Topic 6: THE KERNEL TRICK

STAT 37710/CMSC 25400 Machine Learning
Risi Kondor, The University of Chicago

Enriching linear classifiers

- The hypothesis space of linear classifiers is limited.
- Could be enriched by a **feature mapping** $\phi\colon \mathbf{x} \to \overline{\mathbf{x}}$ that throws in various nonlinear features, e.g.,

$$\overline{\mathbf{x}} = \left(x_1, x_2, \ldots, x_n, x_1x_2, x_3x_5x_9, \sin(x_1), e^{-(x_1-4)^2}, \ldots\right).$$

- However, this is ad-hoc and potentially very expensive.

The kernel trick

Observations:

1. For both the perceptron and the SVM, $\mathbf{w}$ is of the form $\mathbf{w} = \sum_{j=1}^{m} \gamma_j \mathbf{x}_j$.
2. The algorithm only uses $\mathbf{w}$ through inner products such as

$$\mathbf{w} \cdot \mathbf{x}_i = \sum_{j=1}^{m} \gamma_j \left(\mathbf{x}_j \cdot \boldsymbol{x}_i\right)$$

So all that we need are dot products $\mathbf{x}_i \cdot \mathbf{x}_j$, or, in the feature space, $\overline{\mathbf{x}}_i \cdot \overline{\mathbf{x}}_j$.

IDEA: Define the **kernel** $k(\mathbf{x}, \mathbf{x}') = \overline{\mathbf{x}} \cdot \overline{\mathbf{x}}'$, write everything in terms of k , and don't ever bother computing $\overline{\mathbf{x}} = \phi(\mathbf{x})$ or $\overline{\mathbf{x}}' = \phi(\mathbf{x}')$!

Beam me up, Scotty!

Inner product

In infinite dimensional spaces (e.g., function spaces) the notion of dot product $\mathbf{x} \cdot \mathbf{x}' = \mathbf{x}^\top \mathbf{x}'$ does not make sense. Instead, we have inner products.

Definition

An inner product on a vector space V (over $\mathbb{R}$) is a function $\langle \cdot, \cdot \rangle : V \to \mathbb{R}$ satisfying

1. $\langle u, v \rangle = \langle v, u \rangle$ (symmetry)
2. $\langle \alpha u, v \rangle = \alpha \langle u, v \rangle$ (linearity I)
3. $\langle u, v + w \rangle = \langle u, v \rangle + \langle u, w \rangle$ (linearity II)
4. $\langle u, u \rangle \geq 0$ with equality only if $u = 0$ (positivity)

for all $u, v, w \in V$ and $\alpha \in \mathbb{R}$.

Two vectors $u, v \in V$ are said to be **othogonal** if $\langle u, v \rangle = 0$.

Hilbert spaces

Definition

A vector space $\mathcal{H}$ is said to be a Hilbert space if

1. $\mathcal{H}$ has an inner product $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$;
2. $\mathcal{H}$ is complete with respect to the norm

$$\|x\| = \sqrt{\langle x, x \rangle}$$

induced by $\langle \cdot, \cdot \rangle$.

Hilbert spaces sound scary but are really just the natural generalization of $\mathbb{R}^n$ to possibly infinite dimesions. They are inner product in which the inner product works "as expected" and consequently most of linear algebra carries over to them with no problems.

The kernel trick (more exoplicitly)

Algorithms like the Perceptron and the SVM can work in any Hilbert space $\mathcal{H}$.

- To invoke the algorithm in $\mathcal{H}$, use a feature mapping $\phi\colon \mathcal{X} \to \mathcal{H}$.
- However, never compute $\phi(\mathbf{x})$ explicitly. Instead, use the kernel (pull-back of the inner product)

$$k(\mathbf{x}, \mathbf{x}') := \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle \, .$$

- Since $\mathcal{H}$ is typically much higher dimensional than $\mathcal{X}$, the decision surface in $\mathcal{X}$ corresponding to a linear classifier in $\mathcal{H}$ can be much more complex.

Learning algorithms that can exploit this trick are called Hilbert space methods or kernel methods.

