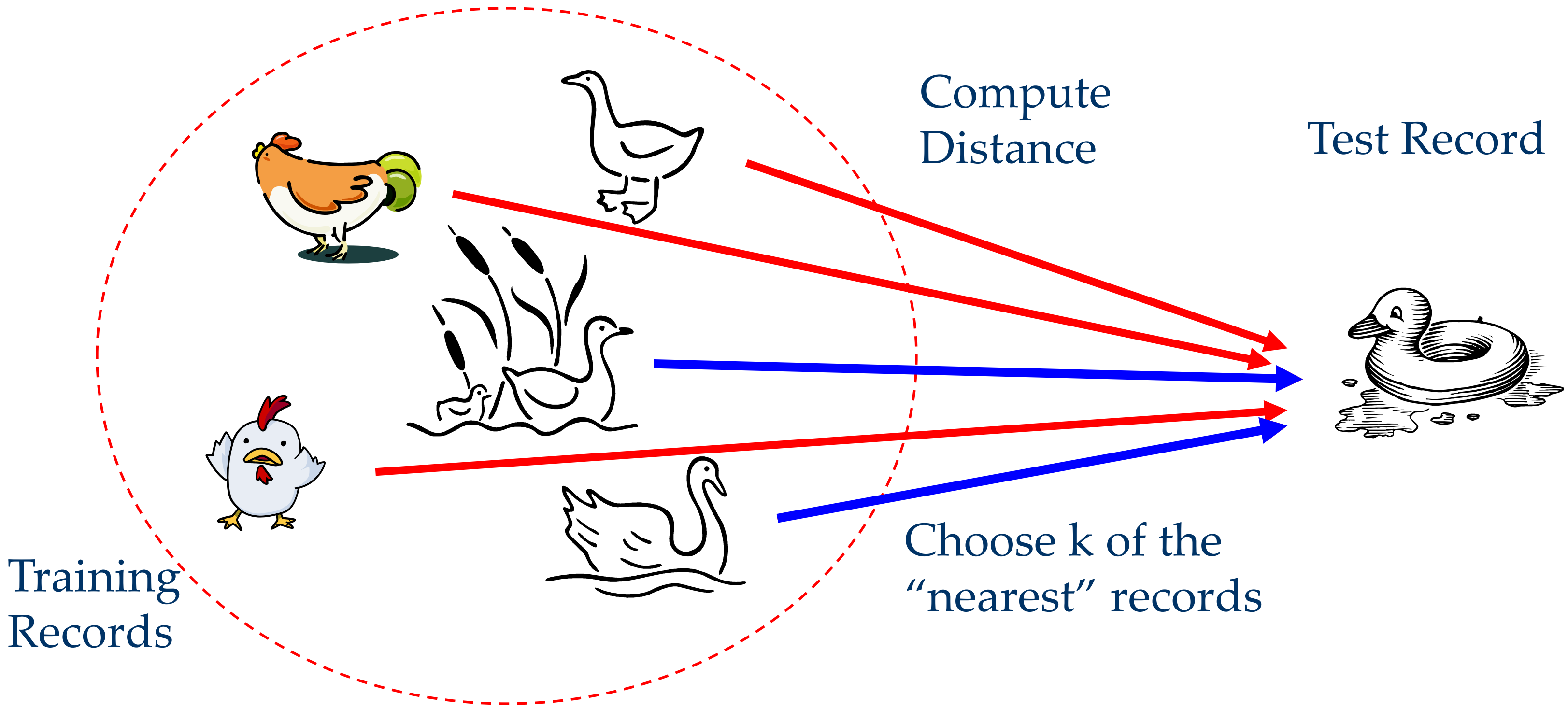
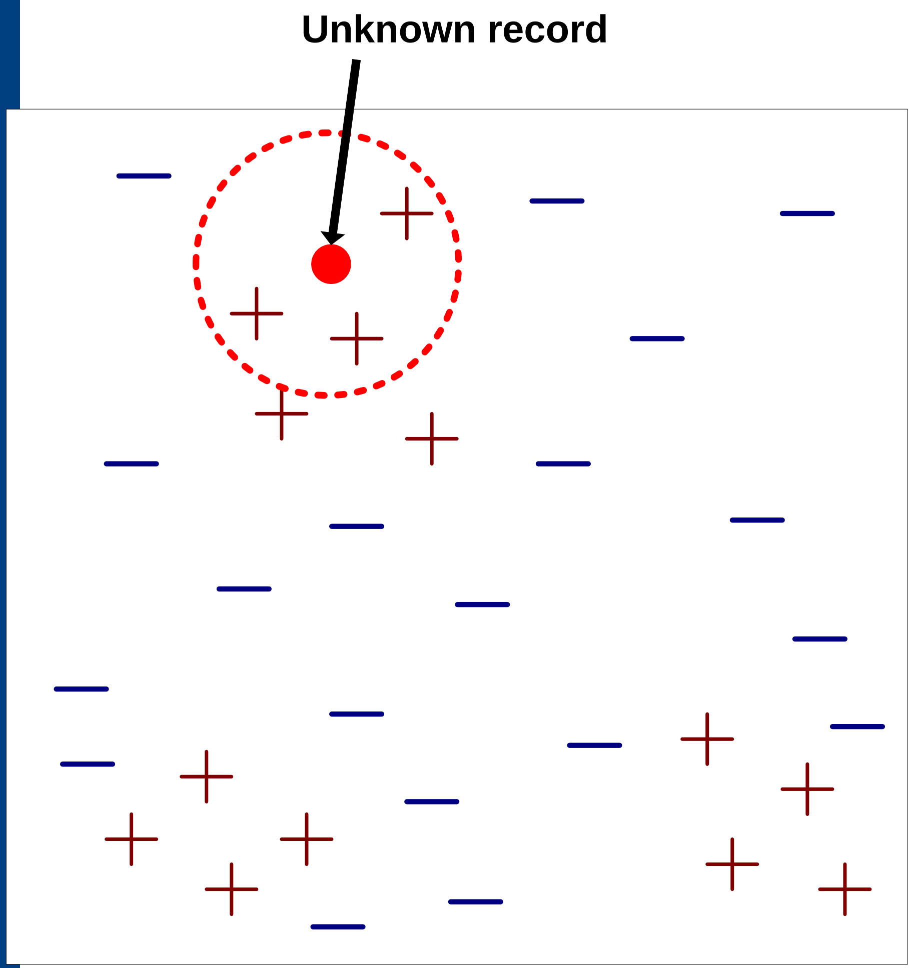
k Nearest Neighbor Classifier

Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



Nearest-Neighbor Classifiers



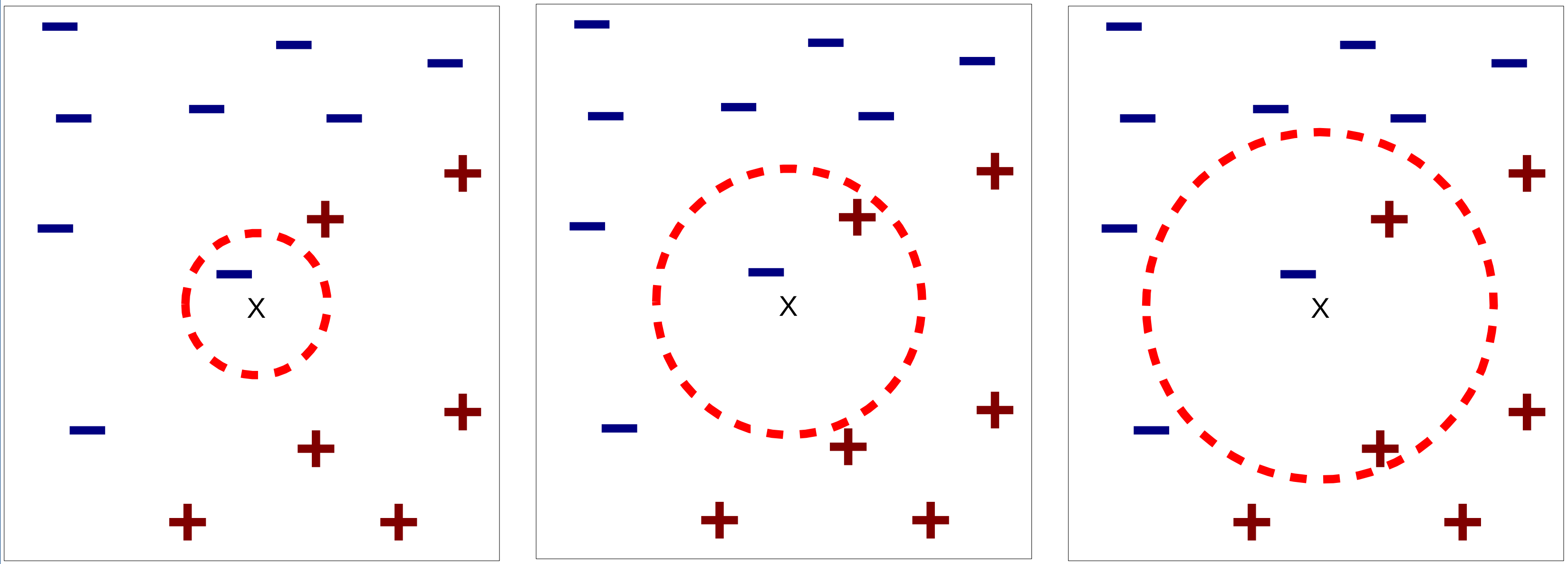
Requires three things

- The set of stored records
- Distance Metric to compute distance between records
- The value of *k*, the number of nearest neighbors to retrieve

To classify an unknown record:

- Compute distance to other training records
- Identify *k* nearest neighbors
- Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record $\boldsymbol{x}$ are data points that have the k smallest distance to $\boldsymbol{x}$

How many parameters in kNN?

