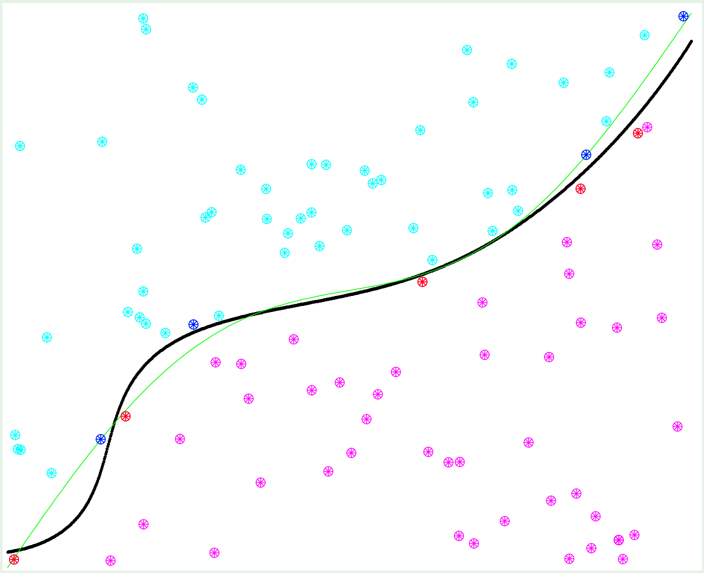
Review of Lecture 15

- Kernel methods

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{z}^{\intercal}\mathbf{z}' \text{ for some } \mathcal{Z} \text{ space}$$

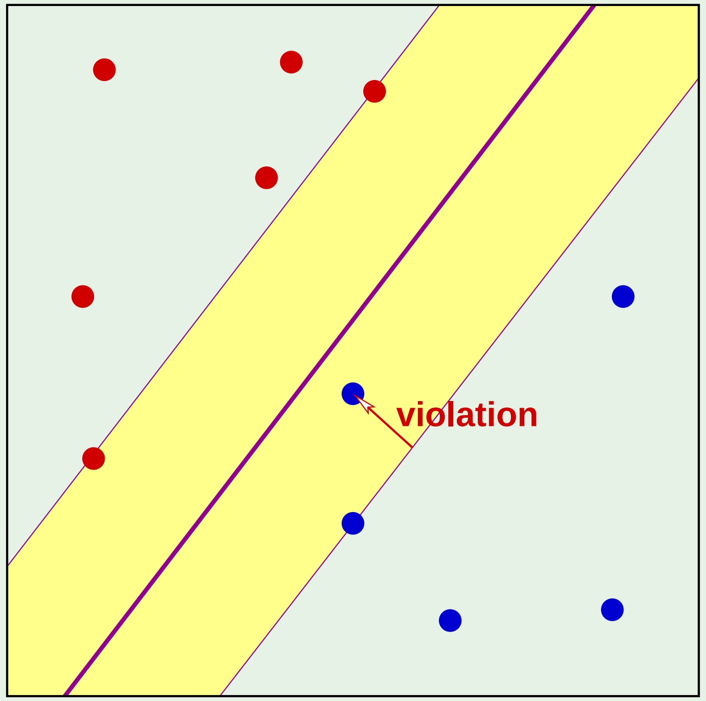


$$\uparrow$$

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma \, \|\mathbf{x} - \mathbf{x}'\|^2\right)$$

- Soft-margin SVM

$$\text{Minimize} \quad \frac{1}{2}\,\mathbf{w}^{\intercal}\mathbf{w} + C\sum_{n=1}^{N}\xi_n$$



Same as hard margin, but $0 \leq \alpha_n \leq C$

Learning From Data

Yaser S. Abu-Mostafa
California Institute of Technology

Lecture 16: Radial Basis Functions

Sponsored by Caltech's Provost Office, E&AS Division, and IST • Thursday, May 24, 2012

Outline

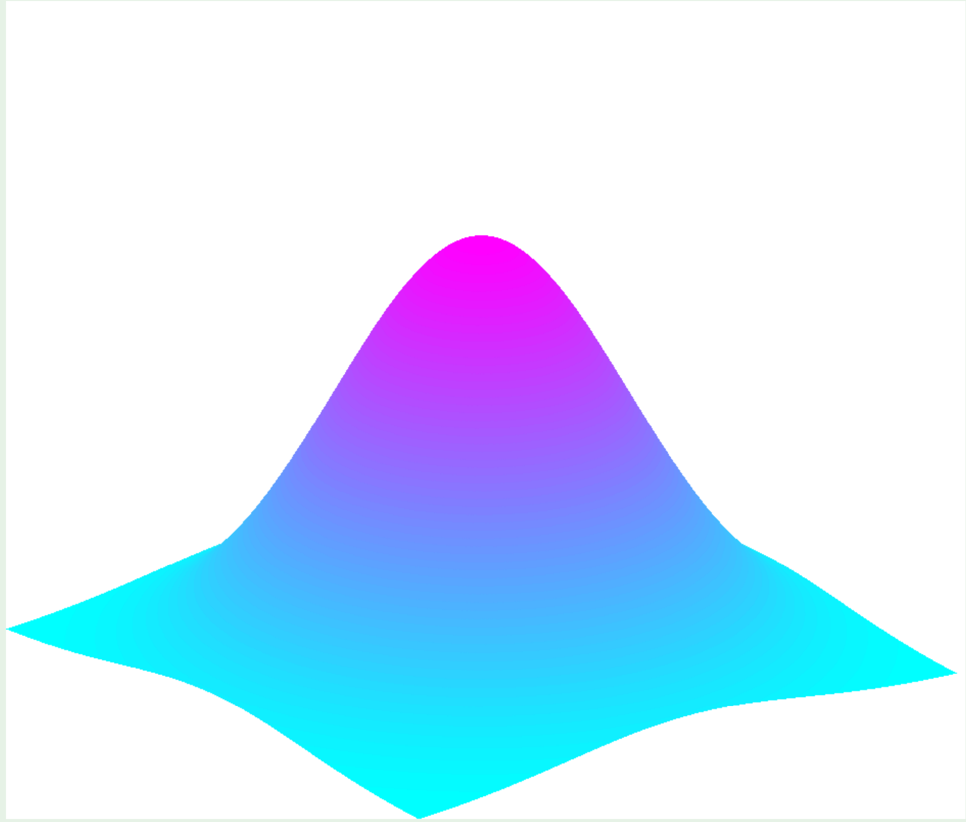
- RBF and nearest neighbors
- RBF and neural networks
- RBF and kernel methods
- RBF and regularization

