Lecture 20 EE 559 Apr 3, 2024

Machine Learning I: Supervised Methods

B. Keith Jenkins

Announcements

- Slido event code: 2198501
- Homework 7 will be posted

Reading

- Bishop 1.5
  - Bayes decision theory
- Bishop 2.3.0 (mostly review)

Today's lecture

- Radial basis function (RBF) networks
  - ANN architecture
  - Learning algorithms
  - Tips for implementing RBF networks
  - ANN and non-neural viewpoints
- K-means clustering
  - Example for finding cluster centers
- Complexity: d.o.f. and constraints in ANN (as time permits)

deferred

* - Error corrected on p.7 post-lecture.

Radial Basis Function (RBF) Networks

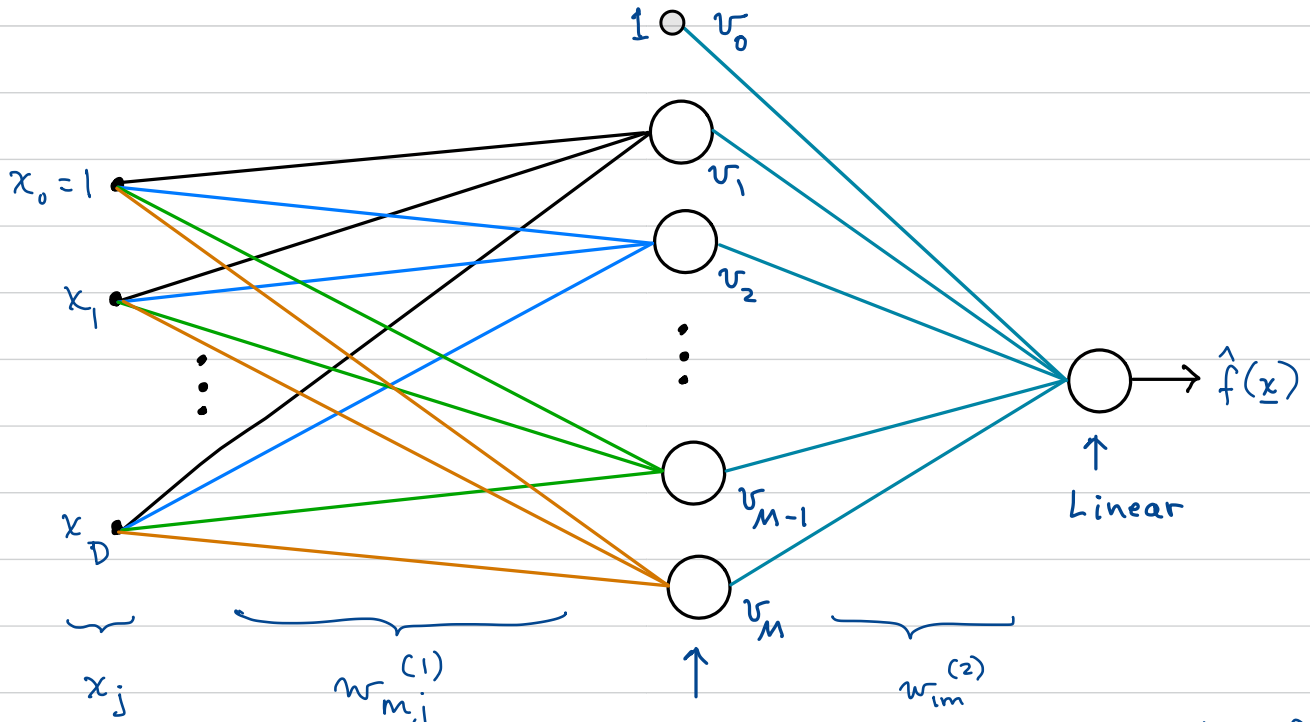
Similar to the universal function approximation network, except uses Radial Basis Functions for "interpolation" instead of piecewise-linear functions. There are a variety of techniques to reduce the # hidden units.

