Small data technique

Lecture 15

Changho Suh

January 26, 2024

Random forests (RFs)

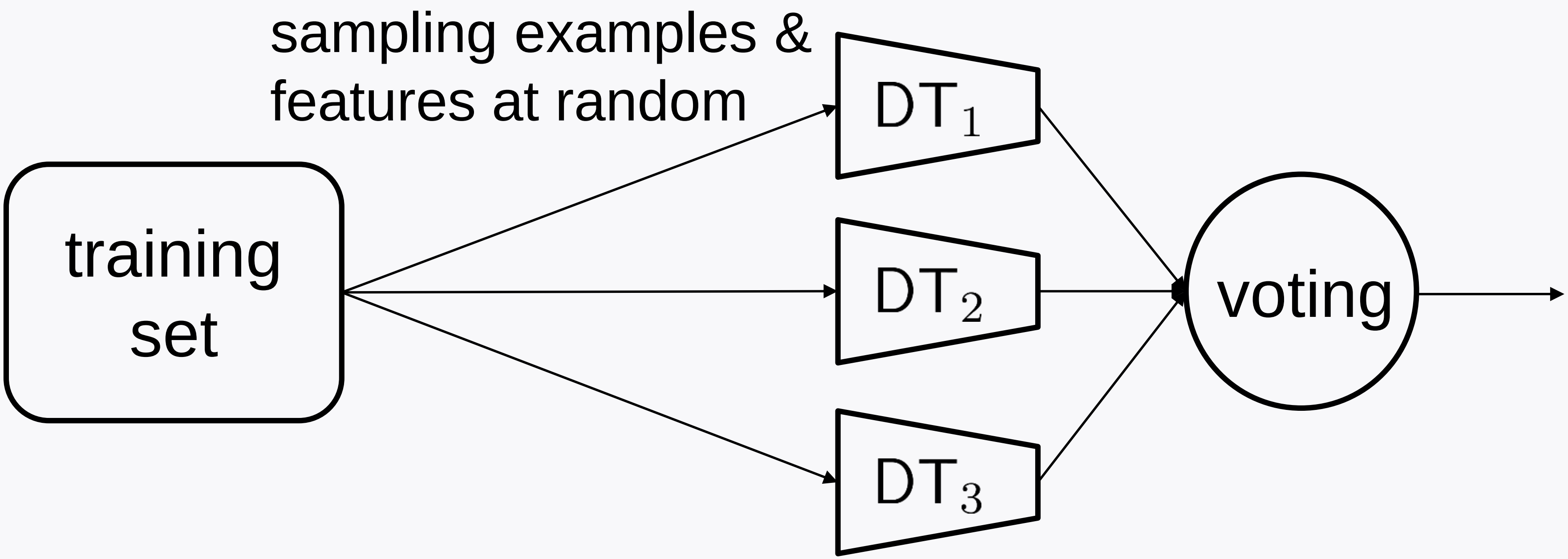
Outline

1. Investigate **hyperparameters.**

2. Study a key measure for model *interpretation*:

Feature Importance

Hyperparameters



Two types:

DT hyperparameters + **additional** hyperparameters

Hyperparameters

DT hyperparameters + **additional** hyperparameters

“max_depth”	“max_features”
“min_samples_split”	“n_estimators”
“min_samples_leaf”	
“max_leaf_nodes”	

Default parameters

DT hyperparameters		+	**additional** hyperparameters	
"max_depth"	none		"max_features"	$\frac{\sqrt{\text{n\_features}}}{\text{n\_features}}$
"min_samples_split"	2		"n_estimators"	100
"min_samples_leaf"	1			
"max_leaf_nodes"	none			

Hyperparameters vs. regularization

DT hyperparameters	+	**additional** hyperparameters	
"max_depth"	↓	"max_features"	↓
"min_samples_split"	↑	"n_estimators"	↑
"min_samples_leaf"	↑		
"max_leaf_nodes"	↓		

→ More regularized.

