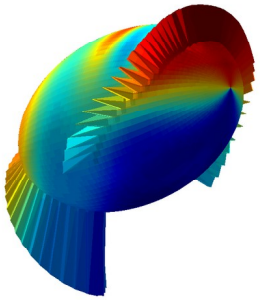
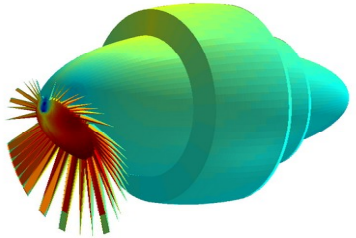
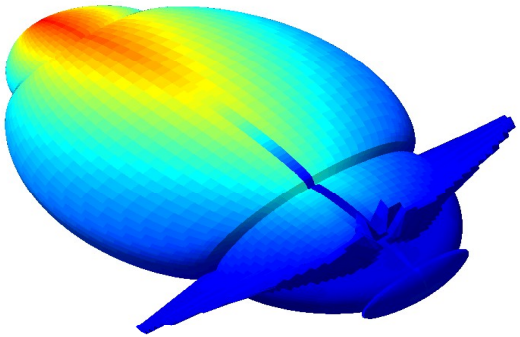
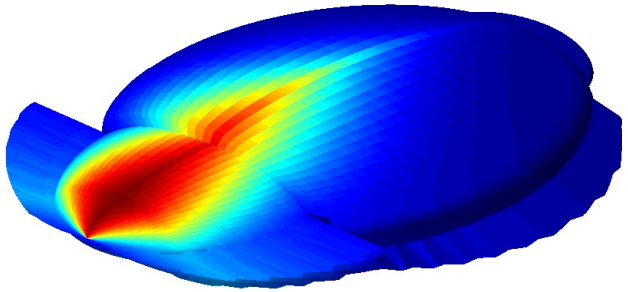




Lecture 3
Basic Concepts





Some Tasks for Machine Learning

- **Classification and recognition**
  Classify inputs into one of several classes – ***many applications***.
- **Estimation and prediction**
  Fit a function to given input-output or current-future data – ***many applications***.
- **Model induction and inference**
  Infer the characteristics of measured quantities and the dependencies between them (including dynamics).
- **Decision and control**
  Produce appropriate decisions / actions in complex environments.
- **Memory**
  Store and recall information based on imperfect cues.
- **Map formation**
  Organize information into useful data structures.
- **Analyzing sensory information (images, video, sound, etc.)**
- **Understanding and using language**

Sensing: Interface with the Environment

- **Multiple modalities:** Visual, auditory, chemical, etc.
- Limited precision / resolution.
- Requires transduction: Internal representation of sensed data.

- Active vs. passive sensing
- Adaptive sensing
  - ✓ The eye anticipates a moving object's future position
  - ✓ Day/night vision trades off sensitivity vs. color

An intelligent system's world is totally constrained by what it can sense.

Representing Information

Information can be represented in many ways:

- **Data structures:** Lists, tables, trees, JSON objects.
- **Linguistic** – words, phrases, sentences, documents, etc.
- **Images**
- **Video**
- **Logic statements:** for-all(x) (tree(x) → has-leaves(x))
- **Neural activity patterns** – spatial and spatiotemporal, spiking, etc.

Machine learning ultimately requires a ***numerical representation*** – even for symbolic data (e.g., words, concepts, etc.) or images.

→ **Feature Space**

Feature Space

Feature ≡ An attribute of data items (e.g., size, color, count, etc.)

Feature Space ≡ The ***metric space*** defined by the features of the data.

- Each data item is a point in the feature space ≡ **data point**.
- If there are *n* features, *n* is called the **dimensionality** of the feature space.

