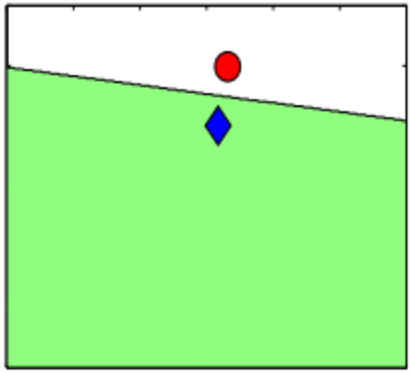
CMSC 510
Regularization Methods for Machine Learning

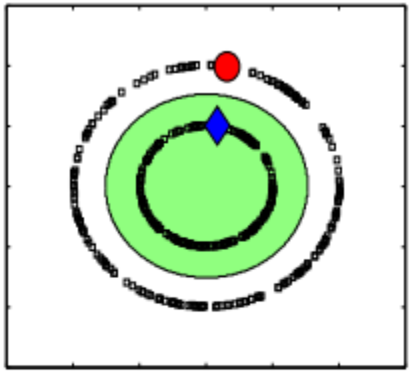
Semi-supervised Learning

Instructor:
Dr. Tom Arodz

Semi-supervised learning

- Sometimes, we knew more about the underlying distribution of samples than what is in the (x,y) training pairs
  - E.g. if we have few points with labels, but we have a lot of other samples without labels, they may tell us something useful:

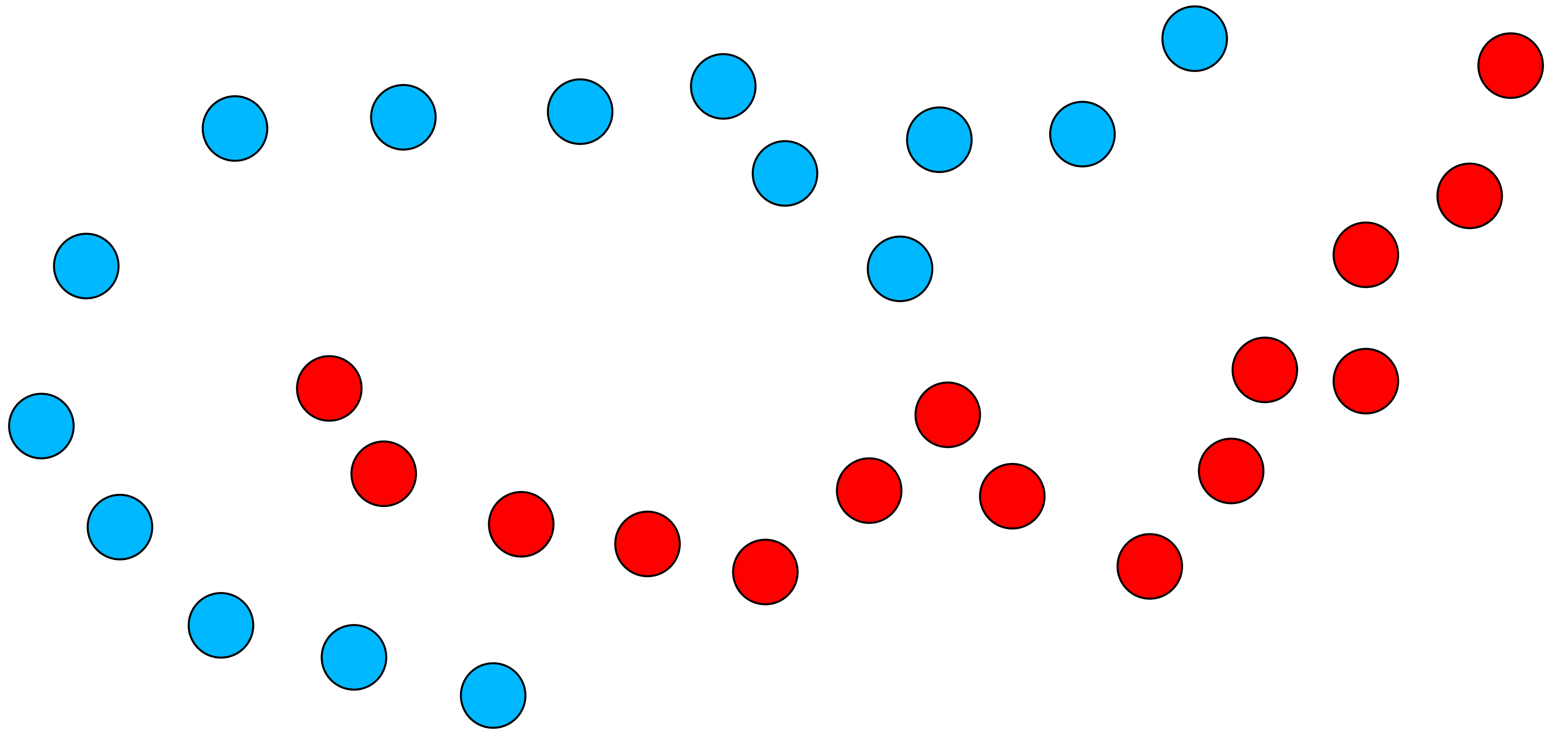




  - Classifier on the right is likely to be better!