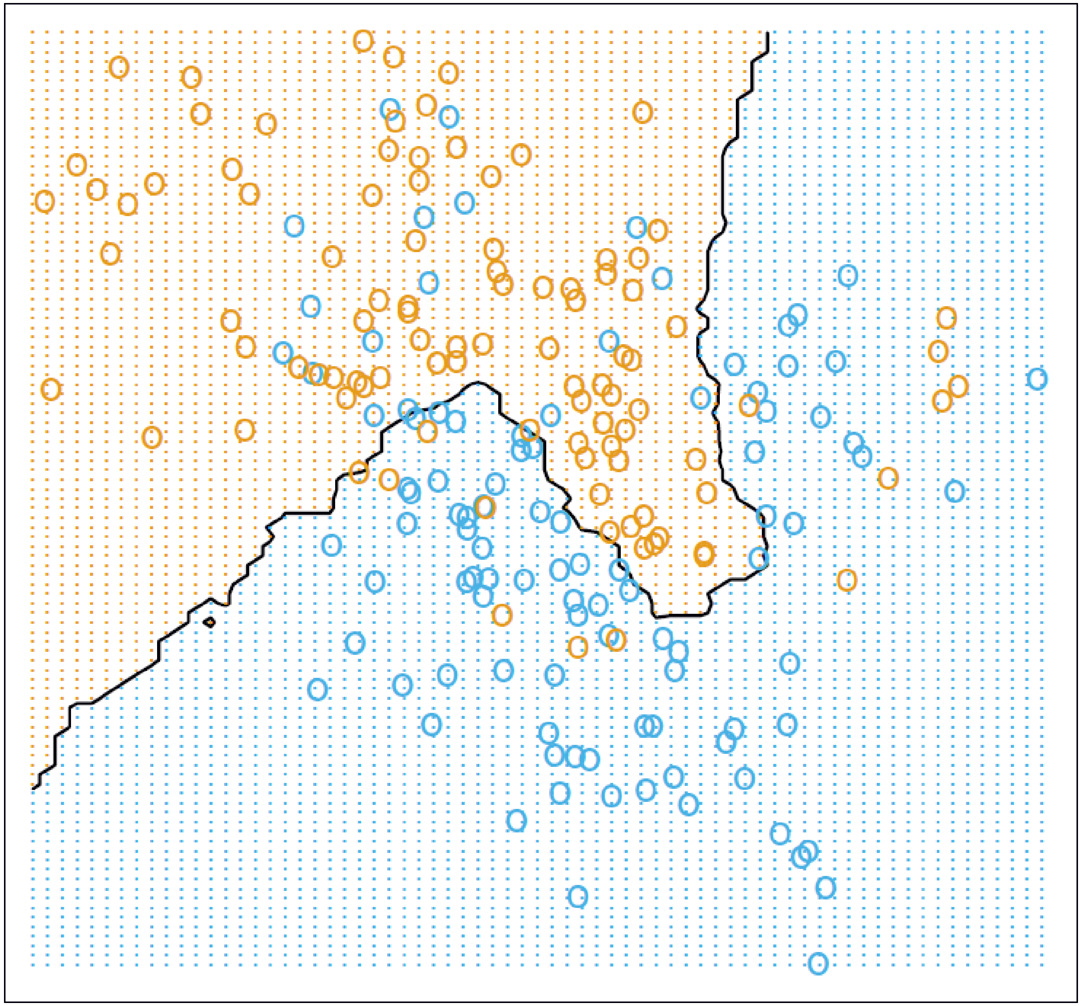
- A Linear Classifier

  $$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$$

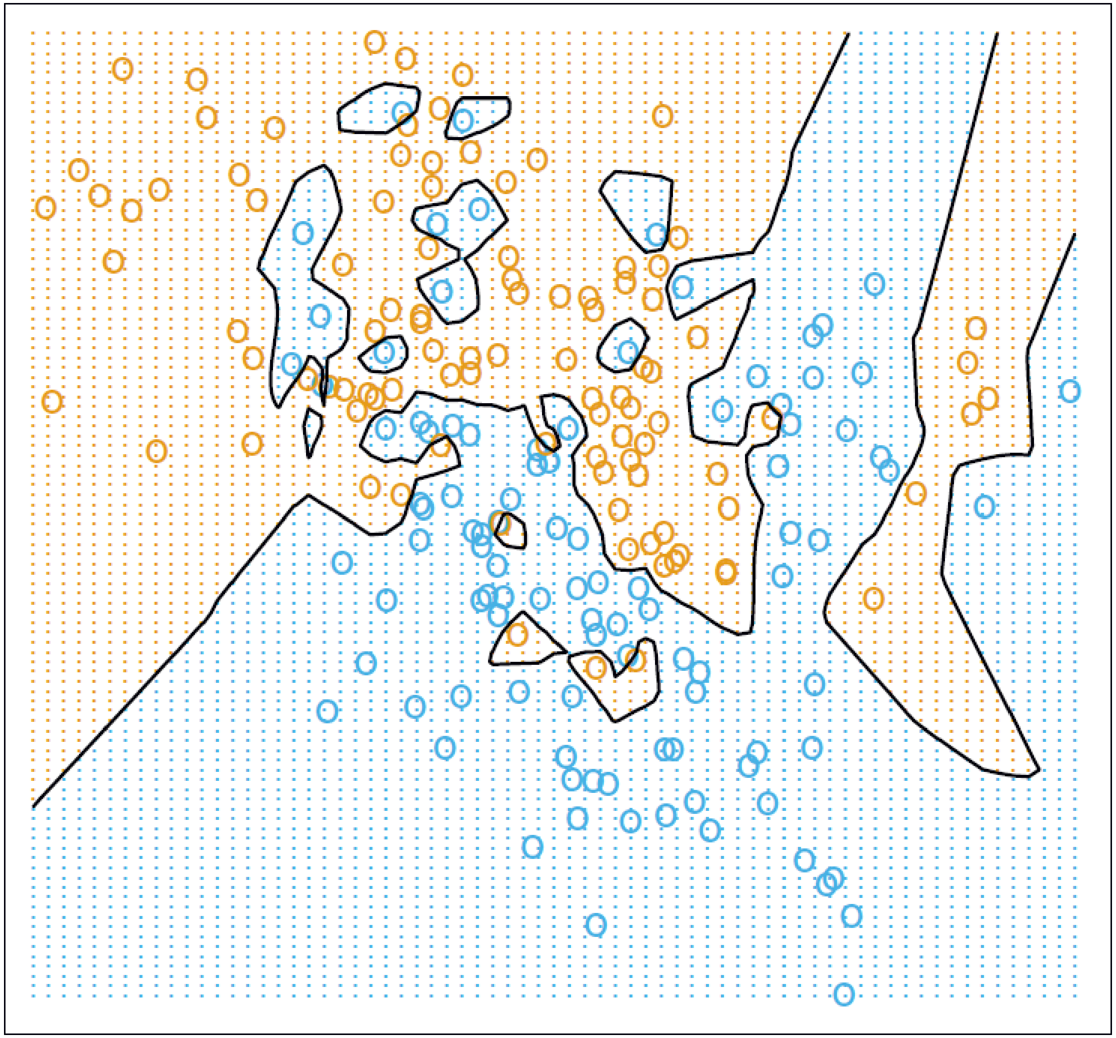
  - The number of parameters?