Basic RBF model

Each $(\mathbf{x}_n, y_n) \in \mathcal{D}$ influences $h(\mathbf{x})$ based on $\underbrace{\|\mathbf{x} - \mathbf{x}_n\|}_{\text{radial}}$

Standard form:

$$h(\mathbf{x}) = \sum_{n=1}^{N} w_n \underbrace{\exp\left(-\gamma \, \|\mathbf{x} - \mathbf{x}_n\|^2\right)}_{\text{basis function}}$$



The learning algorithm

Finding $w_1, \cdots, w_N$:

$$h(\mathbf{x}) \;=\; \sum_{n=1}^{N} w_n \exp\left(-\gamma\,\|\mathbf{x}-\mathbf{x}_n\|^2\right)$$

based on $\mathcal{D} = (\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$

$E_{\text{in}} = 0$: $\quad h(\mathbf{x}_n) = y_n$ for $n = 1, \cdots, N$:

$$\sum_{m=1}^{N} w_m \; \exp\left(-\gamma\,\|\mathbf{x}_n-\mathbf{x}_m\|^2\right) \;=\; y_n$$

The solution

$$\sum_{m=1}^{N} w_m \ \exp\left(-\gamma \ \|\mathbf{x}_n - \mathbf{x}_m\|^2\right) \ = \ y_n$$

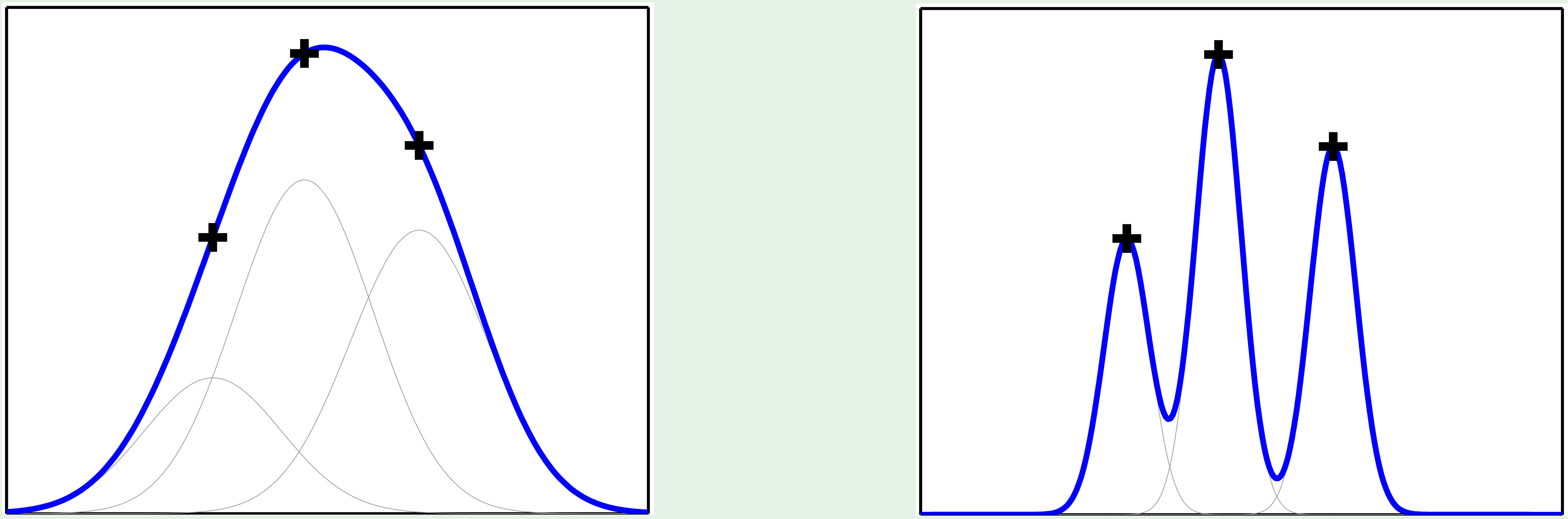
N equations in N unknowns

$$\underbrace{\begin{bmatrix} \exp(-\gamma \ \|\mathbf{x}_1 - \mathbf{x}_1\|^2) & \ldots & \exp(-\gamma \ \|\mathbf{x}_1 - \mathbf{x}_N\|^2) \\ \exp(-\gamma \ \|\mathbf{x}_2 - \mathbf{x}_1\|^2) & \ldots & \exp(-\gamma \ \|\mathbf{x}_2 - \mathbf{x}_N\|^2) \\ \vdots & \vdots & \vdots \\ \exp(-\gamma \ \|\mathbf{x}_N - \mathbf{x}_1\|^2) & \ldots & \exp(-\gamma \ \|\mathbf{x}_N - \mathbf{x}_N\|^2) \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}}_{\mathbf{w}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\mathbf{y}}$$