Suppose we use a nonlinear transformation of feature space using basis functions

$$v_m = \phi_m(\underline{x}), \quad m = 1, 2, \cdots, M$$

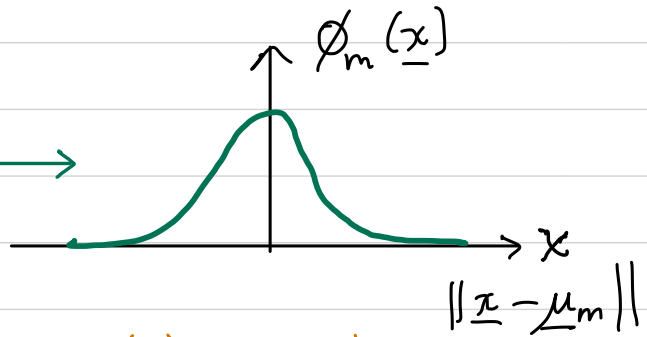
in which the basis functions ϕ_m are "radial": ϕ_m is a function of $\|\underline{x} - \underline{\mu}_m\|$ only, in which $\underline{\mu}_m$ is the m^{th} "prototype" or m^{th} basis-function "centroid"

Fig. 1



Common choice of $\phi_m \Rightarrow$

(1)
$$\begin{cases} \text{For } m = 1, 2, 3, \cdots, M: \\ v_m(\underline{x}) = \phi_m(\underline{x}) = \exp\{-\gamma_m \|\underline{x} - \underline{\mu}_m\|^2\}, \ \gamma_m > 0. \\ \underline{\mu}_m = m^{\text{th}} \text{ basis-function centroid} \\ \hat{f}(\underline{x}) = \sum_{m=0}^{M} w_{im}^{(2)} \phi_m(\underline{x}) \end{cases}$$



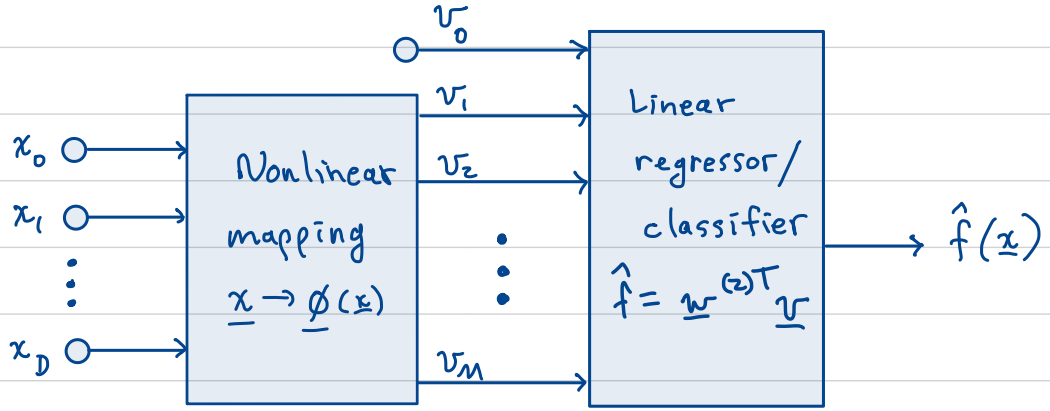
$\phi_m(\underline{x})$ represents similarity between $\underline{x}$ and $\underline{\mu}_m$.

$$\Rightarrow \hat{f}(\underline{x}) = \sum_{m=1}^{M} w_{im}^{(2)} \exp\{-\gamma_m \|\underline{x} - \underline{\mu}_m\|^2\} + w_{i0}^{(2)}$$

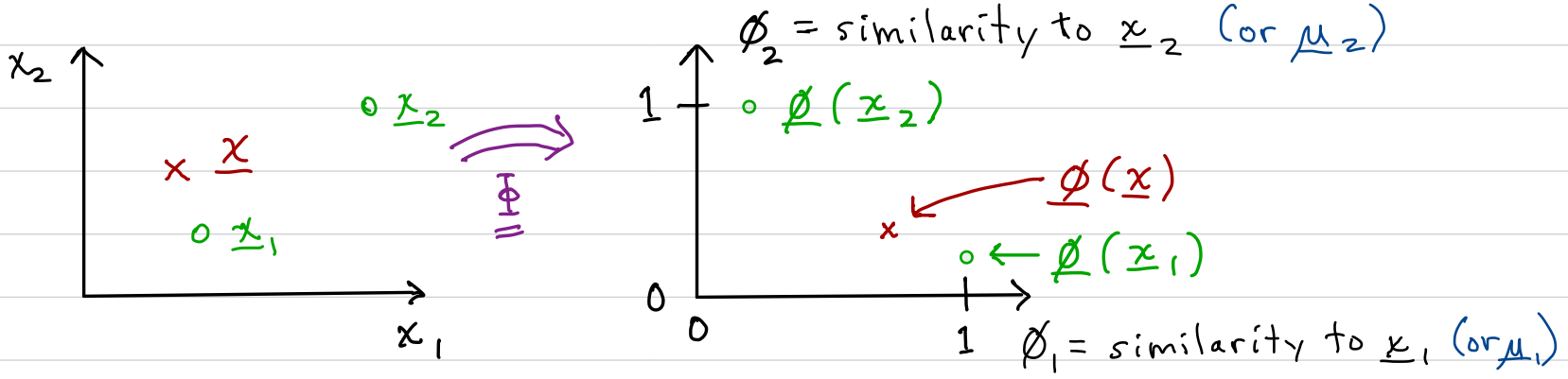
Let $\underline{w}_m^{(1)}$ = non-augmented weight vector from non-augmented input $\underline{x}$ to m^{th} hidden unit: $(\underline{w}_m^{(1)})_j = w_{mj}^{(1)}$, $j = 1, 2, \cdots, D$.