  → Each data point is an *n*-dimensional vector.
- **Metrics** are typically of two types:

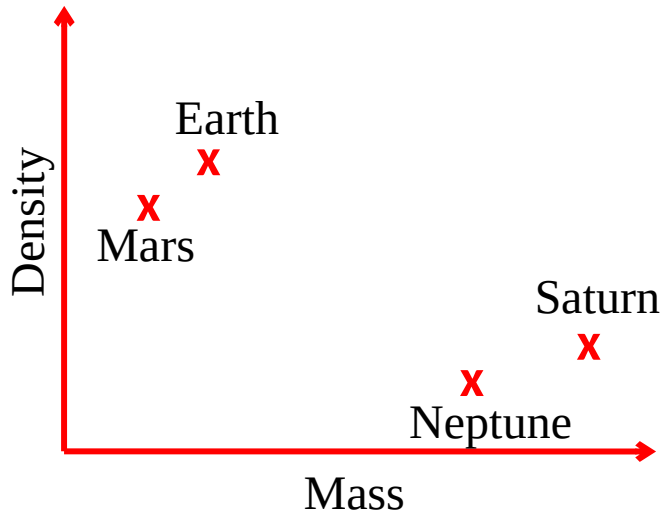
  Distance Metric: The distance between two data points in feature space, e.g. Euclidean distance, Hamming distance, etc.

  Similarity Metric: The similarity/proximity between two points in feature space, e.g., overlap, cosine similarity, etc.

Examples

Example 1:

Given two features: {mass density}, every planet can be placed in a 2-dimensional feature space.



Mars and Earth are ***similar***.
Earth and Saturn are ***distant***.

Example 2:

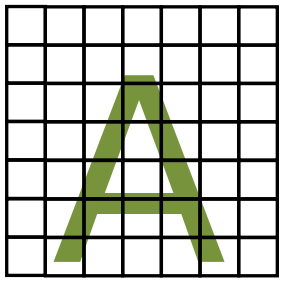
Given a vocabulary of N words, $W = \{w_i\}$ and a document D with m word tokens, the document can be represented by how many times each word occurs in it, e.g.

If

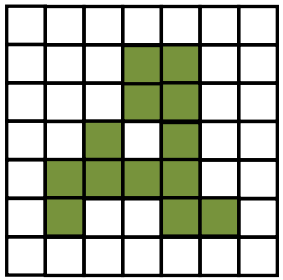
W = {I you is was a an the be not or to this that question answer}
D = *To be or not to be? That is the question*
= [0 0 1 0 0 0 1 2 1 1 2 0 1 1 0]

Example 3:

Representing images as pixel state (intensity, color, etc.)



$\longrightarrow \quad [\, x_1\, x_2\, \ldots .\, x_{49}\,] \qquad x_i \in \{0, 1\}$



= 0000000 0001100 0001100 0010100 0111100 0100110 0000000

Each pixel value is a feature of the data.

Feature spaces can get **very large!**

For example, a 1,000×2,000 pixel color image would have

2,000,000×3×8 = 48 million bits

(2,000,000 pixels, three colors – Red/Green/Blue, 8-bit color code)

If the color is a more realistic 12-bit, we have 72 million bits!

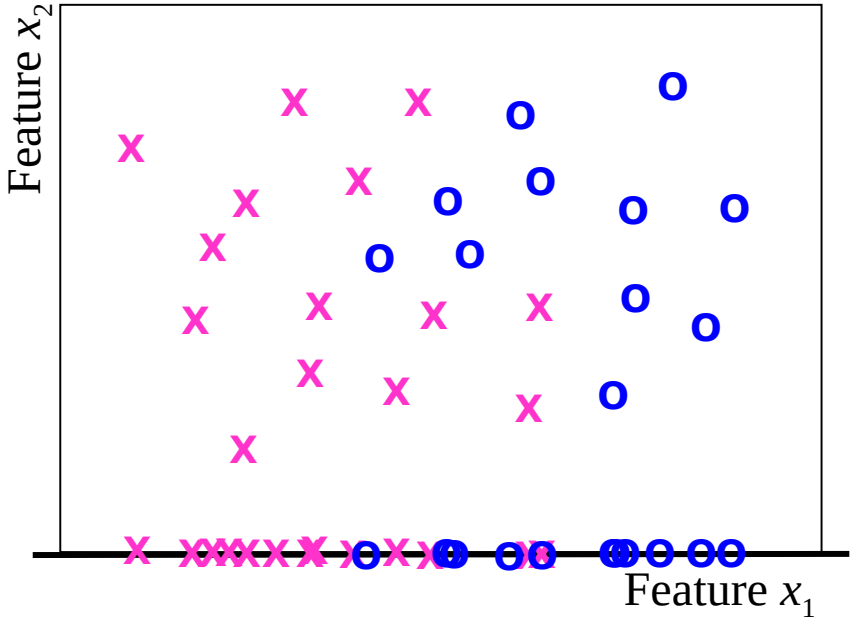
For machine learning, we would generally use the numerical values of the RGB features → 6 million numbers.