If Φ is invertible, $\boxed{\mathbf{w} = \Phi^{-1}\mathbf{y}}$ "exact interpolation"

The effect of γ

$$h(\mathbf{x}) \;=\; \sum_{n=1}^{N} w_n \exp\left(-\gamma \;\|\mathbf{x} - \mathbf{x}_n\|^2\right)$$



small γ large γ

RBF for classification

$$h(\mathbf{x}) \ = \mathrm{sign}\left(\sum_{n=1}^{N} w_n \ \exp\left(-\gamma \ \|\mathbf{x}-\mathbf{x}_n\|^2\right)\right)$$

Learning: $\sim$ linear regression for classification

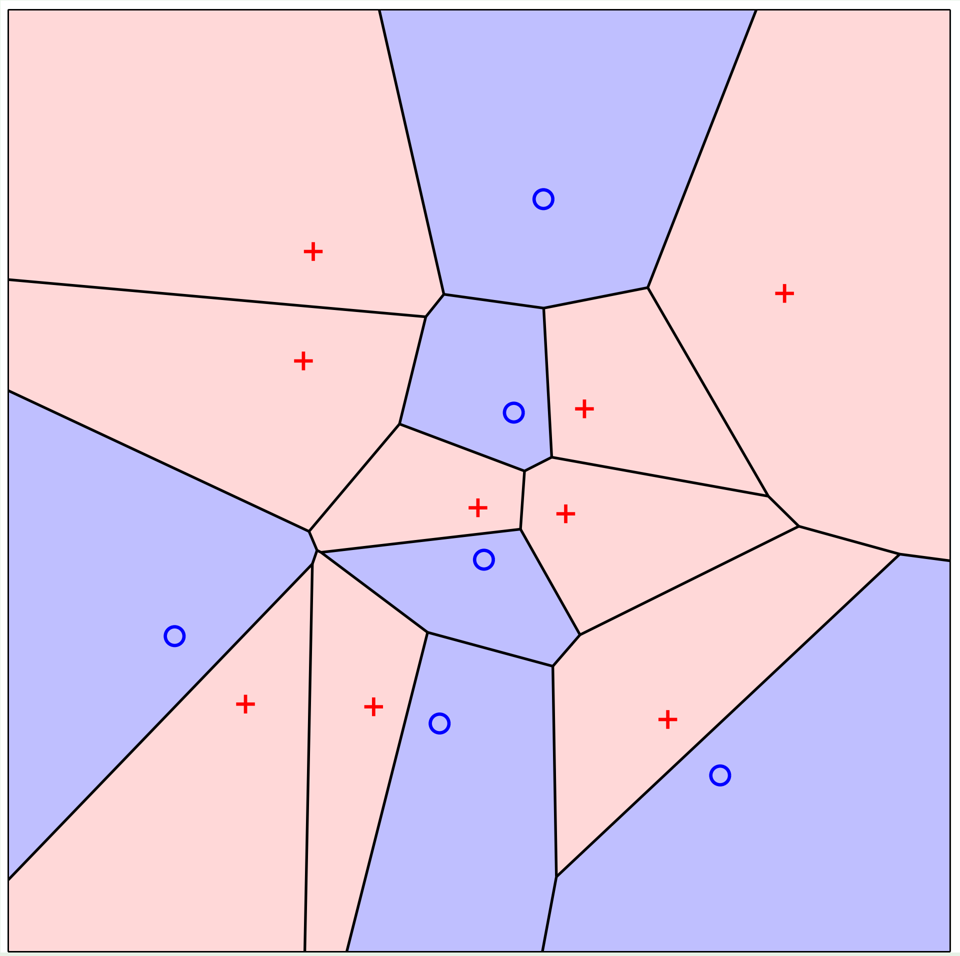
$$s \ = \sum_{n=1}^{N} w_n \ \exp\left(-\gamma \ \|\mathbf{x}-\mathbf{x}_n\|^2\right)$$