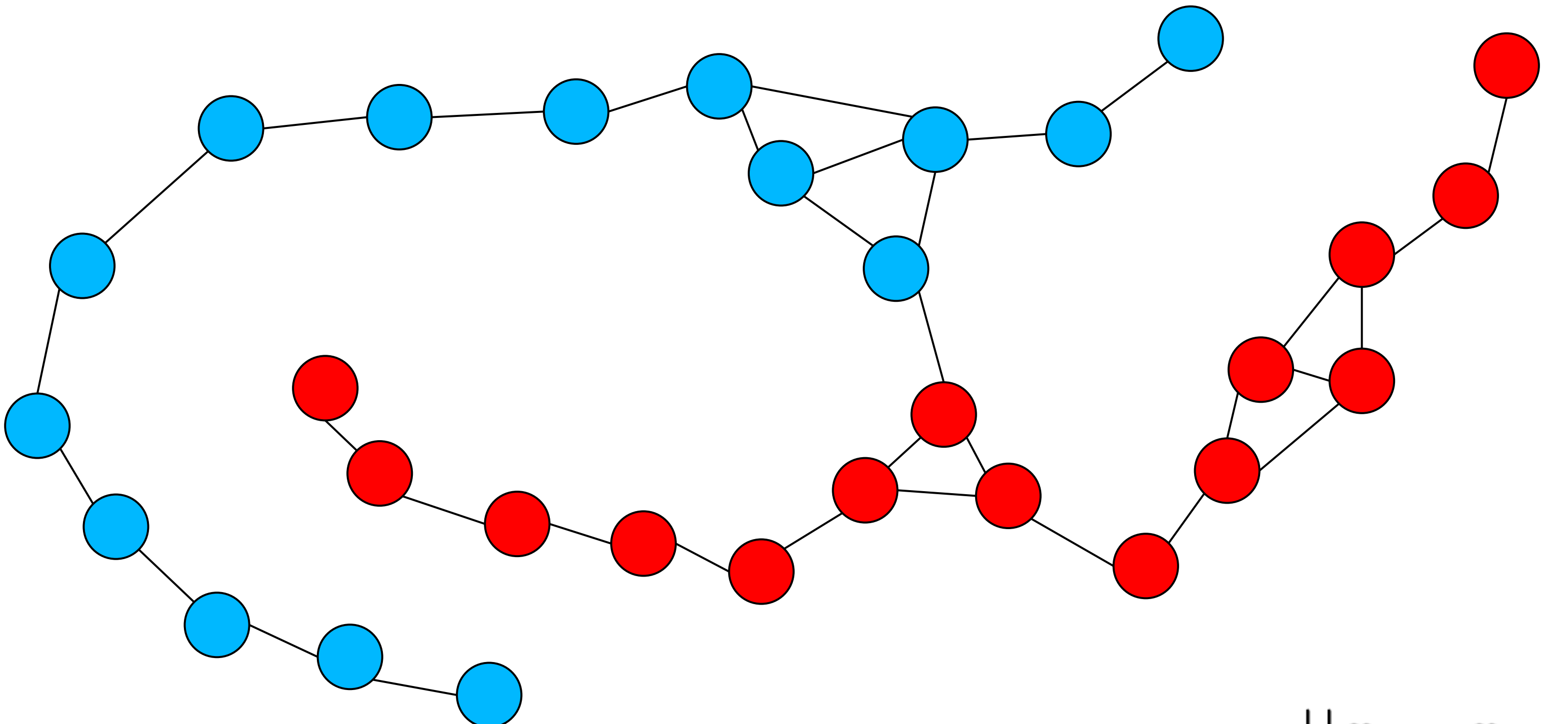
Semi-supervised learning

- We have training samples



Semi-supervised learning

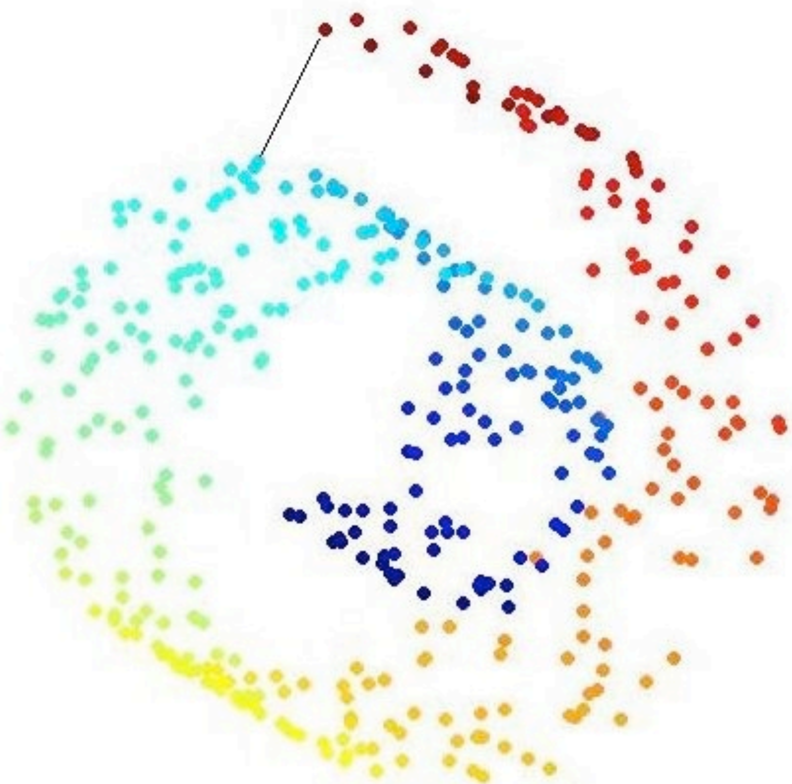
- We have training samples
- And a graph linking samples that are close

