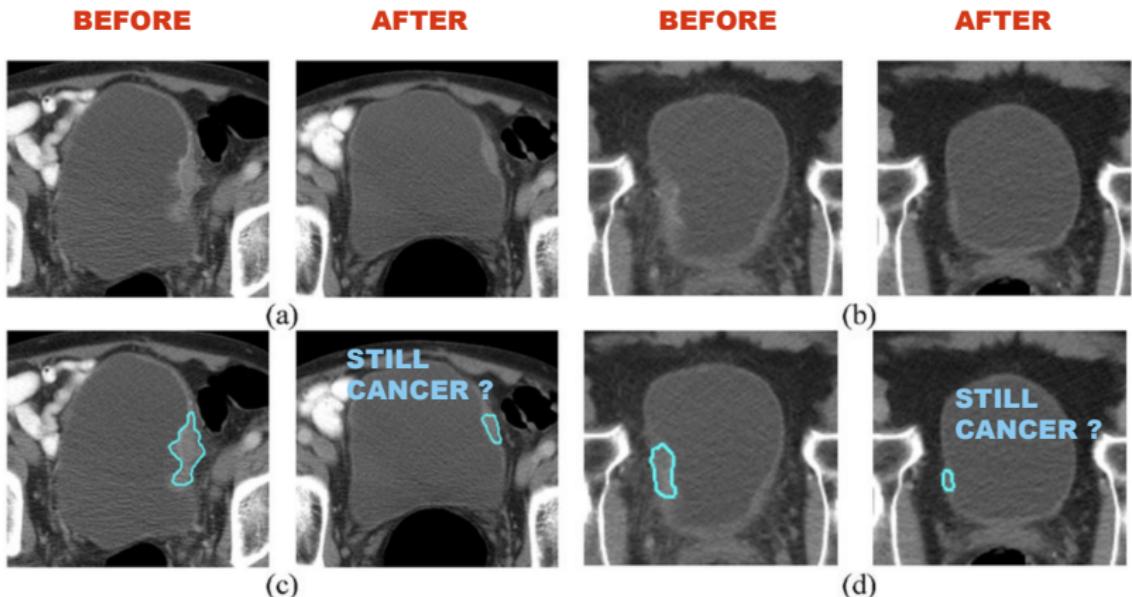
They take X-ray images of the bladder and
use an algorithm to localise the cancer region
before and after treatment

Application 2



Bladder lesion segmentations. Two segmented bladder cancers are illustrated. The lesions in the pre- and post-treatment scan pairs shown in (a,b) are segmented using AI-CALS, as shown in (c,d), respectively. The pre-treatment scan is on the left and the post-treatment scan is located on the right of each pair.

Application 2



Bladder lesion segmentations. Two segmented bladder cancers are illustrated. The lesions in the pre- and post-treatment scan pairs shown in (a,b) are segmented using AI-CALS, as shown in (c,d), respectively. The pre-treatment scan is on the left and the post-treatment scan is located on the right of each pair.

Application 1

Data

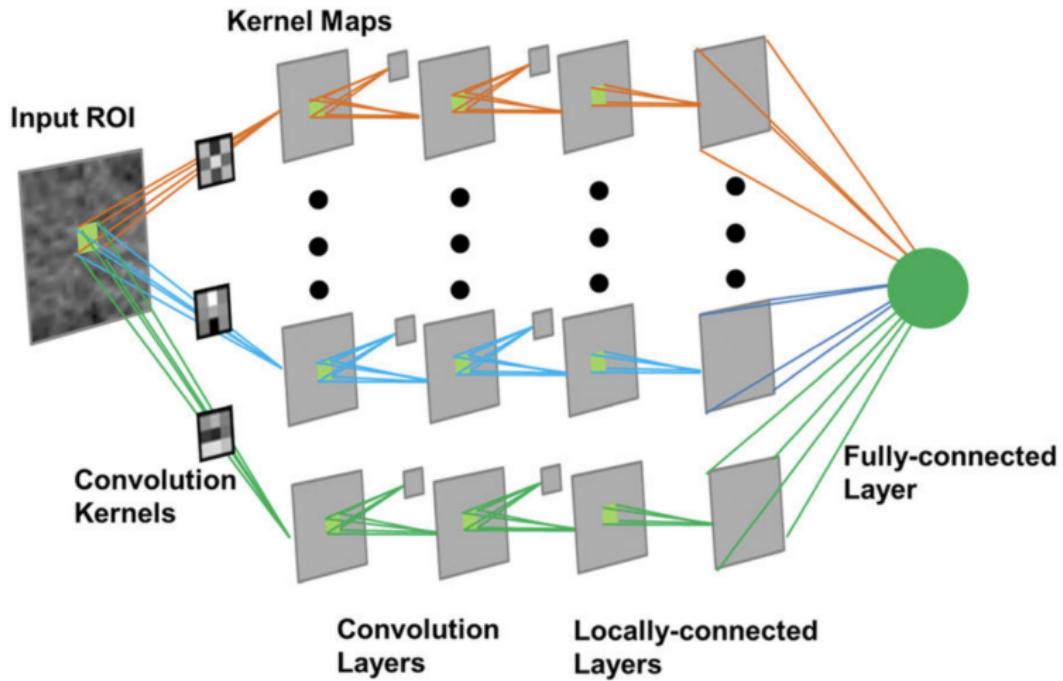
6700 pre-post-treatment paired images
with located cancer region

Application 1

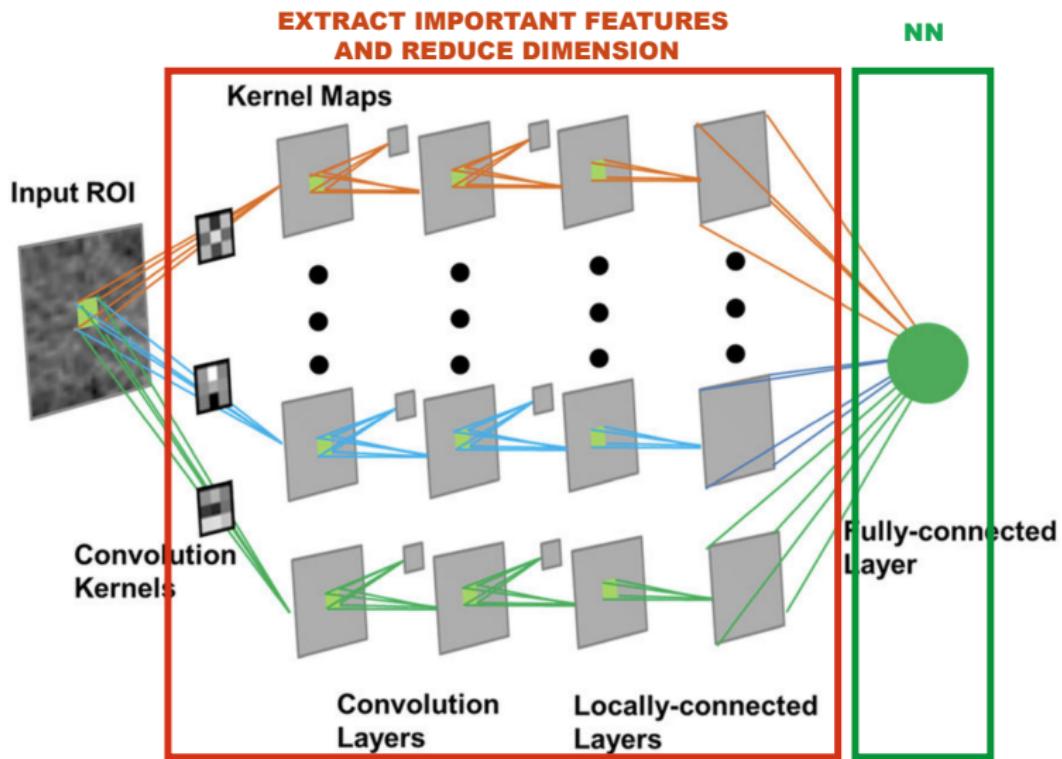
Data

They combined the paired images
into Region Of Interest (ROI) images

Application 2



Application 2



Application 2

Table 2 Number of correctly predicted bladder cancer treatment response assessment of the test set at an operating point determined using the training set.

From: [Bladder Cancer Treatment Response Assessment in CT using Radiomics with Deep-Learning](#)

	DL-CNN	RF-SL	RF-ROI	Radiologist 1	Radiologist 2
Complete Response (Sensitivity)	6/12 (50%)	6/12 (50%)	8/12 (66.7%)	11/12 (91.7%)	11/12 (91.7%)
Non-complete Response (Specificity)	34/42 (81.0%)	33/42 (78.6%)	23/42 (54.8%)	18/42 (42.9%)	16/42 (38.1%)

DL-CNN: Deep-learning convolution neural network. RF-SL: Radiomics features extracted from segmented lesions. RF-ROI: Radiomics features extracted from pre- and post-treatment paired ROIs.

Complete response = No residual cancer

Non-complete response = Residual cancer

Application 2

Table 2 Number of correctly predicted bladder cancer treatment response assessment of the test set at an operating point determined using the training set.

From: [Bladder Cancer Treatment Response Assessment in CT using Radiomics with Deep-Learning](#)

	DL-CNN	RF-SL	RF-ROI	Radiologist 1	Radiologist 2
Complete Response (Sensitivity)	6/12 (50%)	6/12 (50%)	8/12 (66.7%)	11/12 (91.7%)	11/12 (91.7%)
Non-complete Response (Specificity)	34/42 (81.0%)	33/42 (78.6%)	23/42 (54.8%)	18/42 (42.9%)	16/42 (38.1%)

DL-CNN: Deep-learning convolution neural network. RF-SL: Radiomics features extracted from segmented lesions. RF-ROI: Radiomics features extracted from pre- and post-treatment paired ROIs.

Complete response = No residual cancer

Non-complete response = Residual cancer

Application 2

Table 2 Number of correctly predicted bladder cancer treatment response assessment of the test set at an operating point determined using the training set.

From: [Bladder Cancer Treatment Response Assessment in CT using Radiomics with Deep-Learning](#)

	DL-CNN	RF-SL	RF-ROI	Radiologist 1	Radiologist 2
Complete Response (Sensitivity)	6/12 (50%)	6/12 (50%)	8/12 (66.7%)	11/12 (91.7%)	11/12 (91.7%)
Non-complete Response (Specificity)	34/42 (81.0%)	33/42 (78.6%)	23/42 (54.8%)	18/42 (42.9%)	16/42 (38.1%)

DL-CNN: Deep-learning convolution neural network. RF-SL: Radiomics features extracted from segmented lesions. RF-ROI: Radiomics features extracted from pre- and post-treatment paired ROIs.

Complete response = No residual cancer

Non-complete response = Residual cancer

Application 3

Diagnose irregular heart rhythms (arrhythmias)
from single-lead electrocardiography signals

Ref: Cardiologist-Level Arrhythmia Detection With Convolutional Neural Networks, Pranav Rajpurkar, Awni Hannun,
Masoumeh Haghpanahi, Codie Bourn, and Andrew Ng, arXiv:1707.01836

Application 3

Data

60'000 electrocardiography records
(annotated by experts with 14 classes)
from 30'000 patients

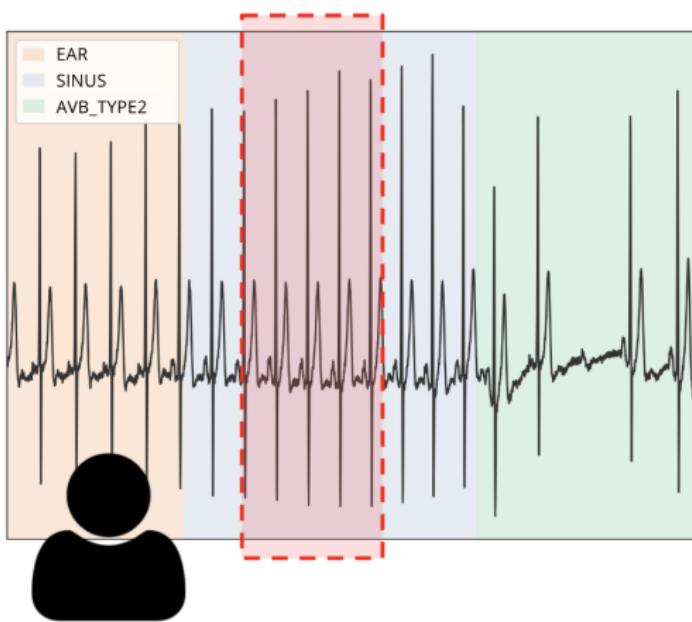
Application 3

Class	Description	Example	Train + Val Patients	Test Patients
AFIB	Atrial Fibrillation		4638	44
AFL	Atrial Flutter		3805	20
AVB.TYPE2	Second degree AV Block Type 2 (Mobitz II)		1905	28
BIGEMINY	Ventricular Bigeminy		2855	22
CHB	Complete Heart Block		843	26
EAR	Ectopic Atrial Rhythm		2623	22
IVR	Idioventricular Rhythm		1962	34

Class	Description	Example	Train + Val Patients	Test Patients
JUNCTIONAL	Junctional Rhythm		2030	36
NOISE	Noise		9940	41
SINUS	Sinus Rhythm		22156	215
SVT	Supraventricular Tachycardia		6301	34
TRIGEMINY	Ventricular Trigeminy		2864	21
VT	Ventricular Tachycardia		4827	17
WENCKEBACH	Wenckebach (Mobitz I)		2051	29

Application 3

GOAL



Application 3

The model outputs a new prediction once every second

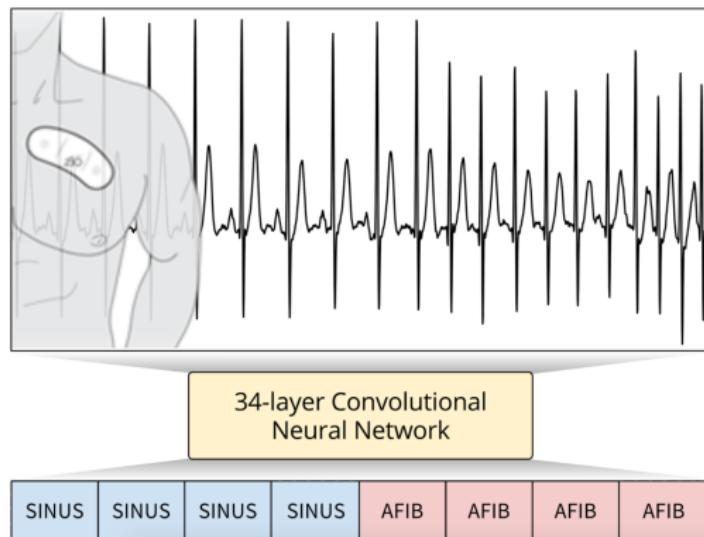
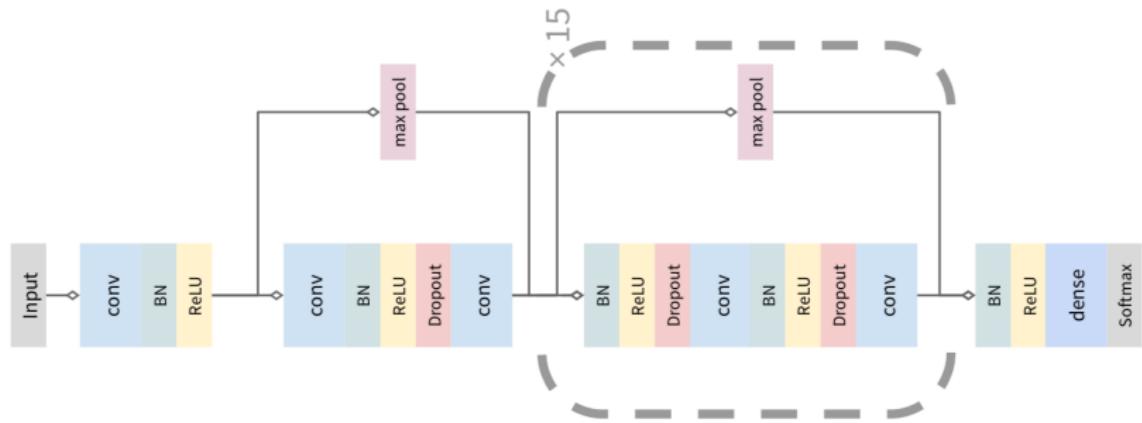


Figure 1. Our trained convolutional neural network correctly detecting the sinus rhythm (SINUS) and Atrial Fibrillation (AFIB) from this ECG recorded with a single-lead wearable heart monitor.

Application 3



33 layers of convolution followed by a fully connected layer

Application 3

The model outperforms the cardiologist

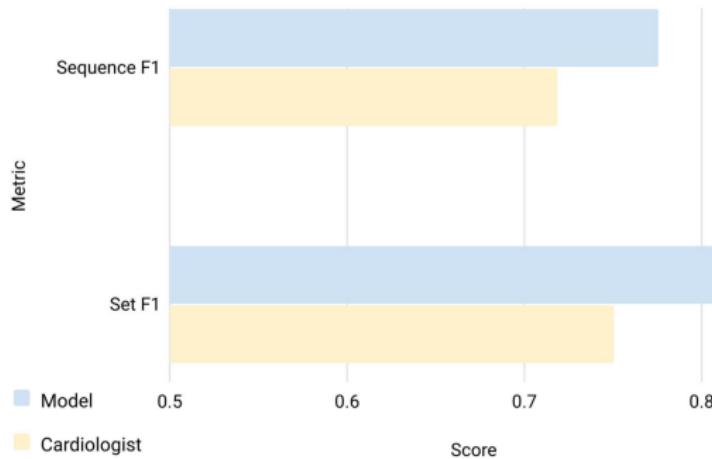


Figure 3. Evaluated on the test set, the model outperforms the average cardiologist score on both the Sequence and the Set F1 metrics.