Let $\underline{w}^{(2)}$ = complete weight vector from hidden units to o/p:
$(\underline{w}^{(2)})_m = w_{im}^{(2)}$, $m = 0, 1, 2, \cdots, M$.

Another view:



Choose
$\underline{\mu}_1 = \underline{x}_1$
$\underline{\mu}_2 = \underline{x}_2$



- The m^{th} dimension in the expanded feature space, $v_m(\underline{x}) = \phi_m(\underline{x}) = \exp\{-\gamma_m \|\underline{x} - \underline{\mu}_m\|^2\}$, is the similarity of $\underline{x}$ to the m^{th} centroid $\underline{\mu}_m$.
- More generally, the m^{th} dimension is a (radial) fcn. of the distance $\underline{x}$ to $\underline{\mu}_m$: $\phi_m(\underline{x}) = \text{fcn.}(\|\underline{x} - \underline{\mu}_m\|)$.

RBF network learning

Assume RBF / Gaussian basis fcns: $\phi_m(\underline{x}) = \exp\{-\gamma_m \|\underline{x} - \underline{\mu}_m\|^2\}$

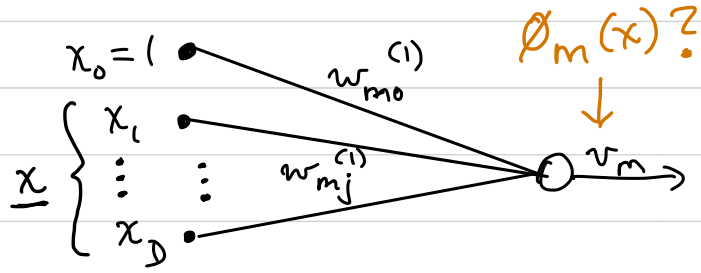
Method 1: Backpropagation on 2-layer ANN

$$\phi_m(\underline{x}) = \exp\{-\gamma_m (\underline{x}-\underline{\mu}_m)^T(\underline{x}-\underline{\mu}_m)\}, \quad \underline{x}, \underline{\mu}_m \text{ non-augmented}$$

("mu")

(2) $$\phi_m(\underline{x}) = \exp\{-\gamma_m (\underbrace{\underline{x}^T\underline{x}}_{t1} \underbrace{-2\underline{\mu}_m^T\underline{x}}_{t2} \underbrace{+\underline{\mu}_m^T\underline{\mu}_m}_{t3})\} \stackrel{?}{=} \exp\{-\gamma_m(a_m)\}$$

$$\underline{w}_m^{(1)T}\underline{x} + w_{m0}^{(1)}$$

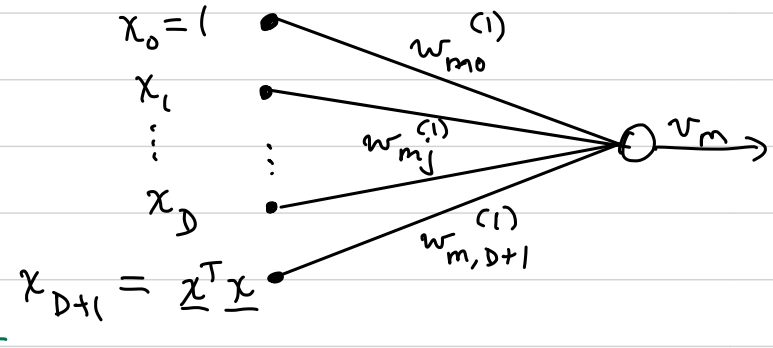


(S)

→ Specify or learn $h(a)$, all weights, for both layers.