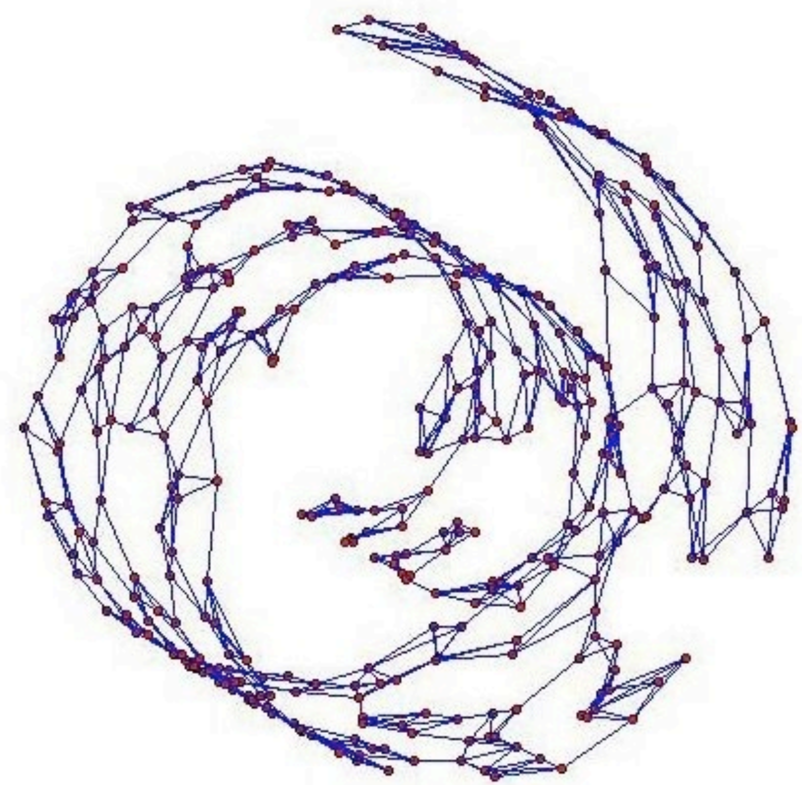


- Edge weights are e.g.: $G_{j,k} = \mathrm{e}^{-\frac{||x_j - x_k||^2}{t}}$

Semi-supervised kernel learning

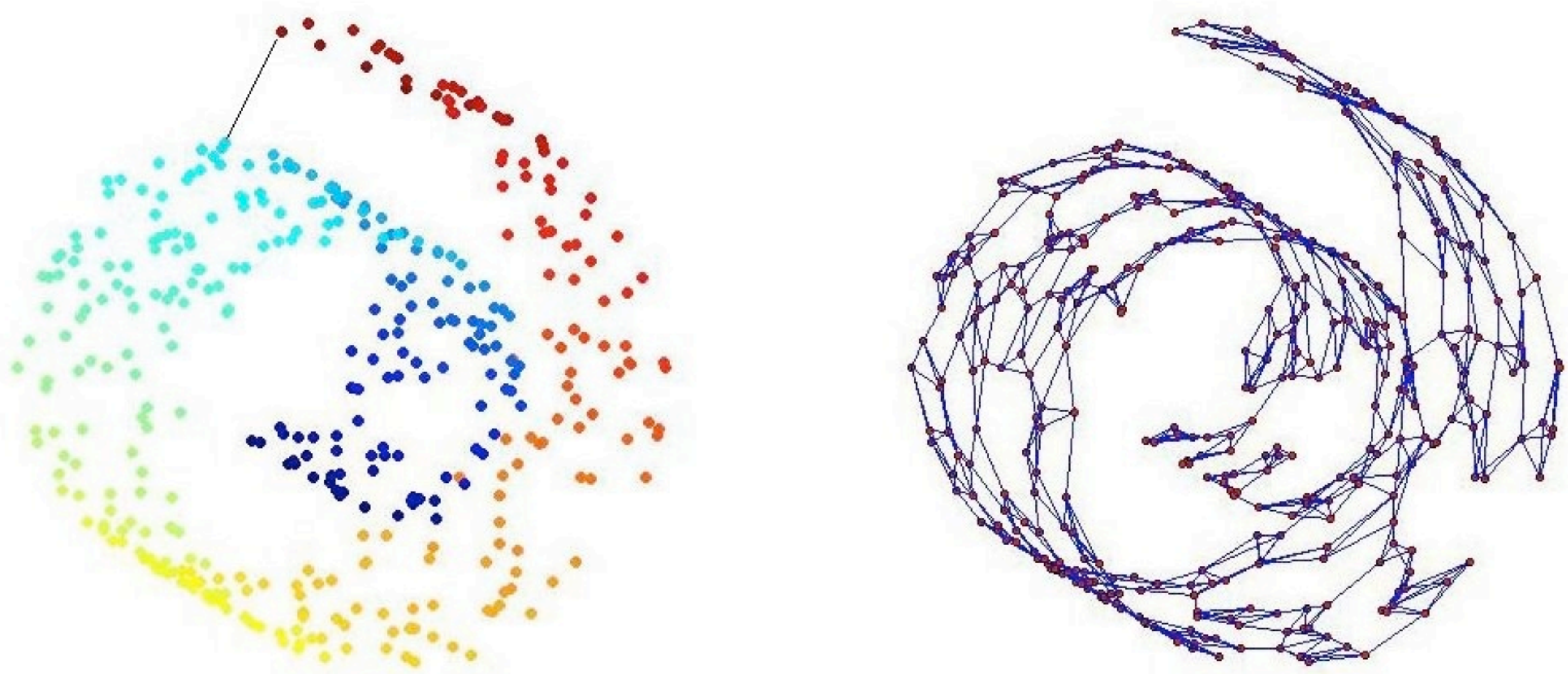
- Data in n-D feature space (here: 3D) may actually reside on a lower-dimensional space (here: 2D)





