Machine Learning

Dr. Thyagaraju G S

Dr.Thyagaraju G S, Professor and HoD, Department of CSE, SDM Institute Of Technology,Ujire-574240.
Source Book : S. Sridhar, M Vijayalakshmi “Machine Learning”. Oxford, 2021

Module2: Syllabus

Understanding Data – 2: Bivariate Data and Multivariate Data, Multivariate Statistics, Essential Mathematics for Multivariate Data, Feature Engineering and Dimensionality Reduction Techniques.

Basic Learning Theory: Design of Learning System, Introduction to Concept of Learning, Modelling in Machine Learning.

Chapter-2 (2.6-2.8, 2.10), Chapter-3 (3.3, 3.4, 3.6)

Module 2.2: Basic Learning Theory

1. Design of Learning System,
2. Introduction to Concept of Learning,
3. Modelling in Machine Learning.

1.What is Learning?

- **Learning** is a process of acquiring knowledge and construct new ideas or concepts based on the **experiences, study and training**.
- ML is way of learning **general concept (hypothesis/formulae/pattern /insight)** from training examples without writing a program.
- A hypothesis is a **statement or proposition that is formulated to explain a set of facts or phenomena**.

Learning types

1. Learn by Memorization
2. Learn by examples (**Experiences**)
3. Learn by being taught (**Passive or Active Learning**)
4. Learning by critical thinking (**Deductive Learning**)
5. Self Learning (**Reinforcement Learning/Self Directed learning**)
6. Learning by solving Problems(**Cognitive Learning**)
7. Learning by generalizing **explanations(Also called as explanation based learning**)

3.Design of Learning system

A system that is built around a **learning algorithm** is called a learning system.

The design of systems focuses on these **4 steps** :

1. **Choosing a training experience**
   - Sample input and output
   - Training Data Set/Supervisor
2. **Choosing the target Function**
   - Type of knowledge required (like legal moves /Move with largest score in chess)

3. **Representation of target function**

- **Table/Collection or Rules or a neural network or a linear combination** of features like $\mathbf{V = w_0 + w_1x_1 + w_2x_2 + w_3x_3}$

4. **Choosing and Approximate Function**

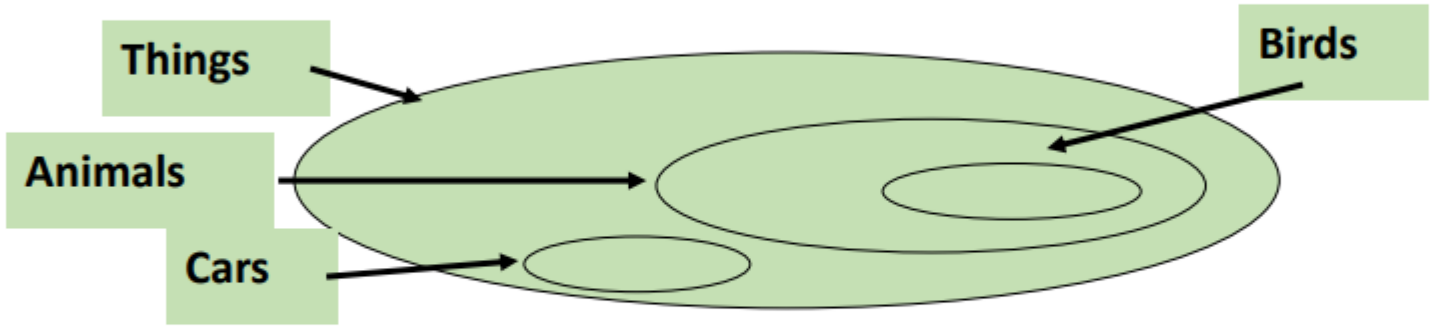
- The focus is to choose weights and fit the given training samples effectively. The aim is to reduce the error given as :
- $\mathbf{E = \sum[V_{train}(b) - V\hat{}(b)]^2}$

Components of Learning systems

1. **Performance** system to allow the game to play against itself
2. A **critic System** to generate the **samples**
3. A **generalizer system** to generate a **hypothesis** based on samples.
4. An **experimenter system** to generate a **new system** based on the currently learnt function.

What is Concept?

- A **Concept** is a subset of objects or events defined over a larger set [Example: The concept of a *bird* is the subset of all objects (i.e., the set of all *things* or all *animals*) that belong to the category of bird.]



- Alternatively, a concept is a boolean-valued function defined over this larger set [**Example:** a function defined over all *animals* whose value is *true* for birds and *false* for every other animal].

What is Concept learning ?

- Concept learning is a learning strategy of acquiring **abstract knowledge or inferring a general concept or deriving a category** from the given training samples.
- It is a process of **abstraction and generalization** from the data and **helps to classify** an object that has set of common and relevant features.
- It is a way of learning categories for **object and to recognize** new instances of those categories .

Concept learning requires 3 things

1. **Input** : Labelled Training Data Set (Set of instances and concept /category)
2. **Output**: Target Concept of Target Function f(x) which maps from input x to output y.
3. **Test –** New instances to test the learning model

Example: Sample Training Instances

Sl.NO	Horns	Tail	Tusks	Paws	Fur	Color	Hooves	Size	Elephant
1	No	Short	Yes	No	No	Black	No	Big	Yes
2	Yes	Short	No	No	No	Brown	Yes	Medium	No
3	No	Short	Yes	No	No	Black	No	Medium	Yes
4	No	Long	No	Yes	Yes	White	No	Medium	No
5	No	Short	Yes	Yes	Yes	Black	No	Big	Yes

Independent attributes: Horns, Tail, Tusks, Paws, Fur, Color, Hooves and Size

Dependent attributes: Elephant

Target Concept is to identify the animal is Elephant

4.2.Representation of hypothesis

- A hypothesis '**h**' approximates a target function '**f**'.
- Each hypothesis is represented as conjunction of attribute conditions Example : **(Tail = Short) ∧ (Color = Black)----**
- The set of hypothesis **(h)** is called hypotheses **(H)**.
- Each attribute of hypothesis can take a value as either '?' or 'ϕ' or can hold a single value
  - "?" denotes that the attribute can take any value [eg: Color = ?]
  - "ϕ" denotes that the attribute cannot take any value,i.e it represents a null value [eg: Horns = ϕ]
  - Single value denotes a specific single value from acceptable values of the attributes i.e. the attribute 'Tail' can take a value a short[eg. Tail = short]