Minimize $(s-y)^2$ on $\mathcal{D}$ $\qquad y = \pm 1$

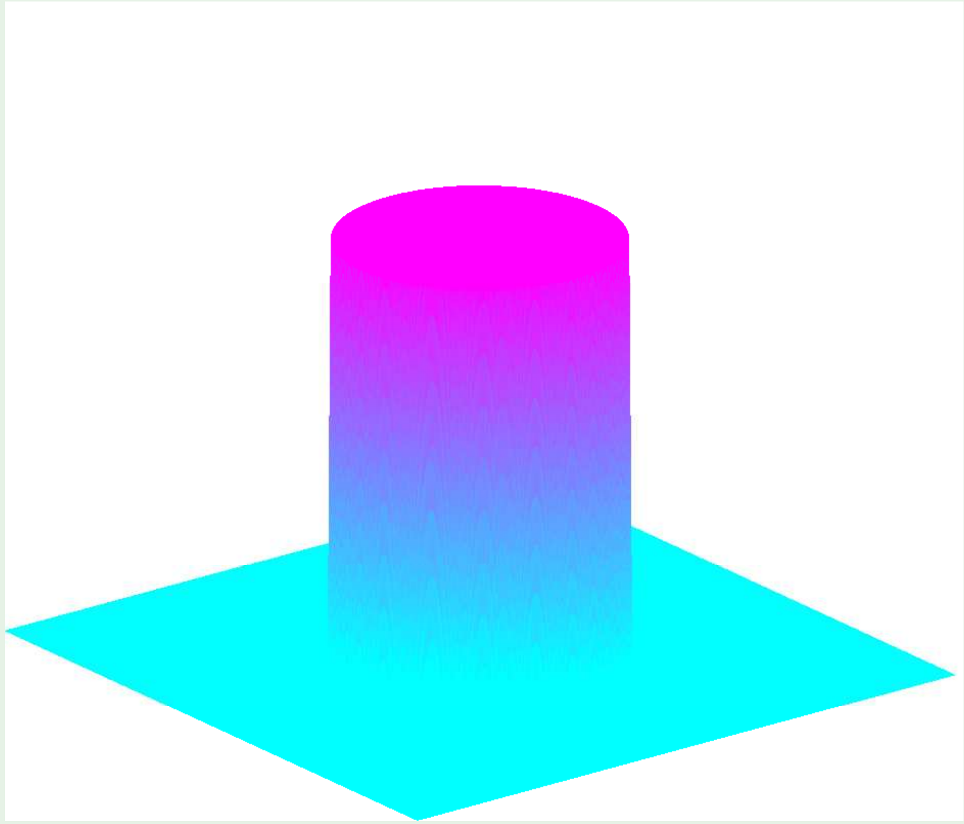
$h(\mathbf{x}) = \mathrm{sign}(s)$

Relationship to nearest-neighbor method

Adopt the y value of a nearby point:



similar effect by a basis function:



RBF with K centers

N parameters $w_1, \cdots, w_N$ based on N data points

Use $K \ll N$ centers: $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K$ instead of $\mathbf{x}_1, \cdots, \mathbf{x}_N$

$$h(\mathbf{x}) = \sum_{k=1}^{K} w_k \exp\left(-\gamma \left\|\mathbf{x} - \boldsymbol{\mu}_k\right\|^2\right)$$

1. How to choose the centers $\boldsymbol{\mu}_k$

2. How to choose the weights w_k

Choosing the centers

Minimize the distance between $\mathbf{x}_n$ and the **closest** center $\boldsymbol{\mu}_k$: $\boxed{K\text{-means clustering}}$

Split $\mathbf{x}_1, \cdots, \mathbf{x}_N$ into clusters $S_1, \cdots, S_K$

Minimize $\displaystyle\sum_{k=1}^{K} \sum_{\mathbf{x}_n \in S_k} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$

Unsupervised learning ☺

NP-hard ☹

An iterative algorithm

Lloyd's algorithm: Iteratively minimize $\displaystyle\sum_{k=1}^{K} \sum_{\mathbf{x}_n \in S_k} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$ w.r.t. $\boldsymbol{\mu}_k, S_k$

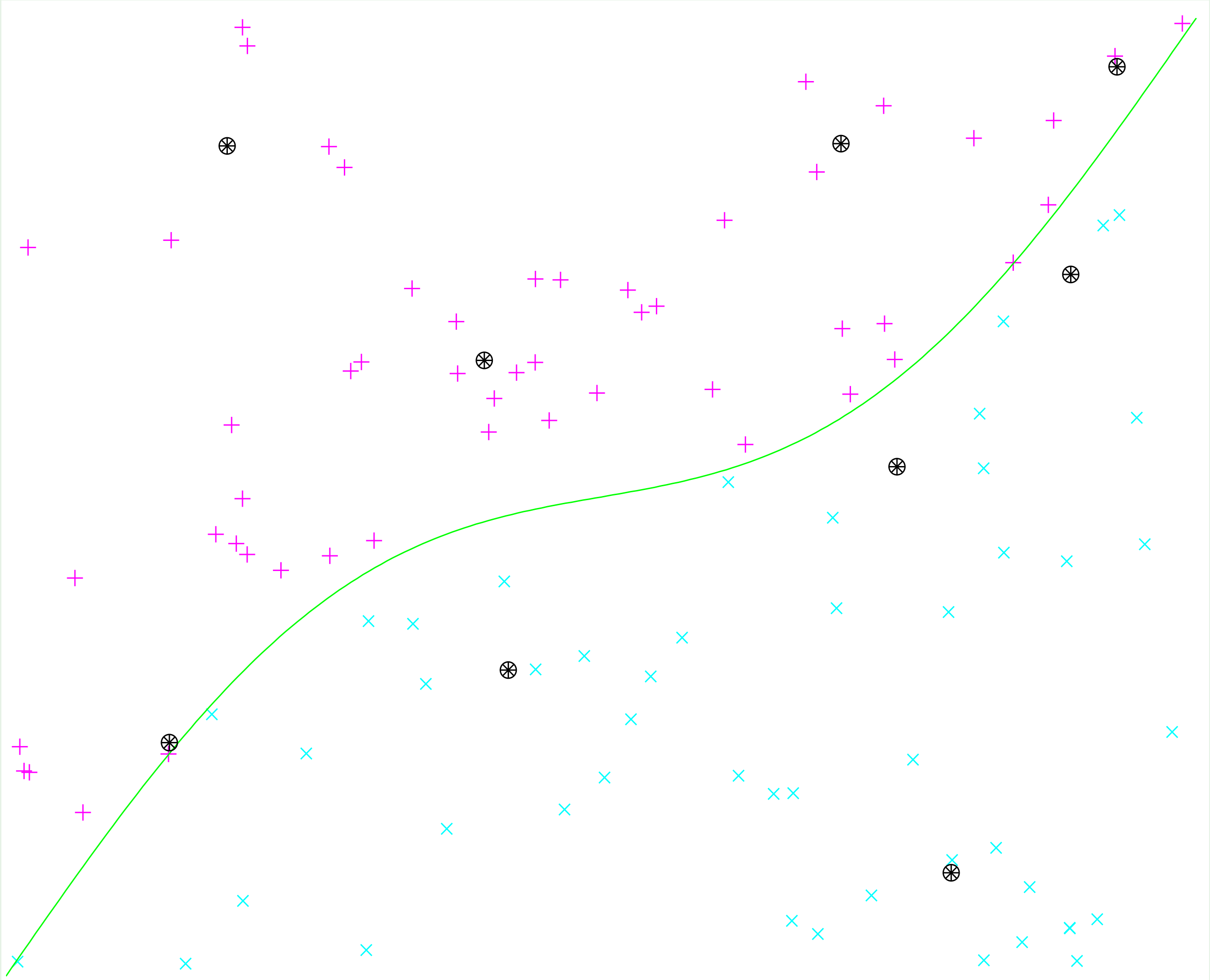
$$\boldsymbol{\mu}_k \leftarrow \frac{1}{|S_k|} \sum_{\mathbf{x}_n \in S_k} \mathbf{x}_n$$

