Instance-Based Learning

Instance-Based Learning: Local approximation to the target function that applies in the neighborhood of the query instance

- Cost of classifying new instances can be high: Nearly all computations take place at classification time
- Examples: *k*-Nearest Neighbors
- Radial Basis Functions and Extreme learning Machines
- Recommendation systems
- One shot learning and metric learning

k-Nearest Neighbor Learning

Instance $\mathbf{x} = [\ a_1(\mathbf{x}), a_2(\mathbf{x}), \ldots, a_n(\mathbf{x})\ ] \in \Re^n$

$d(\mathbf{x}_i,\mathbf{x}_j) = [\ (\mathbf{x}_i-\mathbf{x}_j).(\mathbf{x}_i-\mathbf{x}_j)\ ]^{1/2}$ = Euclidean Distance

(discrete attributes? Other distances?)

– Discrete-Valued Target Functions (classification)

$f : \Re^n \rightarrow V = \{v_1, v_2, \ldots, v_s)$

k-Nearest Neighbor Learning

Prediction for a new query **x**: (*k* nearest neighbors of **x**)

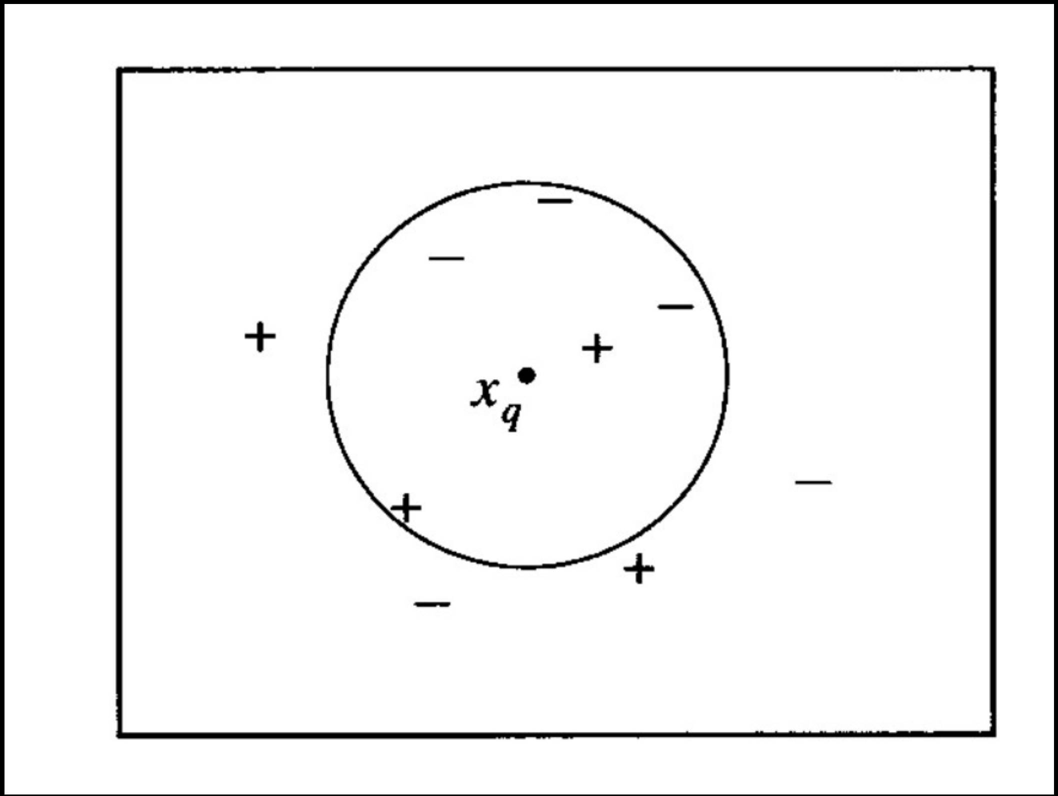
$$f(\mathbf{x}) = \mathrm{argmax}_{v \in V} \sum_{i=1,k} \delta[v, f(\mathbf{x}_i)]$$

$$\delta[v, f(\mathbf{x}_i)] = 1 \text{ if } v = f(\mathbf{x}_i) \text{ , } \delta[v, f(\mathbf{x}_i)] = 0 \text{ otherwise}$$