Application 3

The model outperforms the cardiologist

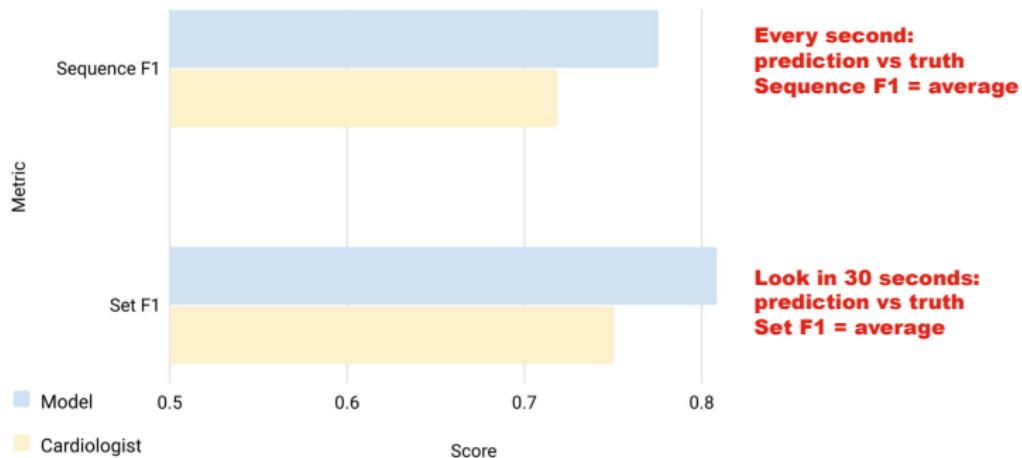


Figure 3. Evaluated on the test set, the model outperforms the average cardiologist score on both the Sequence and the Set F1 metrics.

Application 4

14 Buzz

Algorithmes plus doués que les dermatologues

LOGICIEL Une machine a été capable de détecter 95% des mélanomes sur une série de photos, contre 89% pour l'humain.

Les dermatologues ont du souci à se faire. Un ordinateur a réussi à être meilleur qu'eux pour repérer les cancers de la peau sur des clichés, rapporte la revue «Annals of Oncology». Une équipe germano-franco-américaine a entraîné un système d'intelligence artificielle à distinguer des lésions de la peau et grains de beauté selon qu'ils étaient bénins ou alarmants, en lui montrant plus de 100 000 images. Les performances de la machine (un réseau neuronal convolu-



Chaque année, 55 000 personnes décèdent d'un mélanome malin. -ISTOCK

tif) ont ensuite été comparées à celles de 58 médecins spécialistes de 17 pays. Résultat: «La plupart des dermatologues ont

fait moins bien», écrivent les chercheurs.

Confrontés à 100 photos de cas jugés compliqués, les mé-

decins ont correctement identifié 87% des mélanomes qui leur étaient présentés. Quand ils obtenaient des images en plus gros plan et des infos plus détaillées (âge, sexe du patient, position de la lésion cutanée, par exemple), ce taux montait à 89%. Mais la machine a fait mieux, avec 95% de mélanomes détectés dès la première série de photos.

Pour les chercheurs, la question n'est pas de se passer des médecins au profit de l'intelligence artificielle, mais de faire d'elle «un outil supplémentaire». «Aujourd'hui rien ne remplace un examen clinique approfondi», ont rappelé dans l'étude deux professeurs australiens en dermatologie. -ATS

QUESTIONS ?

BONUS

Bonus 1: Bias node

A simple linear regression model:

$$y_i = \alpha + \beta \cdot x_i + \varepsilon_i$$

where α is called the intercept parameter

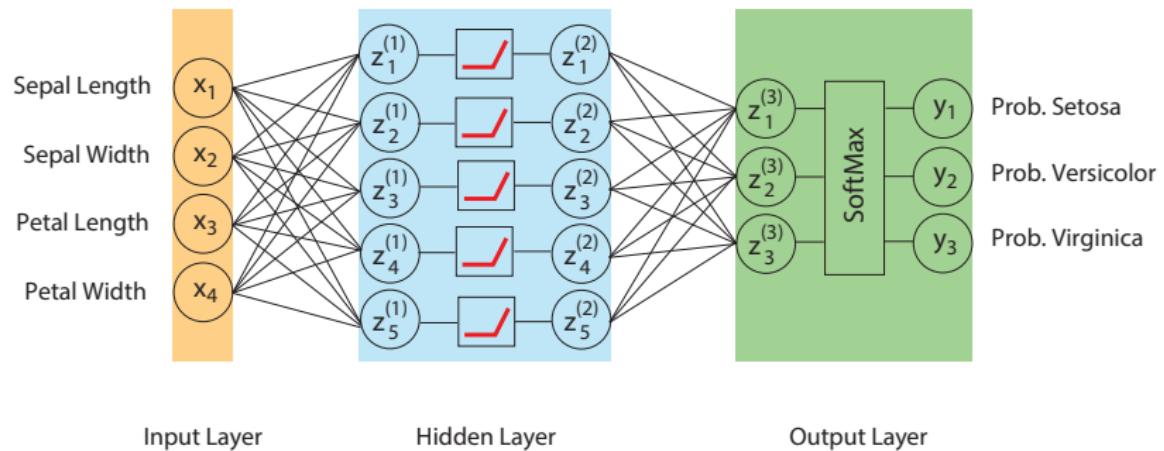
Bonus 1: Bias node

In neural network,

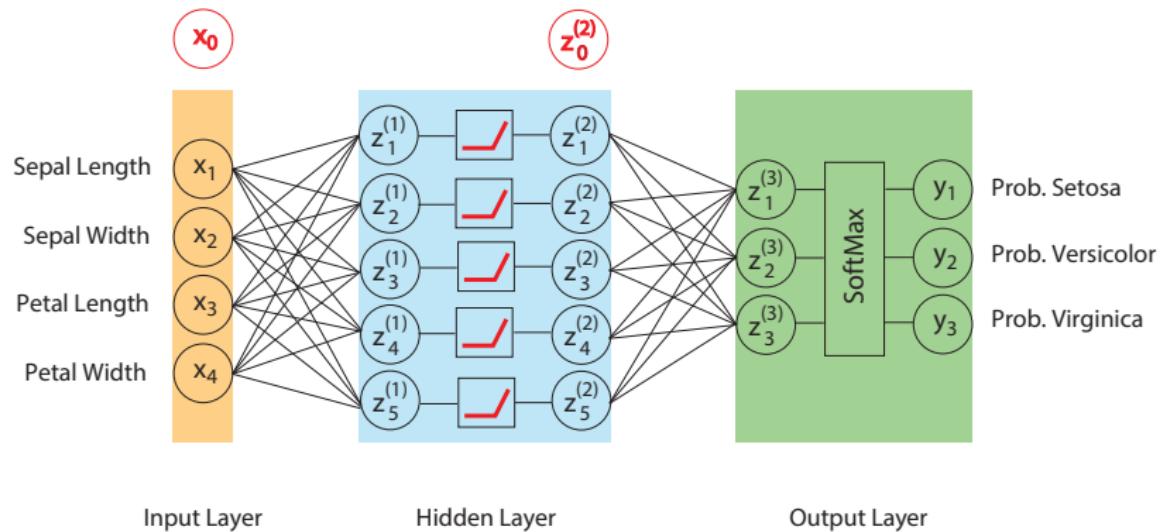
the intercept parameter α

is introduced via the bias node

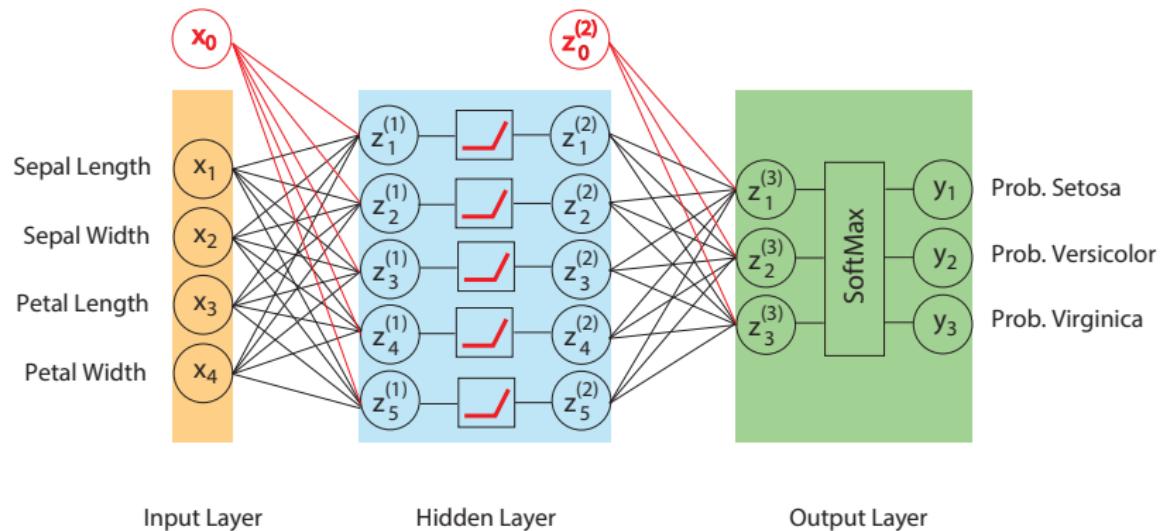
Bonus 1: Bias node



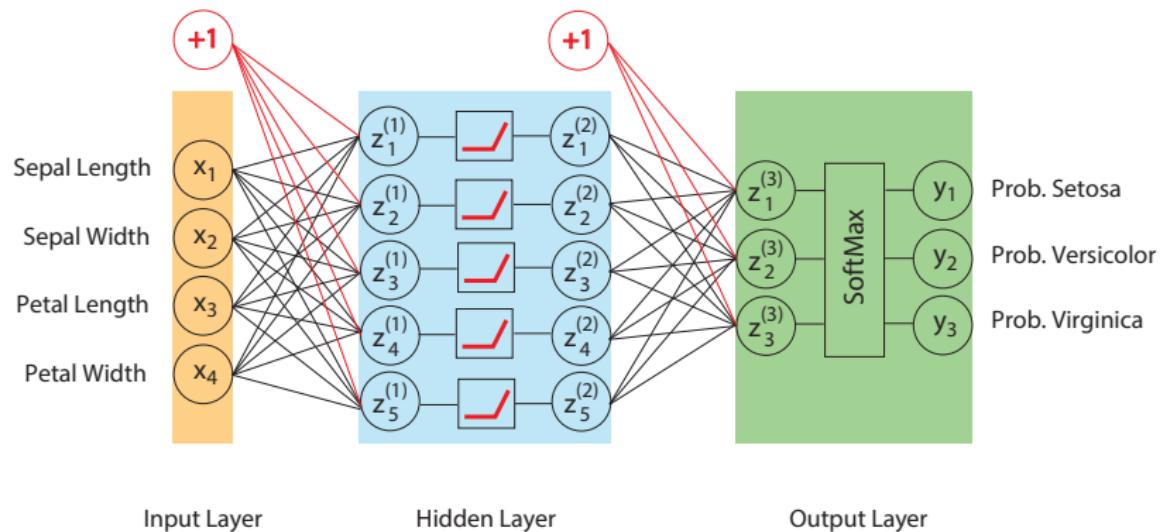
Bonus 1: Bias node



Bonus 1: Bias node



Bonus 1: Bias node



Bonus 1: Bias node

We have

$$\begin{pmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \\ W_{51} & W_{52} & W_{53} & W_{54} \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix} = \begin{pmatrix} W_{11} X_1 + W_{12} X_2 + W_{13} X_3 + W_{14} X_4 \\ W_{21} X_1 + W_{22} X_2 + W_{23} X_3 + W_{24} X_4 \\ W_{31} X_1 + W_{32} X_2 + W_{33} X_3 + W_{34} X_4 \\ W_{41} X_1 + W_{42} X_2 + W_{43} X_3 + W_{44} X_4 \\ W_{51} X_1 + W_{52} X_2 + W_{53} X_3 + W_{54} X_4 \end{pmatrix}$$

Bonus 1: Bias node

We have

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \\ w_{51} & w_{52} & w_{53} & w_{54} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 \\ w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 \\ w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + w_{34}x_4 \\ w_{41}x_1 + w_{42}x_2 + w_{43}x_3 + w_{44}x_4 \\ w_{51}x_1 + w_{52}x_2 + w_{53}x_3 + w_{54}x_4 \end{pmatrix}$$

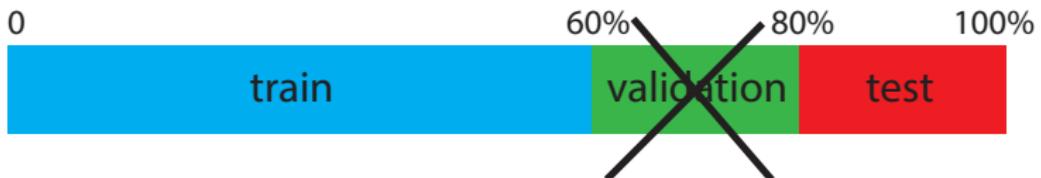
We now add a **bias node**:

$$\begin{pmatrix} w_{10} & w_{11} & w_{12} & w_{13} & w_{14} \\ w_{20} & w_{21} & w_{22} & w_{23} & w_{24} \\ w_{30} & w_{31} & w_{32} & w_{33} & w_{34} \\ w_{40} & w_{41} & w_{42} & w_{43} & w_{44} \\ w_{50} & w_{51} & w_{52} & w_{53} & w_{54} \end{pmatrix} \begin{pmatrix} +1 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} w_{10} + w_{11}x_1 + w_{12}x_2 + \dots \\ w_{20} + w_{21}x_1 + w_{22}x_2 + \dots \\ w_{30} + w_{31}x_1 + w_{32}x_2 + \dots \\ w_{40} + w_{41}x_1 + w_{42}x_2 + \dots \\ w_{50} + w_{51}x_1 + w_{52}x_2 + \dots \end{pmatrix}$$

The bias node introduces the intercepts $w_{10}, w_{20}, w_{30}, w_{40}, w_{50}$

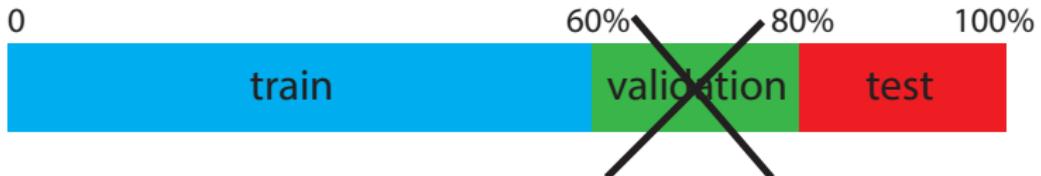
Bonus 2: K-fold cross-validation for model selection

For small datasets, one may not want to keep a separate validation dataset:



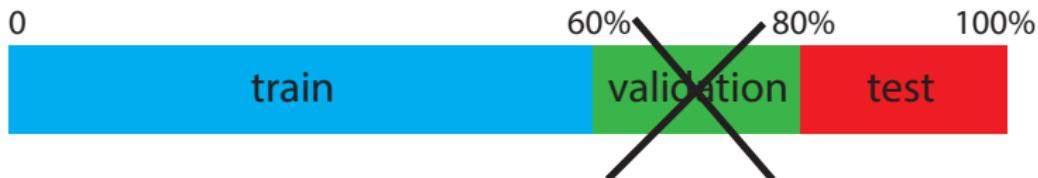
Bonus 2: K-fold cross-validation for model selection

For small datasets, one may not want to keep a separate validation dataset:



Bonus 2: K-fold cross-validation for model selection

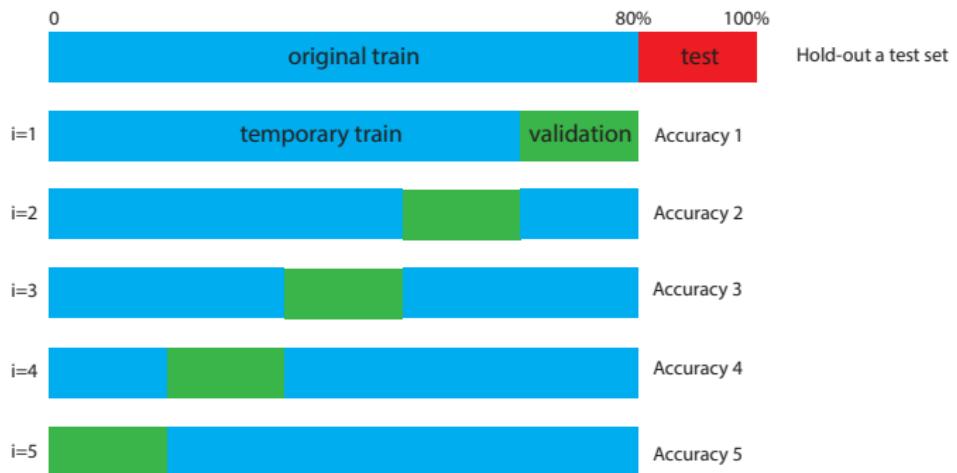
For small datasets, one may not want to keep a separate validation dataset:



- Model selection: Evaluate the validation accuracy of different models by using K-fold cross-validation.
- Model assessment: Use the training set to fit the best selected model and the test set to evaluate its accuracy.