- kNN Classier
  - The number of parameters?

15-Nearest Neighbor Classifier



1-Nearest Neighbor Classifier



How many parameters in kNN?

- A Linear Classifier

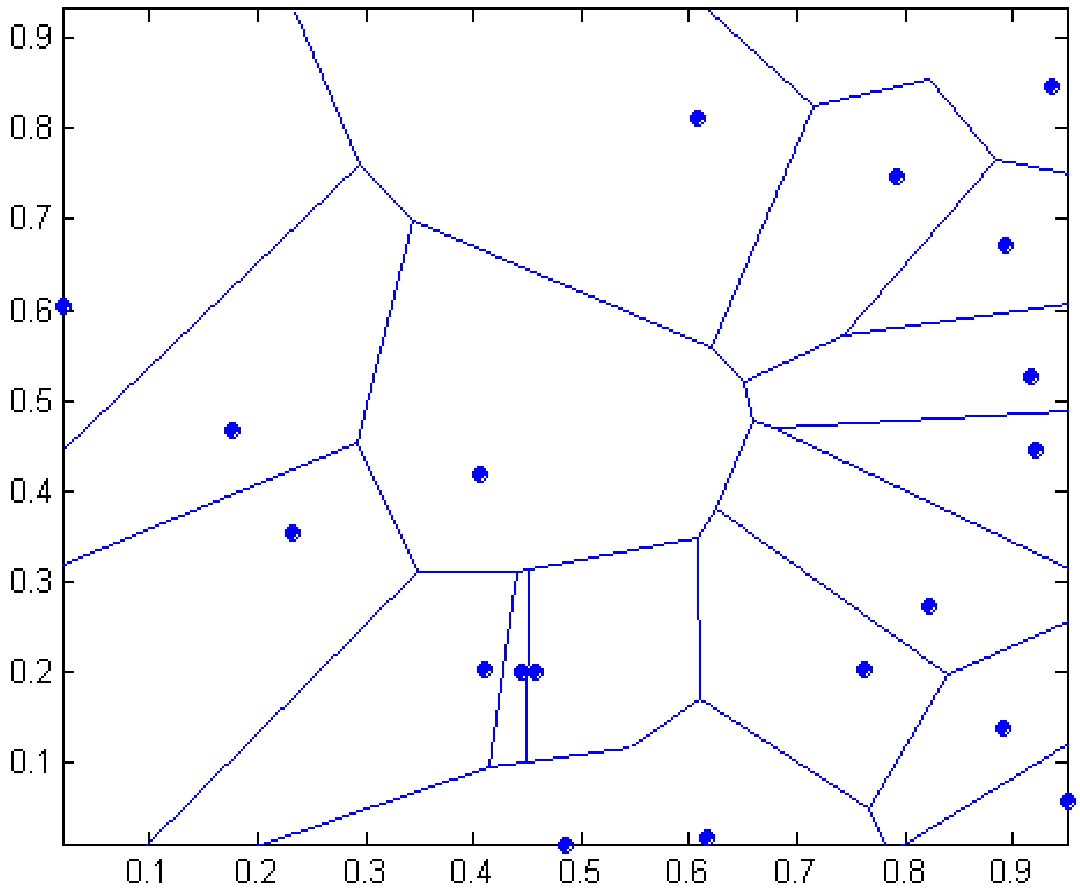
$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$$

  - The number of parameters?

- kNN Classifier
  - Effective number of parameters?

$$\frac{N}{k}$$

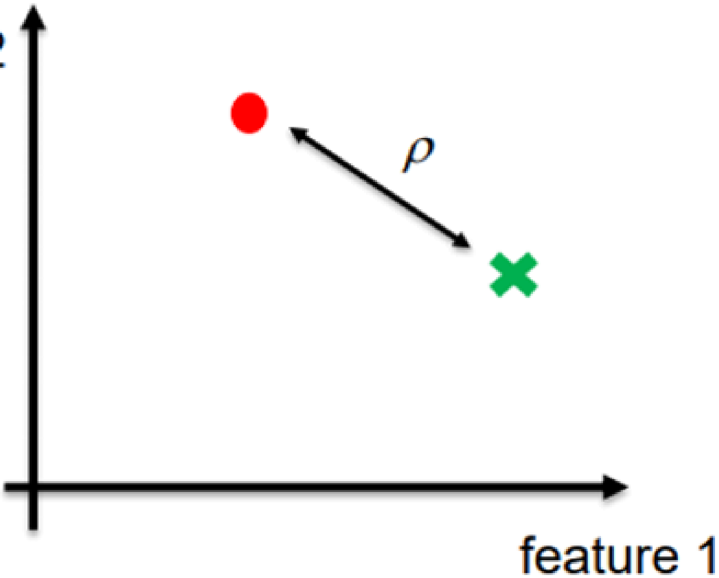
1 nearest-neighbor



Voronoi diagram
(tessellation)