$$S_k \leftarrow \{\mathbf{x}_n : \|\mathbf{x}_n - \boldsymbol{\mu}_k\| \leq \text{all } \|\mathbf{x}_n - \boldsymbol{\mu}_\ell\|\}$$

Convergence $\longrightarrow$ **local minimum**

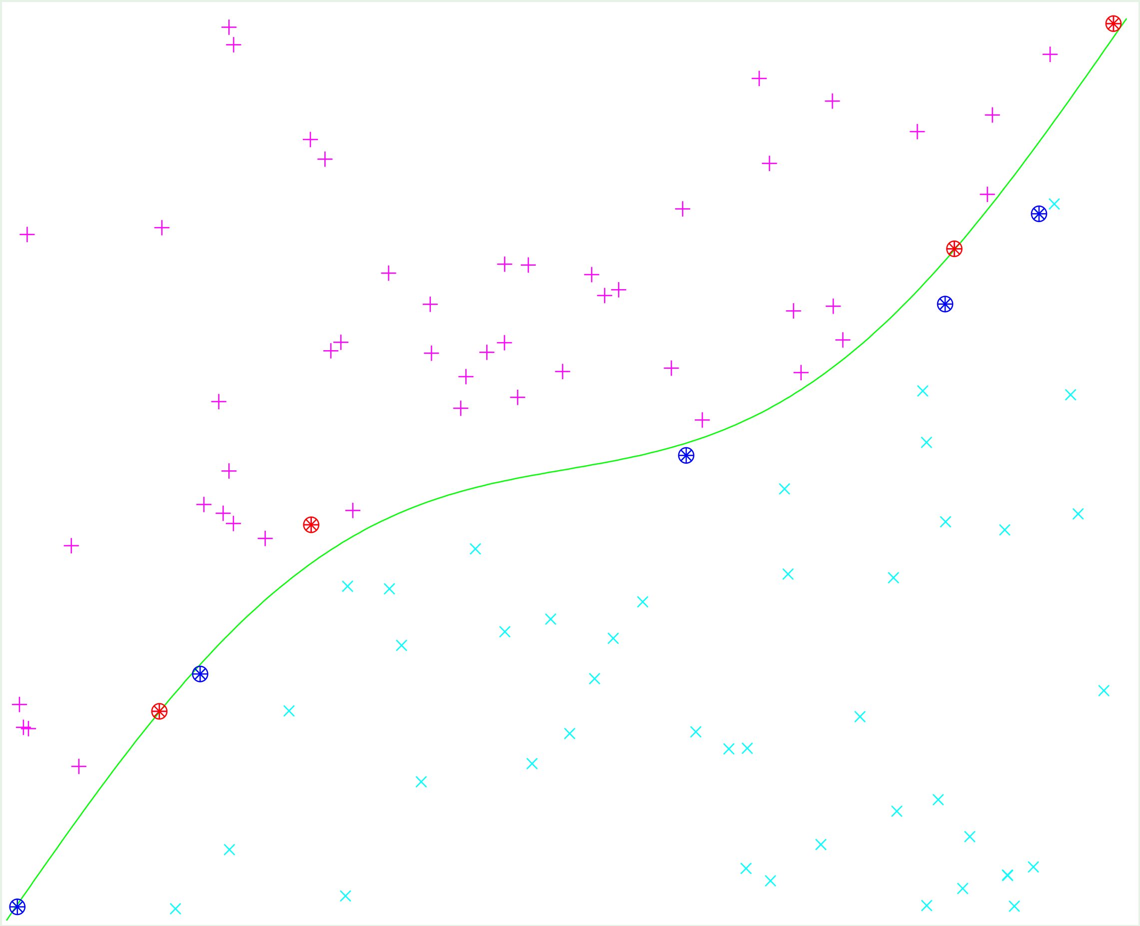
Lloyd's algorithm in action

1. Get the data points
2. Only the inputs!
3. Initialize the centers
4. Iterate
5. These are your $\boldsymbol{\mu}_k$'s

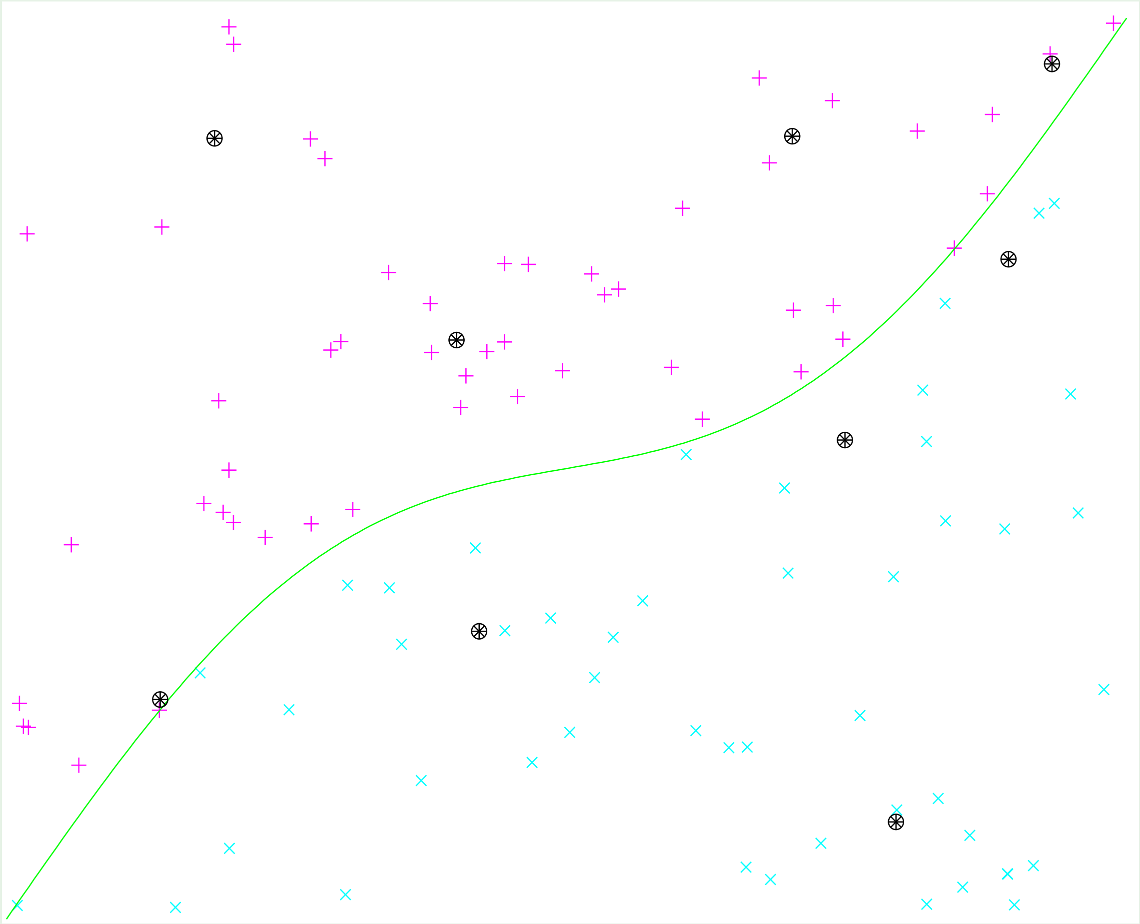


Centers versus support vectors

support vectors



RBF centers



Choosing the weights

$$\sum_{k=1}^{K} w_k \, \exp\left(-\gamma \, \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2\right) \approx y_n$$

N equations in $K < N$ unknowns

$$\underbrace{\begin{bmatrix} \exp(-\gamma\,\|\mathbf{x}_1 - \boldsymbol{\mu}_1\|^2) & \dots & \exp(-\gamma\,\|\mathbf{x}_1 - \boldsymbol{\mu}_K\|^2) \\ \exp(-\gamma\,\|\mathbf{x}_2 - \boldsymbol{\mu}_1\|^2) & \dots & \exp(-\gamma\,\|\mathbf{x}_2 - \boldsymbol{\mu}_K\|^2) \\ \vdots & \vdots & \vdots \\ \exp(-\gamma\,\|\mathbf{x}_N - \boldsymbol{\mu}_1\|^2) & \dots & \exp(-\gamma\,\|\mathbf{x}_N - \boldsymbol{\mu}_K\|^2) \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_K \end{bmatrix}}_{\mathbf{w}} \approx \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\mathbf{y}}$$