Fully ANN implementation, layer $\ell = 1$:

(2.1) input



$$\Rightarrow \begin{cases} \underline{w}_m^{(1)} \triangleq \begin{bmatrix} w_1^{(1)} \\ \vdots \\ w_D^{(1)} \end{bmatrix} = -2\gamma_m \underline{\mu}_m \\ w_{m0}^{(1)} = \gamma_m \underline{\mu}_m^T \underline{\mu}_m \\ w_{m,D+1}^{(1)} = \gamma_m \end{cases}$$

Can train as a 2-layer feedforward ANN

- $\ell = 1$ (hidden layer):

For simplicity, set $\gamma_m = \gamma \ \forall m$.

(3) Activation fcn. $h(a_m) = \exp\{-\gamma a_m\}, \quad \gamma > 0.$

$$a_m = w_0 x_0 + \underline{w}_m^{(1)T} \underline{x} + w_{m,D+1}^{(1)} x_{D+1}$$

Note that $w_m^{(1)}$, $w_{oj}^{(1)}$, $w_{m,D+1}^{(1)}$ are interrelated

For $m = 1, 2, \cdots, M$: (from (2.1) above)

(4)
$$\underline{w}_m^{(1)} = -2\underline{\mu}_m, \quad \text{is learned from data}$$
$$w_{mo}^{(1)} = \underline{\mu}_m^T \underline{\mu}_m \quad \text{is calculated from } \underline{w}_m^{(1)}: \ w_{mo}^{(1)} = \frac{1}{4} \underline{w}_m^{(1)T} \underline{w}_m^{(1)}$$
$$w_{m,D+1}^{(1)} = 1 \quad \text{is fixed before learning.}$$

- $\ell = 2$ (output layer)

(5) activation fcn. $h_{o/p}(a_k) = a_k$ (linear units)

$w_0^{(2)}$, $\underline{w}^{(2)}$ are learned from data

Comments: 1. Some library functions may implement RBF networks directly; no need to cast strictly as an ANN with conventional inner-product units as above. Instead, can use Fig. 1 directly.

2. M needs to be chosen, e.g. by model selection or other algorithm.

3. Method (1) is less common than method (2) below.

Method 2: Unsupervised learning for layer 1, supervised learning for layer 2

[Ref: Christopher Bishop, Neural Networks for Pattern Recognition (Oxford Univ. Press, 1995), Sec. 5.3, 5.9.]

> This is typically much faster than method (1).

* > Let $\gamma_m = \gamma \ \forall m$ again.

- $\ell = 2$ (output layer)

  Supervised learning can be done with any 1-layer ANN algorithm, for example MSE-based algorithms.
  Most criterion functions for 1-layer feedforward ANNs are convex, including MSE.

- $\ell = 1$ (hidden layer) – a number of methods

  (i) A simple method is to choose

  $\underline{\mu}_m = \underline{x}_m$, $m = 1, 2, \cdots, M$, and choose $M = N$. $\gamma_m = \gamma > 0$, $m = 1, 2, \cdots, M$

  Find γ using model selection

  Hidden layer weights

  $$\underline{w}_m^{(1)} = -2\underline{x}_m, \quad w_{m0}^{(1)} = \underline{x}_m^T \underline{x}_m, \quad w_{m,D+1}^{(1)} = 1$$

  Comments:
  - No learning involved
  - Simple, but not optimal
  - M can be large; d.o.f. vs. N_c $\Rightarrow$ need regularization.

Because (i) becomes infeasible for large N, there are many other methods.

Most involve finding good basis-fcn. centers $\underline{\mu}_m$, $m = 1, 2, \cdots, M$, with $M < N$ or $M << N$.