– Continuous-Valued Target Functions (regression)

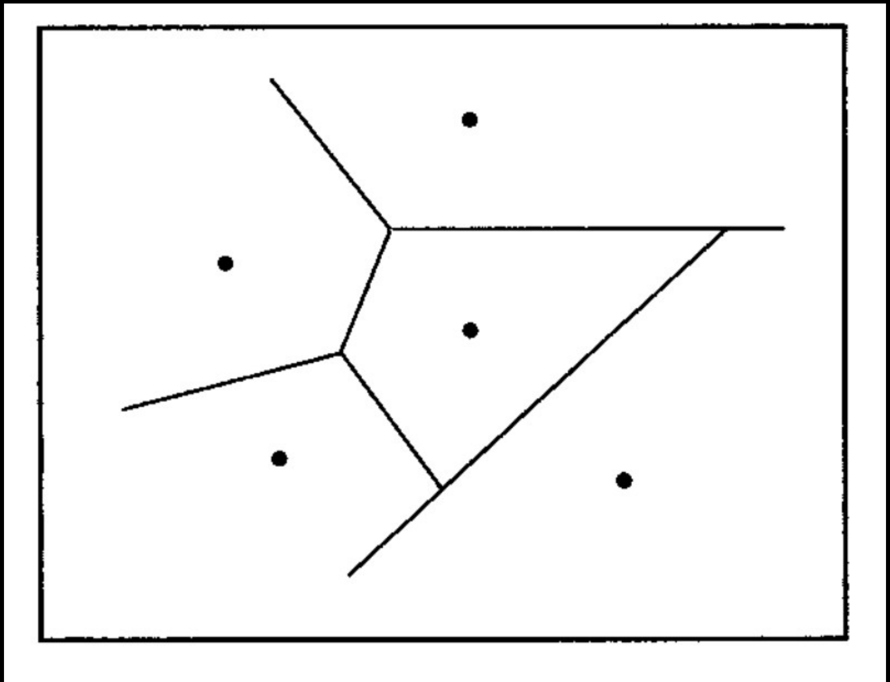
$$f(\mathbf{x}) = (1/k) \sum_{i=1,k} f(\mathbf{x}_i)$$

(+/-) Classification 2D Example



Why k>1?

Hyphotesis Space



Voronoi diagram

Using k=1

(prototypes)

k-Nearest Neighbor Learning

Finding neighbors is costly!

Alternative: use a fixed volume V centered in X

Potential problem: different densities

Distance-Weighted k-NN

$$f(\mathbf{x}) = \text{argmax}_{v \in V} \sum_{i=1,k} w_i \, \delta[v, f(\mathbf{x}_i)]$$

$$f(\mathbf{x}) = \sum_{i=1,k} w_i \, f(\mathbf{x}_i) \; / \; \sum_{i=1,k} w_i$$

$$w_i = [d(\mathbf{x}_i, \mathbf{x})]^{-2}$$

→ Weights more heavily closest neighbors

→ k can be set to “all”

Remarks for k-NN

- Robust to noise in general
- Quite effective for large training sets
- Inductive bias: The classification of an instance will be most similar to the classification of instances that are nearby in Euclidean distance

Remarks for k-NN

- Especially sensitive to the *curse of dimensionality, or to irrelevant features,* as all features are considered in distances.

- Possible elimination of irrelevant attributes by suitably chosen the metric:

$$d(\mathbf{x}_i,\mathbf{x}_j) = [\ (\mathbf{x}_i-\mathbf{x}_j)\mathbf{.Z.}(\mathbf{x}_i-\mathbf{x}_j)\ ]^{1/2}$$

Locally Weighted Regression

Builds an explicit approximation to $f(\mathbf{x})$ over a local region surrounding **x** (usually a linear or quadratic fit to training examples nearest to **x**)

Locally Weighted Linear Regression:

$$f_L(\mathbf{x}) = w_0 + w_1 x_1 + \ldots + w_n x_n$$

$$E(\mathbf{x}) = \Sigma_{i=1,k} [f_L(\mathbf{x}_i) - f(\mathbf{x}_i)]^2 \quad (\mathbf{x}_i \text{ nn of } \mathbf{x})$$