Example

	Horns	Tail	Tusks	Paws	Fur	Color	Hooves	Size
h =	**<No**	**?**	**Yes**	**?**	**?**	**Black**	**No**	**Medium**

The training data set given above has 5 training instances with 8 independent attributes and one dependent attribute. Here the different hypotheses that can be predicted for the target concept are:

	Horns	Tail	Tusks	Paws	Fur	Color	Hooves	Size
h =	**<No**	**?**	**Yes**	**?**	**?**	**Black**	**No**	**Medium>**
				OR				
h=	**<No**	**?**	**Yes**	**?**	**?**	**Black**	**No**	**Big>**

Note : Most General and Most Specific Hypothesis

- **<?,?,?,?,?,?,?,?>** represents the **most general hypothesis** which **allows any value** to the attribute. Indicates **any animal** can be a elephant
- **< ϕ, ϕ, ϕ, ϕ, ϕ, ϕ, ϕ, ϕ>** represents the most specific hypothesis and will **not allow** any value for each of the attribute. This hypothesis indicates that **no animal can be an elephant**.

Concept learning task of an elephant

Sl.NO	Horns	Tail	Tusks	Paws	Fur	Color	Hooves	Size	Elephant
1	No	Short	Yes	No	No	Black	No	Big	**Yes**
2	Yes	Short	No	No	No	Brown	Yes	Medium	**No**
3	No	Short	Yes	No	No	Black	No	Medium	**Yes**
4	No	Long	No	Yes	Yes	White	No	Medium	**No**
5	No	Short	Yes	Yes	Yes	Black	No	Big	**Yes**

Input : 5 Instances with attributes

Target Concept/function ‘c’: Elephant --> {Yes, No}

Hypotheses H: Set of hypothesis each with conjunctions of literals as propositions.

- The hypothesis ‘h’ for the concept learning task of an Elephant is given as :
- **h =<No Short Yes ? ? Black No ?>**
-
- This hypothesis h is expressed in **propositional logic** form as below:
- **(Horns = No) ∧ (Tail = short) ∧ (Tusks = Yes) (Paws = ?) ∧ (Fur = ?) ∧ (Color = Black) ∧ (Hooves = No) ∧ (Size = ?)**

- **Output : Learn hypothesis ‘h’ to predict an ‘Elephant ‘ such that for a given test instance x , h(x) = c(x)**

Note : Concept Learning can also be called as inductive learning that tries to induce a general function from specific training instances.

4.2 Hypothesis Space

- Is the **set of all hypothesis** that approximates the target **function f**.
- The **subset of hypothesis space** that is consistent with all observed training instances is called as **Version Space**.
- Version Space represents the only hypotheses that are used for the **classification**.

4.2 Hypothesis Space

Sl.NO	Horns	Tail	Tusks	Paws	Fur	Color	Hooves	Size	Elephant
1	No	Short	Yes	No	No	Black	No	Big	Yes
2	Yes	Short	No	No	No	Brown	Yes	Medium	No
3	No	Short	Yes	No	No	Black	No	Medium	Yes
4	No	Long	No	Yes	Yes	White	No	Medium	No
5	No	Short	Yes	Yes	Yes	Black	No	Big	Yes

Horns – Yes, No (2)
Tail - Long, short (2)
Tusks – Yes, No (2)
Paws - Yes, No (2)
Fur - Yes, No (2)
Color - Brown, Black, White (3)
Hooves - Yes, No (2)
Size - Medium, Big (2)

Considering these values for each of the attribute there are

(2X2X2X2X2X3X2X2) = 384 distinct instances covering all the 5 instances in the training data set.

So we can generate **(4X4X4X4X4X5X4X4) = 81,920** distinct hypotheses when including two more values [**?**, **ϕ]**

What is **heuristic** Space Search?

- Heuristic space search refers to a problem-solving approach in artificial intelligence and computer science where an algorithm explores a search space using **heuristic information to efficiently find a solution**. A search space is the set of all possible states or configurations that a system can be in, and the goal is to navigate through this space to reach a desirable state or solution.
- Finds a **Solution/hypothesis** to a problem using heuristic functions
- Example : A* Search, Greedy Best FirstSearch, Hill climbing, genetic algorithms ,etc.

Generalization and Specialization

1. **Generalization – Specific to General learning**
2. **Specialization – General to specific Learning**

Find S Algorithm

CGPA	Interactiveness	Practical Knowledge	Communication skills	Logical Thinking	Interest	Job Offer
>=9	Yes	Excellent	Good	Fast	Yes	Yes
>=9	Yes	Good	Good	Fast	Yes	Yes
>=8	No	Good	Good	Fast	No	No
>=9	Yes	Good	Good	Slow	No	Yes

Input: Positive instances in the training data set

Output: Hypothesis '**h**'

S1: Initialize '**h**' to the most specific hypothesis

h = < ϕ ϕ ϕ ϕ ϕ ϕ ϕ ϕ .....>

S2: Generalize the initial hypothesis for the first positive instance

S3: For each subsequent instances:

If it is positive instance :

Check for each attribute value in the instance with the hypothesis **'h'**

If the attribute value is the same as the hypothesis value, then do nothing,

Else if the attribute value is different than the hypothesis value, change it to **'?' in 'h'**

Else if it is a negative instance,

Ignore it

Example

CGPA	Interactiveness	Practical Knowledge	Communication skills	Logical Thinking	Interest	Job Offer
>=9	Yes	Excellent	Good	Fast	Yes	Yes
>=9	Yes	Good	Good	Fast	Yes	Yes
>=8	No	Good	Good	Fast	No	No
>=9	Yes	Good	Good	Slow	No	Yes

CGPA	Interactiveness	Practical Knowledge	Communication skills	Logical Thinking	Interest	Job Offer
>=9	Yes	Excellent	Good	Fast	Yes	Yes
>=9	Yes	Good	Good	Fast	Yes	Yes
>=8	No	Good	Good	Fast	No	No
>=9	Yes	Good	Good	Slow	No	Yes

Solution:

Step1:

Initialize **'h'** to the most specific hypothesis. There are 6 attributes, so for each attribute we initially fill ' ϕ' in the initial hypothesis 'h'

h = < ϕ ϕ ϕ ϕ ϕ ϕ >

Step2:

Generalize the initial hypothesis for the first positive instance. **I1** is a positive instance so generalize the most specific hypothesis **h** to include this positive instance. Hence,

h =	**< ϕ**	**ϕ**	**ϕ**	**ϕ**	**ϕ**	**ϕ >**	
I1:	>=9	Yes	Excellent	Good	Fast	Yes	**Yes (Positive Instance)**
h1=	>=9	Yes	Excellent	Good	Fast	Yes	

CGPA	Interactiveness	Practical Knowledge	Communication skills	Logical Thinking	Interest	Job Offer
>=9	Yes	Excellent	Good	Fast	Yes	Yes
>=9	Yes	Good	Good	Fast	Yes	Yes
>=8	No	Good	Good	Fast	No	No
>=9	Yes	Good	Good	Slow	No	Yes

Step3:

Scan the next instance **I2** since **I2** is positive instance. Generalize **h** to include positive instance **I2**.For each of the non matching attribute value in **'h' put a '?'** to include this positive instance. The third attribute value is mismatching in **h** with **I2 so put a ?**.

h1	**>=9**	**Yes**	**Excellent**	**Good**	**Fast**	**Yes**	
I2	>=9	Yes	Good	Good	Fast	Yes	Yes (Positive Instance)
h2	**>=9**	**Yes**	**?**	**Good**	**Fast**	**Yes**	

Now scan I3 . Since it is a **negative instance ignore it**. Hence the hypothesis remains the same without any change after scanning I3

h2	**>=9**	**Yes**	**?**	**Good**	**Fast**	**Yes**	
I3	>=8	No	Good	Good	Fast	No	No(Negative Instance)
h3	**>=9**	**Yes**	**?**	**Good**	**Fast**	**Yes**	

CGPA	Interactiveness	Practical Knowledge	Communication skills	Logical Thinking	Interest	Job Offer
>=9	Yes	Excellent	Good	Fast	Yes	Yes
>=9	Yes	Good	Good	Fast	Yes	Yes
>=8	No	Good	Good	Fast	No	No
>=9	Yes	Good	Good	Slow	No	Yes

Now scan I4 since it is positive instance, check for mismatch in the hypothesis h with I4. The 5th and 6th attribute value are mismatching, so add? To those attribute in 'h'.

h3	**>=9**	**Yes**	**?**	**Good**	**Fast**	**Yes**	
I4:	>=9	Yes	Good	Good	Slow	No	Yes (Positive Instance)
h4	**>=9**	**Yes**	**?**	**Good**	**?**	**?**	

Thus the final hypothesis generated with find **S** Algorithm is :

h = (>=9 Yes ? Good ? ?)

It includes all **positive instances and obviously ignores any negative instance**.

Limitations of Find S Algorithm

1. Algorithm is **consistent with positive** instances and ignores negative instances.
2. Algorithm Finds only **one unique hypothesis**, wherein there may be many other hypotheses that are consistent with the training dataset.
3. **Erroneous data set can mislead the algorithm** in determining the consistent hypothesis since it ignores negative instances.

Version Space

- The version space is the set of all hypotheses that are consistent with the observed training examples/training data sets.

A hypothesis h is **consistent** with a set of training examples D of target concept c if and only if $h(x)=c(x)$ for each training example in D.

$$Consistent(h, D) \equiv (\forall < x, c(x) > \in D)\, h(x) = c(x)$$

The **version space**, $VS_{H,D}$, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with all training examples in D.

$$VS_{H,D} \equiv \{h \in H \mid Consistent(h, D)\}$$

List then Eliminate Algorithm

Input: Version Space – a list of all hypothesis

Output: Set of consistent hypotheses

1. Initialize the version space with a list of hypotheses
2. For each training instance
    a. **Remove from version space** any hypothesis that is inconsistent

Version Spaces and the Candidate Elimination Algorithm

- Version space learning aims to **generate all hypotheses that are consistent with the given data.** The Candidate Elimination Algorithm constructs the version space by applying two key learning strategies:
- **Specific-to-General Learning**: Expanding the specific boundary (S) to include positive examples.
- **General-to-Specific Learning**: Refining the general boundary (G) to exclude negative examples.

Version Spaces and the Candidate Elimination Algorithm

- The algorithm establishes two boundaries:
- **General Boundary**: Represents the most general hypotheses that remain consistent with the training data.
- **Specific Boundary**: Represents the most specific hypotheses that are still valid given the observed data.

Refining Hypotheses in Candidate Elimination

- **Generating Positive Hypothesis (S):** When encountering a positive example, refine S by generalizing it to include the instance. Initially, S is filled with the first positive instance's attribute values. For subsequent instances, mismatched values are replaced with **'?'** to maintain generalization.
- **Generating Negative Hypothesis (G):** For negative instances, refine **G** by specializing it to exclude the instance. Prune inconsistent hypotheses and update only mismatched attribute values to prevent misclassification. If values match, no update is needed.
- **Generating Version Space**: The version space consists of hypotheses that are consistent with both S and **G**. The final hypothesis set includes only those matching fields in S, ensuring consistency.

Candidate Elimination algorithm

- **Input: Set of instances in the Training dataset**
- **Output: Hypothesis G and S**

1. Initialize G to the **maximally general hypotheses**
2. Initialize S to the **maximally specific hypotheses**
    1. Generalize the initial hypothesis for the positive instance

3. **For each subsequent new training instance**
    1. **If the instance is positive**
        - Generalize **S** to include positive (+ve) instance
            - **Check the attribute value of the positive instance and S,**
                - **If the attribute value of positive instance and S are different fill that field value with ?**
                - **If the attribute value of Positive instance and S are same, then do no change.**
        - Prune **G** to exclude all **inconsistent hypotheses in G with the positive instance**.

2. **If the instance is negative**
   - Specialize **G** to exclude the negative instance,
     - Add to **G** all minimal specializations to exclude the negative example and be consistent with **S**
       - **If the attribute value of S and the negative instance are different, then fill that attribute value with S value**
       - **If the attribute value of S and negative instance are same, no need to update 'G' and fill the attribute value with '?'**
   - Remove from **S** all inconsistent hypothesis with the negative instance.