Semi-supervised kernel learning

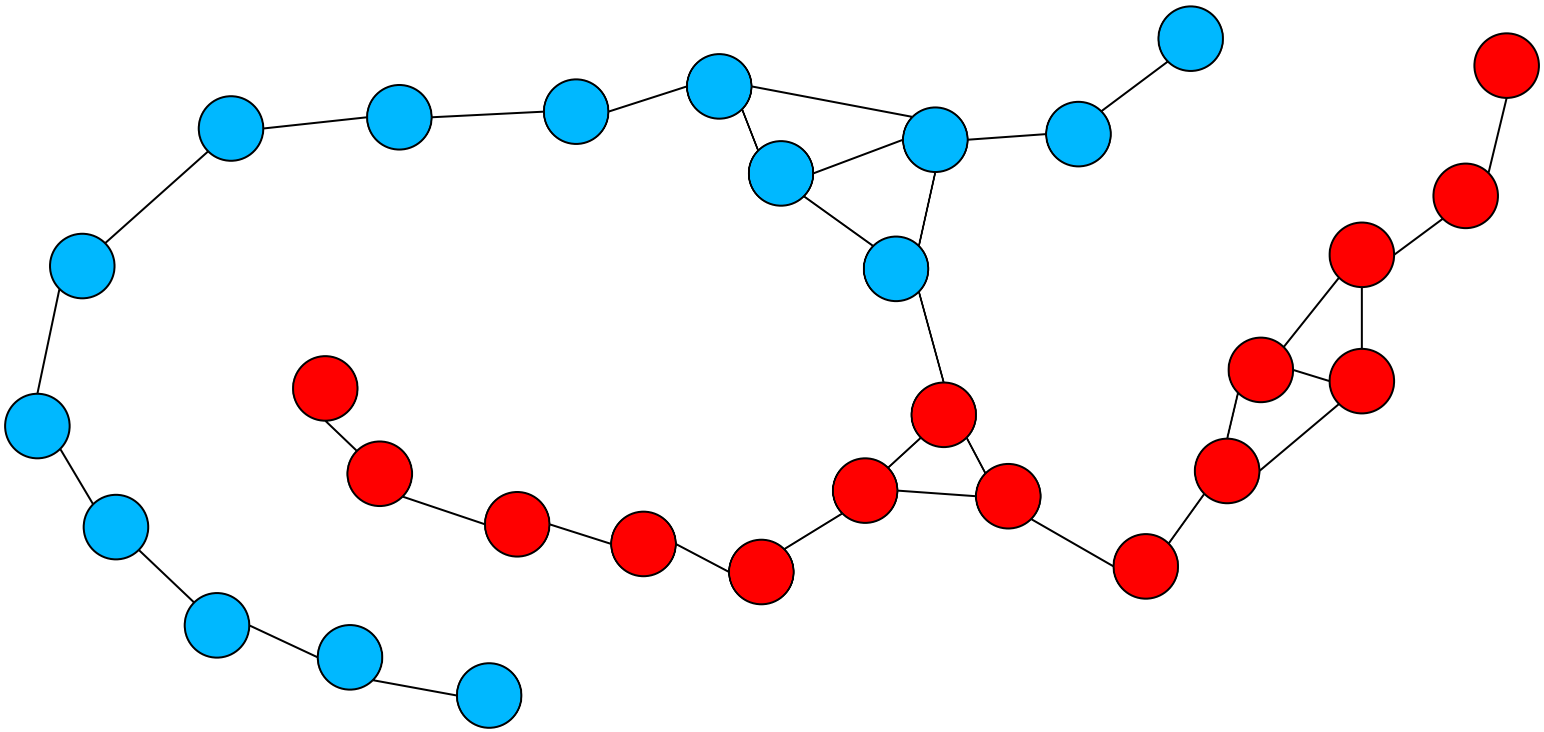
- Samples that may be quite close in original space can be quite far on the graph:



- This new “graph” notion of closeness may be better!