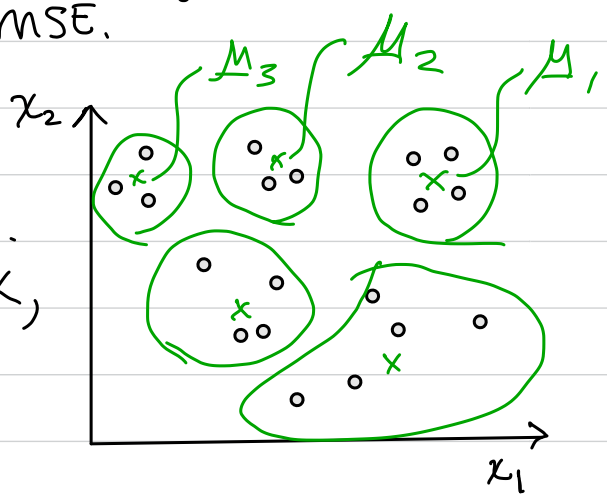
Then hidden layer weights become (same as (4) above):

(6) $$\underline{w}_m^{(1)} = -2\underline{\mu}_m, \quad w_{m0}^{(1)} = \underline{\mu}_m^T \underline{\mu}_m, \quad w_{m,D+1}^{(1)} = 1$$

(ii) Choose $\underline{\mu}_m$ as a random subset of the data-input vectors $\underline{x}_n$.
(Not an optimal choice, but computationally very fast.)

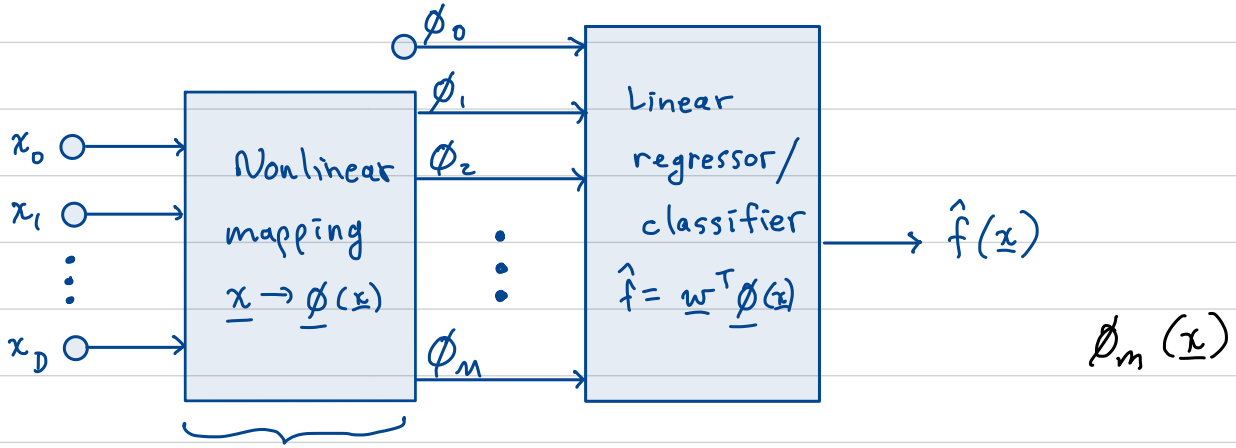
(iii) Orthogonal least squares [Chen et al., 1991, referenced in our textbook Sec. 6.3.0] – sequentially adds 1 basis-fcn. center $\underline{\mu}_m$ at a time, picking the data point that gives the best incremental improvement in MSE.

(iv) Use a clustering algorithm on the dataset's input vectors $\underline{x}_n$, e.g. using K-means clustering. Then use the cluster means $\underline{\mu}_m$, $m = 1, 2, \cdots K$, as the basis-fcn. centers.
Typically $K << N$.



Tips for implementing RBF networks

1. It need not be done using ANN. It could instead be done using our previously discussed tools:



RBF mapping: $\phi_m(\underline{x}) = \text{fcn.}(\|\underline{x} - \underline{\mu}_m\|) = \exp\{-\gamma_m \|\underline{x} - \underline{\mu}_m\|^2\}$; often $\gamma_m = \gamma \ \forall m$.

2. If choice of RBF centers is: $\underline{\mu}_m = \underline{x}_m$, so that $M = N$, then using a regularizer (e.g., ridge regression for the linear regressor) is strongly encouraged. Otherwise, the complexity balance: d.o.f. $\leftrightarrow$ constraints is not good.

3. A simple choice for $\underline{\mu}_m$ is randomly picking data points (as one would to construct a validation set), with $M < N$ but M not too small. M can be chosen by model selection.