For this, we would need a neural network with ***6 million input neurons!***

Dimensionality Reduction

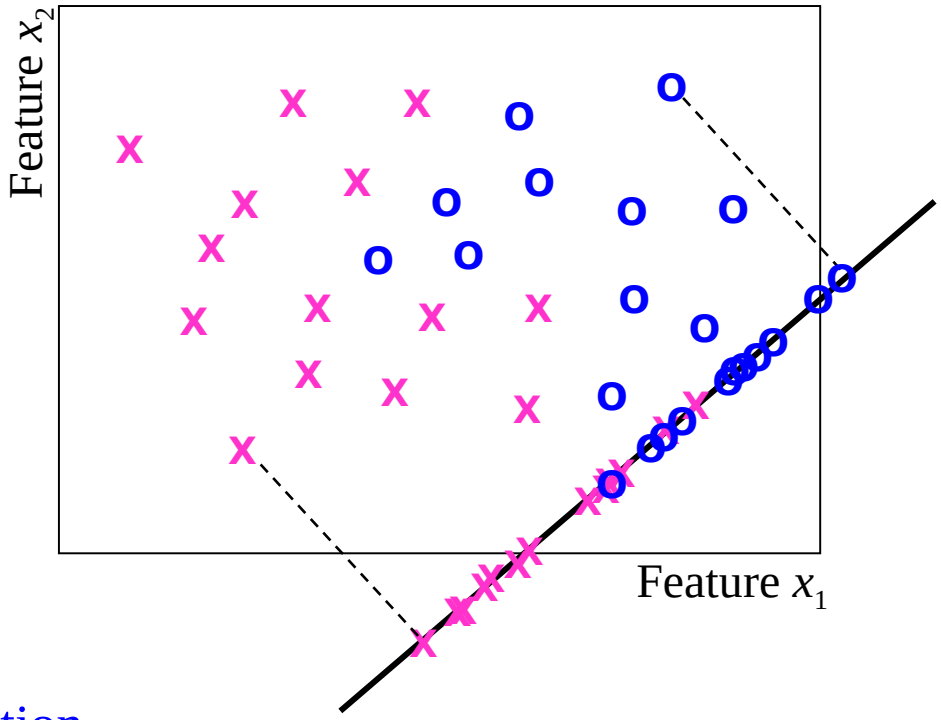
Feature Selection

Project the data on to a subset of features.



Subspace Projection

Project the data on to a subspace of the feature space (e.g., PCA).



Nice tutorial on dimensionality reduction.

Dimensionality Reduction Methods

Feature Selection:
Regression-based methods (e.g. LASSO)
Correlation-based methods
Information theoretic methods (e.g., Conditional Mutual Information)
Tree-based methods (decision trees).

Statistical / Information-Theoretic Methods:
Principal component analysis (PCA)
Independent component analysis (ICA)

Neigborhood-Based Methods:
Multi-dimensional scaling (MDS)
ISOMAP
Stochastic neighbor embedding (SNE, tSNE)

Neural Network Methods:
Self-organized feature maps (SOFM)
Autoencoders

Subspace Projection

Feature Selection vs. Subspace Projection

Feature Selection

- Intuitive and easy to understand.
- Leads to meaningful features.
- May not produce the most efficient dimensionality reduction.