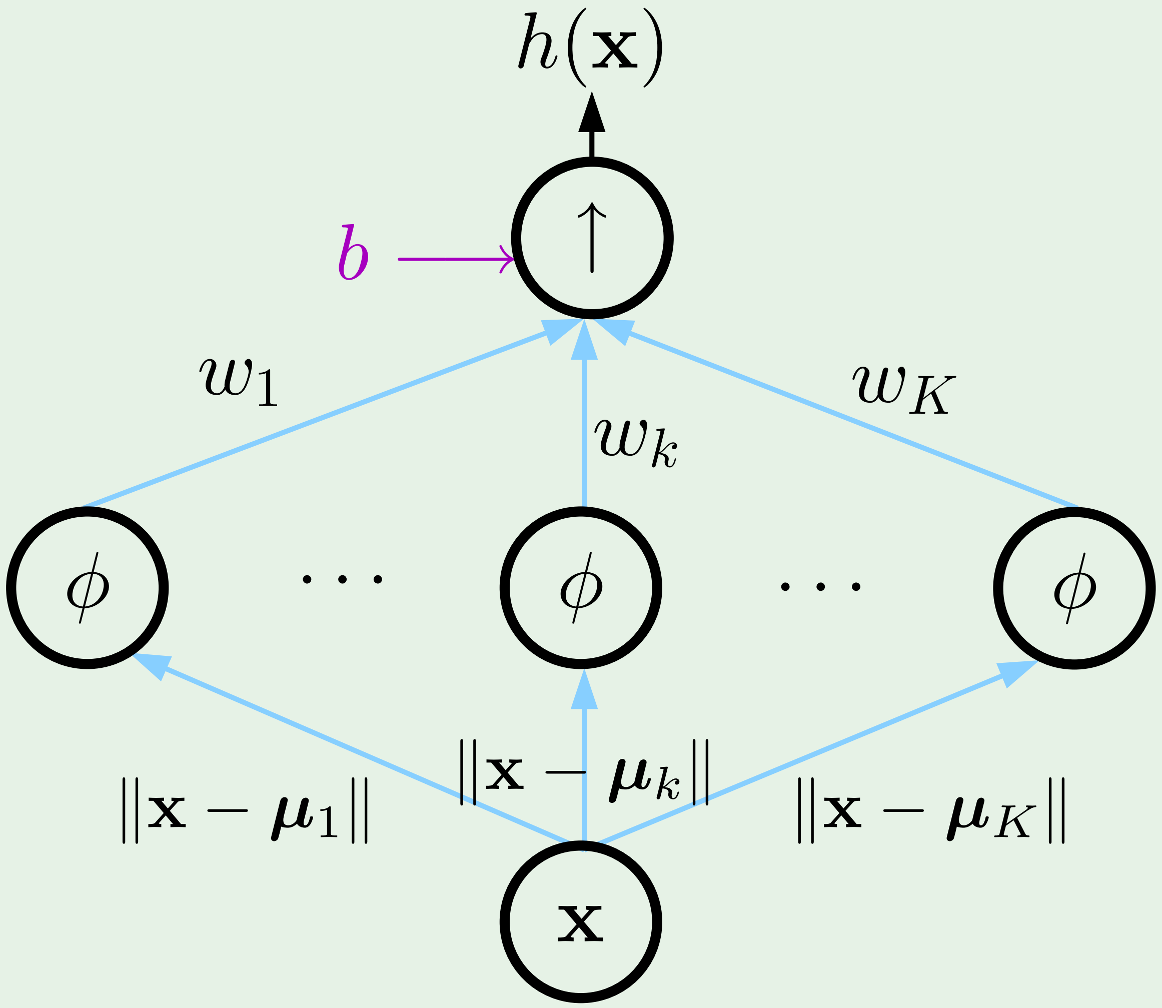
If $\Phi^\top\Phi$ is invertible, $\boxed{\mathbf{w} = (\Phi^\top\Phi)^{-1}\Phi^\top\mathbf{y}}$ **pseudo-inverse**

RBF network

The "features" are $\exp\left(-\gamma\, \|\mathbf{x} - \boldsymbol{\mu}_k\|^2\right)$

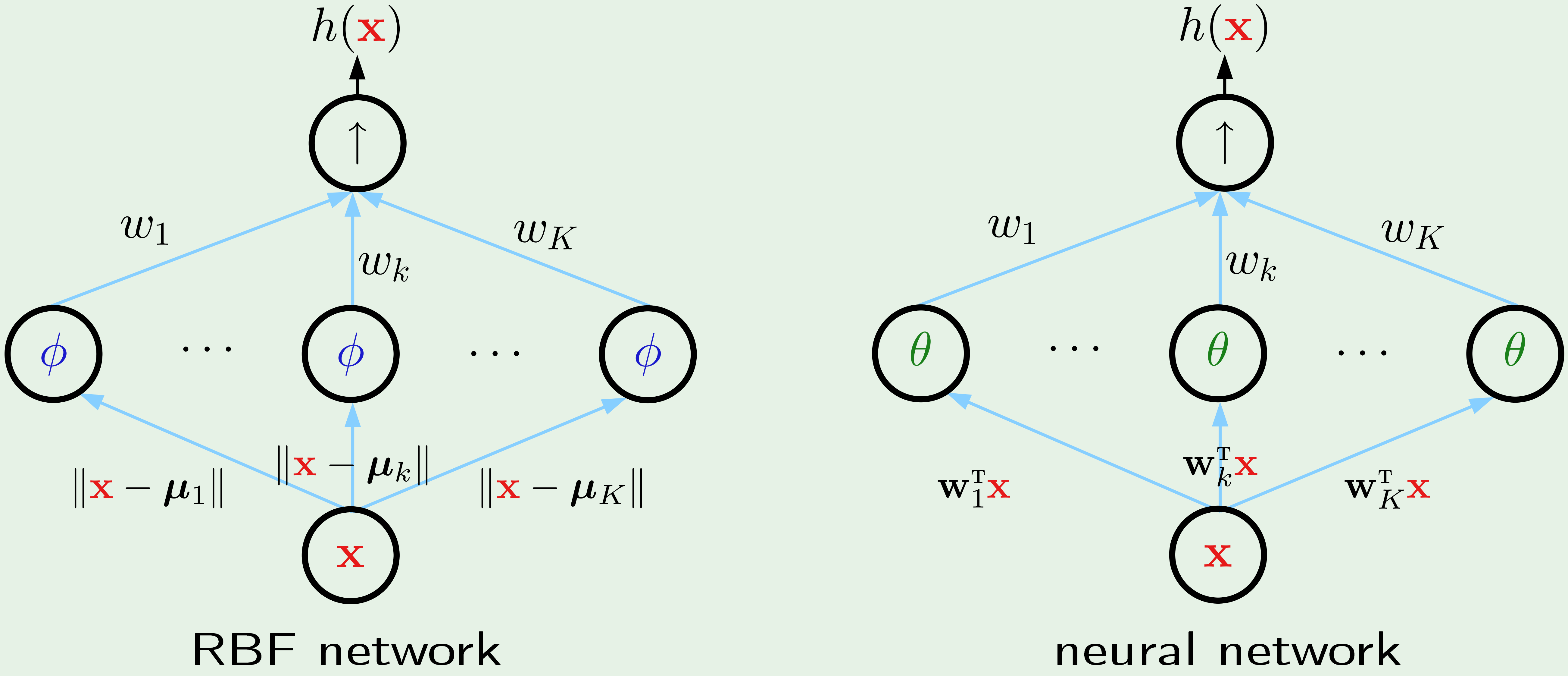
Nonlinear transform depends on $\mathcal{D}$