4. Can typically make $M \ll N$ by using a smarter choice for $\underline{\mu}_m$ (as described above).

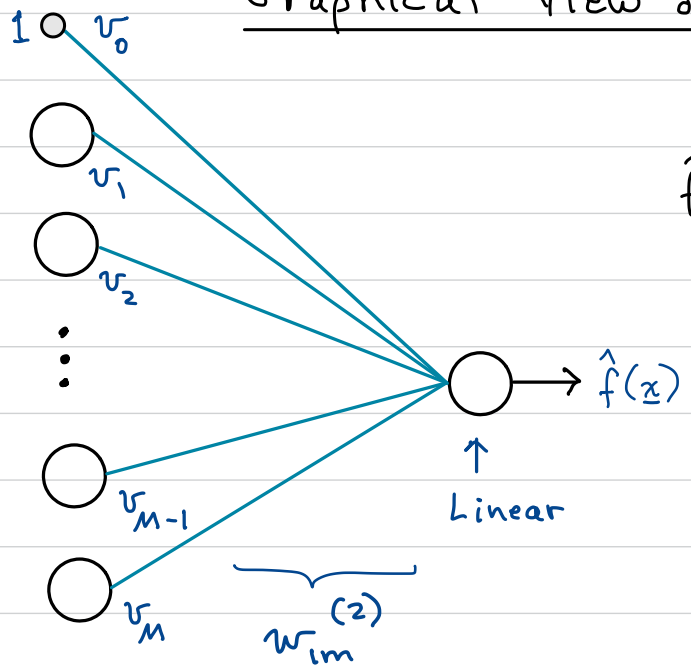
Adv. / disadv. of RBF networks

+ Can be a universal function approximator
+ Only requires 2 layers
+ Does not require non-convex backward error propagation learning

- Number of hidden units can become very large for some approaches
- Irrelevant features can require a lot of extra d.o.f., so if many irrelevant features are suspected, best to do some feature selection by other means first. (Empirical result.)

+/- Basis functions $\phi_m(\underline{x})$ are essentially local in nature to each center $\underline{\mu}_m$. In principle this can provide faster, more reliable convergence than other multilayer ANNs trained using backpropagation. (Representation at hidden layer is less distributed.)

Graphical View of RBF Network Computation (2nd layer)



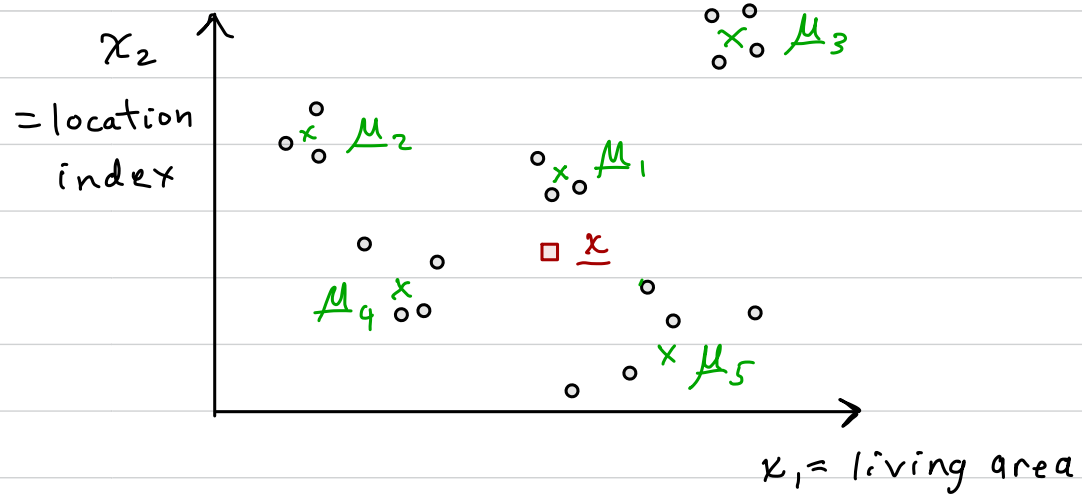
$$\hat{f}(\underline{x}) = \sum_{m=1}^{M} w_{1m}^{(2)} v_m(\underline{x})$$

Let $\gamma_m = 1$

$$= \sum_{m=1}^{M} w_{1m}^{(2)} \exp\{-\gamma_m \|\underline{x} - \underline{\mu}_m\|^2\}$$

= weighted sum of similarity to each RBF center $\underline{\mu}_m$.