Metric Learning

How to compare objects?



$$\rho(x,y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_d - y_d)^2}$$

$$= \sqrt{\left(\begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \end{pmatrix}\right)^T \left(\begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \end{pmatrix}\right)}$$

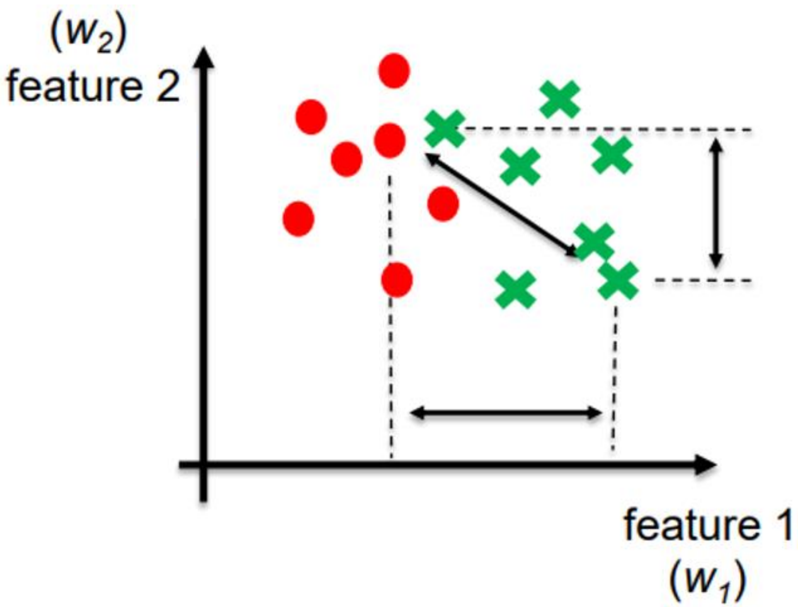
$$= \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T (\boldsymbol{x} - \boldsymbol{y})}$$

- Not all features are equally important.

Problems

$$\rho(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x}-\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y})}$$

- Not all features are equally important.
- Often some features are noisy or uninformative.



What if we already know feature 1 is discriminative than f2 as a prior ?

$$\rho(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{w_1^2(x_1-y_1)^2 + w_2^2(x_2-y_2)^2}$$

$$= \sqrt{\left(\begin{bmatrix} w_1 & 0 \\ 0 & w_2 \end{bmatrix}\left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}\right]\right)^T \left(\begin{bmatrix} w_1 & 0 \\ 0 & w_2 \end{bmatrix}\left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}\right]\right)}$$

$$= \sqrt{(\boldsymbol{x}-\boldsymbol{y})^T[\boldsymbol{w}^T\boldsymbol{w}](\boldsymbol{x}-\boldsymbol{y})}$$

$$= \sqrt{(\boldsymbol{x}-\boldsymbol{y})^T M(\boldsymbol{x}-\boldsymbol{y})}$$

However, we didn't know this prior in a real problem. That is what metric learning does: **To learn a metric M that better distinguishes different classes given the real data.**

Metric Learning

- Given data of interest, learn a metric (M), which helps in the prediction task.

$$\rho_M(x,y) = \sqrt{(x-y)^T M (x-y)}$$

- Basic optimization:
  - Let's create two sets of pairs: similar set S, dissimilar set D. The targets is to find M such that:

$$\rho_M(x,x') \textit{ large for } (x,x') \in D$$

$$\rho_M(x,x') \textit{ small for } (x,x') \in S$$

  - Create cost/energy function $\mathcal{L}(M)$ and minimize with respect to M

$$\mathcal{L}(M) = \sum_{(x,x')\in S} \rho_M^2(x,x') - \lambda \sum_{(x,x')\in D} \rho_M^2(x,x')$$

MMC (From Lagrange Perspective)

- Optimization target: $\min_{M\in\mathbb{R}^{d\times d}, M\succcurlyeq 0} \sum_{(x_i,x_j)\in S} \rho_M^2(x_i,x_j)$ (1)

$$\text{s.t.} \sum_{(x_i,x_j)\in D} \rho_M(x_i,x_j) \geq 1 \qquad (2)$$

(a). Add $-\log(*)$ to (2) and make it a standard inequivalent constrained Lagrange.

$$-\log \sum_D \rho_M(x_i,x_j) \leq 0 \quad (3)$$

(b). Make the optimization target in Lagrange Multiplier formulation:

$$L(M,\lambda) = \sum_S \rho_M^2(x_i,x_j) - \lambda \log \sum_D \rho_M(x_i,x_j)$$

with $M \succcurlyeq 0$ PSD constraints and $\lambda \geq 0$ $et.al$ KKT constraints

(c). According to Lagrange Formulation:

minimize Eq (1) subject to Eq (2) is equivalent to

$$\min_{M\succcurlyeq 0} \max_{\lambda \geq 0} L(M,\lambda)$$

and its dual equation

$$\max_{\lambda\geq 0} \min_{M\succcurlyeq 0} L(M,\lambda)$$

MMC (From Lagrange Perspective)

$$L(M,\lambda) = \sum_S \rho_M^2(x_i, x_j) - \lambda \log \sum_D \rho_M(x_i, x_j) \quad (4)$$

$$\max_{\lambda \geq 0} \min_{M \succcurlyeq 0} L(M,\lambda) \quad (5)$$

Write (4) in its original form:

$$\sum_S (x_i - x_j)^T M (x_i - x_j) - \lambda \log \sum_D \sqrt{(x_i - x_j)^T M (x_i - x_j)}$$