Subspace Projection

- More abstract and mathematical.
- Leads to features that may be hard to interpret.
- Can produce greater compression.

Adaptation and Learning

Goal

To produce useful responses to inputs / stimuli

Why is learning needed?

Because we live in a complex world:

- All situations cannot be foreseen.
- All plans cannot be pre-specified.
- The space of experience is infinite.
- The world is dynamic.

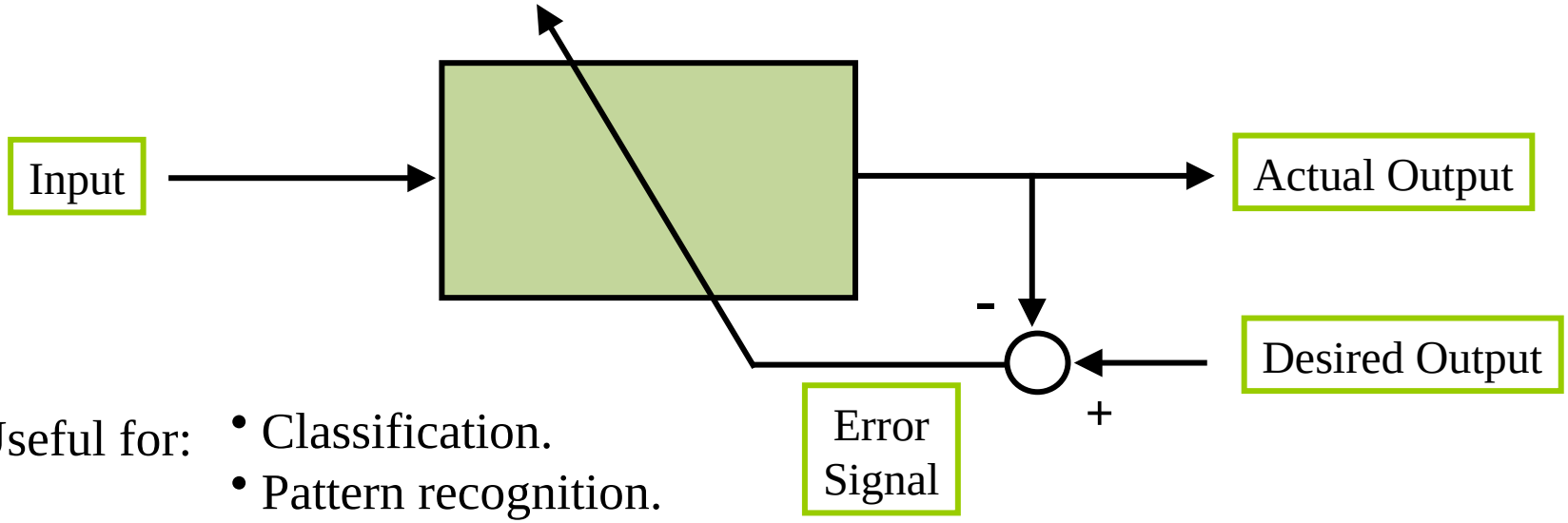
Dumb systems must be told ***how*** to perform their tasks.

Smart systems need only be told ***what*** task is to be performed.

Learning ↔ Determining optimal use of the system's existing set of abilities.

Supervised Learning

- System is shown correct input-output pairs – training set.
- Parameters are adjusted to bring system closer to correct output.
- System is tested on novel inputs – validation/test sets.



Useful for:

- Classification.
- Pattern recognition.
- Approximation.
- Prediction.
- Control.
- Memory.

The goal in supervised learning is to minimize ***empirical risk***.