Hyperparameter search

Scikit-learn provides functions that ease search:

`GridSearchCV`

`RandomizedSearchCV`

Check details in PS.

A measure for model interpretation

RFs have a **measure** that captures **the relative importance of each feature**:

Feature Importance

Can serve model interpretation.

How to compute "feature importance"?

First, for each DT, compute "node importance":

$$\mathsf{NI}_j = G_j - \frac{m_{j,\mathsf{left}}}{m_j} G_{j,\mathsf{left}} - \frac{m_{j,\mathsf{right}}}{m_j} G_{j,\mathsf{right}}$$

→ Quantifies how well node j is split.

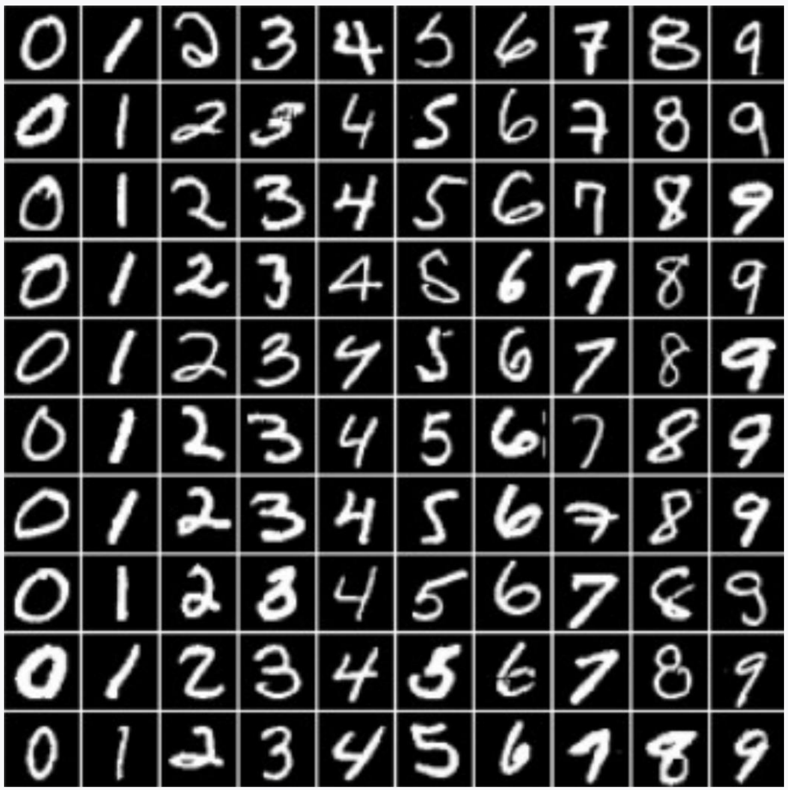
Then compute "feature importance" based on NI_j:

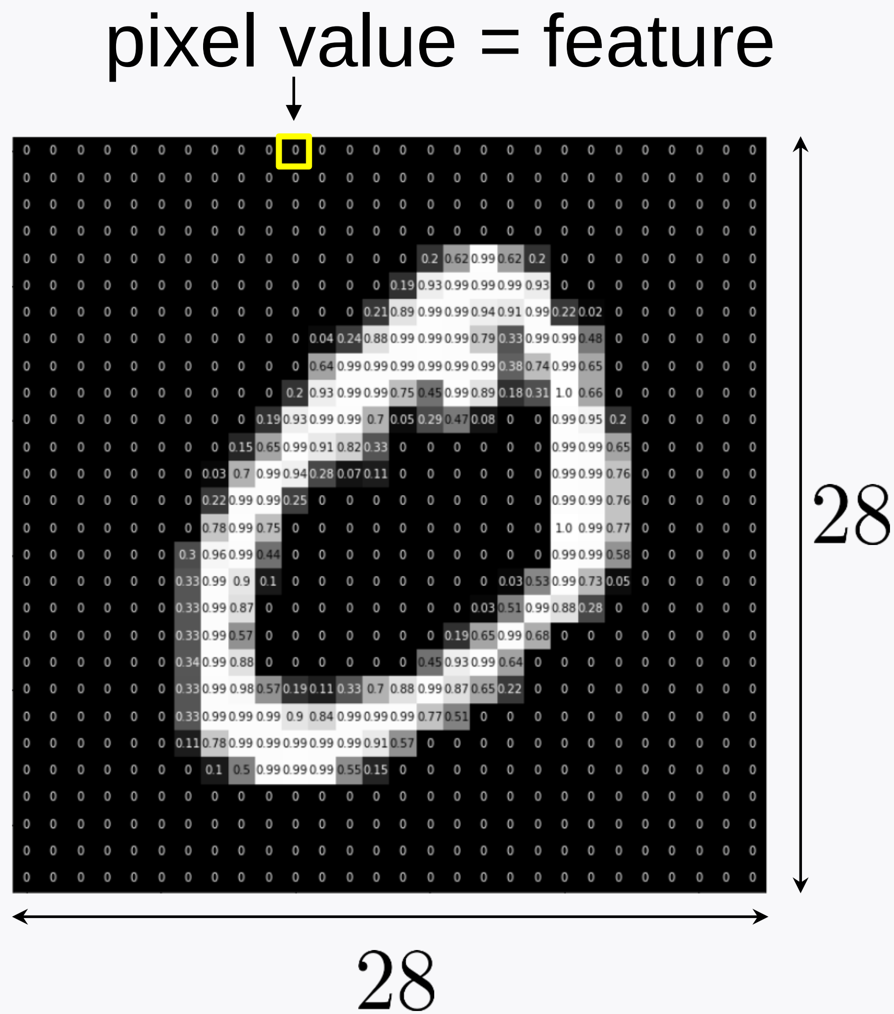
$$\mathsf{FI}_k = \frac{\sum_j \mathsf{NI}_{j,k}}{\sum_j \mathsf{NI}_j}$$

$$\mathsf{NI}_{j,k} = \mathsf{NI}_j \cdot \mathbf{1}\{k = \text{contributer of the node } j \text{ split}\}.$$