Semi-supervised learning

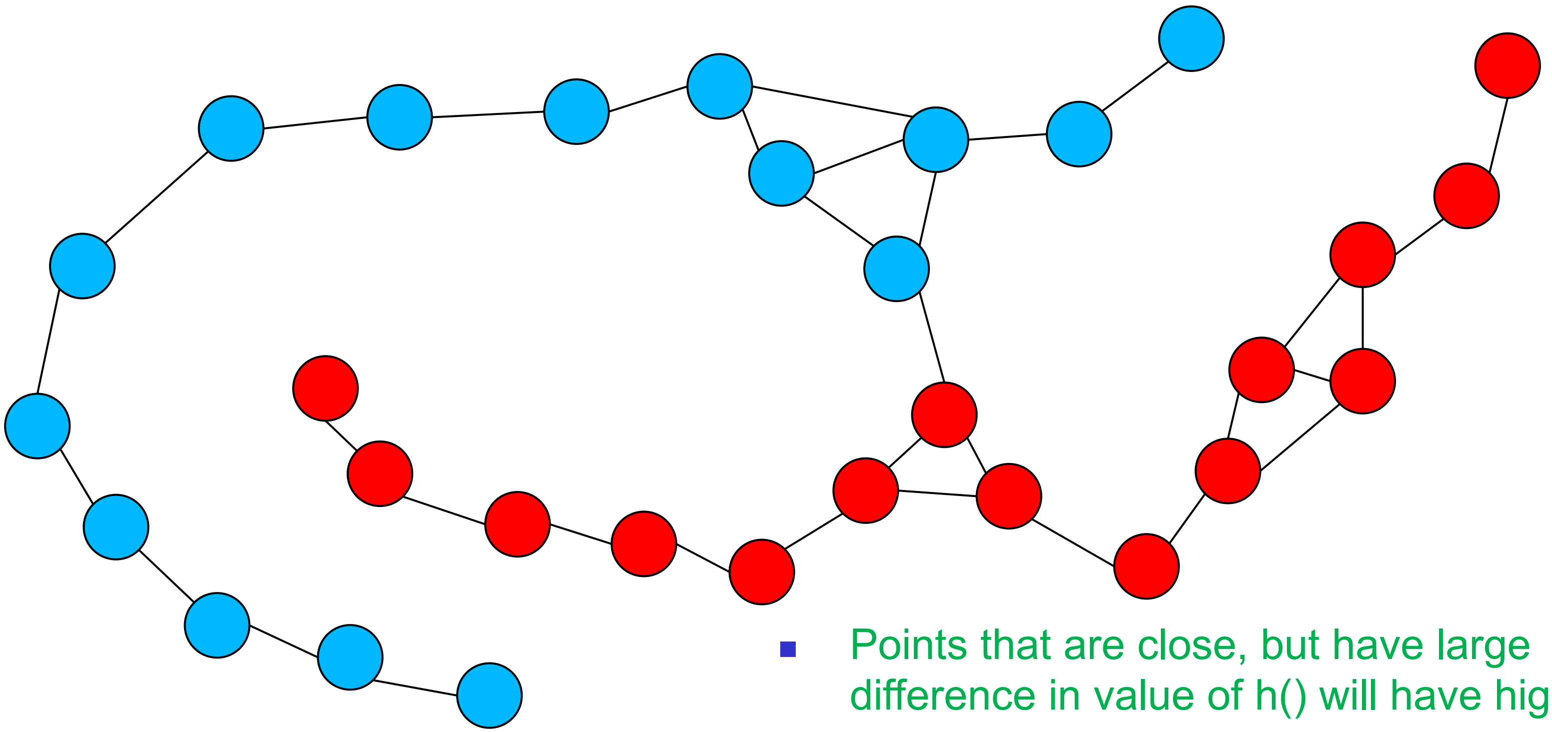
- Training samples => graph G: $G_{j,k} = \mathrm{e}^{-\frac{||x_j - x_k||^2}{t}}$
- We define a penalty $\Omega_G(h)$ for functions h



- Points that are close, but have large difference in value of h() will have high penalty

Semi-supervised learning

- Training samples => graph G: $G_{j,k} = \mathrm{e}^{-\frac{||x_j - x_k||^2}{t}}$
- We define a penalty $\Omega_G(h)$



- Points that are close, but have large difference in value of h() will have high penalty

$$\Omega_G(h) = \frac{1}{2} \sum \sum_{j,k=1}^{F} \mathrm{e}^{-\frac{||x_j - x_k||^2}{t}} \left(h(x_j) - h(x_k)\right)^2$$