Calculate partial derivative of L respect to M and set it to 0:

$$\frac{\partial L(M,\lambda)}{\partial M} = \sum_S (x_i - x_j)(x_i - x_j)^T - \frac{\lambda}{2} \sum_D \frac{(x_i - x_j)(x_i - x_j)^T}{\sqrt{(x_i - x_j)^T M (x_i - x_j)}\ \Sigma_D \left(\sqrt{(x_i - x_j)^T M (x_i - x_j)}\right)} = 0$$

Rewrite above Equation,

$$\sum_S (x_i - x_j)(x_i - x_j)^T - \frac{1}{2} \sum_D \frac{(x_i - x_j)(x_i - x_j)^T}{\sqrt{(x_i - x_j)^T \frac{M}{\lambda} (x_i - x_j)}\ \Sigma_D \left(\sqrt{(x_i - x_j)^T \frac{M}{\lambda} (x_i - x_j)}\right)} = 0$$

MMC (From Lagrange Perspective)

$$L(M,\lambda) = \sum_S \rho_M^2(x_i,x_j) - \lambda \log \sum_D \rho_M(x_i,x_j) \quad (4)$$

$$\max_{\lambda \geq 0} \min_{M \succcurlyeq 0} L(M,\lambda) \quad (5)$$

(e). Now we know $\frac{M}{\lambda}$ is a constant Matrix A, the following steps is the same as solutions in solving Lagrange Equation.

We use $M = \lambda A$ to replace M in (5) and solve the optimal λ.

Then we can find optimal M by using $M = \lambda A$

An interesting thing we can derive from Lagrange perspective is that minimizing (1) is equivalent to

$$\min_{M \succcurlyeq 0} \sum_{(x_i,x_j)\in S} \rho_M^2(x_i,x_j) - \cancel{\lambda}\log\left(\sum_{(x_i,x_j)\in D} \rho_M(x_i,x_j)\right)$$

up to a multiplication of M by a positive const. It makes one of the constraints into the optimization target.

MMC (From Newton Perspective)

$$\min_{M\succcurlyeq 0} F(M) = \sum_{(x_i,x_j)\in S} \rho_M^2(x_i,x_j) - \log\left(\sum_{(x_i,x_j)\in D} \rho_M(x_i,x_j)\right)$$

Recap the unconstrained Newton Method to solve multi-dimensional x

(a) Taylor expansion of optimization target $F(x)$ with last state x^k

$$F(x) = F\left(x^{(k)}\right) + \nabla F\left(x^{(k)}\right)^T \Delta x + \frac{1}{2}\Delta x^T \nabla^2 F\left(x^{(k)}\right)\Delta x + \cdots$$

where $\Delta x = x - x^{(k)}$.

(b) Let $g_k = \nabla F\left(x^{(k)}\right), H_k = \nabla^2 F\left(x^{(k)}\right)$. (Hessian Matrix)

Minimization $F(x)$ with respect to Δx is equivalent to

$$g_k + H_k\left(x - x^{(k)}\right) = 0$$

(c) The iteration becomes

$$x^{(k+1)} = x^{(k)} - H_k^{-1} g_k$$

[Distance Metric Learning, with Application to Clustering with Side-Information, Xing et al., 2002]

MMC (From Newton Perspective)

$$\min_{M\succcurlyeq 0} F(M)=\sum_{(x_i,x_j)\in S}\rho_M^2(x_i,x_j)-\log\left(\sum_{(x_i,x_j)\in D}\rho_M(x_i,x_j)\right)$$

Use the diagonal M=diag($m_{11}, m_{22}, \dots, m_{dd}$) as a simple example

Recap the Newton Method to solve Multi-dimensional $\boldsymbol{x}$

(d) In diagonal MMC, the updates changes to ensure $M \succcurlyeq 0$.

$$M^{(k+1)}=M^{(k)}-H_k^{-1}g_k \rightarrow M^{(k+1)}=M^{(k)}-\alpha H_k^{-1}g_k$$

$$g_k=\nabla F\left(M^{(k)}\right), H_k=\nabla^2 F\left(M^{(k)}\right)$$

where α is a step-size parameter optimized via a line-search to give the largest downhill step subject to $m_{ii}\geq 0$.

And the simplest line-search method is bisection.

[Distance Metric Learning, with Application to Clustering with Side-Information, Xing et al., 2002]

MMC (From Projection Iteration Perspective)

What if the case of full M (require O(d^6) to invert H in Newton)

(a). Pose the equivalent problem:

$$\max_{M\in\mathbb{R}^{d\times d}, M\succcurlyeq 0} g(M) = \sum_{(x_i,x_j)\in D} \rho_M(x_i,x_j)$$

$$\text{s.t.} \quad f(M) = \sum_{(x_i,x_j)\in S} \rho_M^2(x_i,x_j) \le 1,$$

.