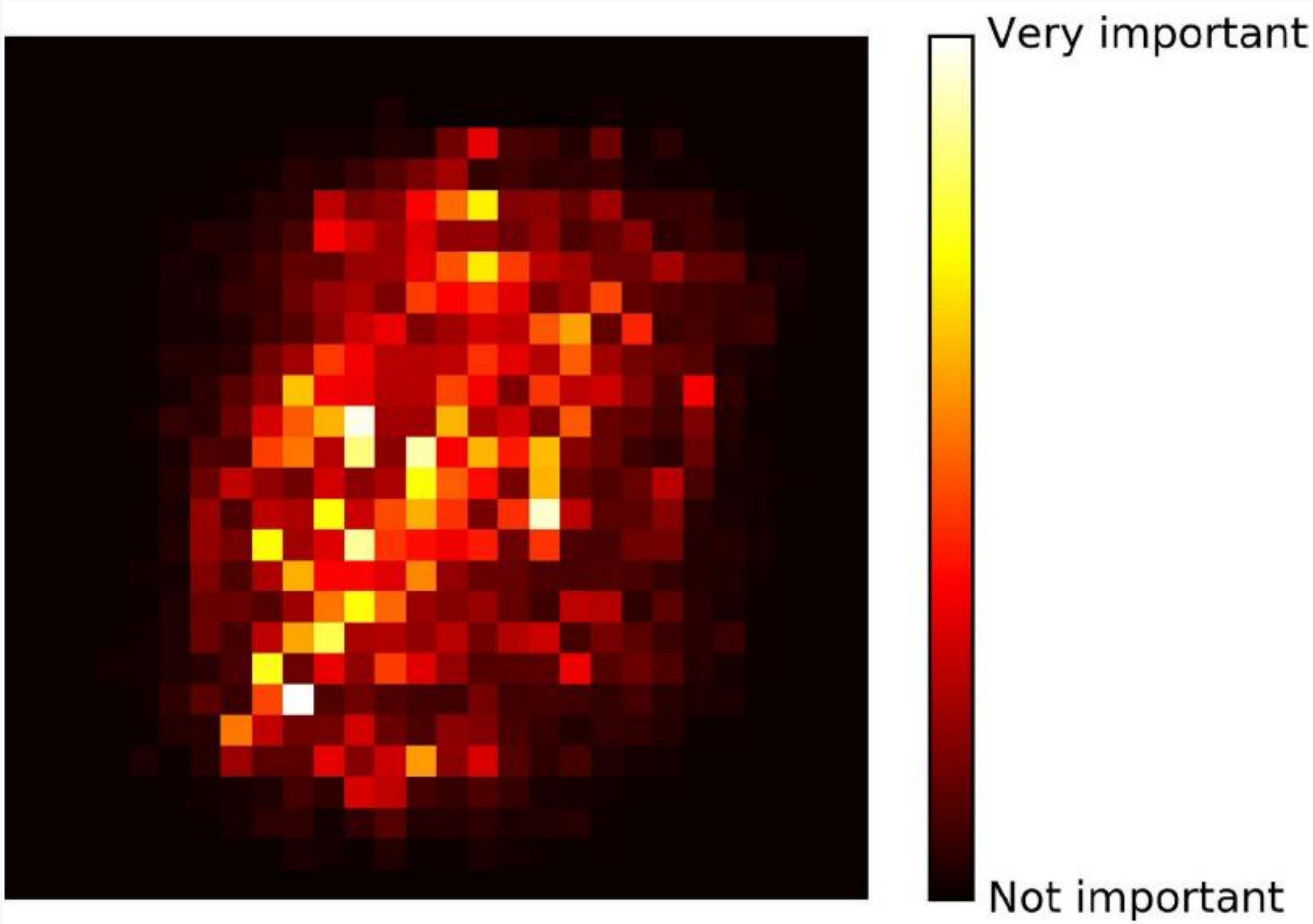
Average over all DTs.