$\Longrightarrow$ No longer a linear model



A bias term (b or w_0) is often added

Compare to neural networks



Choosing γ

Treating γ as a parameter to be learned $\quad h(\mathbf{x}) = \sum_{k=1}^{K} w_k \exp\left(-\gamma \, \|\mathbf{x} - \boldsymbol{\mu}_k\|^2\right)$

Iterative approach ($\sim$ **EM algorithm** in mixture of Gaussians):

1. Fix γ, solve for $w_1, \cdots, w_K$

2. Fix $w_1, \cdots, w_K$, minimize error w.r.t. γ

We can have a different γ_k for each center $\boldsymbol{\mu}_k$

Outline

- RBF and nearest neighbors
- RBF and neural networks
- RBF and kernel methods
- RBF and regularization

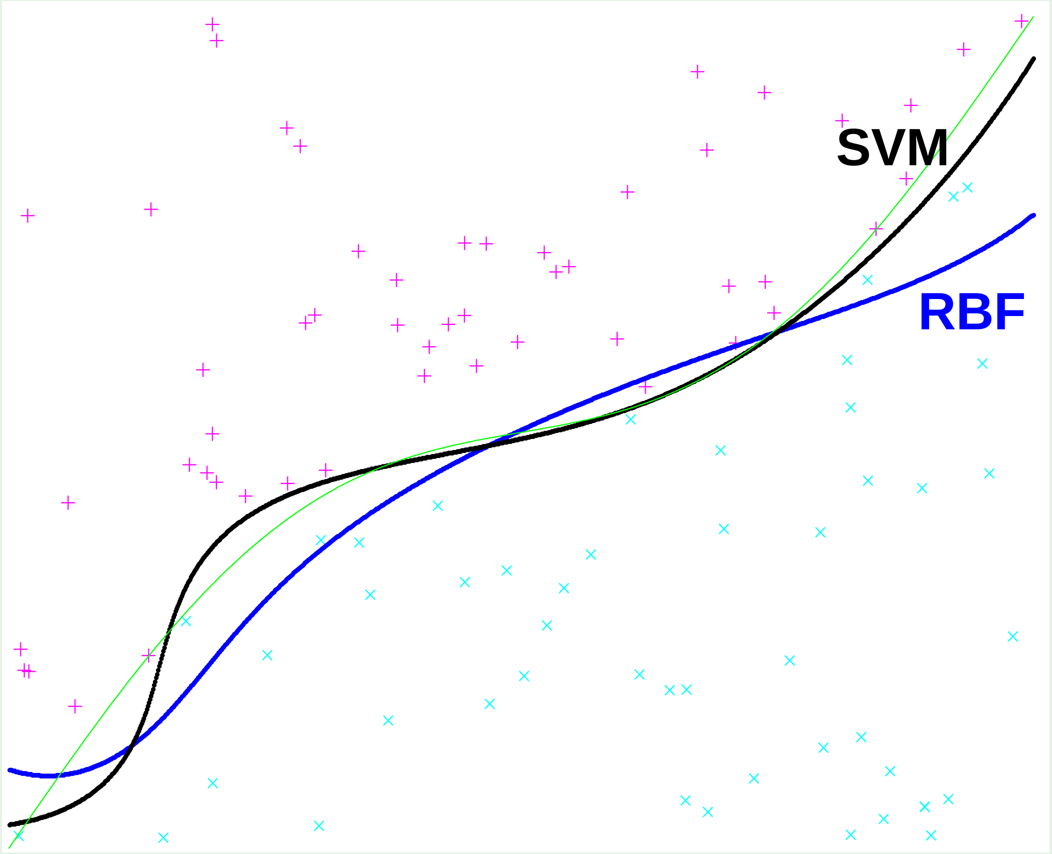
RBF versus its SVM kernel

SVM kernel implements:

$$\mathrm{sign}\left(\sum_{\alpha_n>0} \alpha_n y_n \exp\left(-\gamma \left\|\mathbf{x}-\mathbf{x}_n\right\|^2\right)+b\right)$$

Straight RBF implements:

$$\mathrm{sign}\left(\sum_{k=1}^{K} w_k \exp\left(-\gamma \left\|\mathbf{x}-\boldsymbol{\mu}_k\right\|^2\right)+b\right)$$



RBF and regularization

RBF can be derived based purely on regularization:

$$\sum_{n=1}^{N} \big(h(x_n) - y_n\big)^2 + \lambda \sum_{k=0}^{\infty} a_k \int_{-\infty}^{\infty} \left(\frac{d^k h}{dx^k}\right)^2 dx$$

"smoothest interpolation"