Locally Weighted Regression

Generalization:

$$f_L(\mathbf{x}) = w_0 + w_1 x_1 + \ldots + w_n x_n$$

$$E(\mathbf{x}) = \Sigma_{i=1,N} K[d(\mathbf{x}_i,\mathbf{x})]\,[f_L(\mathbf{x}_i) - f(\mathbf{x}_i)]^2$$

$K[d(\mathbf{x}_i,\mathbf{x})]$ = kernel function

Radial Basis Functions

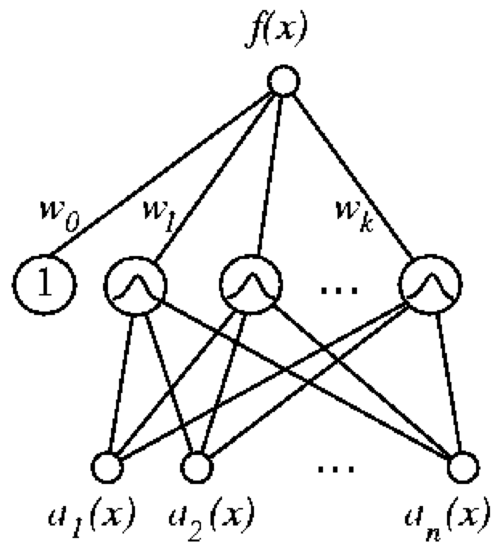
Approach closely related to distance-weighted regression and artificial neural network learning

$$f_{\mathrm{RBF}}(\mathbf{x}) = w_0 + \sum_{\mu=1,k} w_\mu K[d(\mathbf{x}_\mu,\mathbf{x})]$$

$K[d(\mathbf{x}_\mu,\mathbf{x})] = \exp[-d^2(\mathbf{x}_\mu,\mathbf{x})/ 2\sigma^2_\mu]$ = Gaussian kernel function

Radial Basis Functions

Radial Basis Function Networks



Radial Basis Functions

Training RBF Networks

1st Stage:

- Determination of k (=number of basis functions)
- $\mathbf{x}_\mu$ and σ_μ (kernel parameters)

$\rightarrow$ Expectation-Maximization (EM) algorithm

2nd Stage:

- Determination of weights w_μ

$\rightarrow$ Linear Problem

Extreme Learning Machines

Random centers, ultra high dimensionality

1st Stage:

- Set a big number or random kernels

2nd Stage:

- Determination of weights w_μ

$\rightarrow$ Linear Problem

Remarks on Lazy and Eager Learning

Lazy Learning: stores data and postpones decisions until a new query is presented

Eager Learning: generalizes beyond the training data before a new query is presented

- Lazy methods may consider the query instance **x** when deciding how to generalize beyond the training data *D* (local approximation)
- Eager methods cannot (they have already chosen their global approximation to the target function)

Recommendation Systems

- I like these books, can you suggest me something to read?

Recommendation Systems

- I like these books, can you suggest me something to read?
- Different answers:
  - – Look for books that are similar in style to those books
    - Identify some features that characterize the books
    - Give values
    - Search for similar books → distance, neighbors!

Content-based recommendation (CB)

Recommendation Systems

- I like these books, can you suggest me something to read?
- Different answers:
  - Look for users that like books similar to yours, then look for books that they like
    - Define a distance based on similar rankings/purchases
    - Search for similar users → distance, neighbors!
    - Look for books that those users like

Collaborative Filtering recommendation (CF)

Metric Learning

Back:

Possible elimination of irrelevant attributes by suitably chosen the metric:

$$d(\mathbf{x}_i,\mathbf{x}_j) = [\ (\mathbf{x}_i-\mathbf{x}_j)\mathbf{.Z.}(\mathbf{x}_i-\mathbf{x}_j)\ ]^{½}$$

We can learn more general metrics:

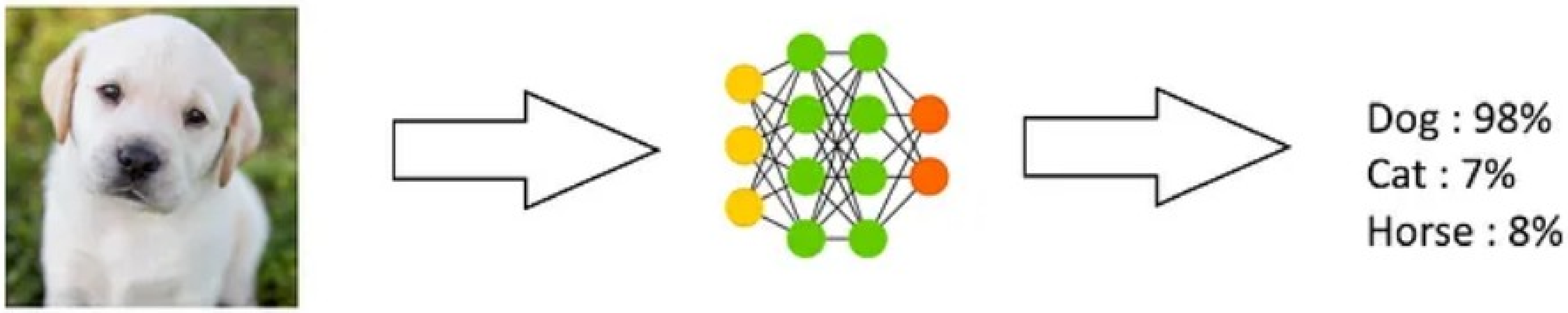
$$d(\mathbf{x}_i,\mathbf{x}_j) = \mathrm{K}(\mathbf{x}_{i,}\mathbf{x}_j)$$

Metric Learning

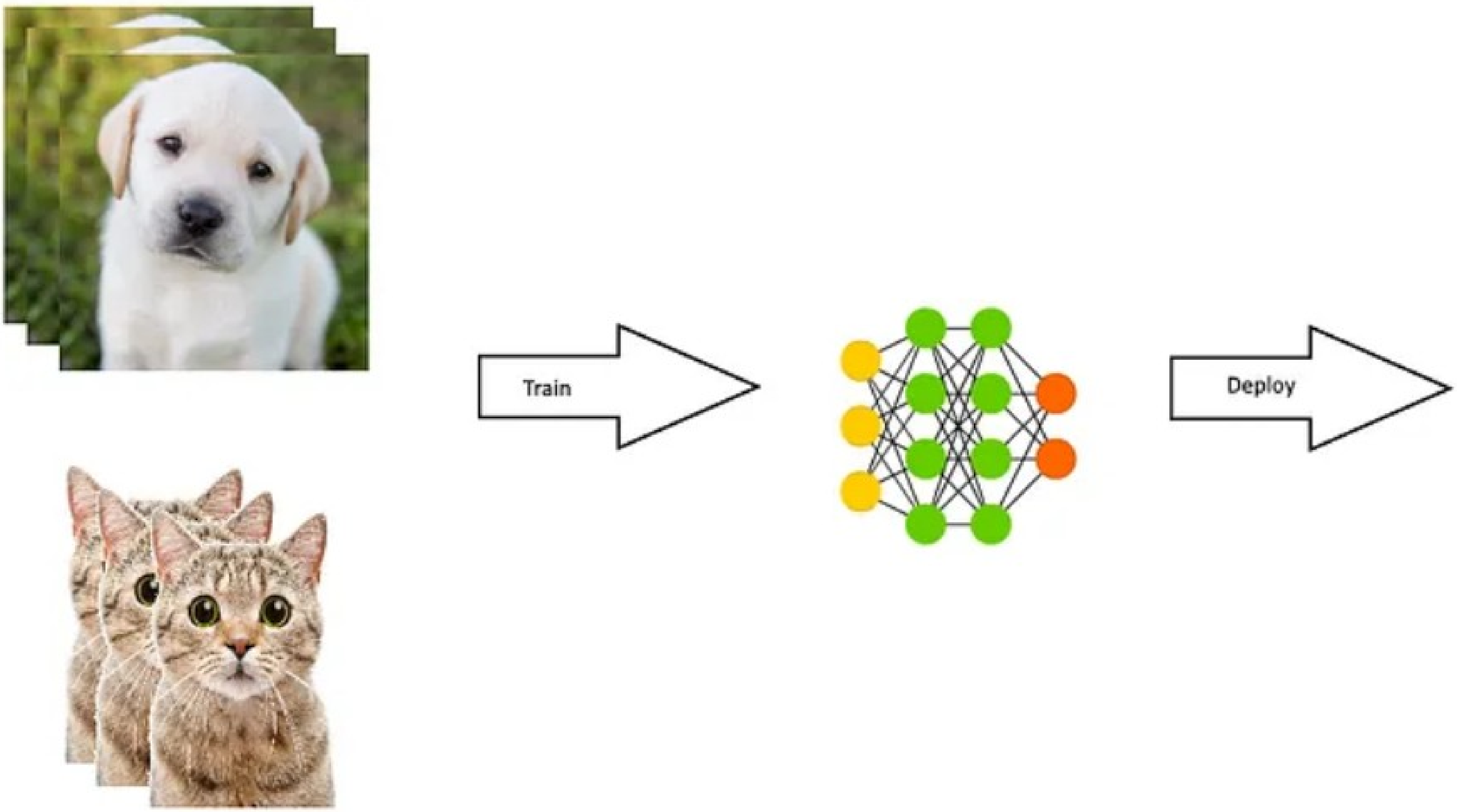
- Lazy learning means no learning before prediction
- We can add classes to a problem easily
- Useful on:
  - many classes problems
  - identification problems
  - novelty detection