Classification Problem:

$x \in \mathbf{X} \subseteq \Re^n \equiv$ **Feature Space**

$y \in \mathbf{Y} = \{C_1, C_2, \ldots, C_m\}$ **Classes**

Ground Truth $G: \mathbf{X} \rightarrow \mathbf{Y}$ Every point $x \in \mathbf{X}$ has a ***true class*** $y = G(x)$

Hypothesis: $H: \mathbf{X} \rightarrow \mathbf{Y}$ Classifier assigns a class $H(x) \in \mathbf{Y}$ to every $x \in \mathbf{X}$

Regression / Approximation / Function-Fitting Problem:

$x \in \mathbf{X} \subseteq \Re^n \equiv$ **Input Space**

$y \in \mathbf{Y} \subseteq \Re^m \equiv$ **Output Space**

Ground Truth $G: \mathbf{X} \rightarrow \mathbf{Y}$ Every point $x \in \mathbf{X}$ maps to a $y = G(x)$

Hypothesis: $H: \mathbf{X} \rightarrow \mathbf{Y}$ Estimator assigns value $H(x) \in \mathbf{Y}$ to every $x \in \mathbf{X}$

The goal is for ***H*** to minimize its error with respect to ***G***

Risk

Data Space $\equiv \mathbf{S} = \mathbf{X} \times \mathbf{Y}$

Loss Function $\boldsymbol{L}(x, y, \boldsymbol{H})$: $\mathbf{S} \times \Re \rightarrow [0,\infty)$

$\equiv$ The loss in generating $H(x)$ in response to x

e.g. $\boldsymbol{L}(x, y, \boldsymbol{H}) = \left(y - \boldsymbol{H}(x) \right)^2$

Since there is often some uncertainty or variation in data, assume that:
Ground Truth $\equiv$ joint probability $P(x, y)$.

Then $P(x,y)$: $\mathbf{S} \rightarrow [0, 1]$ such that the axioms of probability are satisfied.

Risk of $\boldsymbol{H} \equiv$ $\boldsymbol{R}(\boldsymbol{H}) = E\left[\boldsymbol{L}(x, y, \boldsymbol{H}) \right] = \int_{\mathbf{X} \times \mathbf{Y}} \boldsymbol{L}(x, y, \boldsymbol{H}) P(x, y) dy dx$

This is the ***true risk*** of hypothesis $\boldsymbol{H}$, but it cannot be calculated or used because we don't know the ground truth $P(x,y)$ for every point in $\mathbf{X} \times \mathbf{Y}$

Empirical Risk Minimization

In practice, we have ***a finite set of samples from*** $\mathbf{S}$ ↔ Dataset

Dataset: $\mathbf{D} = \{(x^k, y^k)\}; \quad k = 1,\ldots,N$

- **Training set** $\mathbf{D}_{Training}$
- **Validation set** $\mathbf{D}_{Validation}$
- **Test set** $\mathbf{D}_{Test}$