Semi-supervised learning

- We have training samples and graph G
  - Either given from outside, or calculated based on positions of training samples
- We have a penalty on h based on the graph

$$\Omega_G(h) = \frac{1}{2} \sum \sum_{j,k=1}^{F} G_{j,k} \left(h(x_j) - h(x_k) \right)^2$$

- For a fixed training set and fixed graph G, we want to find the best decision function h

$$\min_{h \in \mathcal{H}, b} \quad C \sum_{i=1}^{m} \ell(x_i, y_i, h(x_i), b) + \frac{1}{2} ||h||_{\mathcal{H}}^2 + \Omega_G(h)$$

Kernel learning

$$G_{j,k} = \mathrm{e}^{-\frac{||x_j - x_k||^2}{t}}$$

- We have training samples and graph G
  - Either given from outside, or calculated based on positions of those samples
- We define a view of h as an m-dimensional vector based on training samples:

$$\bar{h} = (h(x_1), h(x_2), \dots, h(x_m))^T \qquad \bar{h}_j = h(x_j)$$

- The penalty can be written as:

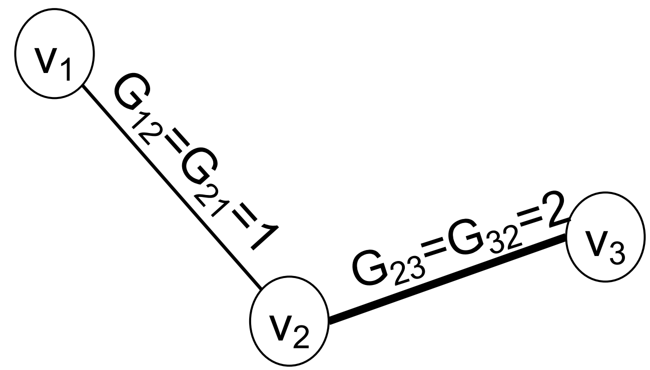
$$\Omega_G(\bar{h}) = \frac{1}{2} \sum \sum_{j,k=1}^{F} G_{j,k} \left(\bar{h}_j - \bar{h}_k\right)^2$$

  - It was:

$$\Omega_G(h) = \frac{1}{2} \sum \sum_{j,k=1}^{F} \mathrm{e}^{-\frac{||x_j - x_k||^2}{t}} \left(h(x_j) - h(x_k)\right)^2$$

Matrices associated with G

- For an undirected graph G with edges G_{jk}
- We can define several matrices:



Graph Laplacian L: $L_G = \underline{D_G} \underline{- A_G}$

Adjacency matrix A: $A_G(j,k) = G_{jk}$

Degree matrix/vector D: $D_G(j,j) = D_j = \sum_{k=1}^{F} G_{jk}$

$D_G(j, k \neq j) = 0$

- For this graph with 3 vertices:

$$L_G = \begin{bmatrix} D_1 & -G_{1,2} & -G_{1,3} \\ -G_{2,1} & D_2 & -G_{2,3} \\ -G_{3,1} & -G_{3,2} & D_3 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 3 & -2 \\ 0 & -2 & 2 \end{bmatrix}$$

Semi-supervised kernel learning

$$\Omega_G(\bar{h}) = \frac{1}{2}\sum\sum_{j,k=1}^{F} G_{j,k}\left(\bar{h}_j - \bar{h}_k\right)^2 = \frac{1}{2}\sum_{j=1}^{F}\sum_{k=1}^{F} G_{j,k}\left(\bar{h}_j^{\,2} + \bar{h}_k^{\,2} - 2\bar{h}_j\bar{h}_k\right)$$

$$= \sum_{j=1}^{F}\sum_{k=1}^{F} -G_{j,k}\bar{h}_j\bar{h}_k + \frac{1}{2}\sum_{j=1}^{F}\bar{h}_j^{\,2}\sum_{k=1}^{F} G_{j,k} + \frac{1}{2}\sum_{k=1}^{F}\bar{h}_k^{\,2}\sum_{j=1}^{F} G_{j,k}$$

$$= \sum_{j=1}^{F}\sum_{k=1}^{F} -G_{j,k}\bar{h}_j\bar{h}_k + \sum_{j=1}^{F}\bar{h}_j^2 D_j$$

$$\boxed{\begin{array}{l} L_G = D_G - A_G \\ A_G(j,k) = G_{jk} \\ D_G(j,j) = D_j = \sum_{k=1}^{F} G_{jk} \\ D_G(j, k \neq j) = 0 \end{array}}$$

$$= \sum_{j=1}^{F}\sum_{k=1}^{F} -G_{j,k}\bar{h}_j\bar{h}_k + \sum_{j=1}^{F}\sum_{k=1}^{F} I(j=k)\bar{h}_j\bar{h}_k D_j$$

$$= \sum_{j=1}^{F}\sum_{k=1}^{F}\bar{h}_j\left[I(j=k)D_j - G_{j,k}\right]\bar{h}_k = \sum_{j=1}^{F}\sum_{k=1}^{F}\bar{h}_j L_G(j,k)\bar{h}_k$$

$$= \bar{h}^T L_G \bar{h}$$

$$\boxed{\Omega_G(\bar{h}) = \bar{h}^T L_G \bar{h}}$$

Kernel learning

$h(x_i) = \sum_{j=1}^{m} c_j K_{x_j}(x_i)$

- With the Laplacian matrix L_G
- And expanding h(x) through kernel K, we see that:

$$\min_{h \in \mathcal{H}, b} \quad C \sum_{i=1}^{m} \ell(x_i, y_i, h(x_i), b) + \frac{1}{2} ||h||^2_{\mathcal{H}} + \Omega_G(h)$$