- $\underline{\mu}_1$ = prototypical condo in downtown L.A.
- $\underline{\mu}_2$ = prototypical small house in Santa Monica
- $\underline{\mu}_3$ = prototypical large house in Beverly Hills
- ⋮



$\underline{x}$ = house near USC.

Price est $\hat{f}(\underline{x}) = w_{11}^{(2)} v_1(\underline{x}) + w_{12}^{(2)} v_2(\underline{x}) + \cdots + w_{15}^{(2)} v_5(\underline{x})$

similarty to: downtown L.A. condo; small house in Santa Monica; ... large house in Pasadena

K-means clustering [Bishop Sec. 9.1]

Given a set of N data points and number of clusters K.
Let C_k denote the k^{th} cluster.
Group the data points into K clusters, each defined by a cluster center $\underline{\mu}_k$, in a way that minimizes the criterion function:

(7)
$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\underline{x}_n - \underline{\mu}_k\|_2^2 , \qquad r_{nk} \triangleq \begin{cases} 1, & \text{if } \underline{x}_n \in C_k \\ 0, & \text{if } \underline{x}_n \notin C_k \end{cases}$$

(For a given data pt. $\underline{x}_n$, r_{nk} is a one-hot encoding of its cluster assignment.)

We want to find $r_{nk}\ \forall n, k$, and $\underline{\mu}_k\ \forall k$, that minimize J.

Typically done iteratively:

1. Choose initial cluster centers $\underline{\mu}_k$, $k = 1, 2, \cdots, K$
2. Assign each data pt. $\underline{x}_n$ to the cluster that will give the smallest contribution to J:

$$J = \sum_{n=1}^{N} J_n , \quad J_n = \sum_{k=1}^{K} r_{nk} \|\underline{x}_n - \underline{\mu}_k\|_2^2 = \|\underline{x}_n - \underline{\mu}_{k'}\|_2^2, \text{ if } \underline{x}_n \in C_{k'}$$

(7)
$$k = \arg\min_{k'} J_n = \arg\min_{k'} \|\underline{x}_n - \underline{\mu}_{k'}\|_2^2 \Rightarrow$$

assign $\underline{x}_n$ to cluster with closest center $\underline{\mu}_k$.

3. Given the cluster assignments r_{nk} from step 2, find new cluster centers $\underline{\mu}_k$ that minimize J:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{N} r_{nk} \|\underline{x}_n - \underline{\mu}_k\|^2 \quad \text{with } r_{nk} \text{ given.}$$

$$\nabla_{\underline{\mu}_{k'}} J = \sum_{n=1}^{N} r_{nk'} \cancel{2} (\underline{x}_n - \underline{\mu}_{k'}) \cancel{(-1)} = \underline{0}$$

$$\sum_{n=1}^{N} r_{nk'} \underline{x}_n = \sum_{n=1}^{N} r_{nk'} \underline{\mu}_{k'}$$

$$\sum_n r_{nk'} \underline{x}_n = \underline{\mu}_{k'} \sum_n r_{nk'}$$

drop primes

$$\underline{\mu}_k = \frac{\sum_n r_{nk} \underline{x}_n}{\sum_n r_{nk}} \quad \Big\} \longrightarrow \sum_{n=1}^{N} r_{nk} = ?$$

$$\underline{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \underline{x}_n$$

= mean of data pts. in cluster C_k.

(8)

$$\boxed{\underline{\mu}_k = \frac{1}{N_k} \sum_{\underline{x}_n \in C_k} \underline{x}_n}$$

N_k = # data points in C_k.

4. Iterate steps 2 and 3 until there are no changes in cluster assignments, or until a maximum # iterations is reached.

Is J convex?

No, because r_{nk} depends on μ_k: $r_{nk} = \text{fcn.}(\mu_k)$.