One Shot Learning

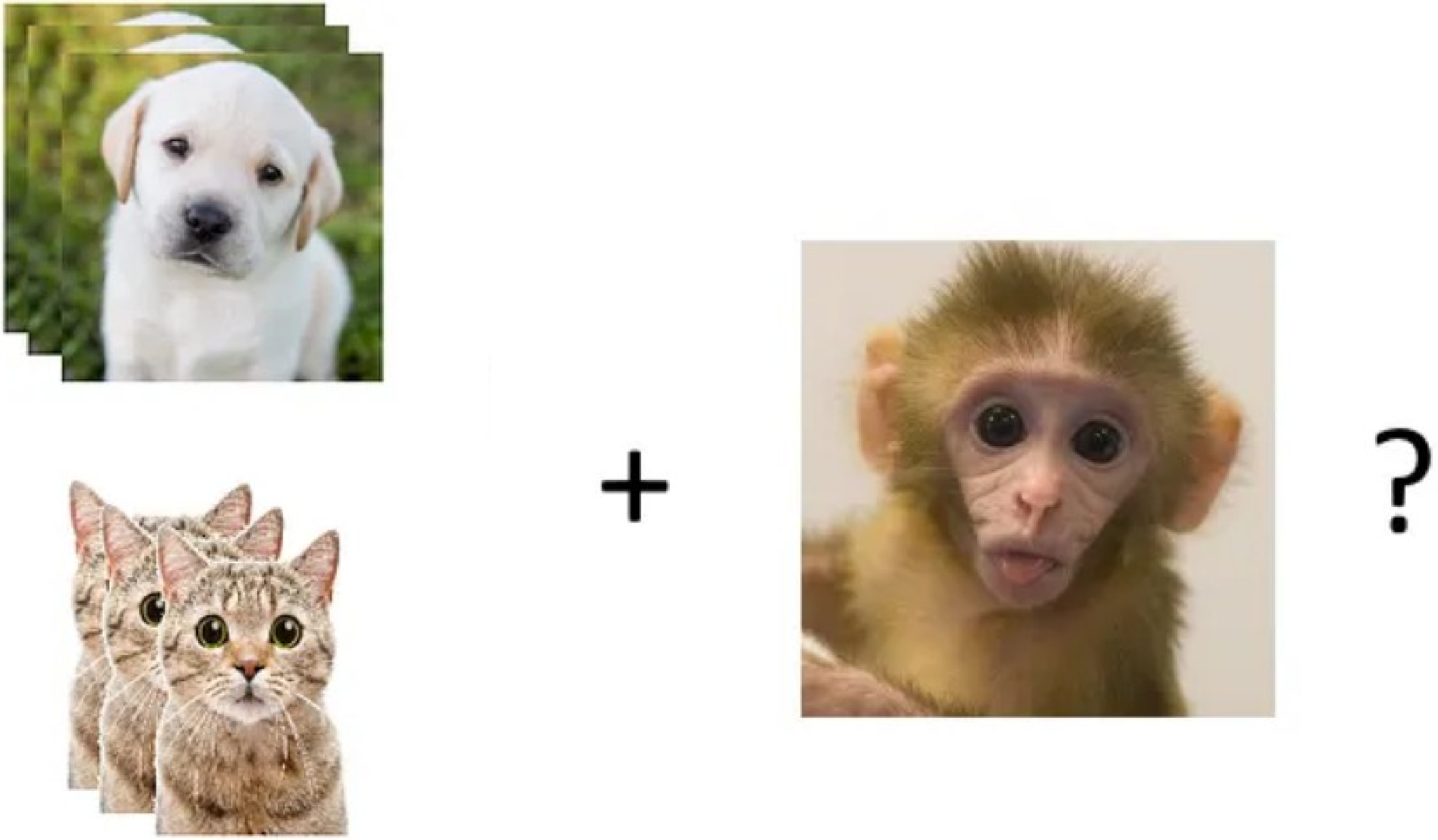
Learning to classify:



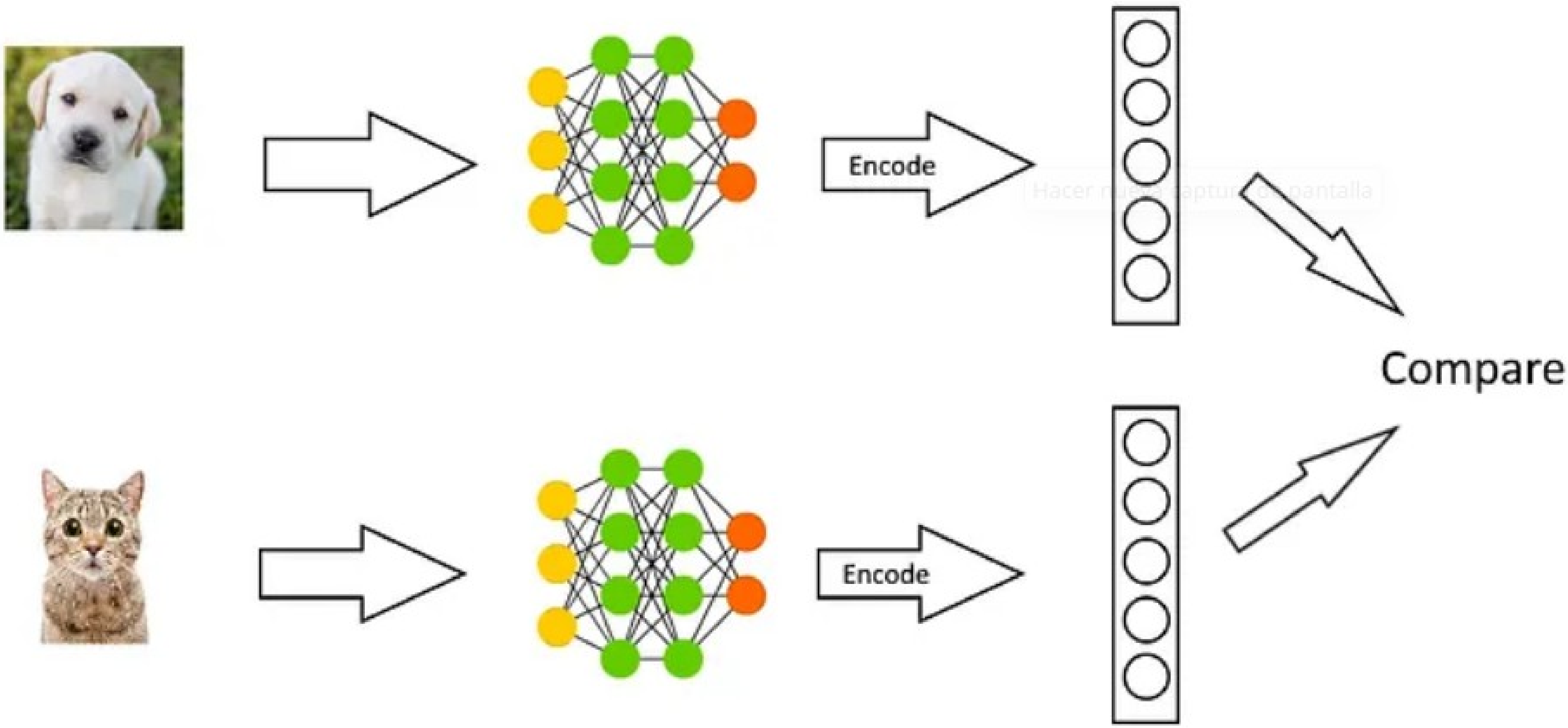
One Shot Learning



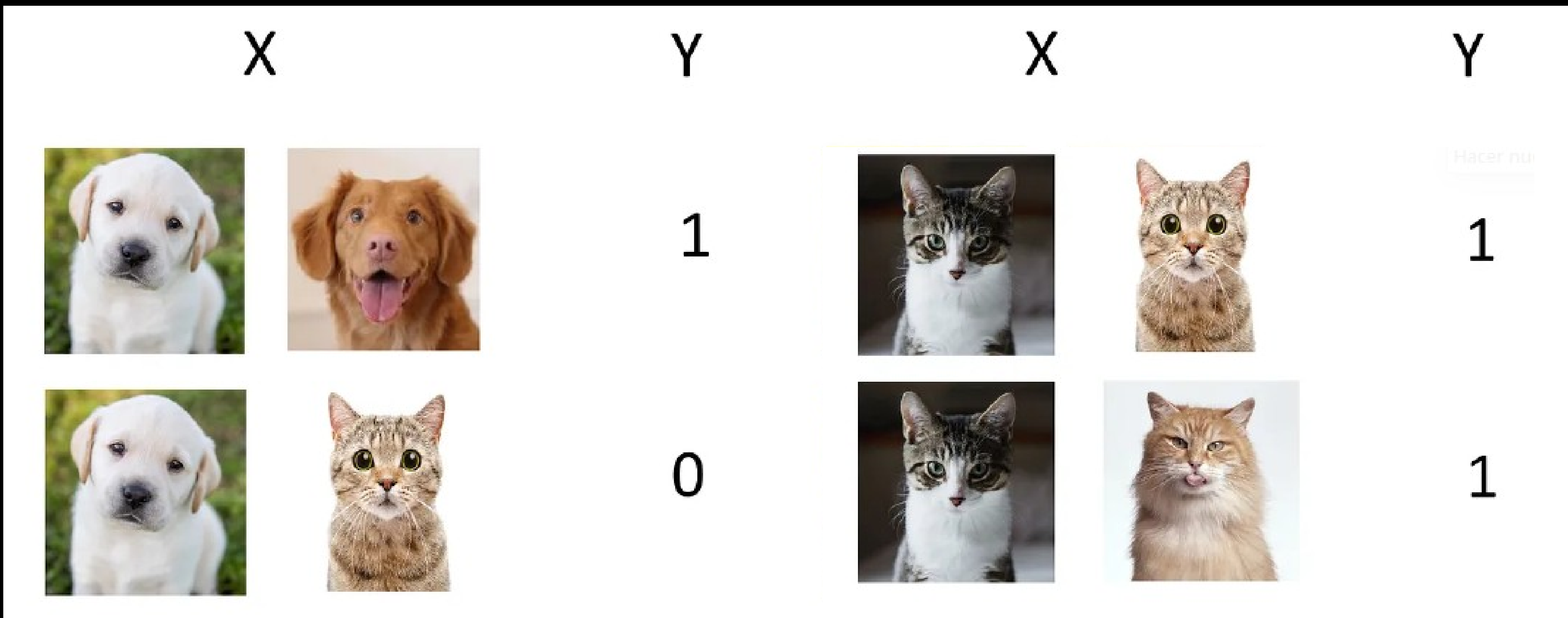
One Shot Learning



Siamese Networks



Metric Learning



Metric Learning

Better strategy:

Metric Learning

$$\text{distance}(A,P) < \text{distance}(A,N)$$

$$\text{distance}(A,P) - \text{distance}(A,N) < 0$$

$$\text{distance}(A,P) - \text{distance}(A,N) + \text{margin} < 0$$

$$\text{Loss} = \max(d(A,P) - d(A,N) + \text{margin}, 0)$$