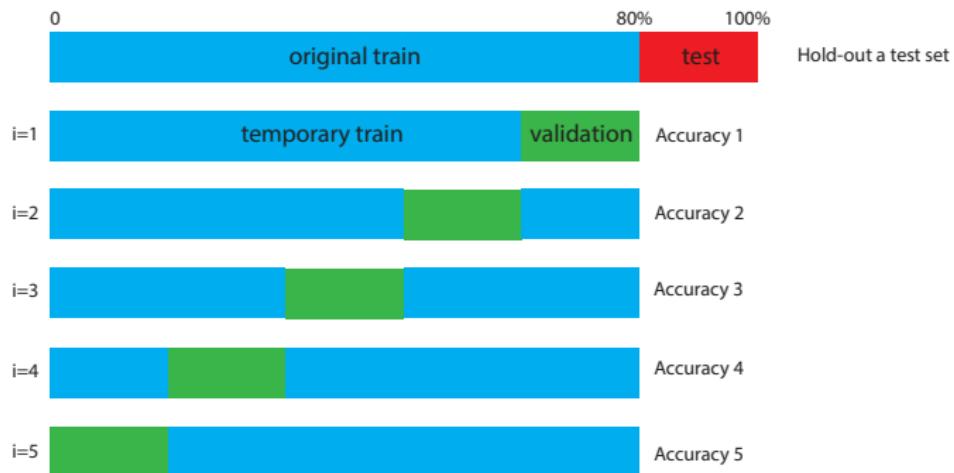
Bonus 2: K-fold cross-validation for model selection

Estimate the validation accuracy of a model by K-fold cross-validation (K=5):



Bonus 2: K-fold cross-validation for model selection

Estimate the validation accuracy of a model by K-fold cross-validation (K=5):

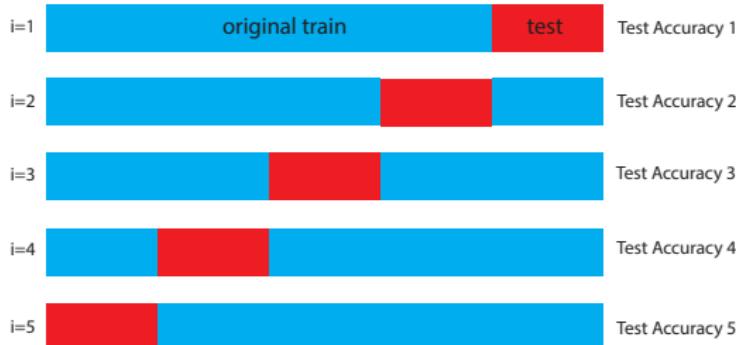


The Cross-Validation accuracy rate:

$$CV = \frac{1}{K} \sum_{i=1}^K \text{Accuracy}_i$$

Bonus 2: K-fold CV for model assessment

The best selected model (obtained by **inner** K-fold CV) and its accuracy may depend on the particular training/test splitting. One may check the stability of the best model and its prediction accuracy by using **outer** K-fold CV ($K=5$):

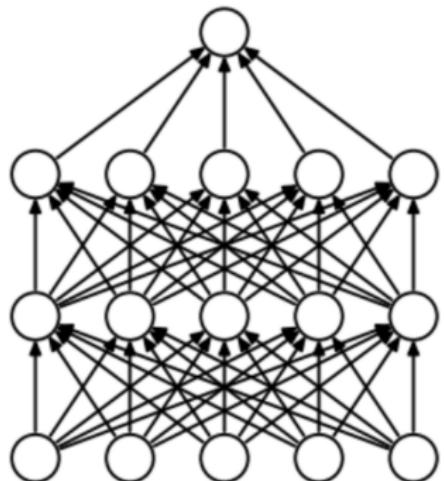


The Cross-Validation Test accuracy rate:

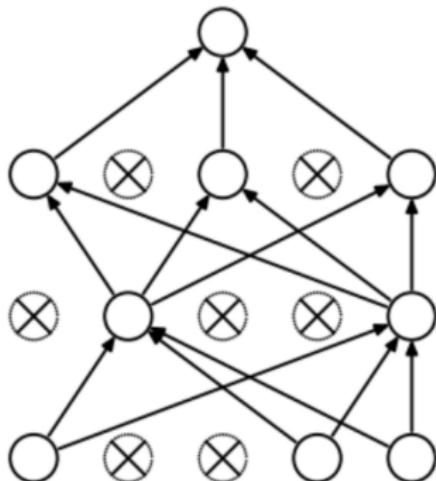
$$\text{CV Test} = \frac{1}{K} \sum_{i=1}^K \text{Test Accuracy}_i$$

Bonus 3: Dropout nodes

Dropout is a regularization technique for reducing overfitting in neural networks. One drops out units (both hidden and visible) during training according to a random distribution. This prevents units from co-adapting too much.



(a) Standard Neural Net



(b) After applying dropout.

Bonus 3: Dropout nodes

- Training Phase: For each selected layer and for each training iteration, ignore a random fraction $1-p$ of nodes. Different nodes will be dropped at each iteration.

Bonus 3: Dropout nodes

- Training Phase: For each selected layer and for each training iteration, ignore a random fraction $1-p$ of nodes. Different nodes will be dropped at each iteration.
- Testing Phase: Use all nodes but make normalization (to account for the missing nodes during training).

Bonus 4: Back-propagation

The **error function**:

$$E_n(W) = - \sum_{k=1}^3 t_{nk} \cdot \log y_{nk}(W)$$

The weights are updated with the **gradient descent algorithm**:

$$(w_{ij}^{(1)})^\tau = (w_{ij}^{(1)})^{\tau-1} - \eta \cdot \frac{\partial E_n}{\partial w_{ij}^{(1)}}$$

$$(w_{ij}^{(2)})^\tau = (w_{ij}^{(2)})^{\tau-1} - \eta \cdot \frac{\partial E_n}{\partial w_{ij}^{(2)}}$$

Bonus 4: Back-propagation

The **error function**:

$$E_n(W) = - \sum_{k=1}^3 t_{nk} \cdot \log y_{nk}(W)$$

The weights are updated with the **gradient descent algorithm**:

$$(w_{ij}^{(1)})^\tau = (w_{ij}^{(1)})^{\tau-1} - \eta \cdot \frac{\partial E_n}{\partial w_{ij}^{(1)}}$$

$$(w_{ij}^{(2)})^\tau = (w_{ij}^{(2)})^{\tau-1} - \eta \cdot \frac{\partial E_n}{\partial w_{ij}^{(2)}}$$

The gradients are computed by **back-propagation**:

$$(1) \frac{\partial E_n}{\partial w_{ij}^{(2)}} \qquad (2) \frac{\partial E_n}{\partial w_{ij}^{(1)}}$$

Bonus 4: Back-propagation

After some analytical computations:

$$\frac{\partial E_n}{\partial w_{ij}^{(2)}} = \Delta_{ni}^{(2)} \cdot z_{nj}^{(2)}$$

$$\frac{\partial E_n}{\partial w_{ij}^{(1)}} = \Delta_{ni}^{(1)} \cdot x_{nj}$$

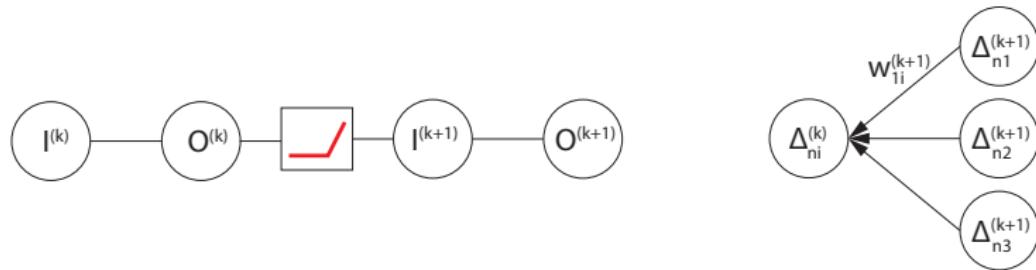
with errors:

$$\Delta_{ni}^{(2)} = (y_{ni} - t_{ni})$$

$$\Delta_{ni}^{(1)} = ReLU'(z_{ni}^{(1)}) \sum_{p=1}^3 w_{pi}^{(2)} \Delta_{np}^{(2)}$$

Bonus 4: Back-propagation

Relations between layer k and $k + 1$:



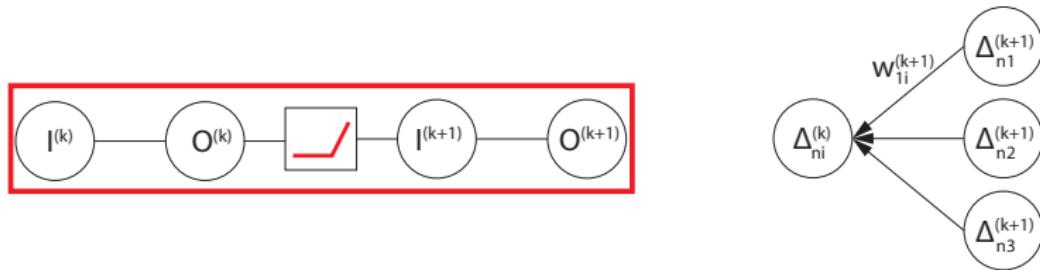
The **back-propagation equations**:

$$\frac{\partial E_n}{\partial w_{ij}^{(k)}} = \Delta_{ni}^{(k)} \cdot I_{nj}^{(k)}$$

$$\Delta_{ni}^{(k)} = \text{ReLU}'(O_{ni}^{(k)}) \sum_{p=1}^3 w_{pi}^{(k+1)} \Delta_{np}^{(k+1)}$$

Bonus 4: Back-propagation

Relations between layer k and $k + 1$:



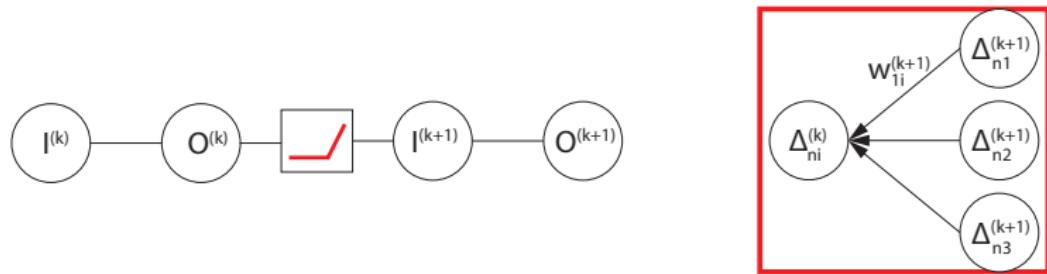
The **back-propagation equations**:

$$\frac{\partial E_n}{\partial w_{ij}^{(k)}} = \Delta_{ni}^{(k)} \cdot I_{nj}^{(k)}$$

$$\Delta_{ni}^{(k)} = \text{ReLU}'(\textcolor{red}{O}_{ni}^{(k)}) \sum_{p=1}^3 w_{pi}^{(k+1)} \Delta_{np}^{(k+1)}$$

Bonus 4: Back-propagation

Relations between layer k and $k + 1$:



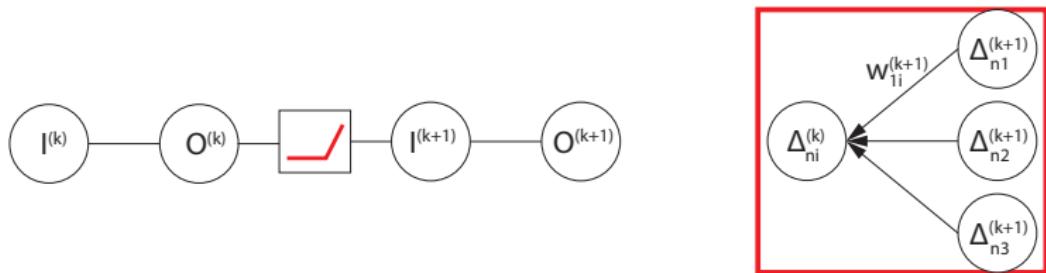
The **back-propagation equations**:

$$\frac{\partial E_n}{\partial w_{ij}^{(k)}} = \Delta_{ni}^{(k)} \cdot I_{nj}^{(k)}$$

$$\Delta_{ni}^{(k)} = \text{ReLU}'(O_{ni}^{(k)}) \sum_{p=1}^3 w_{pi}^{(k+1)} \Delta_{np}^{(k+1)}$$

Bonus 4: Back-propagation

Relations between layer k and $k + 1$:



The **back-propagation equations**:

$$\frac{\partial E_n}{\partial w_{ij}^{(k)}} = \Delta_{ni}^{(k)} \cdot I_{nj}^{(k)}$$

$$\Delta_n^{(k)} = \text{ReLU}'(O_n^{(k)}) \circ \left(W^{(k+1)} \right)^T \Delta_{n+1}^{(k+1)}$$

Bonus 5: Neural Network for Regression

Two modifications to go
from NN classification to NN regression

Bonus 5: Neural Network for Regression

Modification 1

Remove the SoftMax function

Modification 2

Replace the cross-entropy error by
the sum-of-squares error:

$$E(W) = \frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K |t_{nk} - y_{nk}(W)|^2$$

QUESTIONS ?

Coffee break

COFFEE BREAK

The iris decision tree

The iris decision tree

The iris decision tree

input

output

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.3	3.3	6.0	2.5	virginica

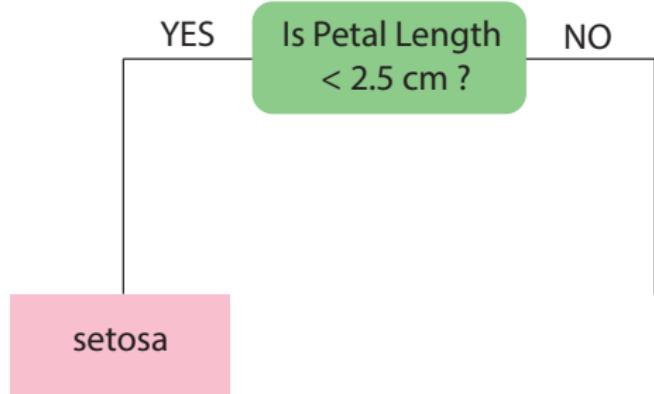
The iris decision tree

A decision tree is a model
that predicts the output
by answering questions on the input

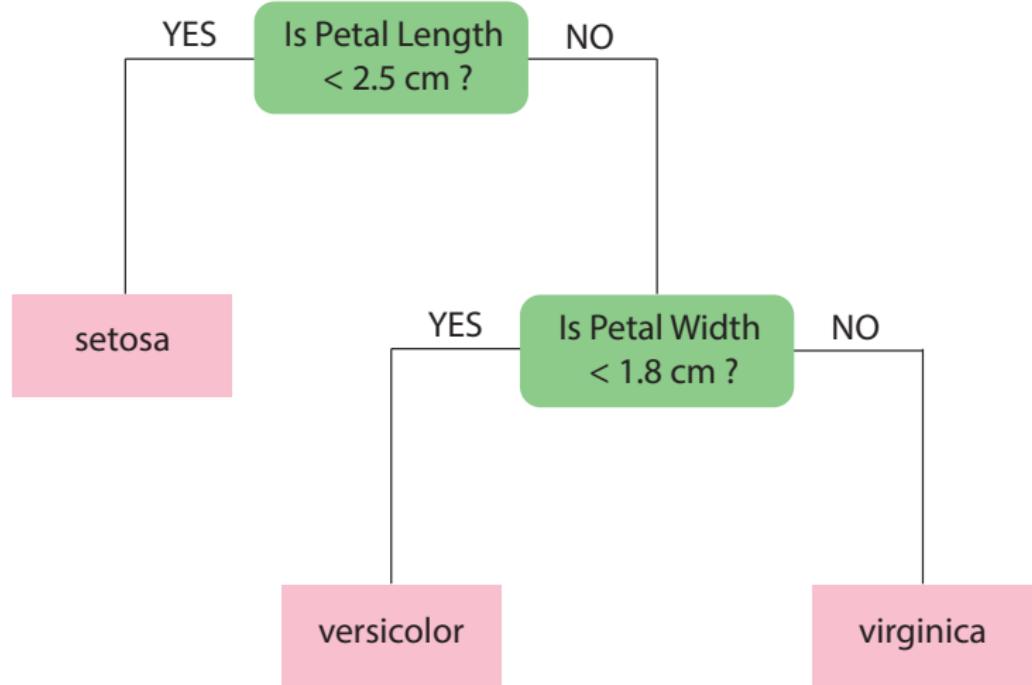
The iris decision tree

EXAMPLE

The iris decision tree

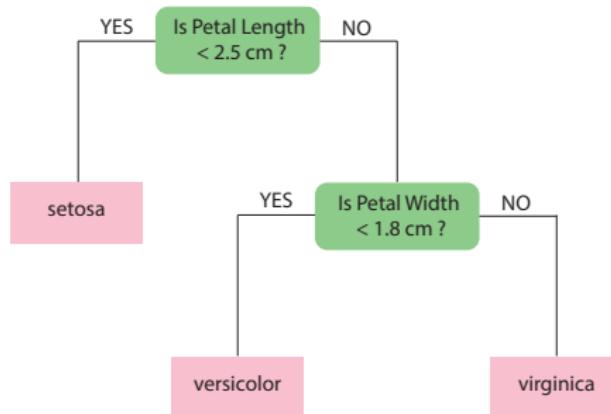


The iris decision tree



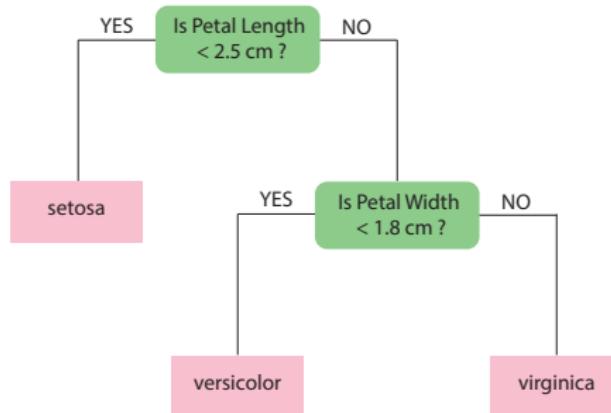
The iris decision tree

Input				output
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.3	3.3	6.0	2.5	virginica



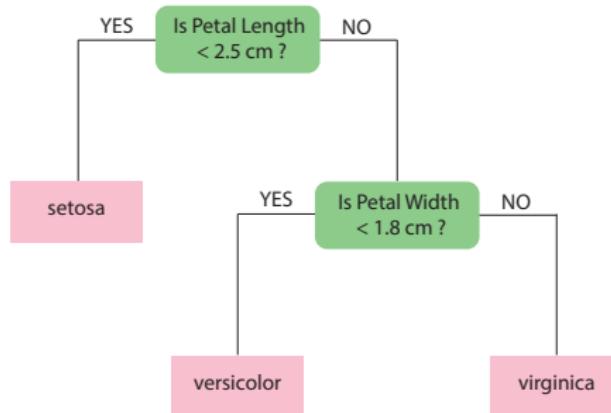
The iris decision tree

Input				output
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.3	3.3	6.0	2.5	virginica



The iris decision tree

Input				output
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.3	3.3	6.0	2.5	virginica



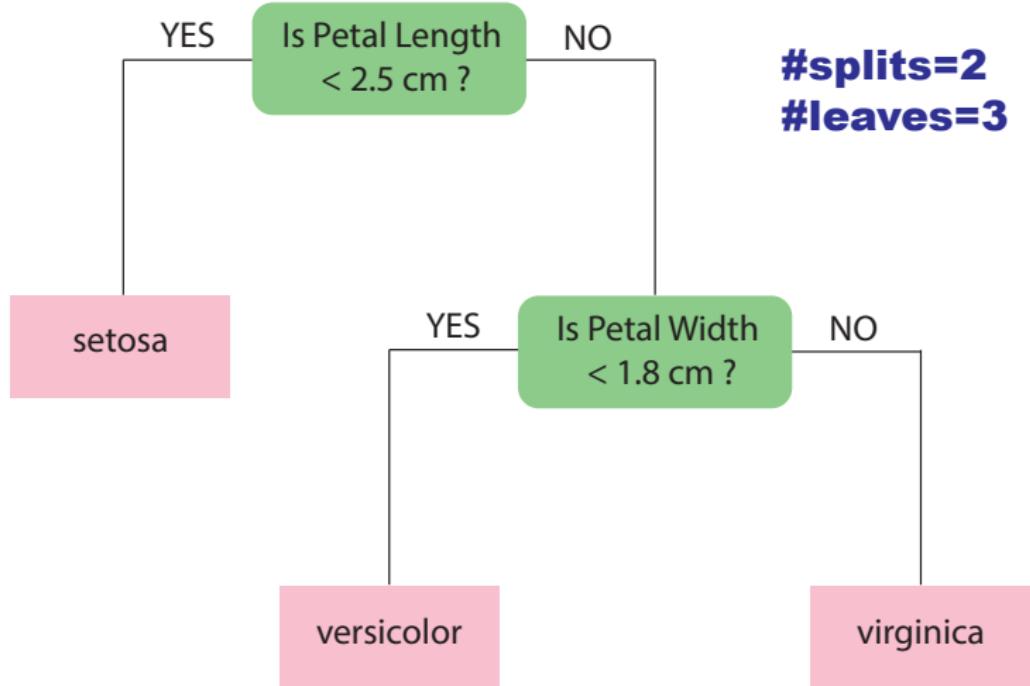
How to choose the splitting features ?