Example: MNIST

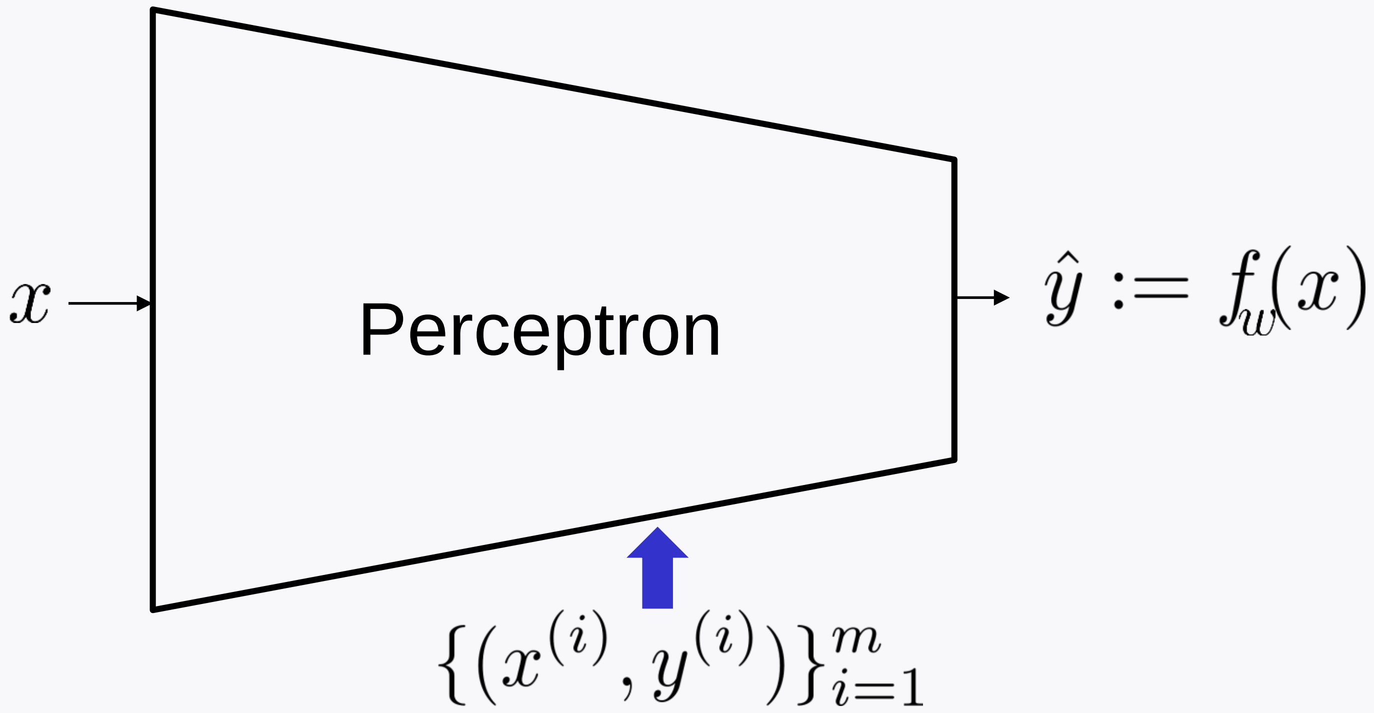




MNIST pixel importance



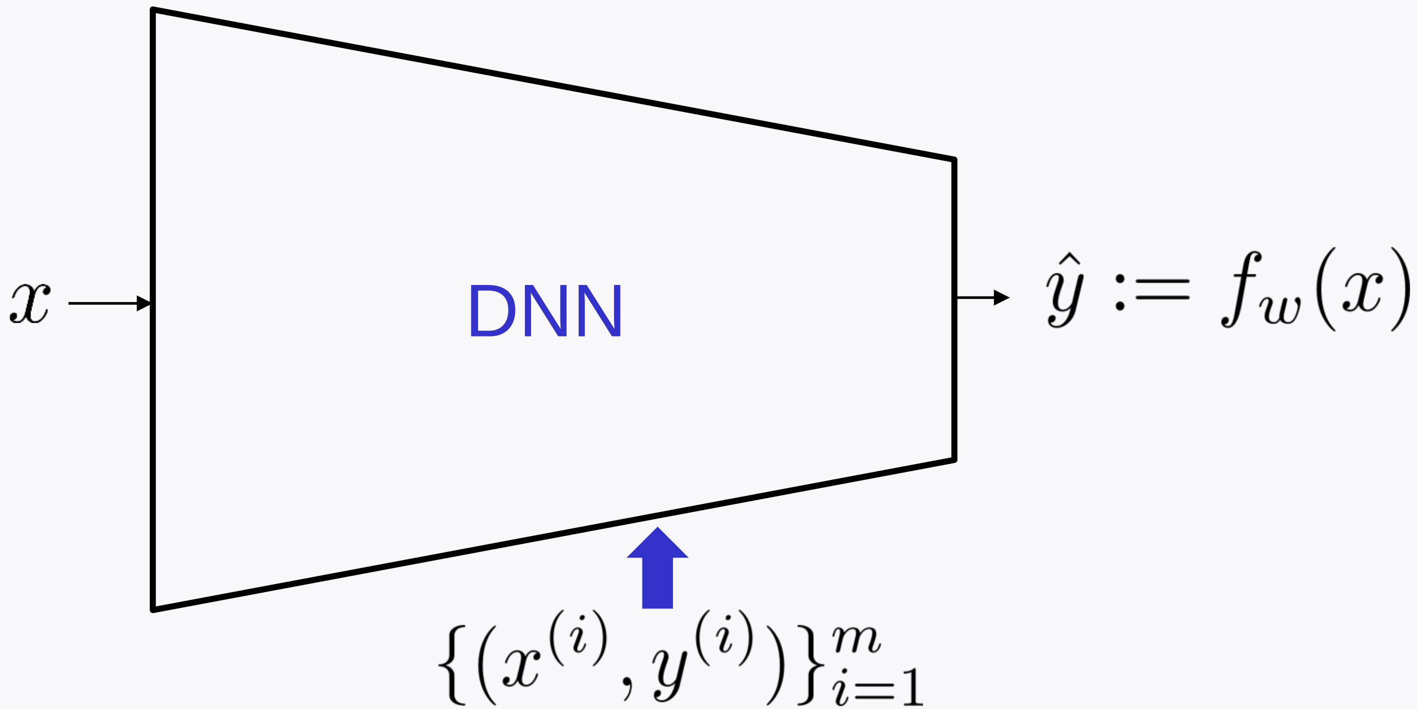
Summary of Day 1 lectures



Linear activation + squared error loss: **LS** classifier

Logistic acti. + cross entropy loss: **Logistic regression**

Summary of Day 1 lectures



ReLU (@hidden); Logistic (@output); Cross entropy loss