Instance	CGPA	Interactiveness	Practical Knowledge	Communication skills	Logical Thinking	Interest	Job Offer
I1	>=9	Yes	Excellent	Good	Fast	Yes	Yes
I2	>=9	Yes	Good	Good	Fast	Yes	Yes
I3	>=8	No	Good	Good	Fast	No	No
I4	>=9	Yes	Good	Good	Slow	No	Yes

Step 1: Initialize Go and So to maximally general hypothesis and maximally specific hypothesis as illustrated below:

- **Go = {(?, ?, ?, ?, ?, ?)}**
- **So = {(ø, ø, ø, ø, ø, ø)}**

Step 2: Generalize the initialize hypothesis for the first positive instance. I1 is a positive instance so generalize

I1	**>=9**	**Yes**	**Excellent**	**Good**	**Fast**	**Yes**	**Positive Instance**
S1 =		**< > =9**	**Yes**	**Excellent**	**Good**	**Fast**	**Yes>**
G1=		**<?**	**?**	**?**	**?**	**?**	**?>**

Step3:

Iteration1 : Scan the next instance **I2**. Since **I2** is a positive instance, generalize **S1** to include positive instance **I2**. For each of the non-matching attribute value in 'S1' put a **?** to include this positive instance . The third attribute value is mismatching in 'S1' with 'I2' so put a '?'. Since G1 is consistent with Positive instance the resulting S2 and G2 is as follows

S1 =	<> =9	Yes	Excellent	Good	Fast	Yes>	
I2	>=9	Yes	Good	Good	Fast	Yes	**Positive Instance**
S2 =	**<> =9**	**Yes**	**?**	**Good**	**Fast**	**Yes>**	
G1=	**<?**	**?**	**?**	**?**	**?**	**?>**	

Step3:

Iteration I2: Scan the next instance I3. Since it is negative instance, S3 remains same as S2 as there is no inconsistent hypothesis in S2. But Specialize G2 to negative example as follows:

Generate for each of the non-matching attribute value in S2 and fill with the attribute value of S2 in set of G3. In those generated hypotheses, for all matching attribute values, put a **'?'**. The **first, second and 6th attribute** values do not match, hence **'3'** hypotheses are generated in G3.

S2 =	**< > =9**	**Yes**	**?**	**Good**	**Fast**	**Yes>**	
G2=	**<?**	**?**	**?**	**?**	**?**	**?>**	
I3=	>=8	No	Good	Good	Fast	No	**Negative Instance**
G3	**< > =9**	**?**	**?**	**?**	**?**	**?>**	
	<?	**Yes**	**?**	**?**	**?**	**?>**	
	<?	**?**	**?**	**?**	**?**	**Yes>**	
S3	**< > =9**	**Yes**	**?**	**Good**	**Fast**	**Yes>**	

Instance	CGPA	Interactiveness	Practical Knowledge	Communication skills	Logical Thinking	Interest	Job Offer
I1	>=9	Yes	Excellent	Good	Fast	Yes	Yes
I2	>=9	Yes	Good	Good	Fast	Yes	Yes
I3	>=8	No	Good	Good	Fast	No	No
I4	>=9	Yes	Good	Good	Slow	No	Yes

Step3:

Iteration 3: Now scan I4. Since it is **a positive instance**, check for mismatch in the hypothesis 'S3' with I4. The 5th and 6th attribute value are mismatching, so add '?' to those attributes in 'S4'.

S3 =	**< > =9**	**Yes**	**?**	**Good**	**Fast**	**Yes>**	
G3	**< > =9**	**?**	**?**	**?**	**?**	**?>**	
	<?	**Yes**	**?**	**?**	**?**	**?>**	
	<?	**?**	**?**	**?**	**?**	**Yes>**	
I4	>=9	Yes	Good	Good	Slow	No	Positive Instance
S4	**< > =9**	**Yes**	?	Good	?	?>	

Prune G3 to exclude all inconsistent hypotheses with positive instance I4.

G3	**< > =9**	**?**	**?**	**?**	**?**	**?>**	
	<?	**Yes**	**?**	**?**	**?**	**?>**	
	<?	**?**	**?**	**?**	**?**	**Yes>**	Inconsistent
I4	>=9	Yes	Good	Good	Slow	No	Positive Instance

Since third hypotheses in G3 is inconsistent with this positive instance, remove the third one. The resulting S4 and G4 are,