How to choose
the questions ?

How to choose the splitting features ?

One wants to answer
as few as possible questions
to determine which species
a given plant belongs to

The iris decision tree



How to choose the splitting features ?

This is a global optimization problem
that cannot be handled computationally

How to choose the splitting features ?

We start with all the training data
and choose a **locally** optimal question
that splits the data at each stage

How to choose the splitting features ?

Information

Information content of a leaf

If I take randomly a sample from the leaf,
how much do I know about its class

The iris decision tree

EXAMPLE

The iris decision tree

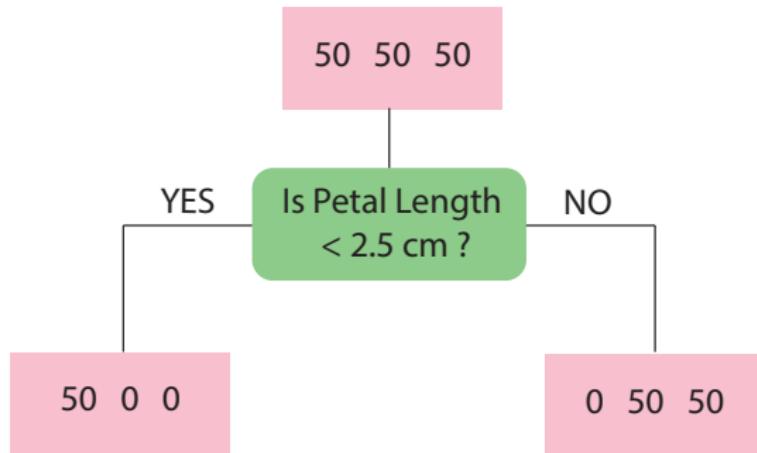
50 50 50

The iris decision tree

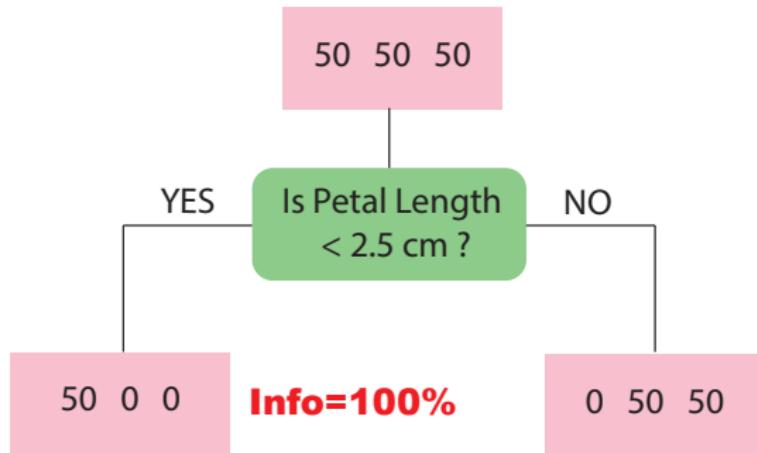
50 50 50

Info=0

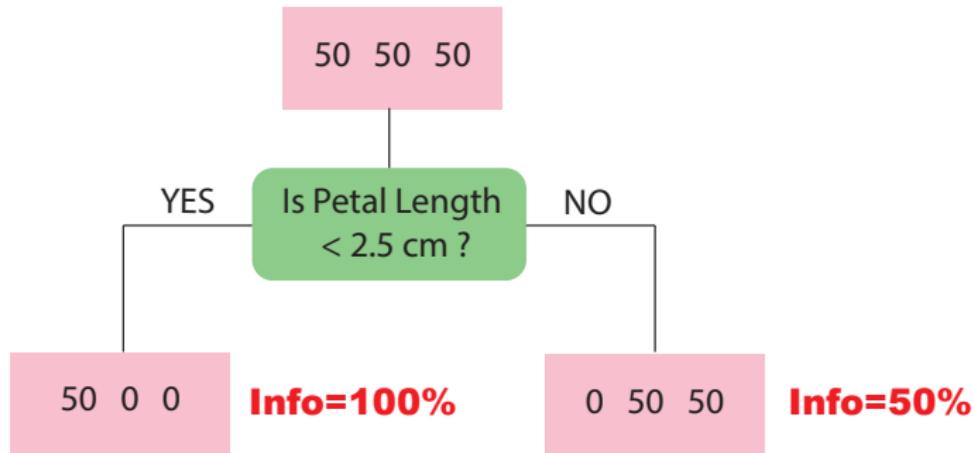
The iris decision tree



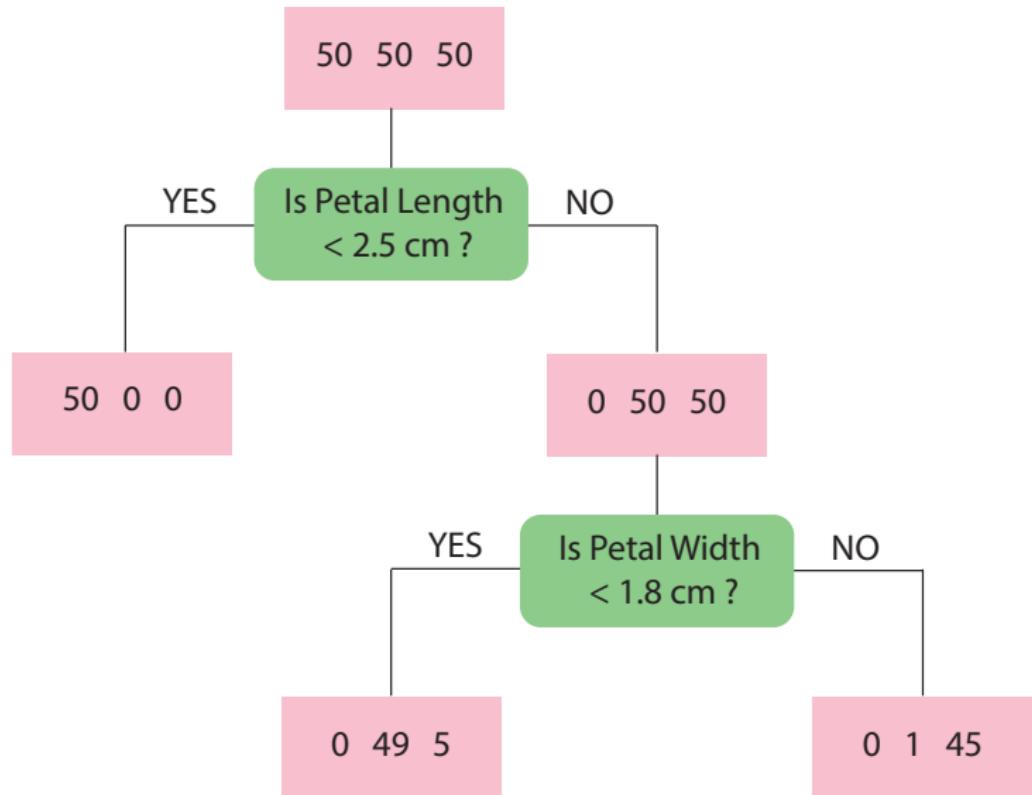
The iris decision tree



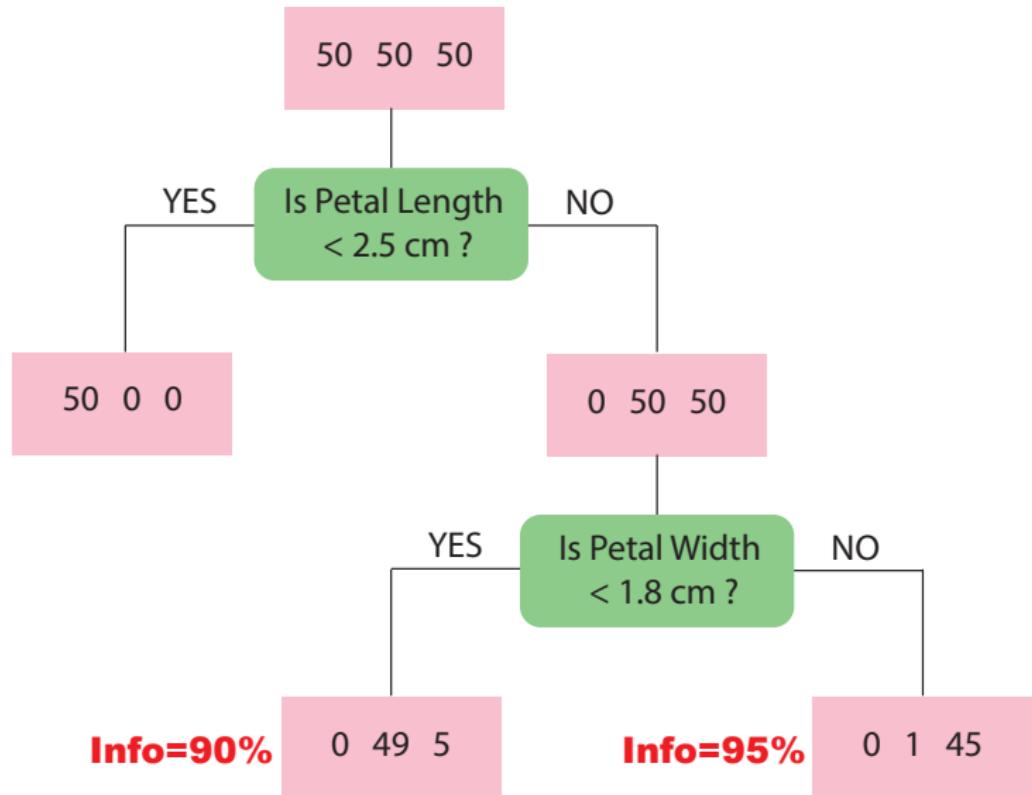
The iris decision tree



The iris decision tree



The iris decision tree

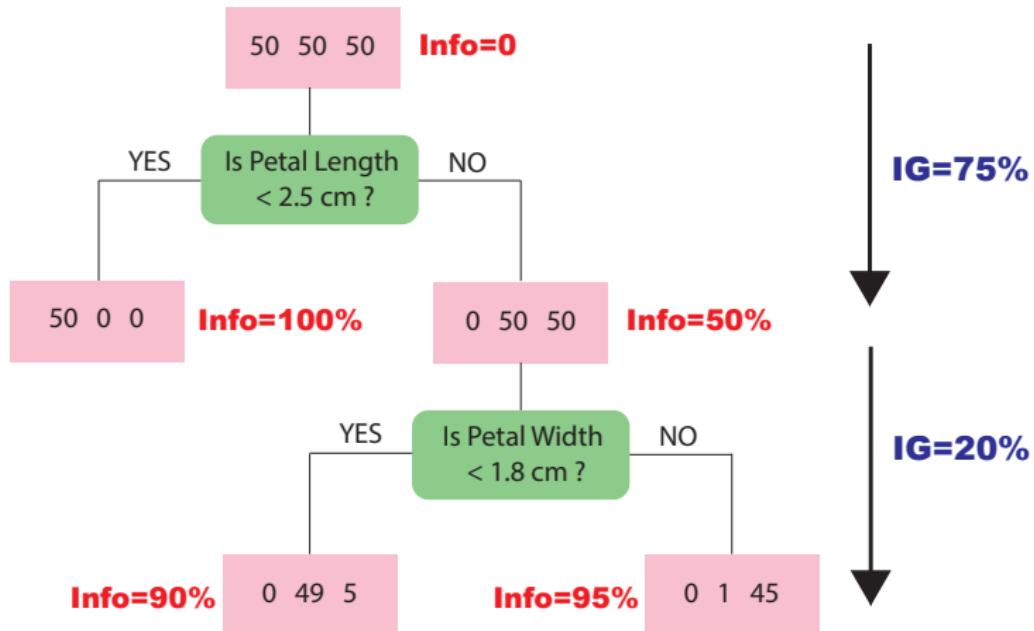


How to choose the splitting features ?

We start at the root
and split the training data on the feature
that results in the largest **information gain**

$$IG = \text{Info}(D_{\text{children}}) - \text{Info}(D_{\text{parent}})$$

How to choose the splitting features ?



How to choose the splitting features ?

Assume $\text{Info}(D_{\text{parent}}) \in [0, 1]$:

$$\begin{aligned} IG &= \text{Info}(D_{\text{children}}) - \text{Info}(D_{\text{parent}}) \\ &= [1 - \text{Info}(D_{\text{parent}})] - [1 - \text{Info}(D_{\text{children}})] \\ &= \text{LackInfo}(D_{\text{parent}}) - \text{LackInfo}(D_{\text{children}}) \end{aligned}$$

How to choose the splitting features ?

LackInfo is called the **impurity index I**:

$$IG = I(D_{\text{parent}}) - \left[\frac{N_{\text{left child}}}{N_{\text{parent}}} I(D_{\text{left child}}) + \frac{N_{\text{right child}}}{N_{\text{parent}}} I(D_{\text{right child}}) \right]$$

How to choose the splitting features ?

The **impurity index**:

$$I_E(D_\ell) = 1 - \max_{c=1,2,3} \{p_{\ell c}\} \quad (\text{Misclassification error})$$

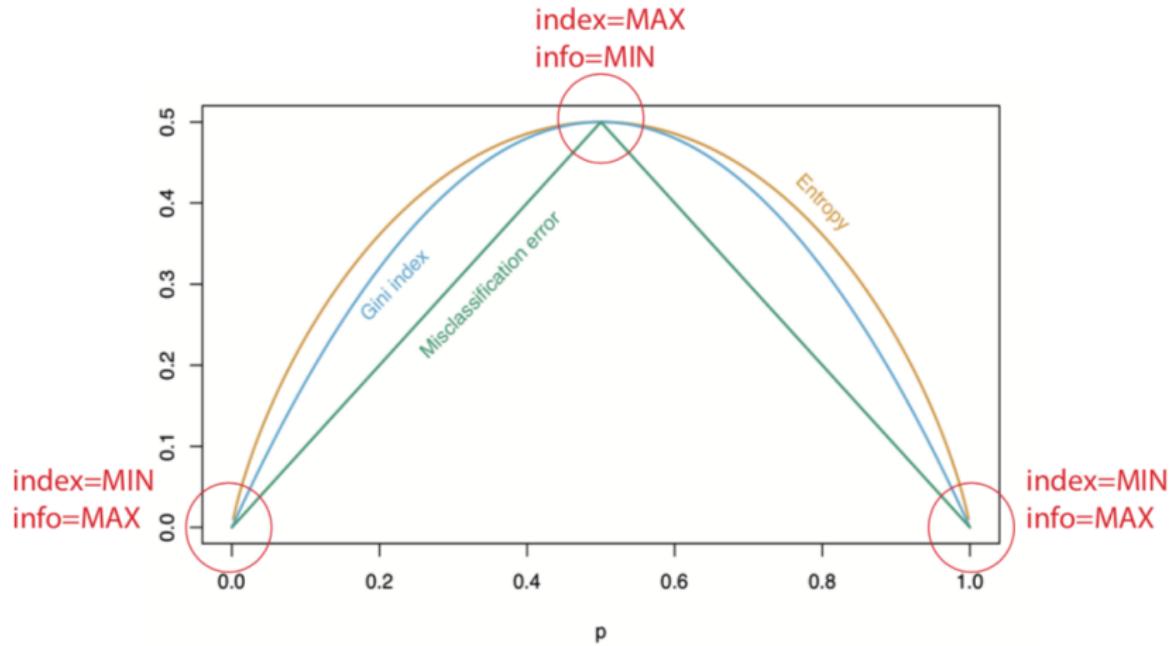
$$I_H(D_\ell) = - \sum_{c=1}^3 p_{\ell c} \log_2(p_{\ell c}) \quad (\text{Entropy or Deviance})$$

$$I_G(D_\ell) = 1 - \sum_{c=1}^3 p_{\ell c}^2 \quad (\text{Gini index})$$

where $p_{\ell c}$ is the proportion of data points at leaf ℓ that belongs to class $c = 1, 2, 3$.

How to choose the splitting features ?

In the case of 2 classes:



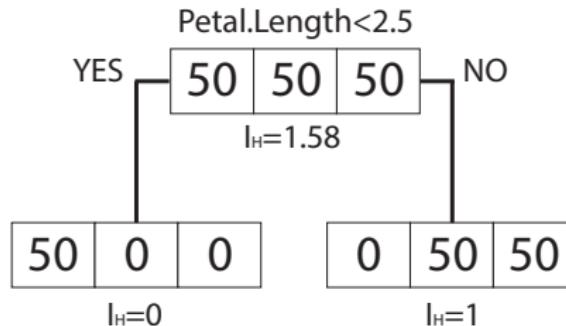
Conclusion: we want $\text{info}=\text{MAX}$ or $I=\text{MIN}$ at the children leaves

How to choose the splitting features ?