(b). Construct two iterative projection sets C_1, C_2

$$C_1 = \{M: \ f(M) \le 1\}, C_2 = \{M: M \succcurlyeq 0\}$$

(c). Solving full M by iterative projection method

Iterate

Iterate

$$M = \arg\min_{M'}\{||M' - M||_F: M' \in C_1\}$$

$$M = \arg\min_{M'}\{||M' - M||_F: M' \in C_2\}$$

Until M **converges**

$$M = M + \alpha\left(\nabla_M g(M)\right)$$

Until M **converges**

[Distance Metric Learning, with Application to Clustering with Side-Information, Xing et al., 2002]

MMC (From Projection Iteration Perspective)

(1) Why it is equivalent:

$$\min_{M\in\mathbb{R}^{d\times d}, M\succcurlyeq 0} \sum_{(x_i,x_j)\in S} \rho_M^2(x_i,x_j) \qquad \max_{M\in\mathbb{R}^{d\times d}, M\succcurlyeq 0} g(M) \sum_{(x_i,x_j)\in D} \rho_M(x_i,x_j)$$

$$\text{s.t.} \sum_{(x_i,x_j)\in D} \rho_M(x_i,x_j) \geq 1, \quad ==> \quad \text{s.t.} \quad f(M) = \sum_{(x_i,x_j)\in S} \rho_M^2(x_i,x_j) \leq 1,$$

They are actual the same from Lagrange perspective.

They are equivalent to (if you apply log(*) on optimization target)

$$L(M,\lambda) = \log \sum_S \rho_M^2(x_i,x_j) - \log \sum_D \rho_M(x_i,x_j)$$

(2) Why need to pose this equivalent equation? (Related to projection set)

$$C_1 = \{M:\ f(M) \leq 1\}, C_2 = \{M: M \succcurlyeq 0\}$$

To make it easier to find projection set C_1 (simply linear system)

[Distance Metric Learning, with Application to Clustering with Side-Information, Xing et al., 2002]

MMC (From Projection Iteration Perspective)

$$\max_{M\in\mathbb{R}^{d\times d}, M\succcurlyeq 0} g(M) \sum_{(x_i,x_j)\in D} \rho_M(x_i,x_j)$$

$$\text{s.t.} \quad f(M) = \sum_{(x_i,x_j)\in S} \rho_M^2(x_i,x_j) \le 1,$$

(1) Projection M on to C_1 and C_2 can be done inexpensively.

$$C_1 = \{M: \ f(M) \le 1\}, C_2 = \{M: M \succcurlyeq 0\}$$

(2) Iterative projection is also fast.

Iteration (a):

$$M = \arg\min_{M'}\{||M' - M||_F: M' \in C_1\}$$

Find M by solving a sparse system of linear equations in O(d^2)

Iteration (b):

$$M = \arg\min_{M'}\{||M' - M||_F: M' \in C_2\}$$

Find the diagonalization $M = X^T \Lambda X$ with eigenvalues $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_d)$ and eigenvectors. Then $M' = X^T \Lambda' X$ where $\Lambda' = \mathrm{diag}(\max\{0, \lambda_1\}, \dots, \{0, \lambda_d\})$

[Distance Metric Learning, with Application to Clustering with Side-Information, Xing et al., 2002]

MMC (Mahalanobis Metric for Clustering)

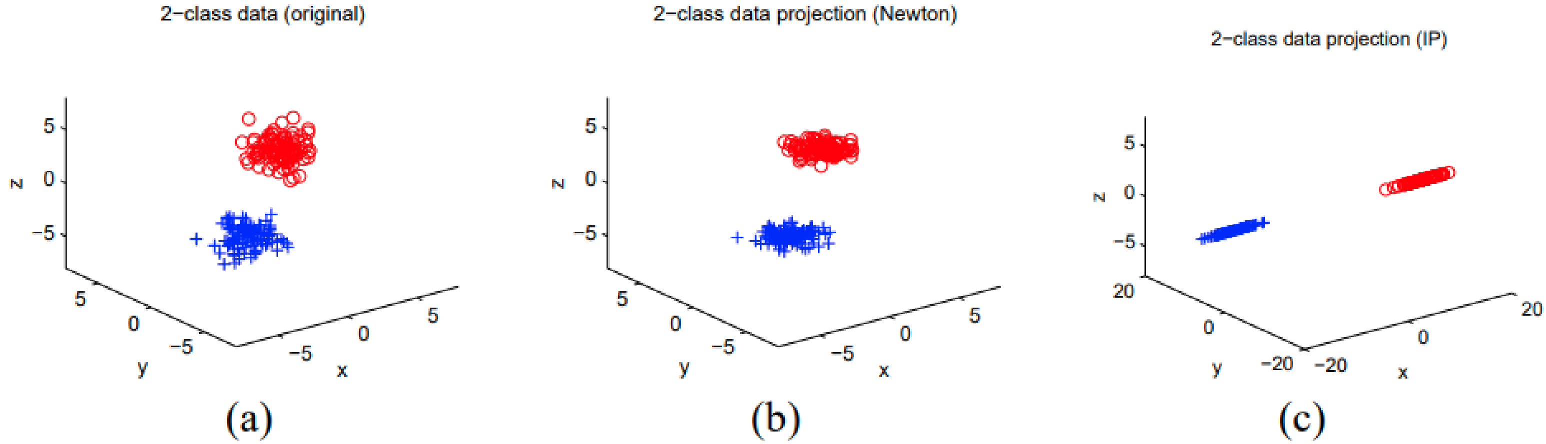


Figure 2: (a) Original data, with the different classes indicated by the different symbols (and colors, where available). (b) Rescaling of data corresponding to learned diagonal A. (c) Rescaling corresponding to full A.