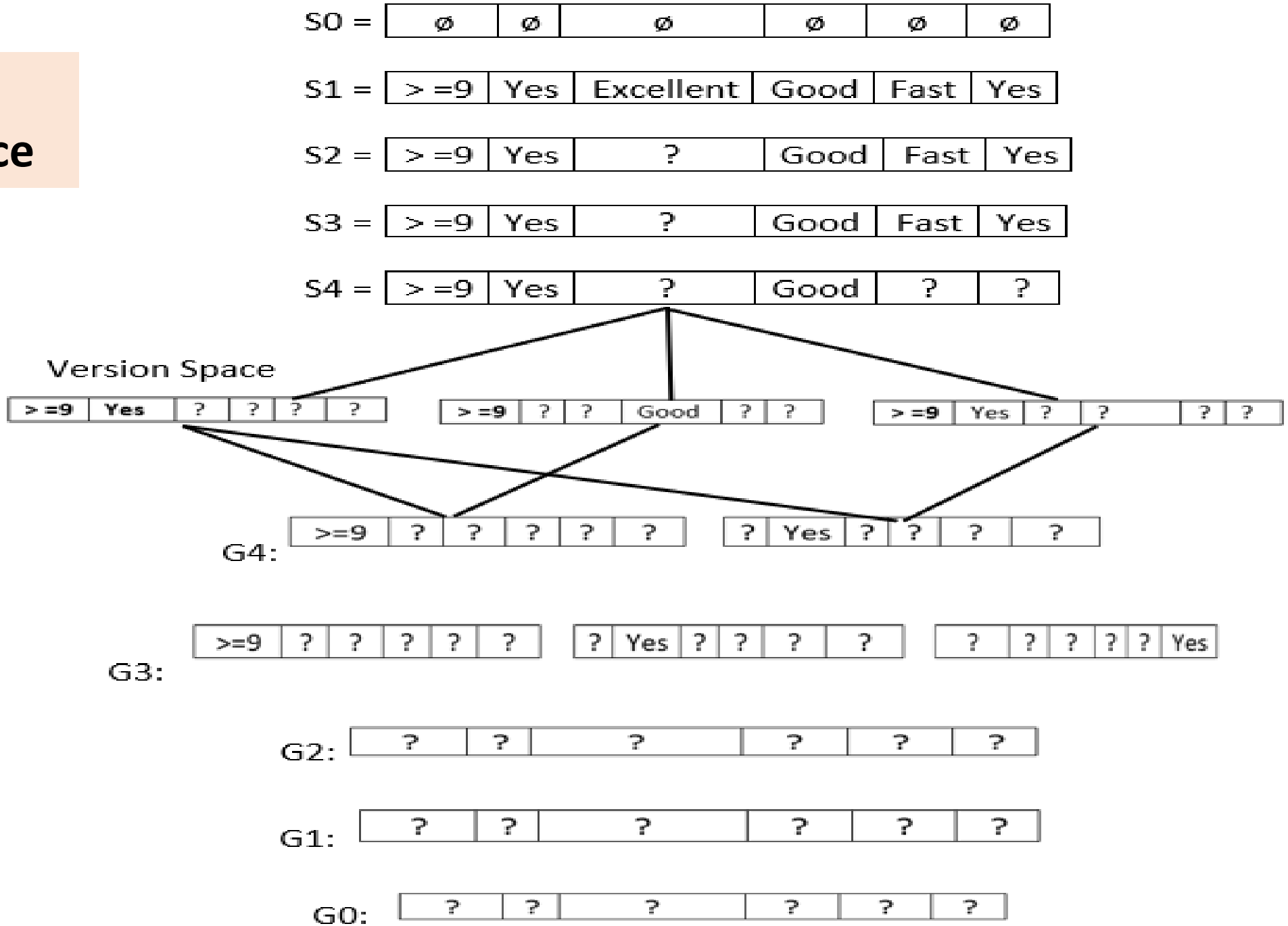
S4	**< > =9**	**Yes**	**?**	Good	**?**	**?>**

G4	**< > =9**	**?**	**?**	**?**	**?**	**?>**
	<?	**Yes**	**?**	**?**	**?**	**?>**

The Final version Space is:

S4	**< > =9**	**Yes**	**?**	Good	**?**	**?>**
G4	**< > =9**	**?**	**?**	**?**	**?**	**?>**
	<?	**Yes**	**?**	**?**	**?**	**?>**

Deriving the Version Space



What is Model based learning ?

- Model based machine learning describes all assumptions about the **problem domain in the form of a model**.
- These algorithms basically learn in two phases called **training and testing phase**.
- In **training phase a model is built from the training dataset** and is used to classify a **test instance during the testing phase**.
- **Examples :** Decision trees, Neural Networks and SVM

Modeling in Machine Learning

- A machine learning model is **a representation of a training dataset.**
- A machine learning model **learns from a training dataset** to make predictions on new data.
- The training process involves **feeding data into an algorithm** to **optimize and fine-tune the model**.
- Once **trained, the model can generate predictions for unseen data.**
- Key factors include selecting the **right model, training method, dataset, and setting performance expectations**.

Model Parameters and Hyper parameters

- In machine learning, learning model parameters is essential.
- **Model parameters**, like coefficients in regression or weights in neural networks, are **learned from training data**.
- **Hyperparameters**, such as regularization parameter (λ) or the number of decision trees, are **set before training and cannot be learned from** the data.

Model Evaluation and Overfitting

- Evaluating a machine learning model is crucial and involves splitting the dataset into **two parts: the training dataset, used for training, and the test dataset**, used to assess the model's performance on unseen data.
- The test dataset remains **hidden during** training to **ensure an unbiased evaluation.**
- If the datasets are too similar, the model may **overfit, performing well on known data but poorly on new data**.

Error Measurement and Loss Function

- During **prediction, errors occur** when the **estimated output differs from the actual output.** These errors are classified as:
- **Training Error (In-Sample Error):** Error measured when predicting data from the training set.
- **Test Error (Out-of-Sample Error):** Error observed when making predictions on new, unseen data.

Error Measurement and Loss Function

The overall error is captured using a loss function, which quantifies the difference between true values and predicted values. A common loss function is the **Mean Squared Error (MSE):**

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - f(X_i))^2$$

where Y_i represents the actual value and $f(X_i)$ represents the predicted value for an input X_i. A lower MSE indicates a more accurate model.

Machine Learning Process

The machine learning process consists of **four fundamental** steps:

1. **Algorithm Selection:** Choose a suitable machine learning algorithm based on the training data and problem domain.
2. **Model Training:** Input the training dataset and train the algorithm to learn patterns from the data.
3. **Parameter Tuning:** Adjust model parameters to enhance learning accuracy.
4. **Model Evaluation:** Assess the trained model's performance to ensure its effectiveness.

Model Selection and Model Evaluation

A major challenge in machine learning is selecting the right algorithm for a problem, focusing on two aspects:

- **Model performance (how well it performs on training data)** and
- **Model complexity (how complex the model is post-training).**
- Model selection involves choosing the **best model or adjusting features and hyperparameters**.
- The goal is to pick a model that performs well with the dataset, as no model is perfect.

Approaches for Machine Learning Model Selection

- Several methods can be used to select a suitable machine learning model:
- **Resampling Methods:** The dataset is split into **training, testing, and validation subsets**. The model's performance is evaluated across all phases, making this approach suitable for smaller datasets.
- **Basic Fitting Approach:** A model is trained using the training dataset, and performance metrics such as accuracy or error are computed.
- **Probabilistic Framework:** The model's performance is quantified using probability-based scoring methods.

3.6.2 Re-sampling Methods

- Re-sampling improves model accuracy by creating different training and test datasets through **random selection**.
- It aids in model evaluation and selection.
- Common **re-sampling methods include**
  - **random train/test splits**,
  - **cross-validation (K-fold, LOOCV), and**
  - **bootstrap.**

Cross-Validation

- Cross-validation is a model evaluation technique where a portion of the **training data is set aside for validation.** The model is trained on the remaining data, and its performance is estimated by averaging errors across different test sets. Popular cross-validation methods include:

1. **Holdout Method**
2. **K-fold Cross-Validation**
3. **Stratified Cross-Validation**
4. **Leave-One-Out Cross-Validation (LOOCV)**

Holdout Method

- The holdout method is the simplest form of cross-validation. The **dataset is split into two subsets: a training dataset and a test dataset**. The model is trained using the training set and then evaluated on the test set.
- **Single-use Holdout:** The dataset is **split once, and the model** is evaluated.
- **Repeated Holdout:** The splitting process is **repeated multiple times for better accuracy estimation.**
- Although straightforward, this method may **lead to high variance**, as performance depends on how the dataset is divided.