EXAMPLE

How to choose the splitting features ?

Case A

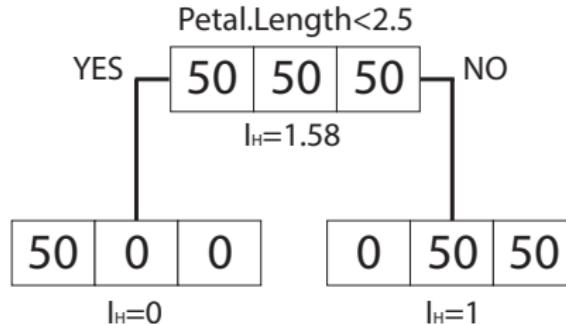


Information Gain:

$$IG = 1.58 - [(50/150)*0 + (100/150)*1] \\ = 0.92$$

How to choose the splitting features ?

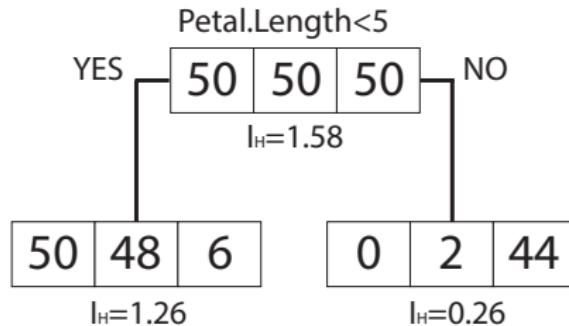
Case A



Information Gain:

$$IG = 1.58 - [(50/150)*0 + (100/150)*1] \\ = 0.92$$

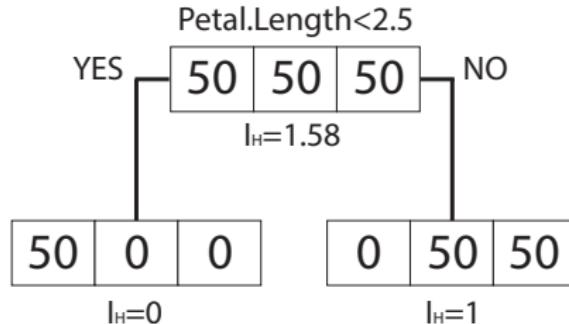
Case B



$$IG = 1.58 - [(104/150)*1.26 + (46/150)*0.26] \\ = 0.63$$

How to choose the splitting features ?

Case A

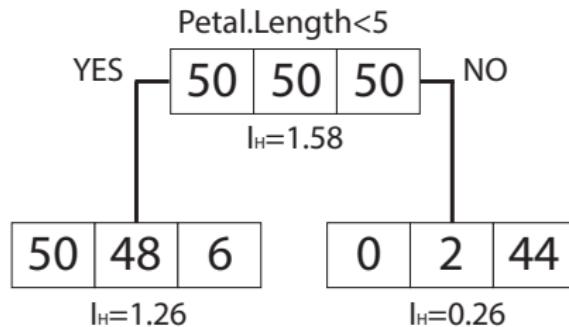


Information Gain:

$$IG = 1.58 - [(50/150)*0 + (100/150)*1] \\ = 0.92$$

IG(Case A) > IG(Case B)

Case B



How to choose the splitting features ?

How many splitting features
should be tested ?

How to choose the splitting features ?

Order the values from smallest to biggest:

n	1	2	...	149	150
Petal Length	1.0	1.1	...	6.7	6.9
Petal Width	0.1	0.1	...	2.5	2.5
Sepal Length	4.3	4.4	...	7.7	7.9
Sepal Width	2.0	2.2	...	4.2	4.4

There are $4 \times 150 = 600$ possible questions (splitting features)

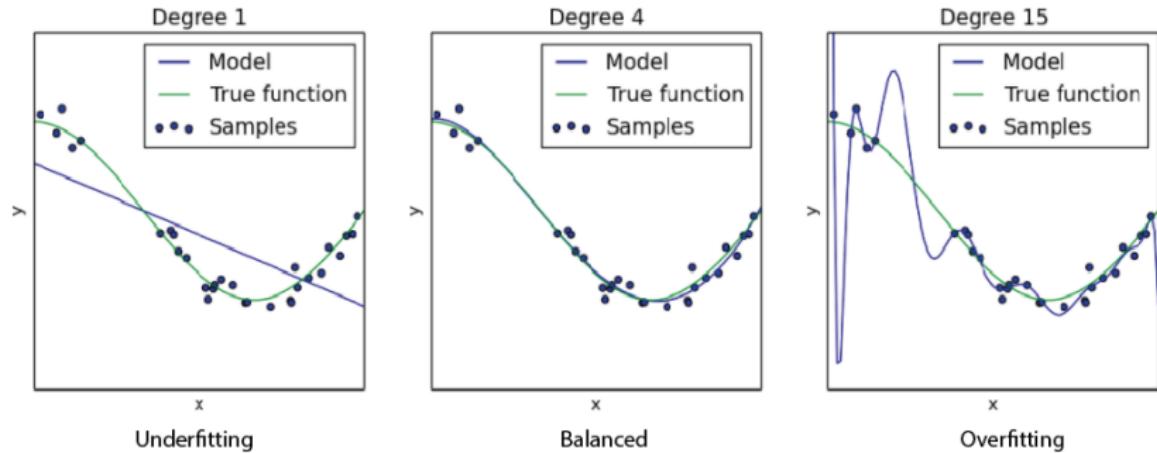
When to stop splitting features ?

When to stop splitting features ?

When to stop splitting features ?

The answer is related to the problem of
under-fitting and over-fitting

When to stop splitting features ?



Ref: scikit-learn 0.18 documentation

When to stop splitting features ?

If the tree is too small
(too few splits),
it may under-fit the data

When to stop splitting features ?

If the tree is too big

(too many splits),

it may over-fit the data

When to stop splitting features ?

Method 1

Use stopping rules

that will prevent any node being split

if those conditions are not met

When to stop splitting features ?

minsplit

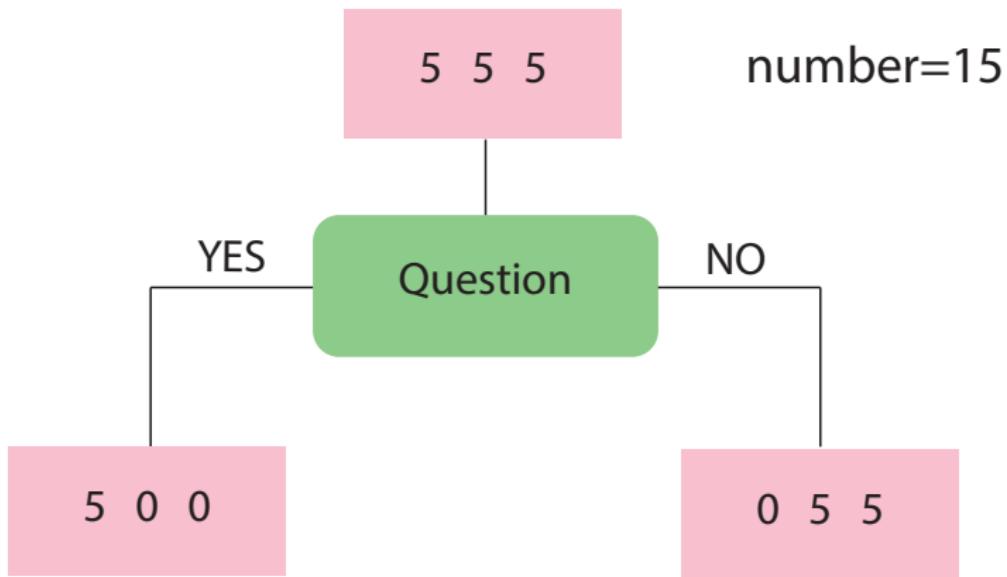
The minimum number of samples

that must exist in a node

in order for a split to be attempted

When to stop splitting features ?

minsplit=10



When to stop splitting features ?

minsplit=10

5 0 4

number=9

DO NOT SPLIT

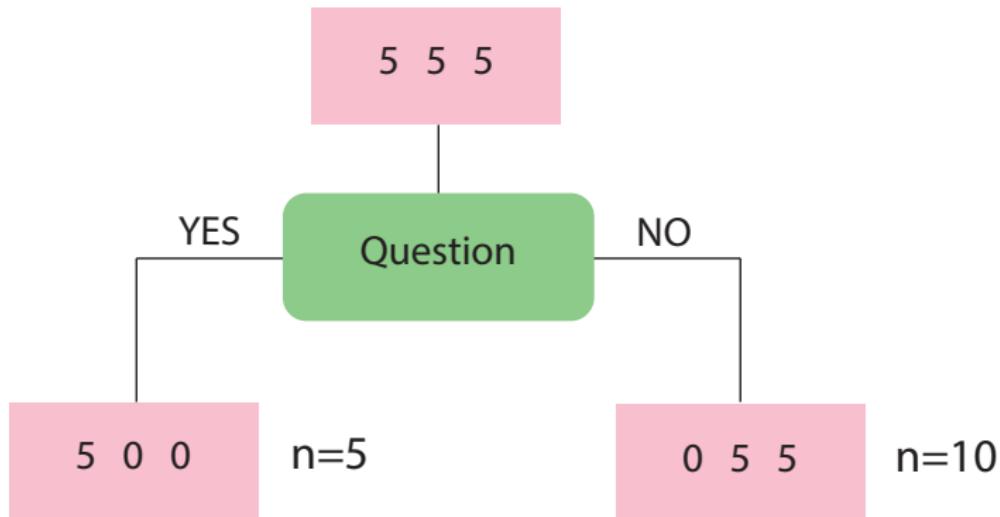
When to stop splitting features ?

minbucket

The minimum number of samples
in any terminal leaf

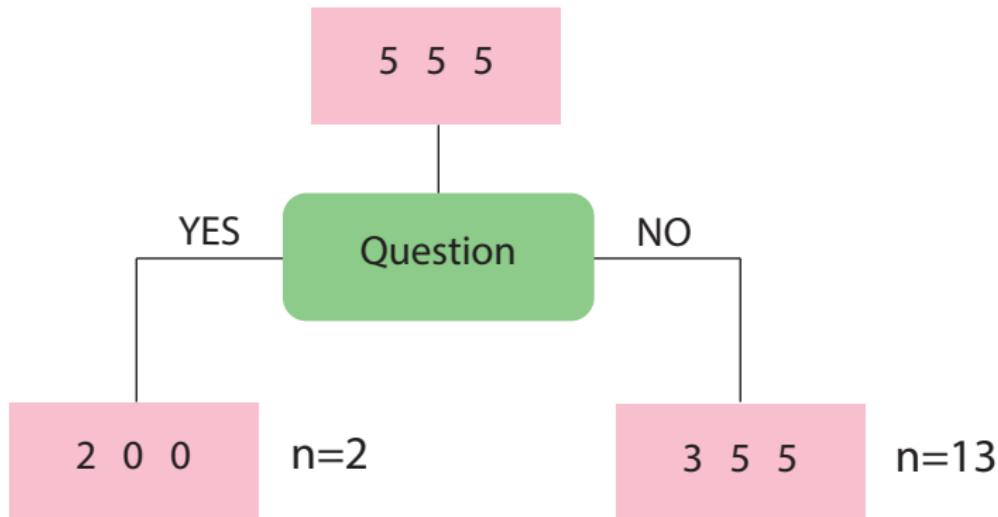
When to stop splitting features ?

minbucket=5



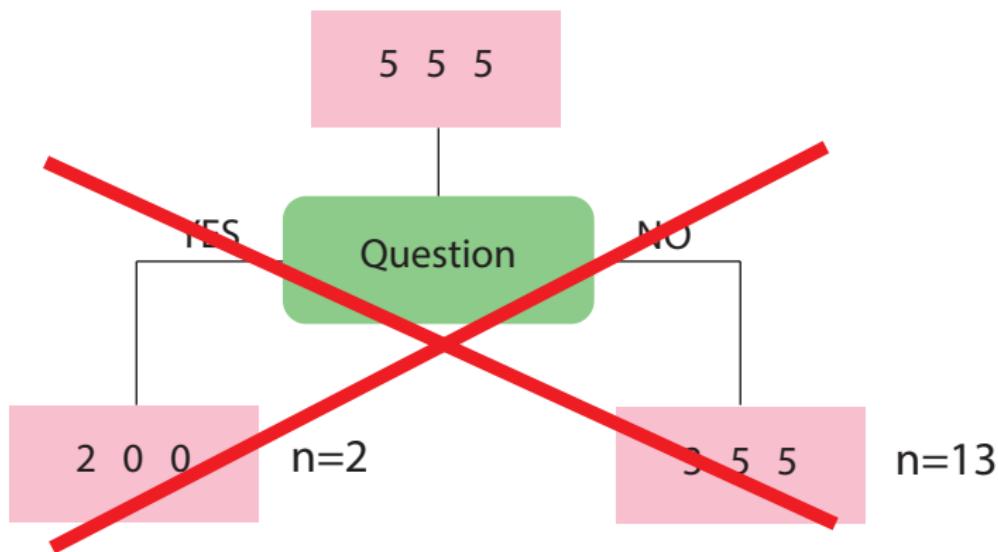
When to stop splitting features ?

minbucket=5



When to stop splitting features ?

minbucket=5



When to stop splitting features ?

Method 2

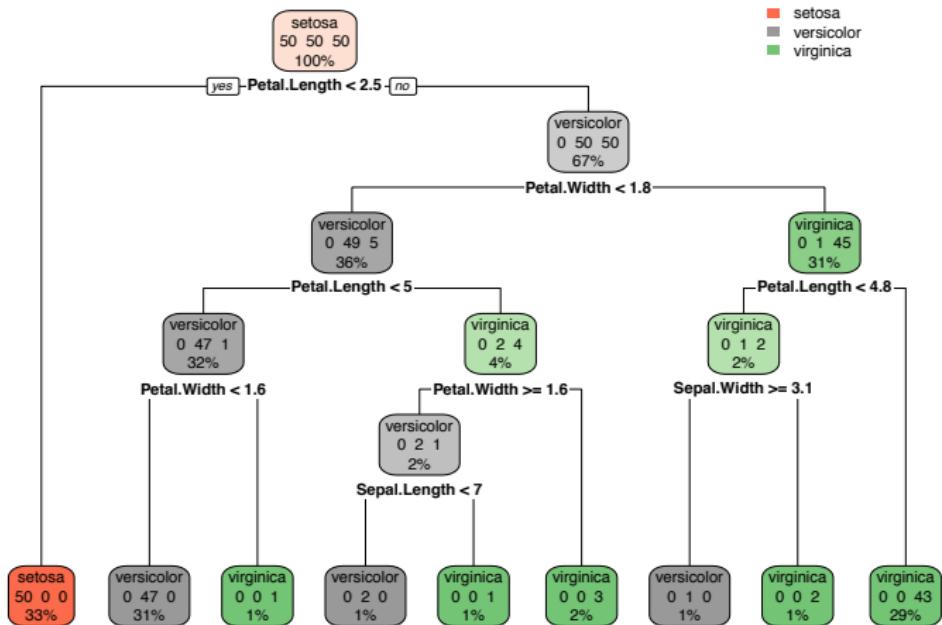
Prune the biggest tree from bottom-up:
stop when reaching minimum validation error

When to stop splitting features ?

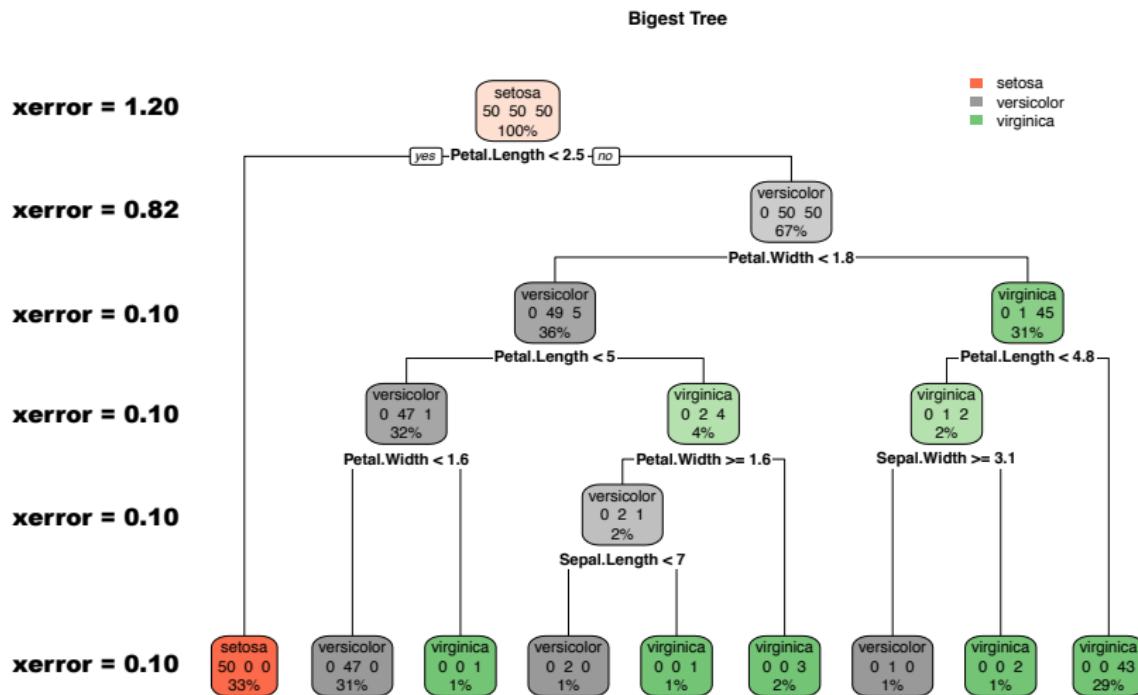
EXAMPLE

When to stop splitting features ?

