Application: Few-shot classification

Classification with dynamically changing class labels. Do not need to decide on the set of classes during training time.

Example:

Test - 1

few = 2

Classes: {dog, panther, cat}

x^1, dog

x^2, dog

x^3, panther

x^4, panther

x^5, cat

x^6, cat

x^*

Test - 2

classes: {upma, pongal}

x^1 upma

x^2 upma

x^3 pongal

x^4 pongal

x^* ? upma / pongal ?

Learning distance functions

Many methods have been proposed.

Assume we have supervision on pairs of objects that are similar and dissimilar.

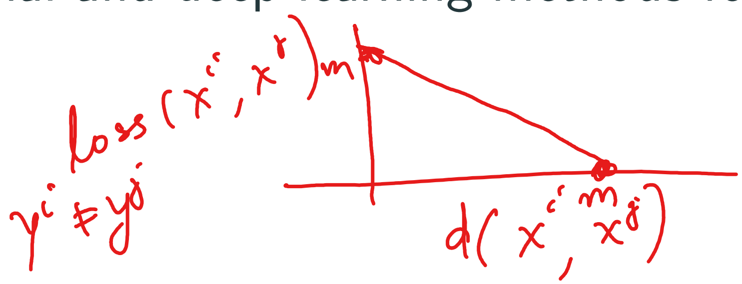
Learn a distance function that brings together similar objects and keeps apart dissimilar objects.

Default: $\min \left[\delta(y_i = y_j)\, d(x^i, x^j) - \delta(y_i \neq y_j)\, d(x^i, x^j) \right]$

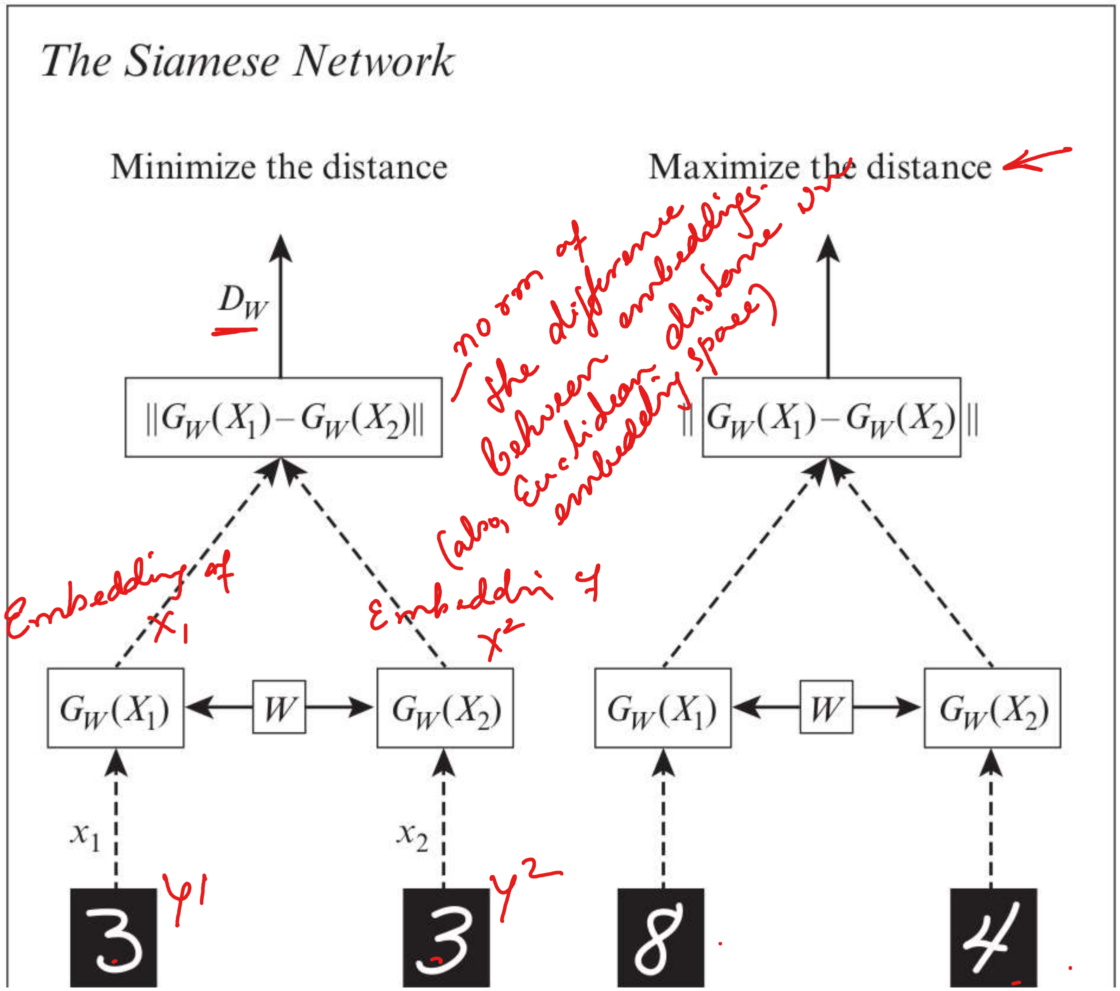
margin = hyper-parameter

$$\min L(\mathbf{x}^i, \mathbf{x}^j) = \delta(y_i = y_j) d(\mathbf{x}^i, \mathbf{x}^j) + \delta(y_i \neq y_j) \max(0, m - d(\mathbf{x}^i, \mathbf{x}^j))$$

Many different models proposed in traditional and deep learning methods for representing distances.