- Translates to a problem:

$$\min_{c \in \mathbb{R}^m, b} C \sum_{i=1}^{m} \ell(x_i, y_i, \sum_{j=1}^{m} c_j K(x_j, x_i), b) + \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} c_i c_j K(x_i, x_j) + \bar{h}^T L_G \bar{h}$$

$$\bar{h} = (\sum_{j=1}^{m} c_j K(x_j, x_1), \sum_{j=1}^{m} c_j K(x_j, x_2), ..., \sum_{j=1}^{m} c_j K(x_j, x_m))^T$$

Kernel learning

$h(x_i) = \sum_{j=1}^{m} c_j K_{x_j}(x_i)$

- We have an optimization problem:

$$\min_{c\in\mathbb{R}^m, b} C \sum_{i=1}^{m} \ell(x_i, y_i, \sum_{j=1}^{m} c_j K(x_j, x_i), b) + \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} c_i c_j K(x_i, x_j) + \bar{h}^T L_G \bar{h}$$

$$\bar{h} = (\sum_{j=1}^{m} c_j K(x_j, x_1), \sum_{j=1}^{m} c_j K(x_j, x_2), ..., \sum_{j=1}^{m} c_j K(x_j, x_m))^T$$

- But then: $\bar{h} = Kc$
- We can move further into matrix notation:

$$\min_{c\in\mathbb{R}^m, b} C \sum_{i=1}^{m} \ell(x_i, y_i, \sum_{j=1}^{m} c_j K(x_j, x_i), b) + \frac{1}{2} c^T K c + c^T K L_G K c$$

Kernel learning

- Laplacian SVM:

$$\min_{c \in \mathbb{R}^m, b} C \sum_{i=1}^{m} \ell(x_i, y_i, \sum_{j=1}^{m} c_j K(x_j, x_i), b) + \frac{1}{2} c^T K c + c^T K L_G K c$$

- What if we don't have labels y for some samples (the semi-supervised case)?
  - E.g. if only first k out of m samples have labels:

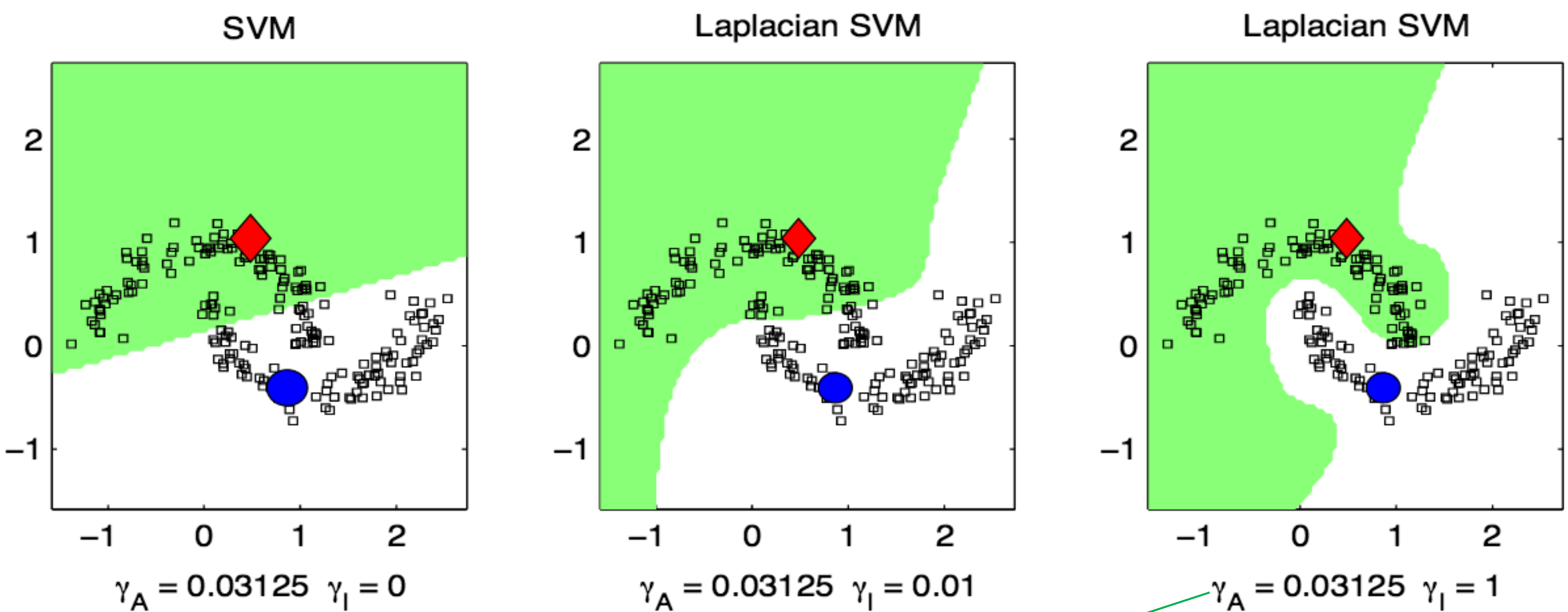
calculate loss only over the k samples with labels