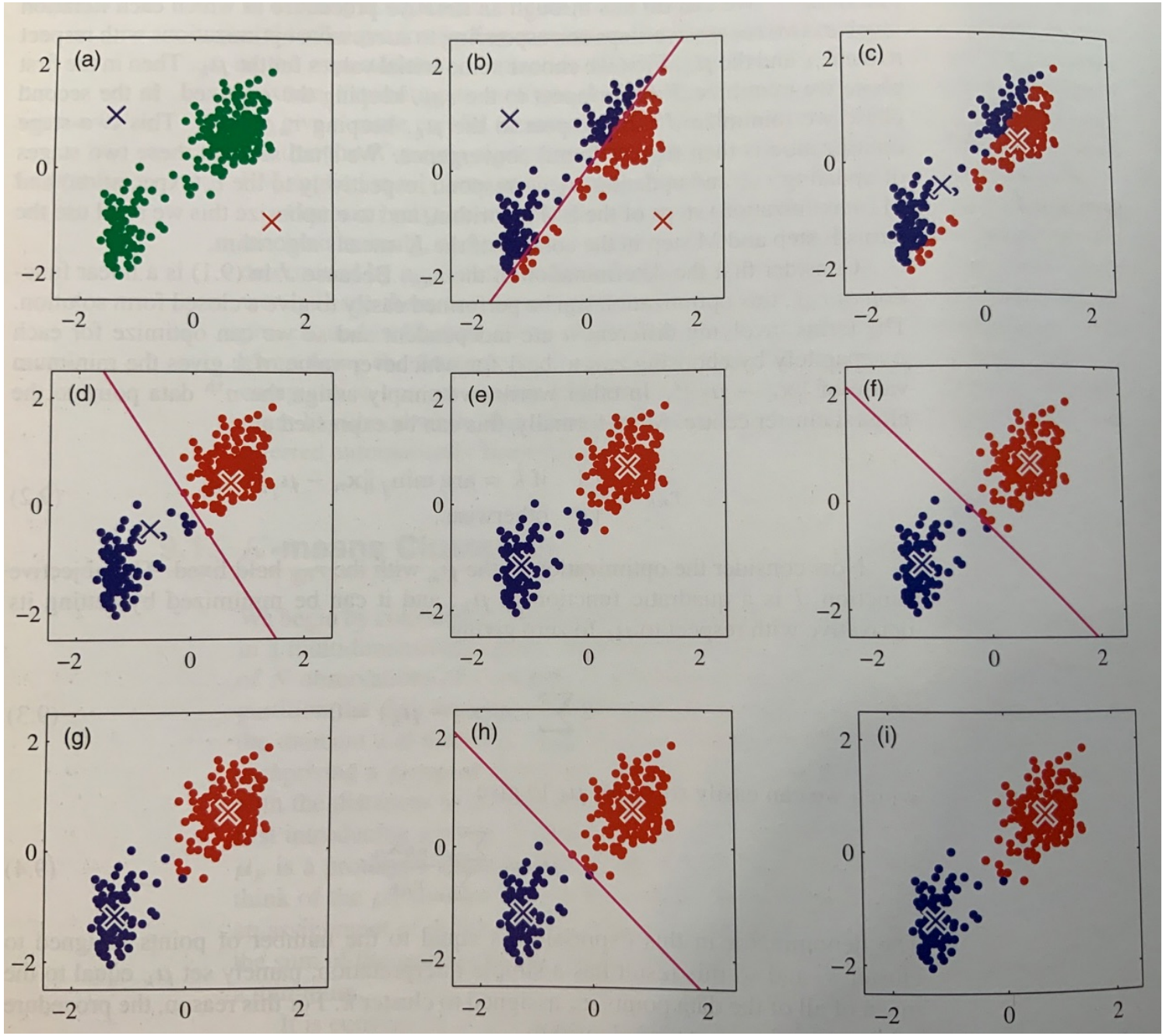
But: J is a monitonically decreasing fcn. of iteration number (until it converges), so it will converge to a local minimum.

$K=2$

Initial μ_k:

× μ_1 →

× μ_2 →



Converged result ←

Bishop Fig. 9.1 — example of K-means clustering.

Comments on K-means clustering

1. Can be used to find good RBF centers $\underline{\mu}_m$ in a RBF network.
   ⇒ Gives weights in first layer of RBF network, learned from data (Eq. (5) above).

2. Computational time complexity is $O(N^2)$.
   There are techniques to reduce it. Can also use parallelism.

3. Can run K-means multiple times with different initial centers $\underline{\mu}_m$; pick the best result.

4. For our case (find $\underline{\mu}_m$ for RBF network), we can save computation time by:

   (i) first subsampling the training data, then run K-means on the subsampled data.

   (ii) first partioning the training data, then run K-means on each partition independently to get K' clusters from each partition.

5. Can set $\gamma_m = \gamma_m' \gamma$, with γ_m' determined by variance of m^{th} cluster. Then γ is a global RBF-width parameter which can be chosen by model selection.