Siamese networks for learning distance functions



$\max(0, m - D(G_w(x^1) - G_w(x^2)))$

if $y^1 \neq y^2$

<u>During training</u>:

Use labeled data to learn $\underline{W}$

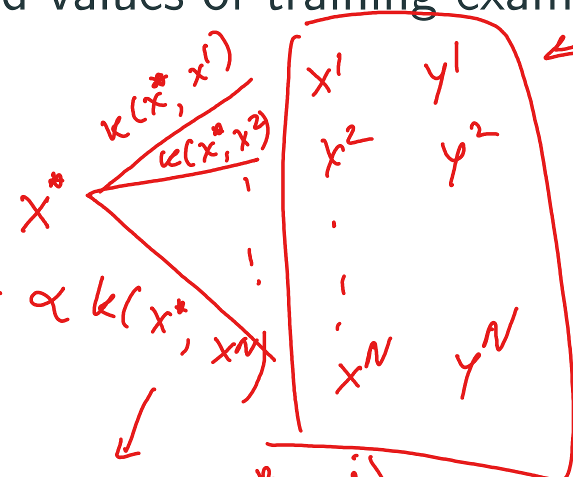
<u>During test-time</u>:

Can support few shot classification with $\|G_w(x^i) - G_w(x^*)\|$ as distance function $d(x^i, x^*)$

Kernel Regression

Predict real-values based on similarity weighted values of training examples.

$$f(\mathbf{x}|D) = \sum_{i=1}^{N} \frac{K(\mathbf{x},\mathbf{x}^i)}{\sum_{j=1}^{N} K(\mathbf{x},\mathbf{x}_j)} y_i$$