Algorithm: Gradient descent

Efficient method: backprop

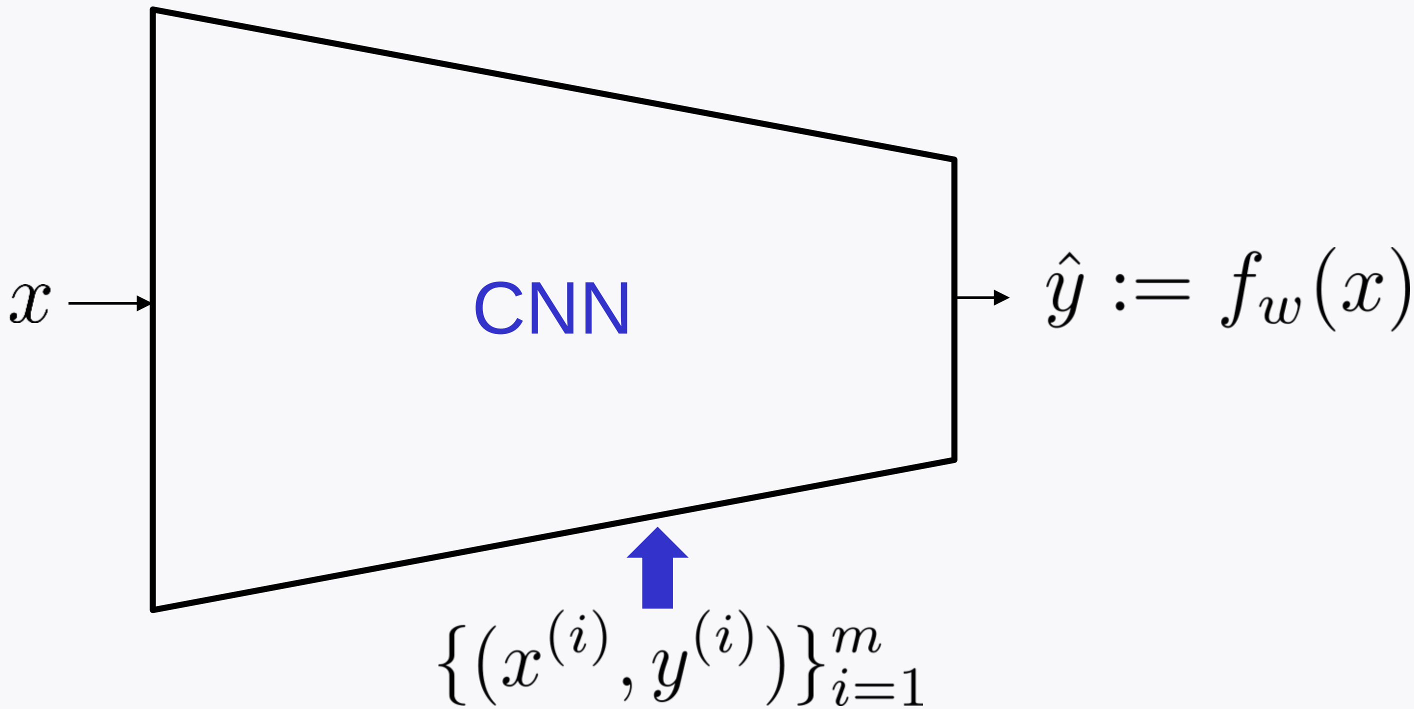
Practical variant: Adam optimizer

Summary of Day 2 lectures

Advanced techniques:

1. Data organization
2. Generalization techniques
3. Weight initialization
4. Techniques for training stability
5. Hyperparameter search
6. Cross validation