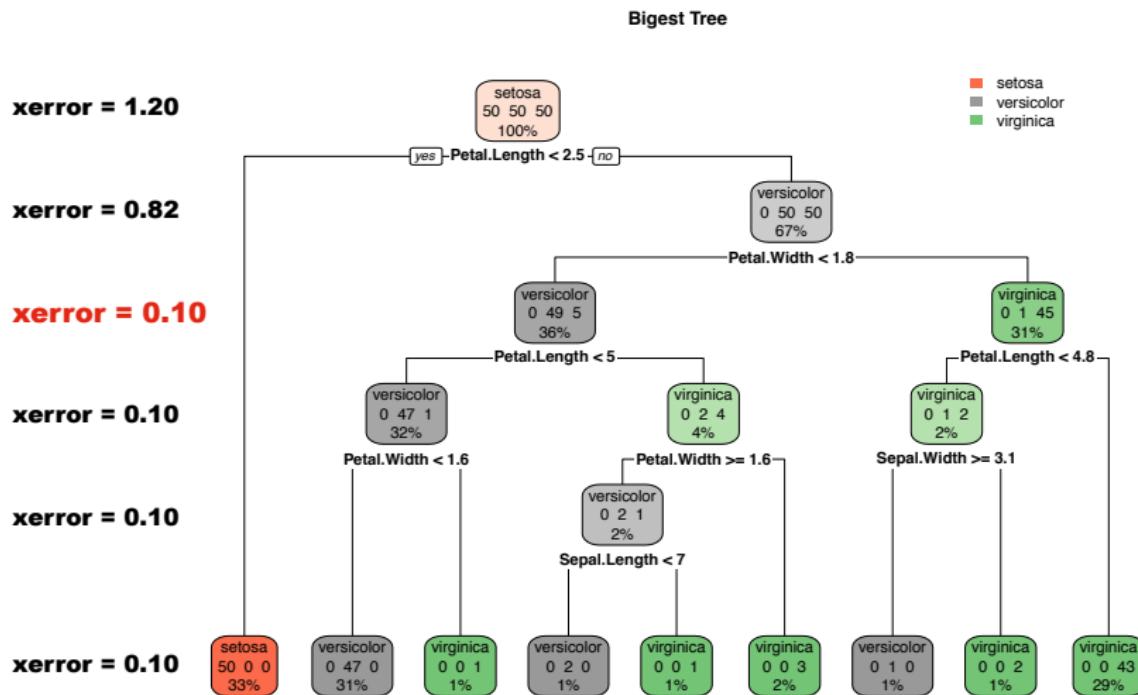
Bigest Tree



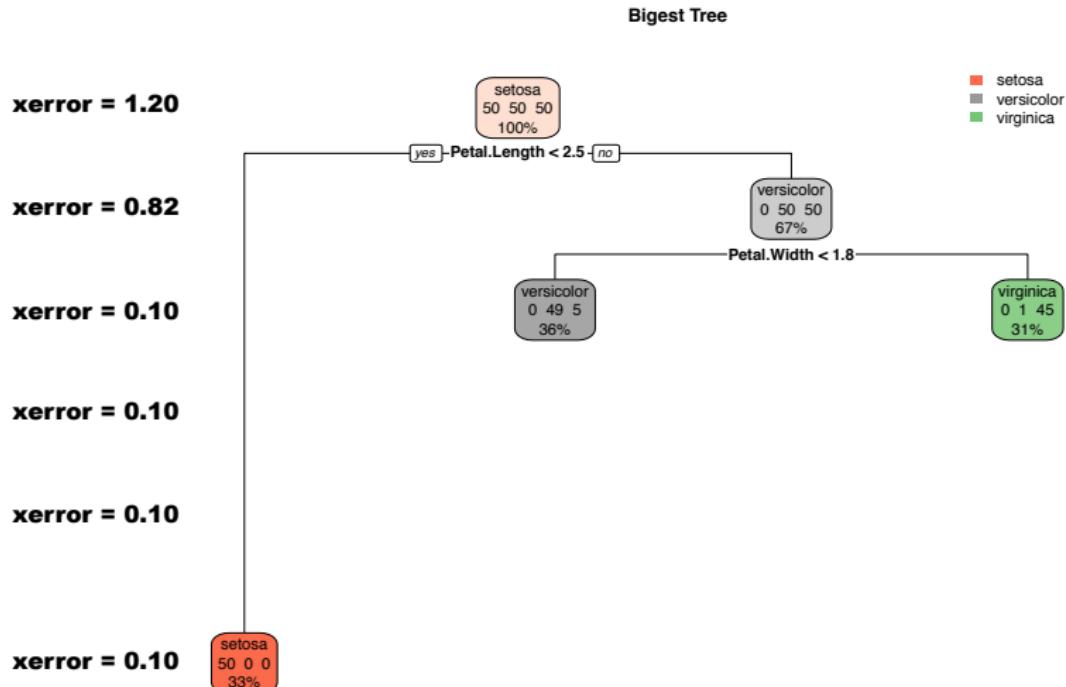
When to stop splitting features ?



When to stop splitting features ?

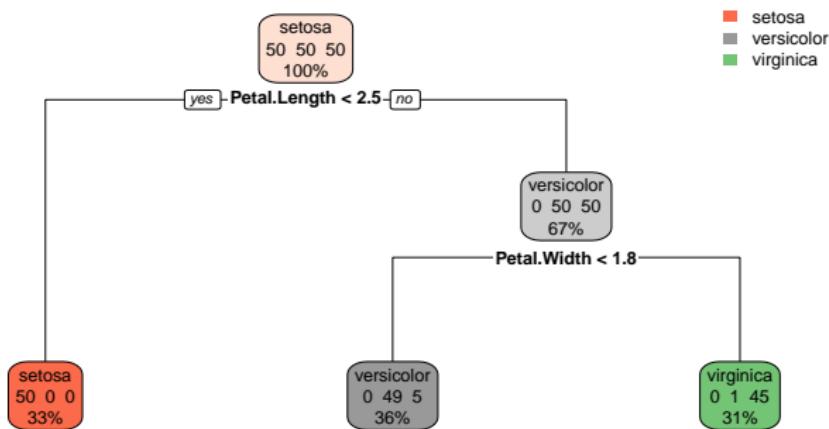


When to stop splitting features ?



When to stop splitting features ?

Pruned Tree

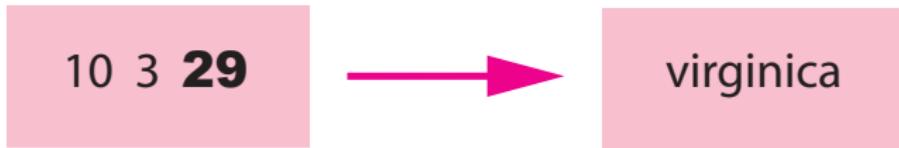
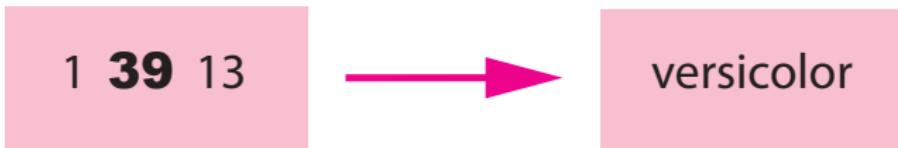
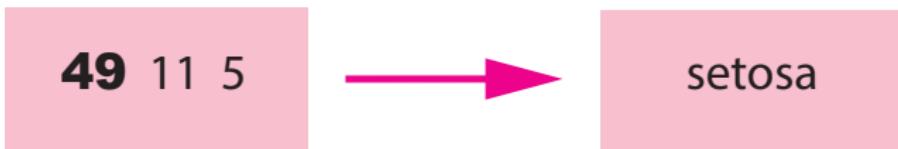


How to evaluate the decision tree performances ?

How to evaluate
the decision tree performances ?

How to evaluate the decision tree performances ?

At first we apply the **majority class rule** at each end node:



How to evaluate the decision tree performances ?

Apply the decision tree to test data to get **confusion matrix**:

		predictions		
		setosa	versicolor	virginica
actuals	setosa	14	0	0
	versicolor	0	9	0
virginica	0	2	10	

Use the following **performance metrics**:

$$\text{Accuracy rate} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{33}{35} = 94\%$$

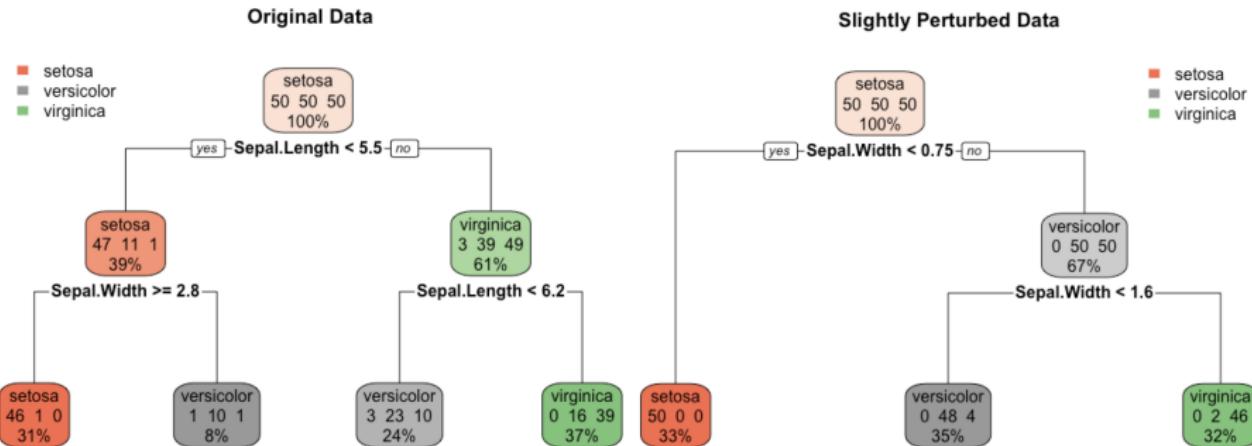
$$\text{Error rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}} = \frac{2}{35} = 6\%$$

Weakness

A decision tree is sensitive
to small data modifications
(a small change may give a very different tree)

Weakness

We perturb the data with ± 0.1 random numbers



Conclusion on decision tree

Hyperparameters:

- Splitting rule: impurity index
- Stopping rule: minsplit, minbucket, prune

Conclusion on decision tree

Hyperparameters:

- Splitting rule: impurity index
- Stopping rule: minsplit, minbucket, prune

Main advantage:

- Easy to visualise graphically and to interpret

Conclusion on decision tree

Hyperparameters:

- Splitting rule: impurity index
- Stopping rule: minsplit, minbucket, prune

Main advantage:

- Easy to visualise graphically and to interpret

Main disadvantage:

- Sensitive to training data (a small change in data may give a very different tree). This may be solved partially using a random forest, which is an ensemble of decision trees.

QUESTIONS ?

Random forest

Random Forest

Random forest

A decision tree has an important weakness:

a small change in the training dataset

may give a very different tree

EXAMPLE

If we split randomly the training data
into two parts and fit a DT to both halves,
they may be very different

Random forest

Decision tree has high variance

Random forest

How to reduce variance ?

Random forest

Average over a set of predictions

Random forest

Probabilistic setting

1 tree: X and Y are random variables with $V(Y) = \sigma^2$

Random forest

Probabilistic setting

1 tree: X and Y are random variables with $V(Y) = \sigma^2$

Random forest with M trees:

$$Y = \frac{1}{M} \sum_{i=1}^M Y_i \quad \text{Corr}(Y_i, Y_j) = \varrho$$

Then

$$V(Y) = \frac{1}{M}\sigma^2 + \frac{M-1}{M}\varrho\sigma^2$$

If M is large and ϱ is small, the RF output has small variance

How to generate
many uncorrelated trees ?

Random forest

- Generate many training datasets

Random forest

- Generate many training datasets
- Build a decision tree for each training dataset

Random forest

- Generate many training datasets
- Build a decision tree for each training dataset
- Apply the majority rule for the prediction

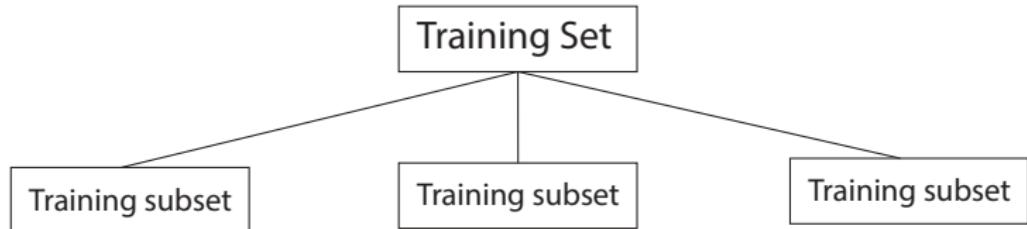
How to choose the splitting features ?

EXAMPLE

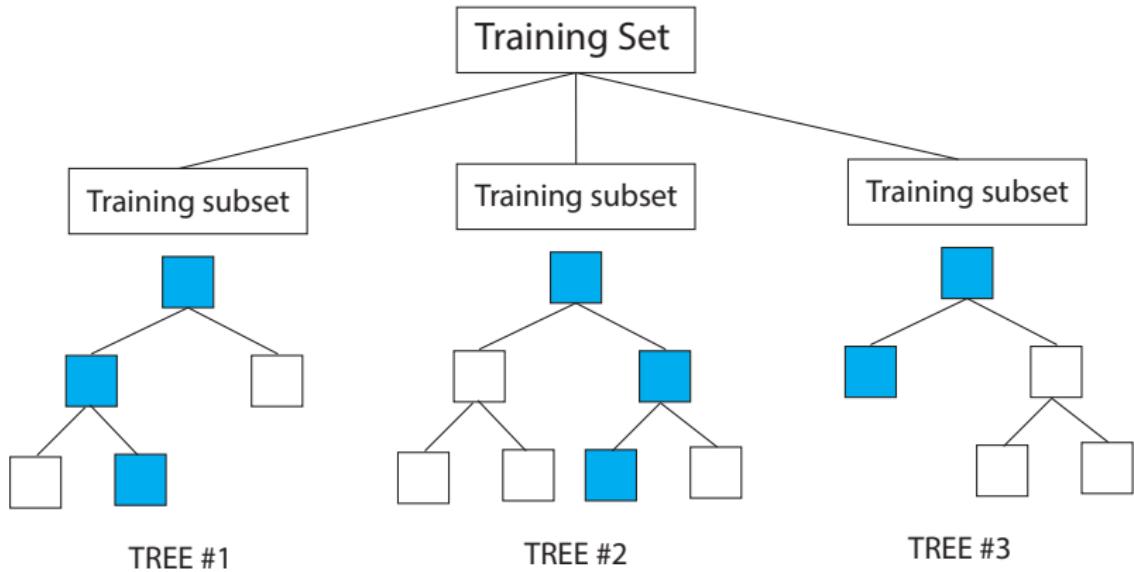
Random forest with 3 decision trees

Training Set

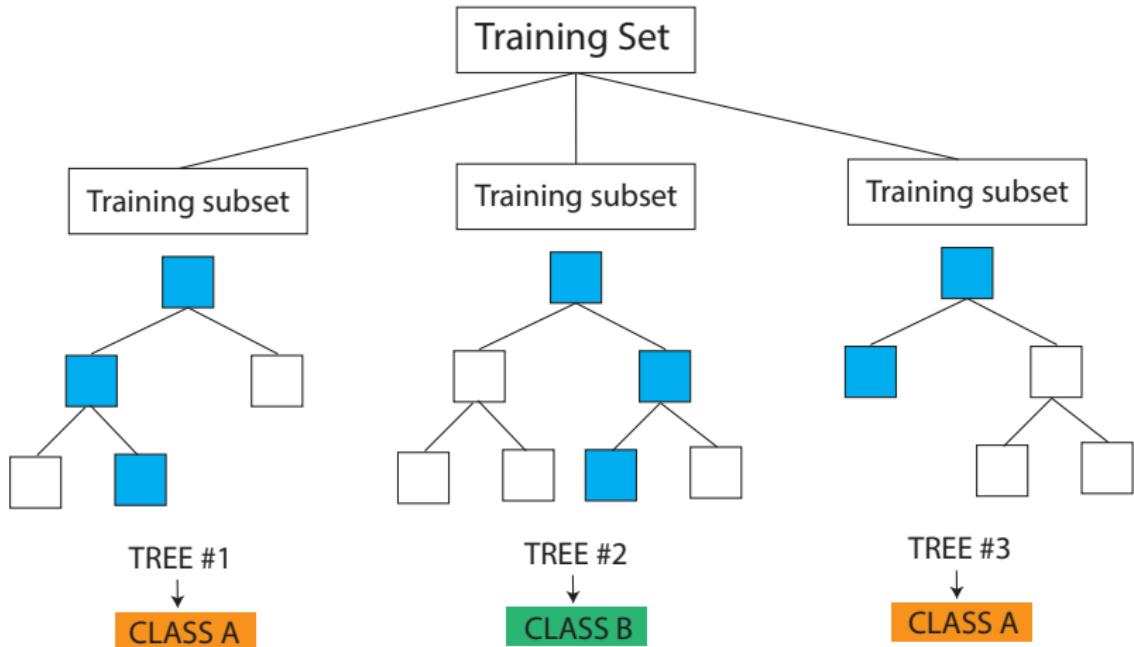
Random forest with 3 decision trees



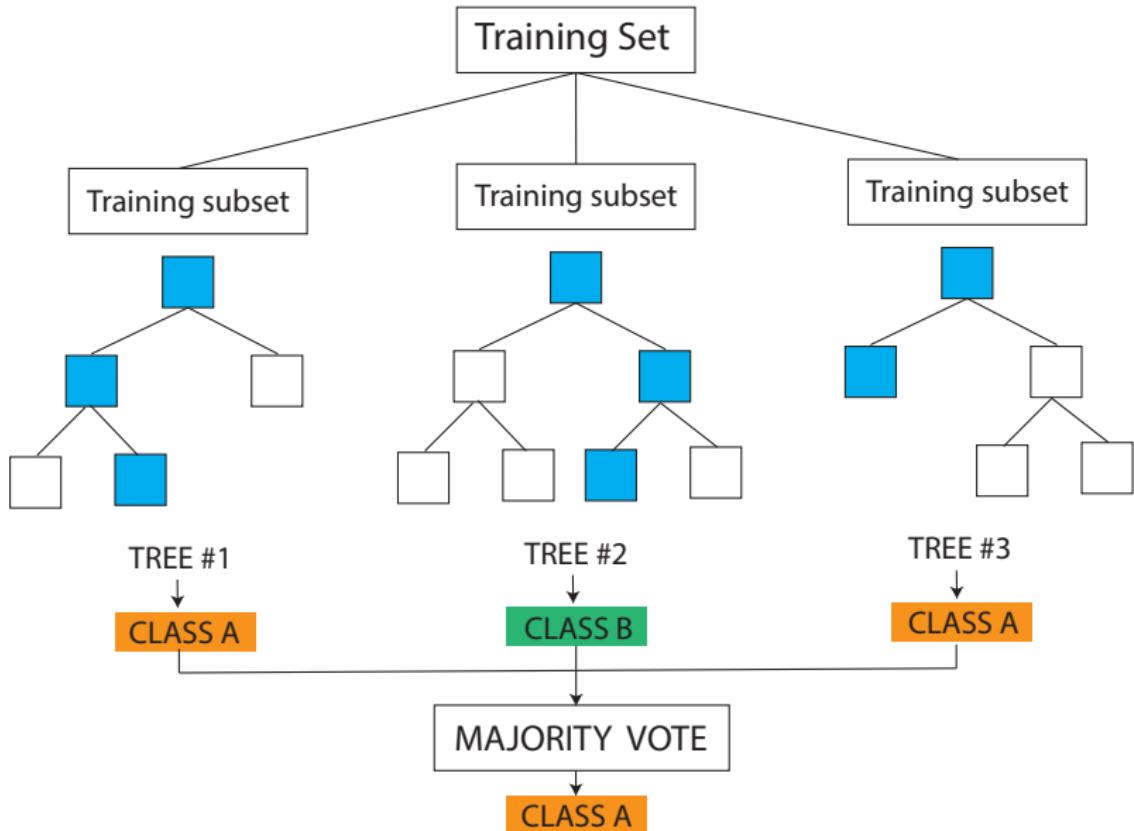
Random forest with 3 decision trees



Random forest with 3 decision trees



Random forest with 3 decision trees



Generate many training datasets

Algorithm: Draw B bootstrap samples of size $n < N$ with or without replacement ($B = 4, n = 4, N = 10$, no replacement):