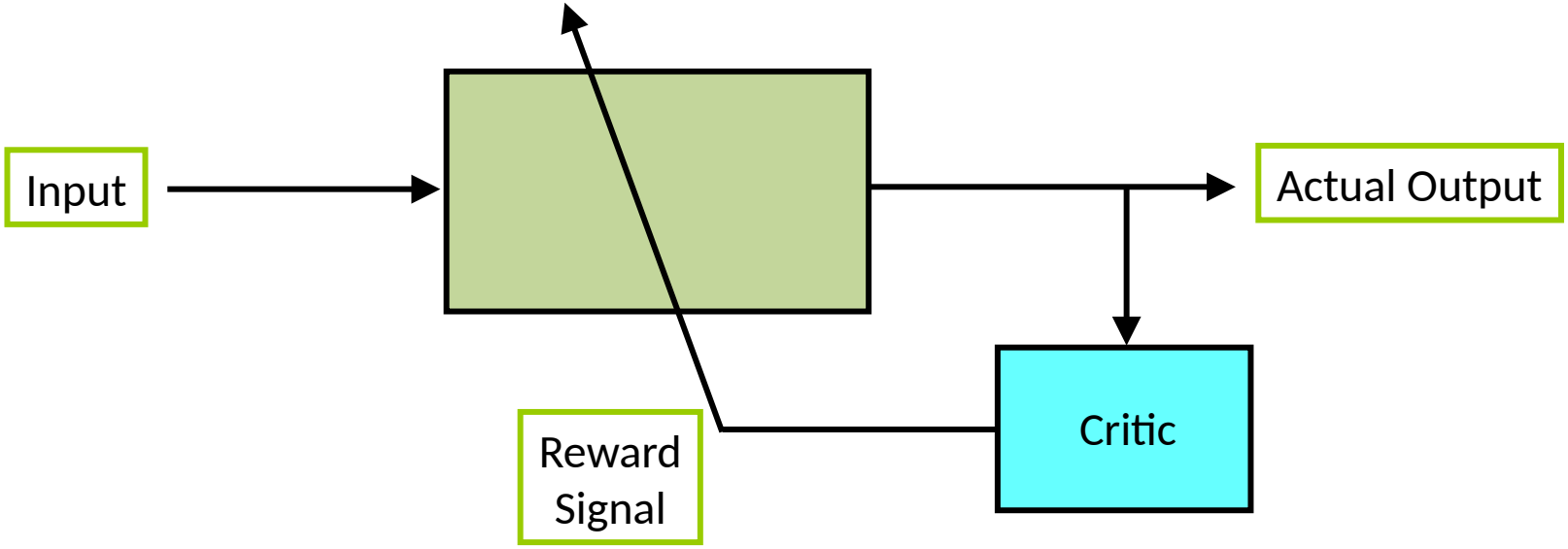
Empirical Risk $\equiv J(\boldsymbol{H}) = \frac{1}{|\mathbf{D}_q|} \sum_k \boldsymbol{L}(x^k, y^k, \boldsymbol{H})$ where $(x^k, y^k) \in \mathbf{D}_q$

This becomes the objective function for supervised learning, which tries to find the hypothesis $\boldsymbol{H}^*$ that minimize $J(\boldsymbol{H})$.

$$\boldsymbol{H}^* = \arg\min_{\phi} \left[\frac{1}{|\mathbf{D}_q|} \sum_k \boldsymbol{L}(x^k, y^k, \boldsymbol{H}) \right]$$ where ϕ are the parameters of $\boldsymbol{H}$.

Reinforcement Learning

- System is shown inputs → responses.
- Responses are rewarded (+1), punished (-1) or unrewarded (0).
- System changes parameters to increase expected future reward.



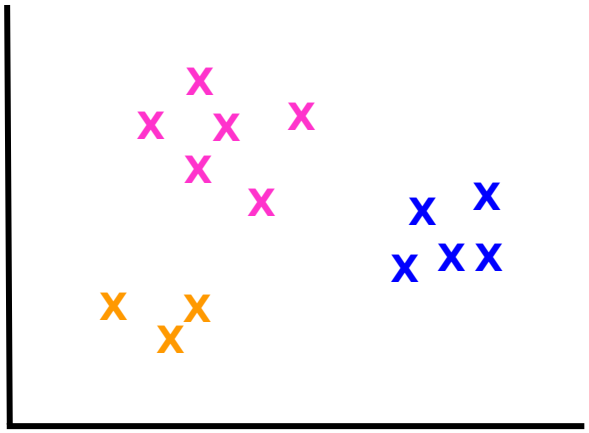
Useful for:

- Control.
- Decision-making.
- Behavior learning.

For a great simple Tutorial on RL:
https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

Unsupervised Learning / Self-Organization

- System is shown inputs → outputs
- Parameters are adjusted so the structure of the data becomes apparent → clustering, density estimation.