K-Fold Cross-Validation

- K-fold cross-validation is a more robust evaluation technique. The dataset is divided **into *k* equal parts (folds).** The process involves:
- **Using *k-1* folds for training and one fold for testing.**
- Repeating this process *k* times, each time selecting a **different fold** for testing.
- **Averaging the performance across** all iterations to obtain the final model evaluation.
- This method provides a more reliable assessment of model performance by ensuring that **every data point is used for both training and testing**.

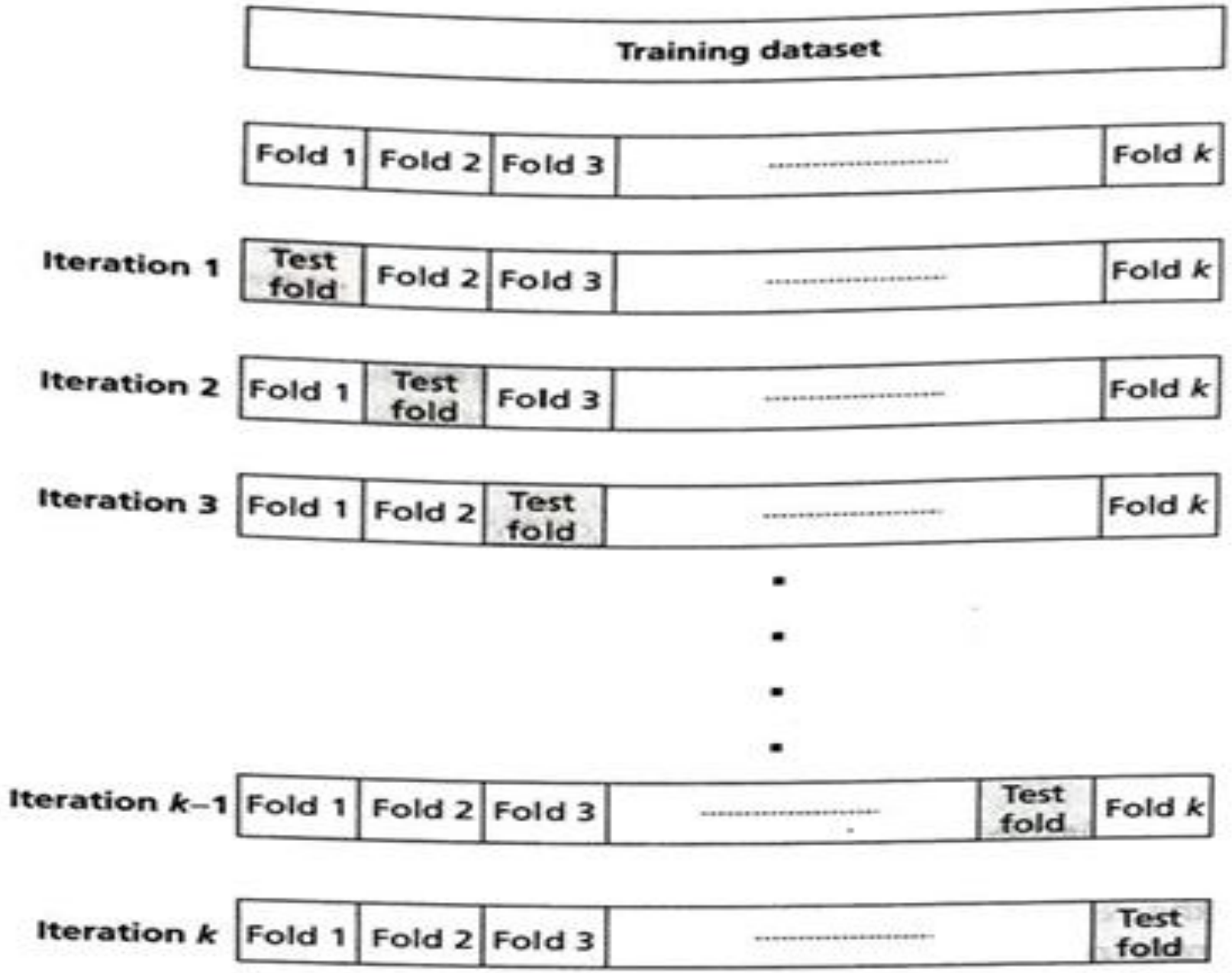


Figure 3.4: Illustration of *K*-fold Cross-Validation

Algorithm : K-Fold Cross-Validation

- **Input:**
- A dataset containing *n* instances
- *K* – Number of folds
- **Steps:**
- Perform the following steps *K* times, using a different "**hold-out**" fold for evaluation each time:
  - **Divide the dataset into *K* equal-sized folds.**
  - **Train the model using *(K-1)* folds.**
  - **Test the model on the remaining "hold-out" fold.**
- Calculate the average **performance across all *K* hold-out folds.**

Stratified K-Fold Cross-Validation

- This method is similar to ***k*-fold cross-validation** but with a slight difference. Here, it is ensured that while splitting the dataset into *k* folds, **each fold should contain the same proportion of instances** with a given categorical value.
- This is called **stratified cross-validation**.
- This helps improve the reliability of model evaluation, especially for imbalanced datasets.

Leave-One-Out Cross-Validation (LOOCV)

- LOOCV **(Leave-One-Out Cross-Validation)** splits a dataset of n instances into **training (n-1) and test (1) sets**, repeating the process **n times,** each time testing on a different instance.
- The model's performance is evaluated by averaging the test errors. While LOOCV is computationally expensive, it tends to have lower bias.
- For example, with **100 instances, it trains on 99 instances** and **tests on one, repeating this process 100 times**.
- The illustration in **Figure 3.5** depicts this iterative resampling process.