$$\min_{c \in \mathbb{R}^m, b} C \sum_{i=1}^{\overset{k}{\cancel{m}}} \ell(x_i, y_i, \sum_{j=1}^{m} c_j K(x_j, x_i), b) + \frac{1}{2} c^T K c + c^T K L_G K c$$

But using kernel and laplacian for all m samples

Laplacian SVM

- Examples:

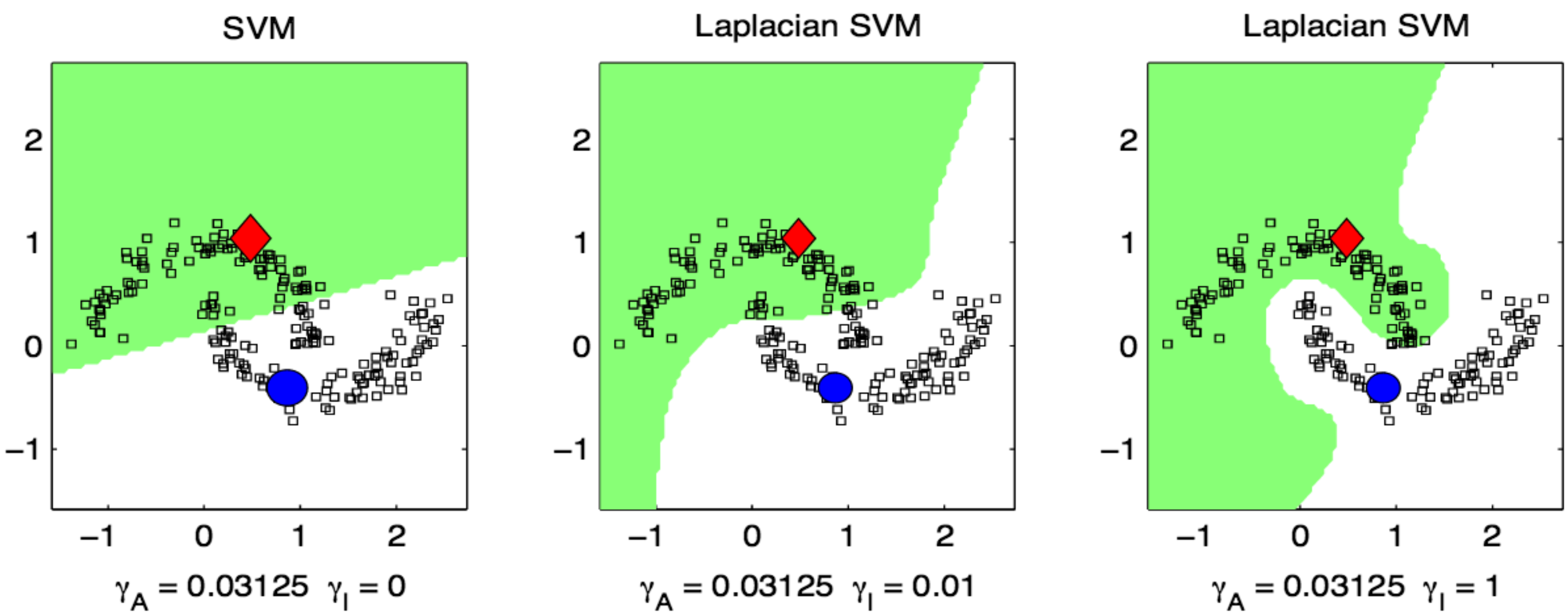


$$\min_{h \in \mathcal{H}, b} \quad C \sum_{i=1}^{l} \ell(x_i, y_i, h(x_i), b) + \frac{1}{2}||h||_{\mathcal{H}}^2 + 1/2||\nabla h||_E^2$$

$$\min_{c \in \mathbb{R}^m, b} \quad C \sum_{i=1}^{l} \ell(x_i, y_i, \sum_{j=1}^{m} c_j K[j,i], b) + \frac{1}{2} c^T K c + \frac{1}{2} c^T K L_G K c$$

Laplacian SVM

- Examples:



$$\Omega_G(\bar{h}) = \frac{1}{2}\sum\sum_{j,k=1}^{F} G_{j,k}\left(\bar{h}_j - \bar{h}_k\right)^2$$

Norm of "graph gradient" of h() along edges
how much h() changes along an edge?

$$\min_{h\in\mathcal{H},b} \quad C\sum_{i=1}^{l}\ell(x_i, y_i, h(x_i), b) + \frac{1}{2}||h||_{\mathcal{H}}^2 + 1/2||\nabla h||_E^2$$