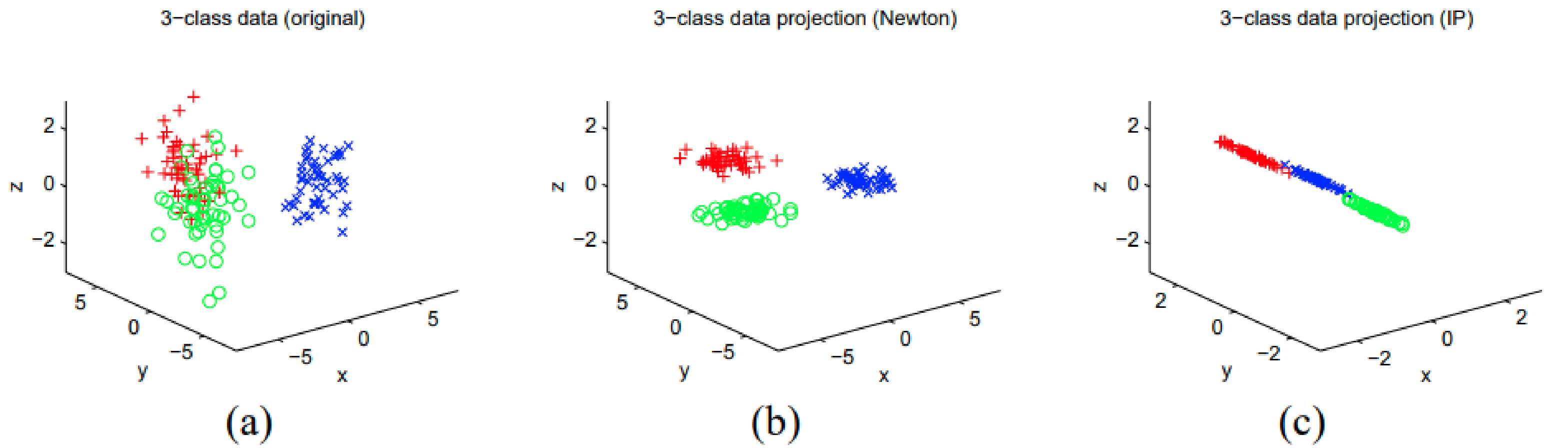
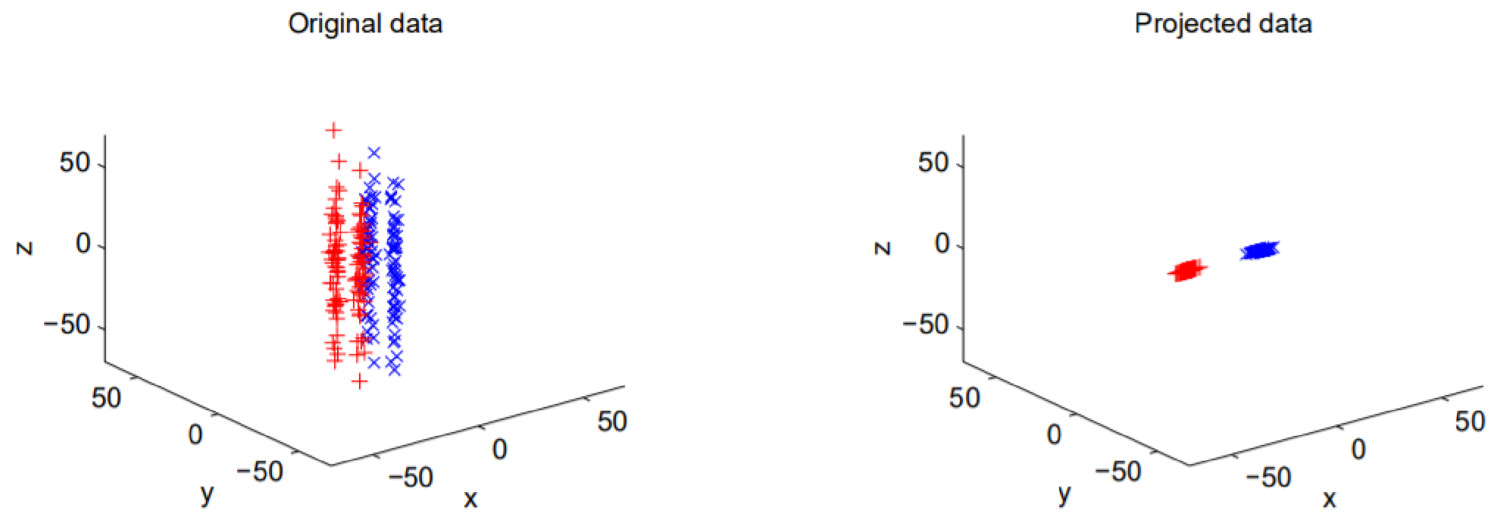


Figure 3: (a) Original data. (b) Rescaling corresponding to learned diagonal A. (c) Rescaling corresponding to full A.

[Distance Metric Learning, with Application to Clustering with Side-Information, Xing et al., 2002]

Empirical Performance for MMC



(a) (b)

1. K-means: Accuracy = 0.4993
2. Constrained K-means: Accuracy = 0.5701
3. K-means + metric: Accuracy = 1
4. Constrained K-means + metric: Accuracy = 1

Figure 5: (a) Original dataset (b) Data scaled according to learned metric. (A_{diagonal}'s result is shown, but A_{full} gave visually indistinguishable results.)

[Distance Metric Learning, with Application to Clustering with Side-Information, Xing et al., 2002]