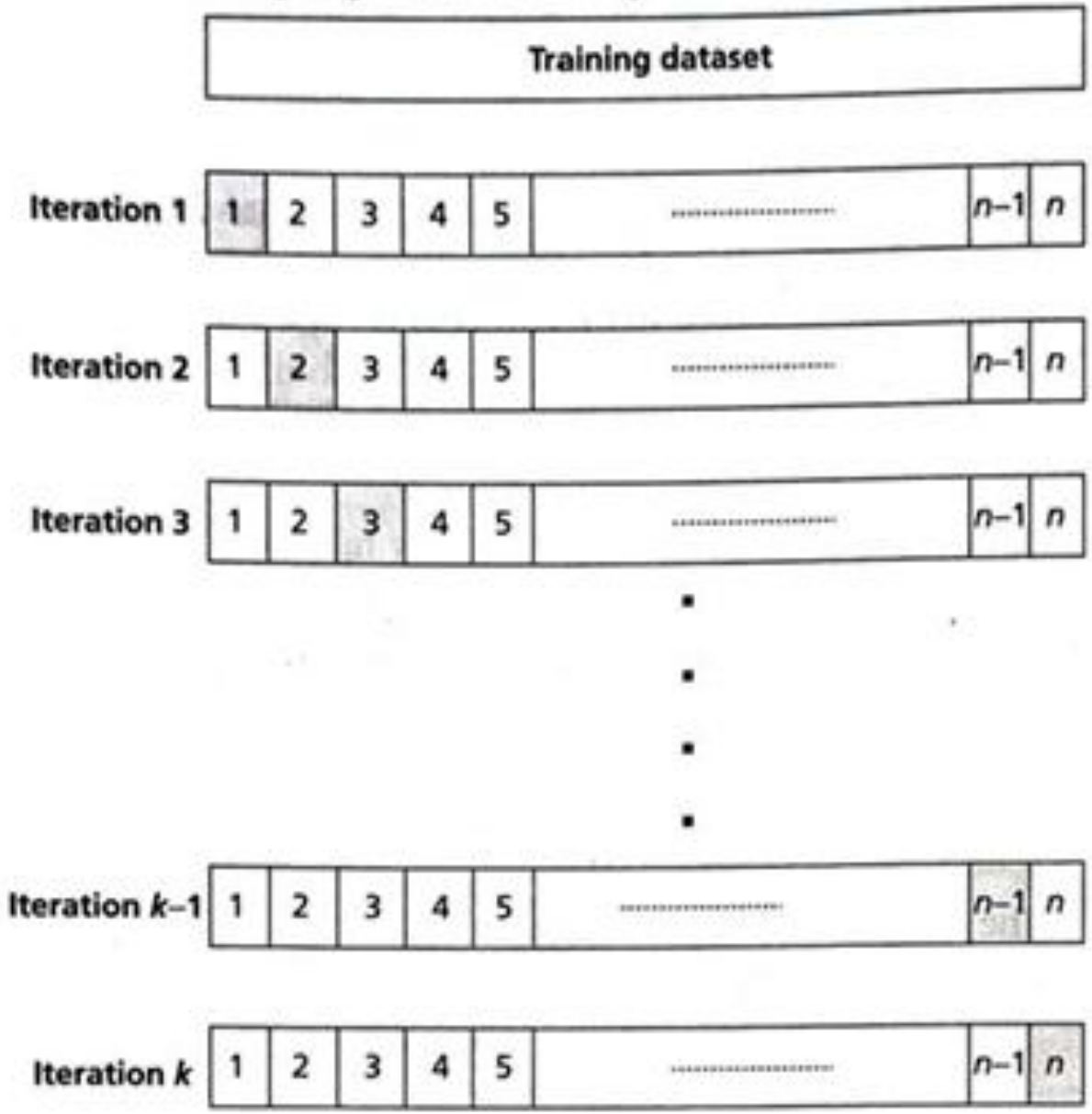


Figure 3.5: Illustration of Leave-One-Out Cross-Validation

Algorithm 3.5: LOOCV

- **Input: Dataset with *n* instances**
- Repeat the following steps *n* times, selecting a different "**hold-out**" data instance in each iteration:
  - Train the model using ***(n-1)*** **data instances**.
  - Evaluate the model on the remaining single **"hold-out"** instance.
- Compute the average performance across all ***n*** **holdouts**.

Model Performance

- Classifiers can be unstable, as small input changes can significantly affect the output.
- A solid evaluation framework is crucial, with various metrics available to assess a classifier's quality.
- One common approach to compute these metrics is using a **contingency table**.
- For instance, consider a disease detection test, like for cancer, with the example shown in Table 3.4.

Table 3.4: Contingency Table

Test vs Disease	Has Disease (Cancer)	Does Not Have Disease (No Cancer)
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negative (FN)	True Negative (TN)

In this table:

- **True Positive (TP):** The number of cancer patients correctly classified by the test.
- **True Negative (TN):** The number of normal patients without cancer correctly detected.
- **False Positive (FP):** A test result that wrongly indicates the presence of cancer when the patient is healthy.
- **False Negative (FN):** A test result that wrongly indicates no cancer when the patient actually has cancer.

False Positives (FP) and False Negatives (FN) are costly errors in classification.

Derived Metrics from the Contingency Table

1. **Sensitivity (True Positive Rate)**
   - Measures the probability of correctly identifying positive cases (i.e., detecting cancer when it is present).
   - **Formula:**

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

Derived Metrics from the Contingency Table

2. **Specificity (True Negative Rate)**

   - Measures the probability of correctly identifying negative cases (i.e., detecting no cancer when it is absent).
   - **Formula:**

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Derived Metrics from the Contingency Table

3. Positive Predictive Value (Precision)

- The probability that an object classified as positive is actually positive.
- Formula:

$$\text{PPV} = \frac{TP}{TP + FP}$$

Derived Metrics from the Contingency Table

4. **Negative Predictive Value (NPV)**

- The probability that an object classified as negative is actually negative.
- **Formula:**

$$\text{NPV} = \frac{TN}{TN + FN}$$

Derived Metrics from the Contingency Table

5. **Accuracy**

   - Measures the overall correctness of the classifier, considering both positive and negative classifications.
   - **Formula:**

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Derived Metrics from the Contingency Table

6. **Precision** – Also known as **positive predictive power**, precision is defined as the ratio of true positives to the sum of true positives and false positives.