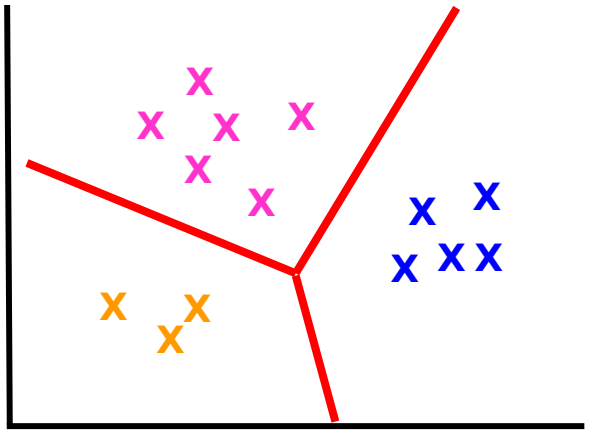


Useful for:

- Clustering.
- Feature detection.
- Classification.
- Pattern recognition.
- Memory.
- Dimensionality reduction.
- Density estimation.
- Map building.
- Control organization.

Unsupervised Learning / Self-Organization

- System is shown inputs → outputs
- Parameters are adjusted so the structure of the data becomes apparent → clustering, density estimation.



Useful for:

- Clustering.
- Feature detection.
- Classification.
- Pattern recognition.
- Memory.
- Dimensionality reduction.
- Density estimation.
- Map building.
- Control organization.

Other Possibilities

Semi-Supervised Learning:

When most of the data is unlabeled but part of it is labeled.

- Learn from the labeled data as in supervised learning.
- Use the results to make intelligent inferences about the unlabeled data, e.g., assuming distributions, smoothness and/or clustering.
- Learn more from the inferred data.

Non-Iterative Supervised Methods:

When labels available for the training data are applied to novel test data through a calculation or heuristic rather than by fitting a model, e.g., k-nearest neighbors classifier.

Transfer Learning

Training a system for Task A and then using all or part of it for Task B.

Task A is used because if one or more of these:

- It has more data available.
- It has labeled data available for supervised learning.
- It is easier to learn.

But transfer learning is useful only if:

- The information learned in Task A is useful for Task B
- Task B is significantly more difficult to learn.

Example:

Train a neural network to recognize digits. Then use the features learned by that network to train a system to recognize letters with just a little more training.

Generalization

Inducing/hypothesizing relationships over a large space based on experience with a limited subset of it.

Useful learning assumes/requires generalization

If generalization is not needed
i.e. if all relevant data are already available

→ use a lookup table!

In most eases, generalization requires implicit assumptions.

Why?

Because an infinite number (or very large number)
of hypothesis are consistent with a finite subset of data

→ choice → assumptions

Cross-Validation

Cross validation is a way to obtain greater confidence in the performance of a learning-based model such as neural networks, k-NN, etc.

Basic Approach – K-Fold Cross-Validation:

- Given a dataset, $\boldsymbol{D}$, partition it into K subsets, $\boldsymbol{D}_j$ $j = 1, 2, \ldots, K$.
- For $j = 1$ to K
  - Keep set $\boldsymbol{D}_j$ for validation and train the model on the rest of the data.
  - Test the model on $\boldsymbol{D}_j$ to get performance value P_j.
- Combine the P_j values (e.g., averaging) to get overall performance.

Variants:

- Leave-one-out cross-validation: One data point is left out for validation and the rest is used for training → train n times for n data points. This is useful when data is limited and hard to get.
- Random sample cross-validation: Here, the subsets are not mutually exclusive but obtained by randomly sampling $\boldsymbol{D}$. This can allow for more folds with larger sets.

Uses of Cross-Validation

Cross-validation provides *general information* about the potential for a model to work in a particular application – independent of the specific dataset used to train the model (but, of course, not independent of the global dataset ***D***).

Cross-validation *does not* produce *one specific instance* of the model to be used for all data.

Cross-validation is used for:

- Deciding whether a model is likely to work for a dataset or application.
- Comparing two or more types of models to decide which one is best for a dataset.
- Determining the optimal size of a model for a given dataset.

For example, one can determine whether a 10 hidden neuron neural network works better than a 20 hidden-neuron neural network.

Or

Whether k-NN works better than a multi-layer neural network for a dataset.

Example: 5-Fold Cross-Validation