Plant 1
Plant 2
Plant 3
Plant 4
Plant 5
Plant 6
Plant 7
Plant 8
Plant 9
Plant 10

Generate many training datasets

Algorithm: Draw B bootstrap samples of size $n < N$ with or without replacement ($B = 4, n = 4, N = 10$, no replacement):

Plant 1
Plant 2
Plant 3
Plant 4
Plant 5
Plant 6
Plant 7
Plant 8
Plant 9
Plant 10

Plant 1
Plant 2
Plant 3
Plant 4
Plant 5
Plant 6
Plant 7
Plant 8
Plant 9
Plant 10

Plant 1
Plant 2
Plant 3
Plant 4
Plant 5
Plant 6
Plant 7
Plant 8
Plant 9
Plant 10

Plant 1
Plant 2
Plant 3
Plant 4
Plant 5
Plant 6
Plant 7
Plant 8
Plant 9
Plant 10

Generate many training datasets

Algorithm: Draw B bootstrap samples of size $n < N$ with or without replacement ($B = 4, n = 4, N = 10$, no replacement):

Plant 1
Plant 2
Plant 3
Plant 4
Plant 5
Plant 6
Plant 7
Plant 8
Plant 9
Plant 10

Plant 1
Plant 2
Plant 3
Plant 4
Plant 5
Plant 6
Plant 7
Plant 8
Plant 9
Plant 10

Plant 1
Plant 2
Plant 3
Plant 4
Plant 5
Plant 6
Plant 7
Plant 8
Plant 9
Plant 10

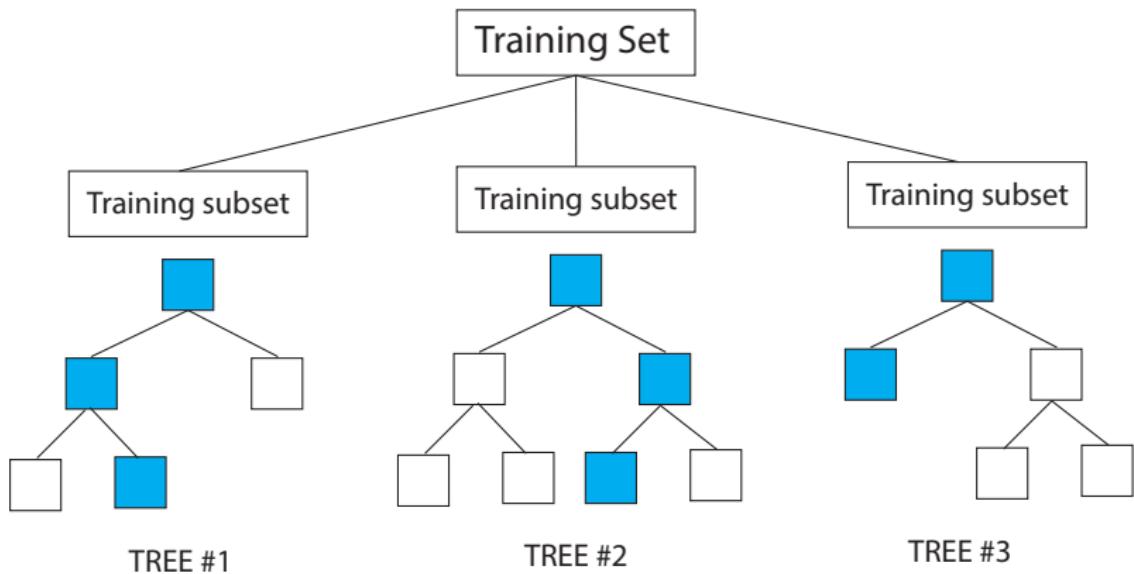
Plant 1
Plant 2
Plant 3
Plant 4
Plant 5
Plant 6
Plant 7
Plant 8
Plant 9
Plant 10

Build a decision tree for each training dataset

For each bootstrap sample:

- ① Select randomly m variables from the p variables.
Important to obtain **decorrelated** trees
Typically $m = \sqrt{p}$ ($p = 4$ so $m = 2$ for iris)
- ② Split the node into two daughter nodes based on the m variables (the value m is same at each node)
- ③ Continue until no more split is possible

Build a decision tree for each training dataset



What is the importance of each variable ?

PROBLEM

What is the importance of each variable ?

It is not clear in a random forest

which variables are important in predicting
the species a given plant belongs to

What is the importance of each variable ?

SOLUTION

What is the importance of each variable ?

The **importance** of a given predictor

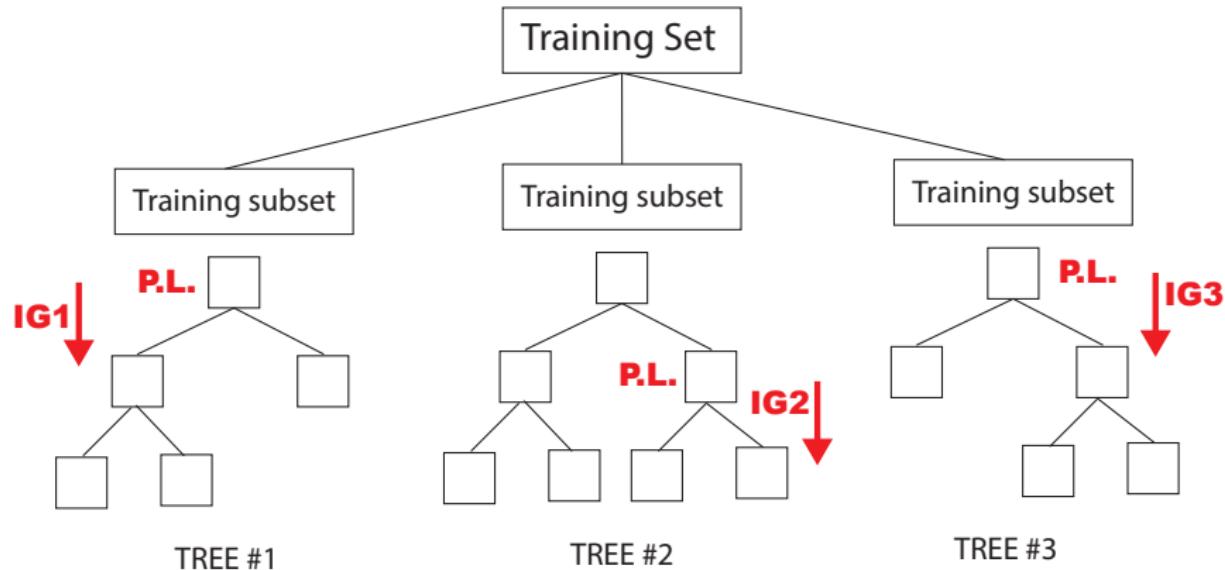
may be computed by adding up

the information gain increases averaged

over all splits in the random forest involving

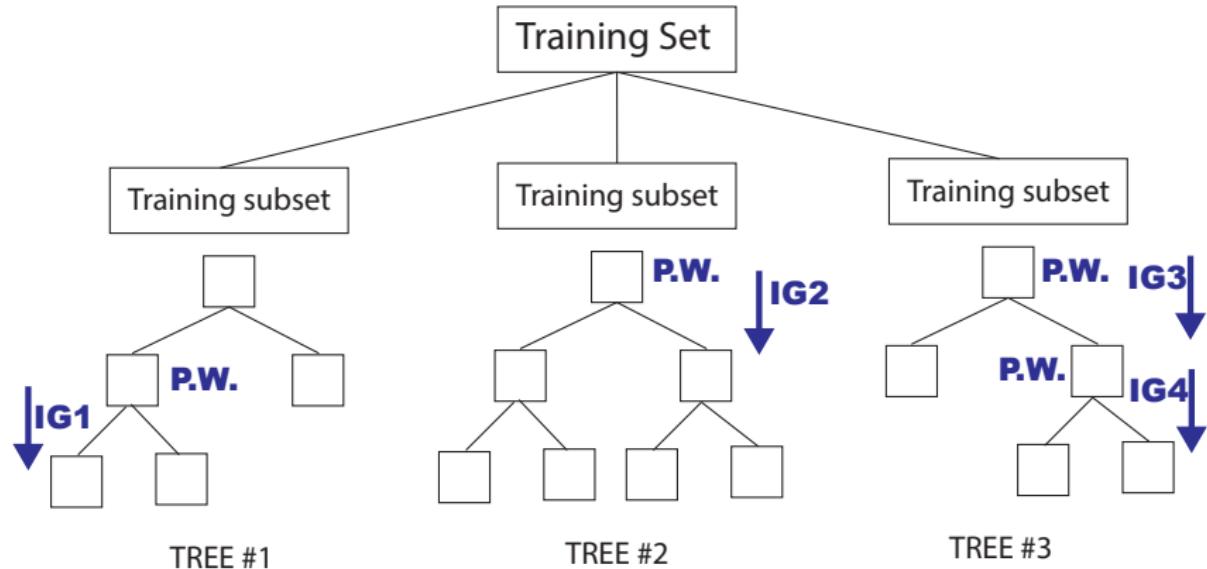
the predictor in question

Build a decision tree for each training dataset



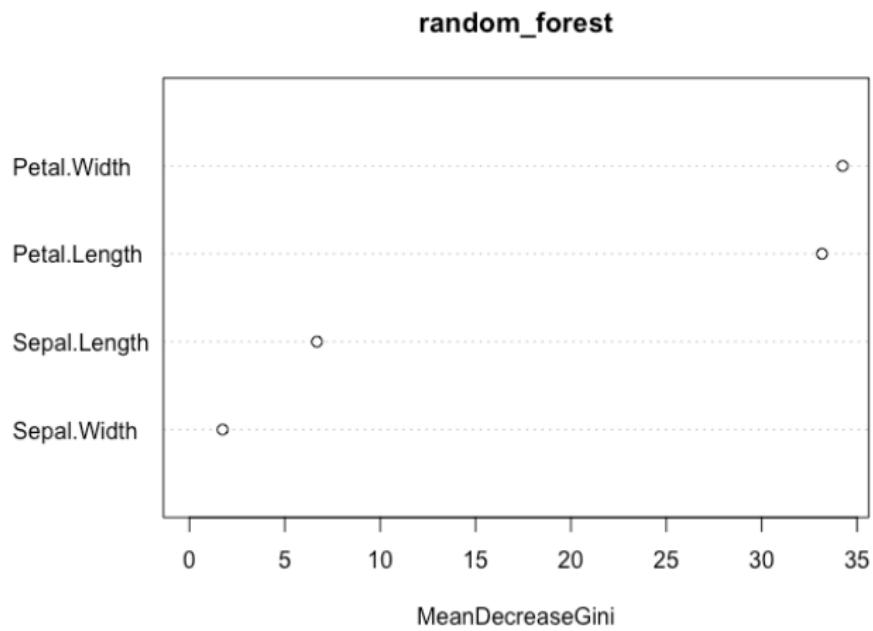
$$\text{Importance} = (\text{IG1} + \text{IG2} + \text{IG3}) / 3$$

Build a decision tree for each training dataset



$$\text{Importance} = (\text{IG1} + \text{IG2} + \text{IG3} + \text{IG4}) / 3$$

What is the importance of each variable ?



Conclusion on random forest

New hyperparameters compared to decision tree:

- Number of trees, size of train subset (with or without replacement), number of variables used at each split

Conclusion on random forest

New hyperparameters compared to decision tree:

- Number of trees, size of train subset (with or without replacement), number of variables used at each split

Main advantage compared to decision tree:

- Robust to training data change

Conclusion on random forest

New hyperparameters compared to decision tree:

- Number of trees, size of train subset (with or without replacement), number of variables used at each split

Main advantage compared to decision tree:

- Robust to training data change

Main disadvantage compared to decision tree:

- Not easy to visualise graphically and to interpret

QUESTIONS ?

Applications

Applications

Application 1

Predict the risk of kidney transplantation rejection

Ref: Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation,
Torgyn Shaikhina, DaveLowe, Sunil Dagade, David Briggs, Robert Higgins, Natasha Khovanov, Biomedical Signal
Processing and Control (2017)

Application 1

Some kidney diseases require transplantation
to save the life of the patient

Application 1

But the kidney may be rejected

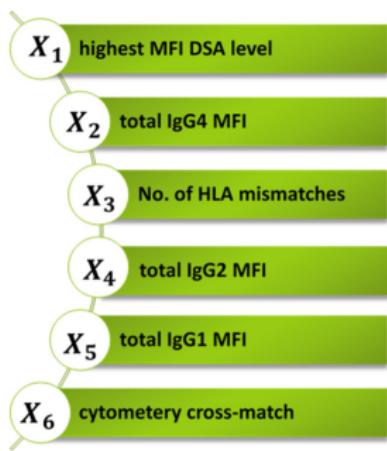
Application 1

What are the risk factors
associated with rejection ?

Application 1

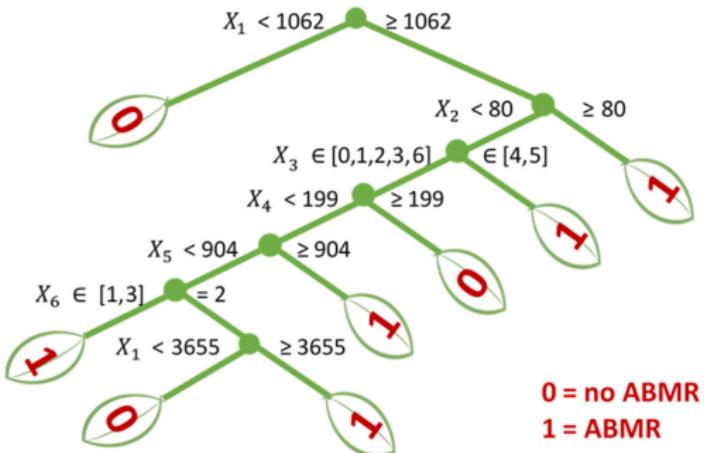
They used 6 predictors in a decision tree
(data: 80 patients)

Application 1



0 = No Kidney Rejection

1 = Kidney Rejection

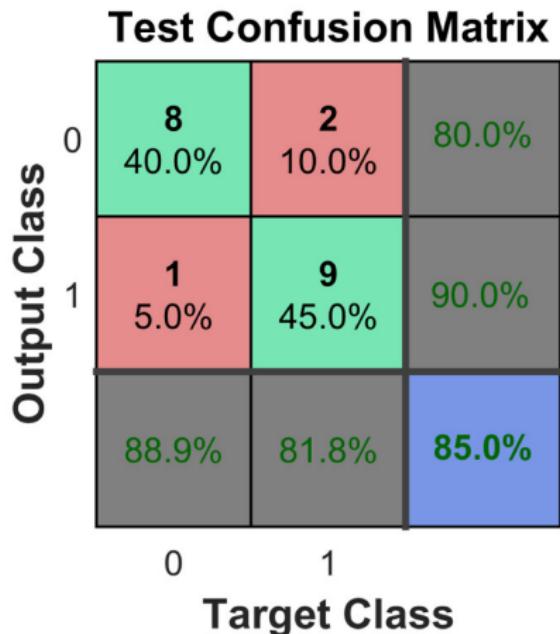
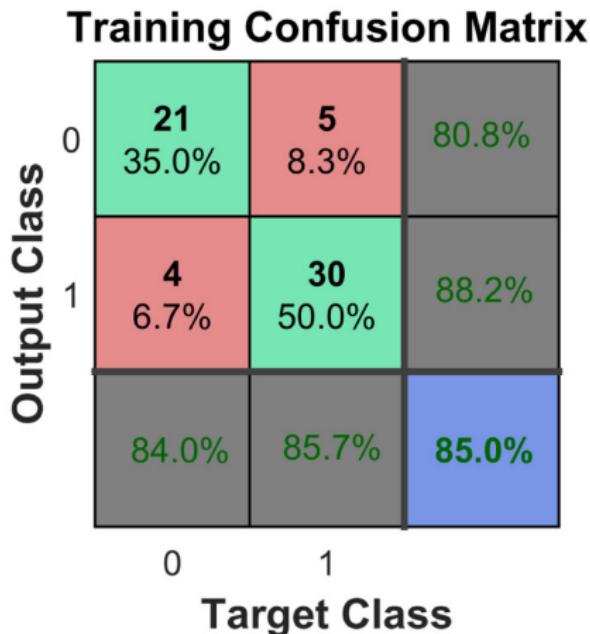