Formula:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Precision indicates how well the classifier predicts the positive classes.

Derived Metrics from the Contingency Table

7. **Recall** – Also referred to as **sensitivity**, recall is the probability of correctly identifying positive cases.

Formula:

$$\text{Recall} = \text{Sensitivity} = \frac{TP}{TP + FN}$$

F- Measure or F1 Score

The **F1 Score** is the harmonic mean of **Precision** and **Recall**, and it is calculated using the following formula:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Substituting the formulas for **Precision** and **Recall**:

$$F1 = 2 \times \frac{\frac{TP}{TP+FP} \times \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}}$$

This metric is useful when you need a balance between **Precision** and **Recall**, especially in cases where false positives and false negatives have significant consequences.

Visual Classifier Performance

- The **Receiver Operating Characteristic (ROC) curve** and the **Precision-Recall curve** are used to visually evaluate the performance of classifiers.
- **ROC curves** provide a visual method to assess the accuracy and compare different classifiers.
- An ROC curve is a plot of **sensitivity** (True Positive Rate) against **1-specificity** (False Positive Rate) for a given model.

Sample ROC Curve

- Figure 3.6 shows a sample ROC curve with the results of five classifiers labeled A to E:
- **A** – Represents the ROC of an average classifier.
- **E** – Represents the ideal classifier with an area under the curve (AUC) of 1.0. Theoretically, AUC values range from **0.9 to 1** for a good classifier.
- **B, C, and D** – Represent classifiers that are categorized as **good, better,** and **still better** based on their area under the curve.
- A larger area under the curve indicates a better classifier.

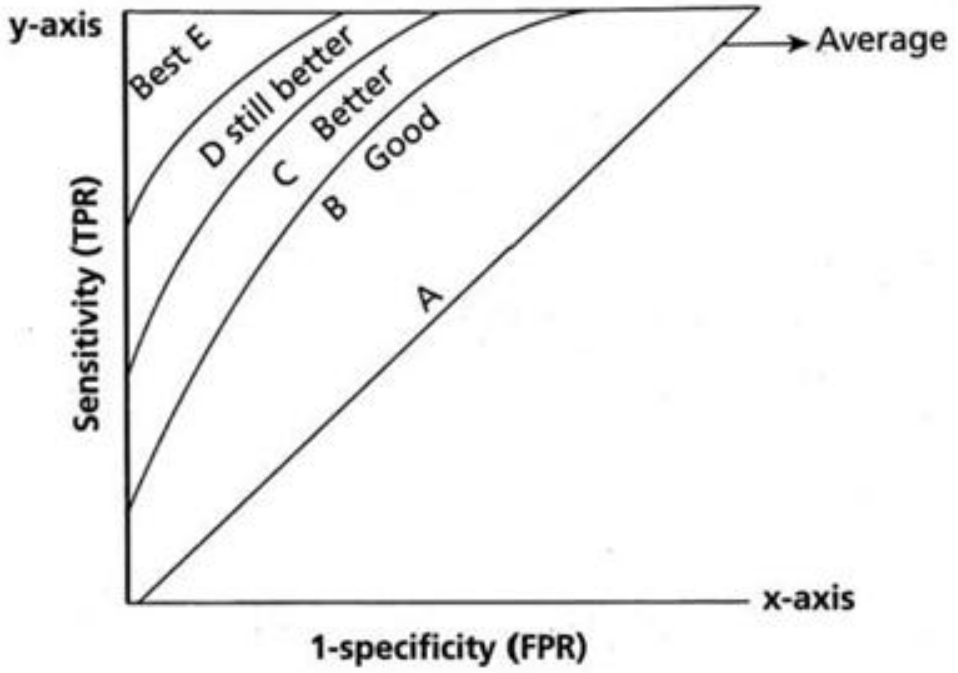


Figure 3.6: A Sample ROC Curve

How Classifier Prediction Works

The classifier prediction depends on the **threshold value**:

- A threshold of **0.5** maps the probability to either **class 0 or class 1**.
- Adjusting the threshold helps in controlling the **False Positive (FP) and False Negative (FN) rates.**
- This is useful when focusing on reducing error – **either FP or FN** – during an experiment.

Plotting the ROC Curve

1. **Start at the bottom-left corner of the plot**.
2. **For each true positive case, move upward and plot a point**.
3. **For each false positive case, move right and plot a point.**
4. **Repeat this process until the complete curve is drawn**.

Plotting the ROC Curve

If the ROC curve follows the **diagonal line, the classifier has no skill and** performs like a random classifier.

A better classifier will have a **curve above the diagonal line**.

A curve closer to the **diagonal indicates a less accurate classifier**.

Significance of AUC

- AUC measures the accuracy of the model.
- A model with an **AUC of 1** is considered perfect.
- A model with an **AUC of 0** indicates that the model is entirely wrong.
- The approximate **AUC under a precision-recall** curve reflects the model's effectiveness across thresholds.

Precision-Recall Curve vs ROC Curve

- A **precision-recall curve** is a plot of **precision** and **recall** for different threshold values.
- Precision-recall curves **are useful when there is a class imbalance**, where one class has significantly more labels than the other.
- **ROC curves** are more suitable when there **is no class imbalance**.
- Precision-recall curves are preferred for **moderate-to-large class imbalances**.

Scoring Methods

- Another approach for model selection involves combining the **complexity** of the model with its **performance** into a single score.
- The model is selected based on which one **maximizes** or **minimizes** the score.

Minimum Description Length (MDL)

- MDL is a method that describes **the target variable and the model in terms of the number of bits required**.
- The principle of MDL is to use the **minimum number of bits to represent the data and the model.**
- It is based on **Occam's Razor, which states that the simplest explanation is usually the best.**
- MDL recommends selecting the hypothesis that minimizes the sum of two parts:
  - **Model complexity**
  - **Data description**

MDL Formula

Let **h** be the learning model:

- $L(h)$ = Number of bits to represent the model
- $L(D|h)$ = Number of bits to represent predictions based on training data

$$\text{MDL} = L(h) + L(D|h)$$

MDL can also be written using **negative log-likelihood**:

$$\text{MDL} = -\log(p(\Theta)) - \log(p(y|x, \Theta))$$

where:

- **y** = Target variable
- **x** = Input
- **Θ** = Model parameters

End of Module2