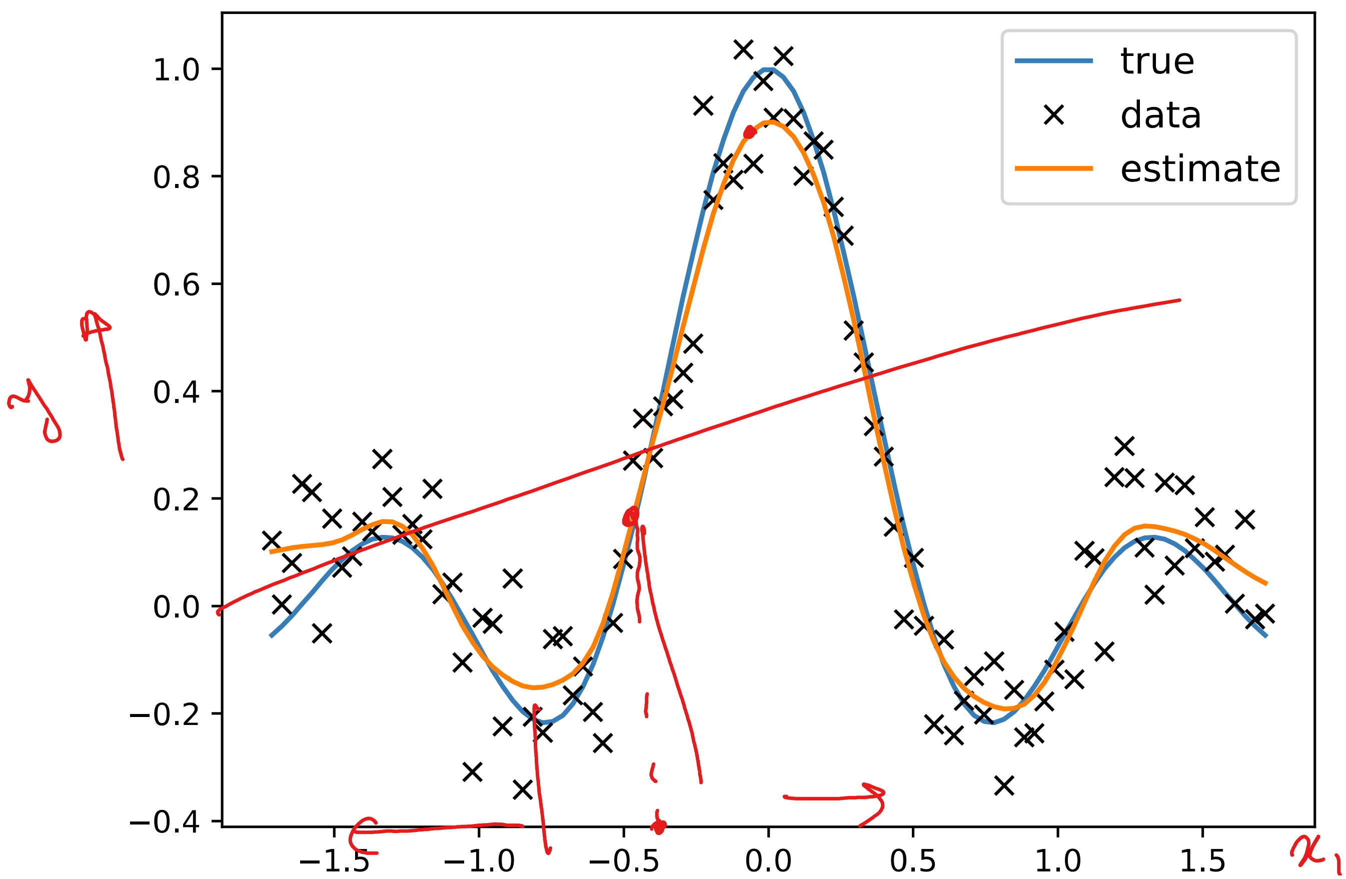


$w_N \propto k(x^*, x^N)$

$$\hat{y} = \sum_{i=1}^{N} w_i y_i$$

$$w_i = \frac{k(x^*, x^i)}{\sum_{j=1}^{N} k(x^*, x^j)}$$

Kernel regression example



$d = 1$

$$k(x^i, x^j) = e^{-\gamma \underbrace{\|x^i - x^j\|^2}_{d(x^i, x^j)^2}}$$

$$\boxed{w^i = \frac{e^{-\gamma\|x^* - x^i\|^2}}{\sum_k e^{-\gamma\|x^* - x^k\|^2}}}$$

if $\gamma = 0$, $\hat{y} = \dfrac{\sum_{i=1}^{N} y_i}{N}$

if $\gamma \to \infty$ $\hat{y} = y_{i^*}$

i^* = nearest example to x^*

Mercer's Kernel function

The (Mercer's) kernel function $K : \mathcal{X} \times \mathcal{X} \mapsto R$ is a symmetric function such that for any set of N points and any choice of numbers $c_i \in R$

$k(x^i, x^j) = k(x^j, x^i)$

$$\sum_{i=1}^{N} \sum_{j=1}^{N} K(\mathbf{x}^i, \mathbf{x}^j) c_i c_j \geq 0$$

Equivalent to matrix of kernel values (Gram) matrix being positive semi-definite.

$$\begin{array}{cc} N & \begin{bmatrix} k(x^i, x^j) \end{bmatrix} \\ & N \end{array}$$

Commonly Used Kernels

- Linear Kernel: $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$
- Polynomial Kernel: $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + 1)^d$ ← degree of polynomial.
- Radial Basis Function (RBF) or Gaussian Kernel:
  $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2)$
- Sigmoid Kernel: $K(x, y) = \tanh(\alpha x^T y + c)$

Kernel embedding

For every valid Mercer's Kernel, there exists an embedding function $\phi(\mathbf{x})$ such that $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}).\phi(\mathbf{x}')$ (Mercer's theorem)

(1cm) Example: Quadratic Kernel: $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}.\mathbf{x}')^2$.

If original data is 2-d, then $K(\mathbf{x}, \mathbf{x}') = (x_1 x_1' + x_2 x_2')^2 = (x_1 x_1')^2 + (x_2 x_2')^2 + 2 x_1 x_1' x_2 x_2'$

$\phi(\mathbf{x}) = [x_1^2, \sqrt{2} x_1 x_2, x_2^2] \in R^3$

$\phi(x') = [x_1'^2, \sqrt{2} x_1' x_2', x_2'^2] \in R^3$

$= x_1^2 \cdot x_1'^2 + x_2^2 \cdot x_2'^2 + (\sqrt{2} x_1 x_2) \cdot (\sqrt{2} x_1' x_2')$

$\overset{?}{=}$

$$\boxed{\phi(x) \cdot \phi(x') = x_1^2 \cdot x_1'^2 + (\sqrt{2} x_1 x_2)(\sqrt{2} x_1' x_2') + x_2^2 \cdot (x_2')^2}$$

Applications in Machine Learning

- Support Vector Machines (SVM)
- Kernel Principal Component Analysis (PCA)
- Gaussian Processes
- Radial Basis Function Networks (RBFN)

Support Vector Machines

Non-probabilistic models for classification of the form:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}, \mathbf{x}^i) + w_0$$

Learning fixes values of α_i. These are chosen in such a way that only a few is have non-zero α_i. These are called support vectors.

Assume $y_i \in \{-1, +1\}$, binary classification model.

Predicted class label $= \text{sign}(f(\mathbf{x}))$