Impurity index: Gini

`minsplit=10, minbucket=1`

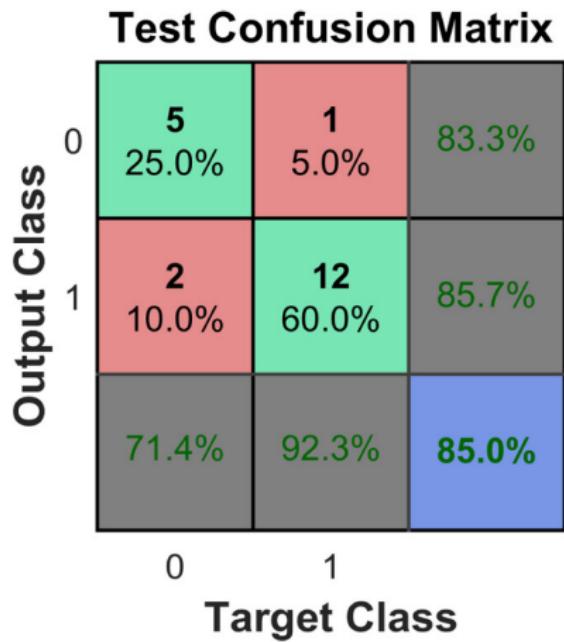
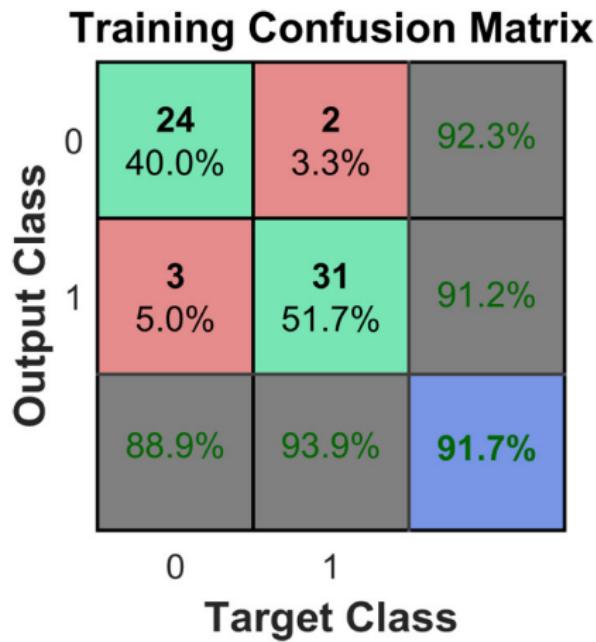
Application 1

Decision Tree:



Application 1

Random Forest (600 trees):



Application 2

Predict the risk of
type 2 diabetes

Ref: Type 2 Diabetes Mellitus Screening and Risk Factors Using Decision Tree: Results of Data Mining, Shafi Habibi, Maryam Ahmadi, and Somayeh Alizadeh, Glob J Health Sci. (2015)

Application 2

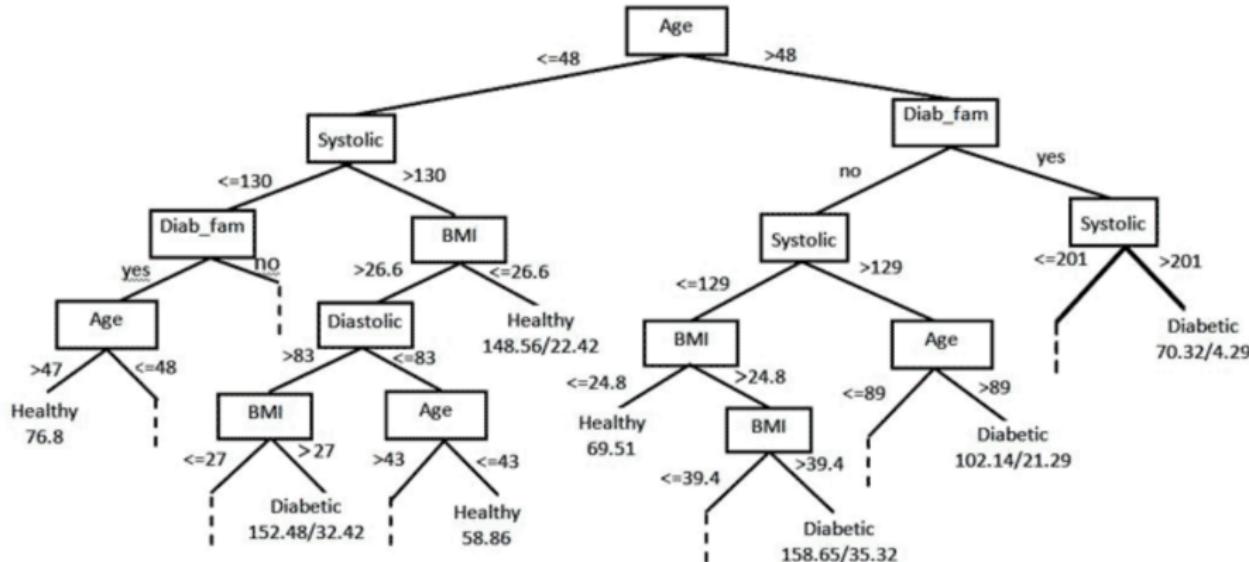
DATA

20'000 patient records

(age, gender, BMI, etc)

are classified as diabetic or healthy

Application 2



Rem: Gender was used but does not appear in the tree

Application 2

PREDICTION

Table 2. Confusion matrix of the decision tree model

	Classes	Diabetic	Healthy
TRUE	Healthy	253	21221
	Diabetic	641	283

Accuracy = 98 %

Application 2

PREDICTION

Table 2. Confusion matrix of the decision tree model

	Classes	Diabetic	Healthy
TRUE	Healthy	253	21221
	Diabetic	641	283

Accuracy = 98 %

You may want to minimize FNR

QUESTIONS ?

BONUS

Bonus 1: The best learning algorithm

Which is
the best learning algorithm ?

Bonus 1: The best learning algorithm

No universal machine learning algorithm:

- No free lunch theorem: In a nutshell, it states that there is no learning algorithm that works best for all problems.

Bonus 1: The best learning algorithm

No universal machine learning algorithm:

- No free lunch theorem: In a nutshell, it states that there is no learning algorithm that works best for all problems.
- As a consequence, one should try several reasonable learning algorithms based on the nature of the problem, type and amount of data, error function, etc.

Bonus 1: The best learning algorithm

No universal machine learning algorithm:

- No free lunch theorem: In a nutshell, it states that there is no learning algorithm that works best for all problems.
- As a consequence, one should try several reasonable learning algorithms based on the nature of the problem, type and amount of data, error function, etc.
- And use the validation set accuracy as a performance measure to select the best one.

Bonus 1: The best learning algorithm

No universal machine learning algorithm:

- No free lunch theorem: In a nutshell, it states that there is no learning algorithm that works best for all problems.
- As a consequence, one should try several reasonable learning algorithms based on the nature of the problem, type and amount of data, error function, etc.
- And use the validation set accuracy as a performance measure to select the best one.

Example: Let us put aside the interpretability and robustness. Then, the accuracy for the iris data:

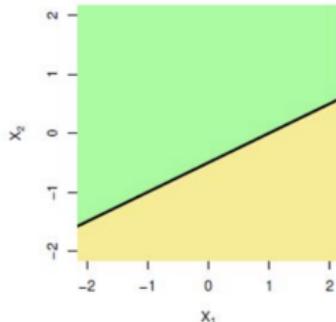
NN= 0.94, Tree= 0.94, RF= 0.91 → Best: NN=Tree

Bonus 2: Decision Tree versus Linear Model

Decision Trees vs. Linear Models – Depends on underlying data structure

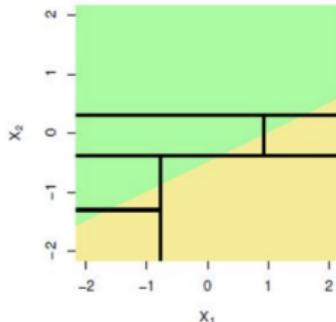
1. True linear boundary:

Linear model fits perfectly



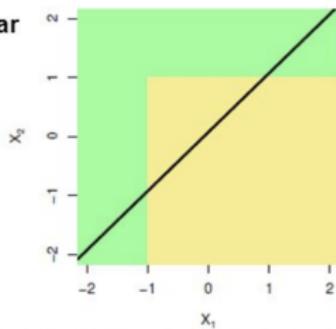
3. True linear boundary:

Flexible decision tree can approximate



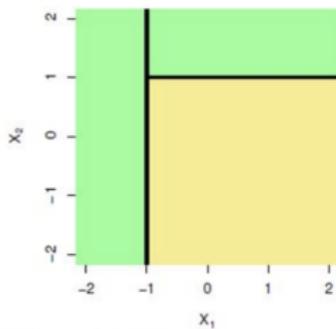
2. True nonlinear boundary:

Linear model fits poorly



4. True nonlinear boundary:

Simple decision tree fits perfectly



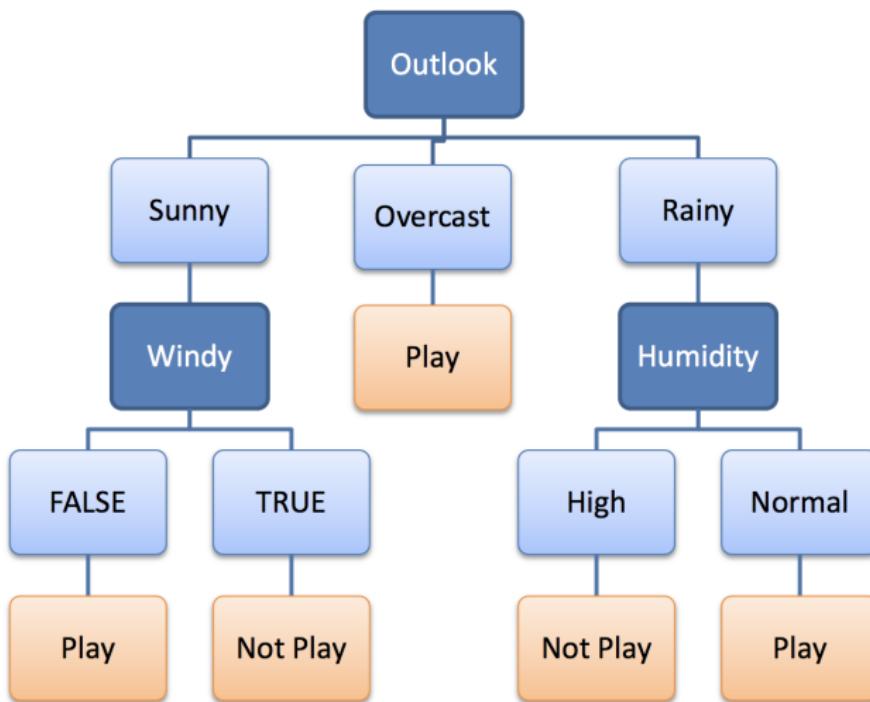
Bonus 3: Decision Tree for Regression

Decision Tree for Classification:

Predictors				Target
Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Bonus 3: Decision Tree for Regression

Decision Tree for Classification:



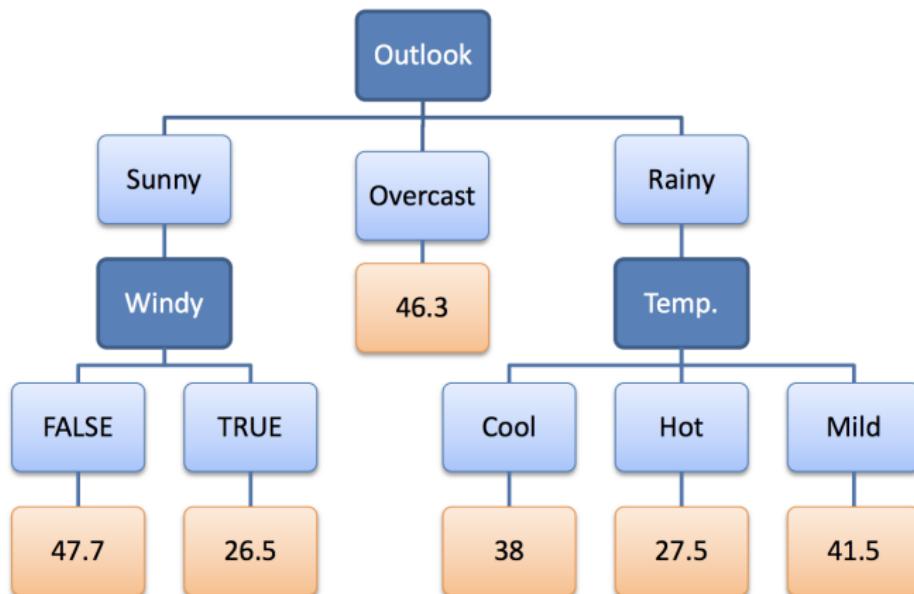
Bonus 3: Decision Tree for Regression

Decision Tree for Regression:

Predictors				Target
Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	25
Rainy	Hot	High	True	30
Overcast	Hot	High	False	46
Sunny	Mild	High	False	45
Sunny	Cool	Normal	False	52
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	35
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	46
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

Bonus 3: Decision Tree for Regression

Decision Tree for Regression:



Bonus 3: Decision Tree for Regression

Two modifications to go
from DT classification to DT regression

Modification 1

Information Gain

becomes

Standard Deviation Reduction

Modification 2

Most commonly occurring class in each leaf

becomes

Mean response of the training output values

Bonus 3: Decision Tree for Regression

① Standard Deviation Reduction:

		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR=0.17		

		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		

		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		

Bonus 3: Decision Tree for Regression

① Standard Deviation Reduction:

		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

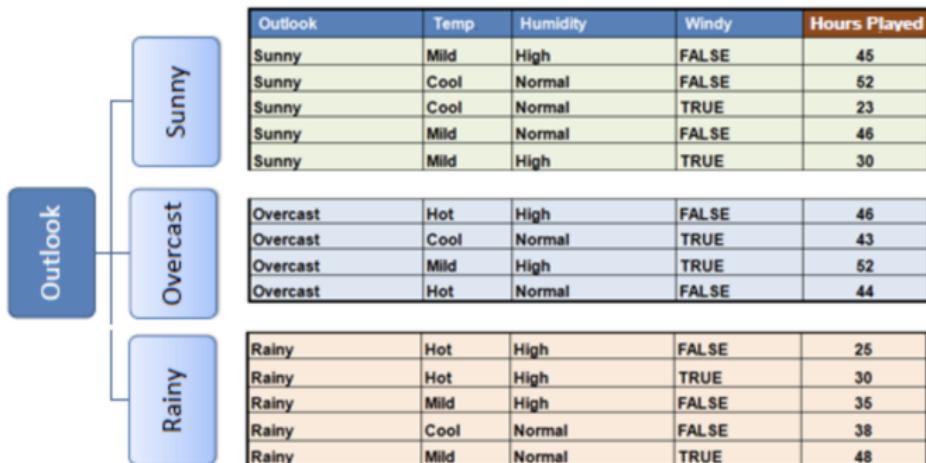
		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR=0.17		

		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		

		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		

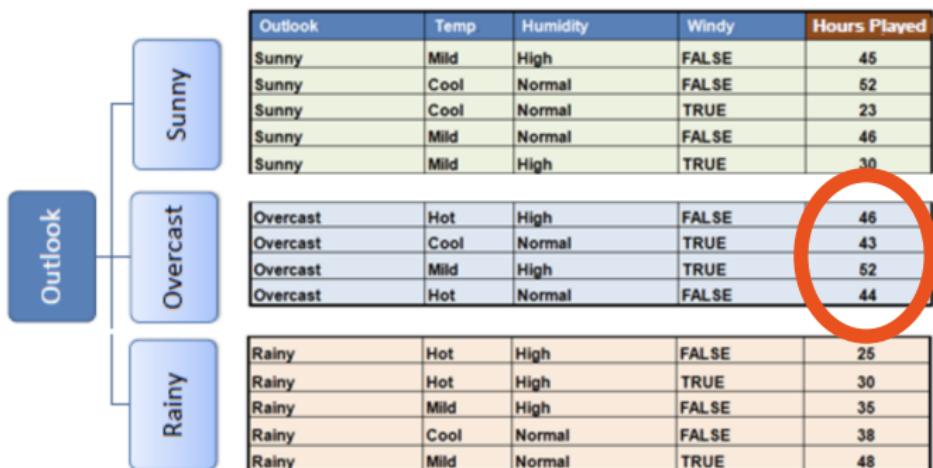
Bonus 3: Decision Tree for Regression

① Standard Deviation Reduction:



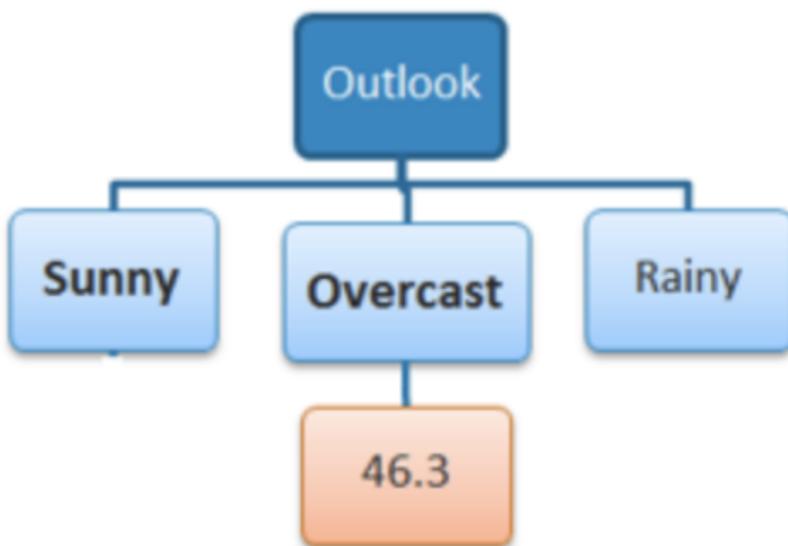
Bonus 3: Decision Tree for Regression

1 Standard Deviation Reduction:



Bonus 3: Decision Tree for Regression

- ② Mean response of the training output values:



Bonus 4: Multicollinearity

Multicollinearity

occurs when two or more predictor variables
are intercorrelated

Bonus 4: Multicollinearity

Is multicollinearity a problem ?

Bonus 4: Multicollinearity

Yes for interpretation

(to know which variables are important)

(or to know the effects of individual predictors)

No really for prediction

(more variables may give better accuracy)

THANK YOU
FOR YOUR ATTENTION