Summary of Day 3 lectures

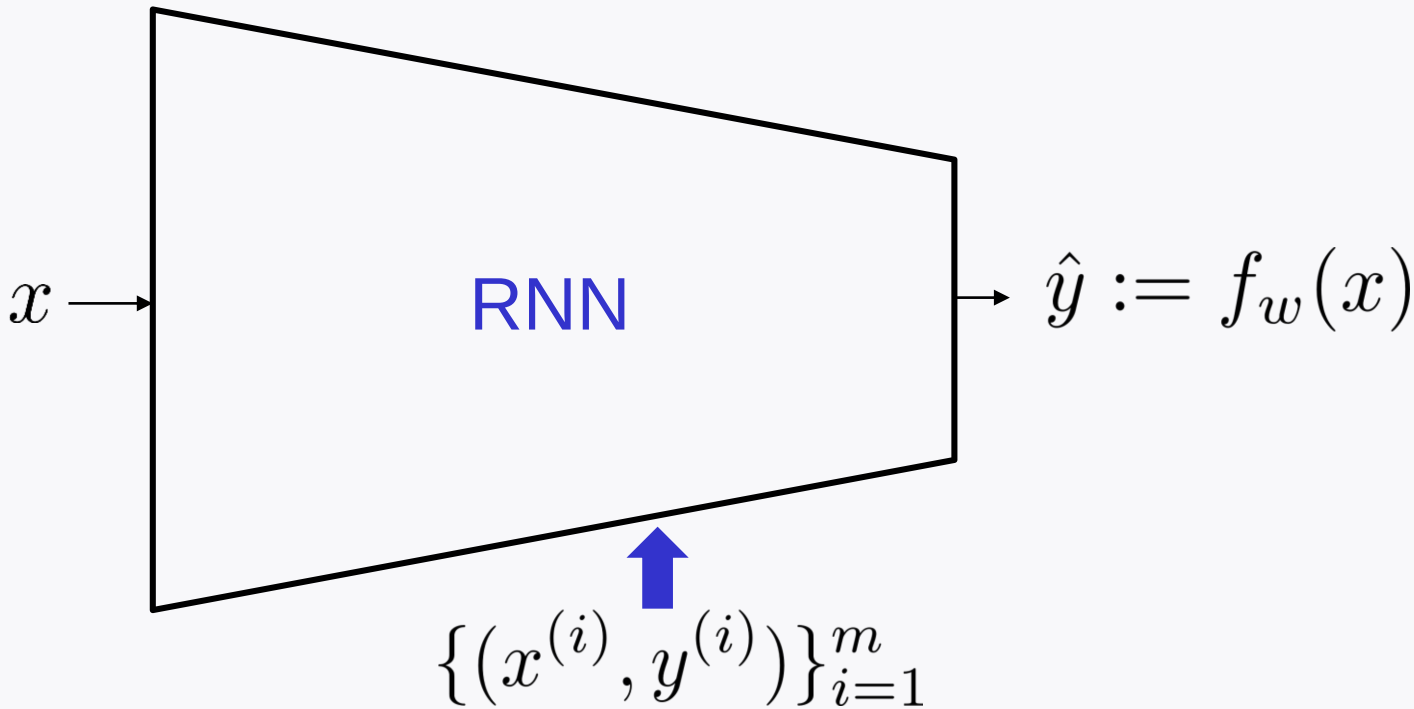


Two building blocks: Conv layer & Pooling layer

Design principles: As a network is deeper,

1. Feature map sizes gets smaller.
2. # of feature maps gets bigger.

Summary of Day 4 lectures



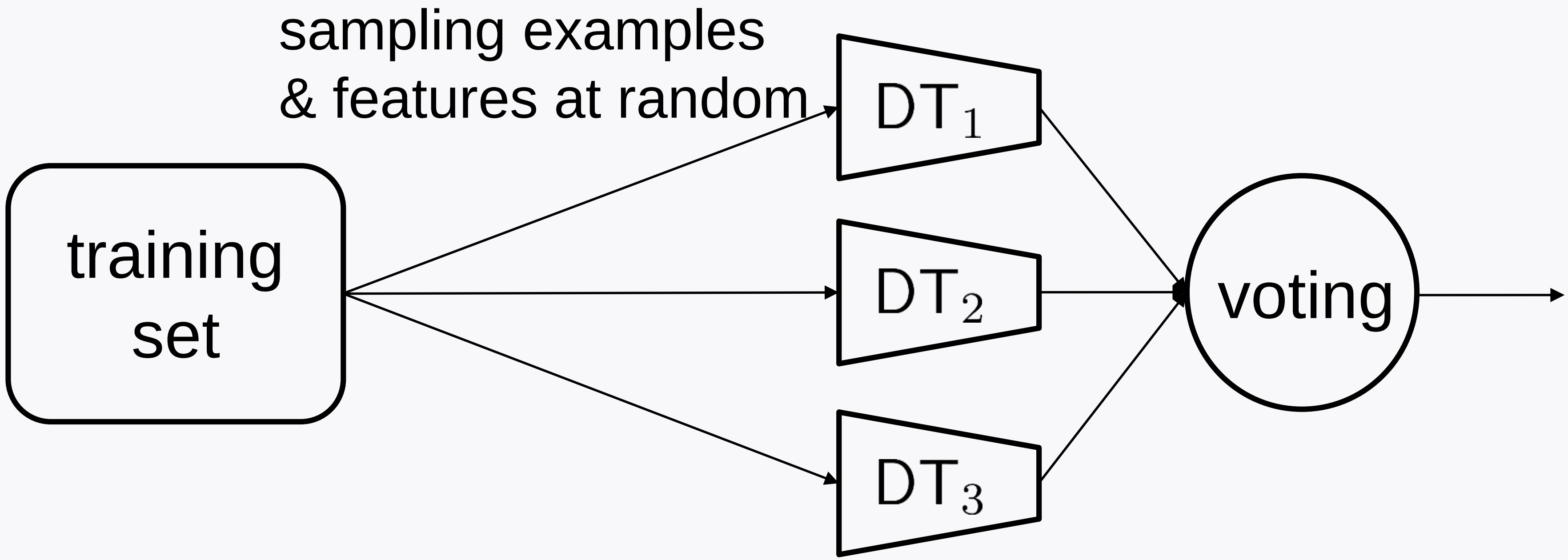
Two building blocks: Recurrent neurons & memory cell

Basic RNNs: Trained via truncated BTTP;
Memory fades quickly.

LSTM: Offers great performance and fast training.

Summary of today's lectures

RF: An ensemble of DTs, each trained on the random subspace method



A key hyperparameter: **"max_features"**

A measure for *interpretation*: **Feature importance**

Many remaining issues

What if labels are not available? $\{(x^{(i)}, \cancel{y^{(i)}})\}_{i=1}^{m}$

Unsupervised learning:

Clustering, anomaly detection

Principal component analysis (PCA), autoencoder

Generative Adversarial Networks (GANs)

Many remaining issues

Advanced small data techniques:

Semi-supervised learning

Transfer learning

Simulator-based learning