The vanilla perceptron

$\mathbf{w} \leftarrow 0$;
$t \leftarrow 1$;
while(true){
  if $\mathbf{w} \cdot \mathbf{x}_t \geq 0$ predict $\hat{y}_t = 1$; else predict $\hat{y}_t = -1$;
  if $((\hat{y}_t = -1)$ and $(y_t = 1))$ let $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_t$;
  if $((\hat{y}_t = 1)$ and $(y_t = -1))$ let $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_t$;
  $t \leftarrow t + 1$;
}

At any time t, the weight vector is of the form

$$\mathbf{w} = \sum_{i=1}^{t-1} c_i \, \mathbf{x}_i \qquad \text{where} \qquad c_i \in \{-1, 0, +1\} \, .$$

The kernel perceptron

$t \leftarrow 1$;
while(1){
if $\sum_{i=1}^{t-1} c_i k(\mathbf{x}_i, \mathbf{x}_t) \geq 0$ predict $\hat{y}_t = 1$; else predict $\hat{y}_t = -1$;
$c_t \leftarrow 0$;
if $((\hat{y}_t = -1)$ and $(y_t = 1))$ let $c_t = 1$;
if $((\hat{y}_t = 1)$ and $(y_t = -1))$ let $c_t = -1$;
$t \leftarrow t + 1$;
}

The kernel SVM

Primal (forget about b)

$$\underset{w\in\mathcal{H},\xi_1,\ldots,\xi_m}{\text{minimize}}\ \frac{1}{2}\parallel w\parallel^2+\frac{C}{m}\sum_i \xi_i \qquad \text{s.t.} \qquad y_i\left\langle w,\phi(\mathbf{x}_i)\right\rangle \geq 1-\xi_i \quad \xi_i\geq 0$$

Dual

$$\underset{\alpha_1,\ldots,\alpha_m}{\text{maximize}}\, L(\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j)$$

$$\text{subject to} \quad \sum_i y_i\alpha_i = 0 \quad \text{and} \quad 0\leq \alpha_i\leq \frac{C}{m}\ \forall i$$

Predict according to

$$\widehat{y} = h(\mathbf{x}) = \text{sgn}(\langle w,\phi(x)\rangle) = \text{sgn}\Big(\sum_i \underbrace{y_i\alpha_i}_{\gamma_i} k(\mathbf{x},\mathbf{x}_i)\Big).$$

Kernels

So what does k need to satisfy?

1. Symmetry: $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = \langle \phi(\mathbf{x}'), \phi(\mathbf{x}) \rangle \implies k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$.
2. For any $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ and $c_1, \ldots, c_\ell \in \mathbb{R}$, letting $\xi = \sum_{i=1}^{\ell} c_i \, \phi(\mathbf{x}_i)$, must have $\langle \xi, \xi \rangle \geq 0 \implies$

$$\sum_{i=1}^{\ell} \sum_{j=1}^{\ell} c_i \, c_j \, \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} c_i \, c_j \, k(\mathbf{x}_i, \mathbf{x}_j) \geq 0.$$

In general, any function $k \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ satisfying 1. and 2. is called a symmetric, positive semi-definite (SPSD) kernel.

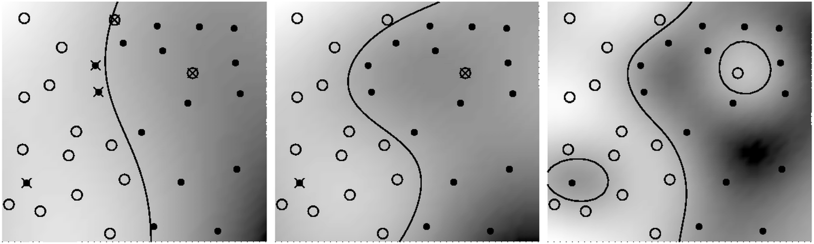
It turns out that this is all. Any SPSD kernel k has a corresponding $\mathcal{H}$ and $\phi \colon \mathcal{X} \to \mathcal{H}$ such that $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = k(\mathbf{x}, \mathbf{x}')$ for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ (will see this below).

Some typical kernels

- Linear: $k(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$ (boring!)
- Polynomial: $k(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x} \cdot \mathbf{x}')^p$
- Gaussian:
$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\| \mathbf{x} - \mathbf{x}' \|^2}{2\sigma^2}\right)$$
- Laplacian: $k(x, x') = e^{-|x-x'|/\lambda}$
- String, graph, etc…

Since the hypothesis is $h(\mathbf{x}) = \sum_i \alpha_i \, k(\mathbf{x}_i, \mathbf{x})$, the kernel is like a similarity measure.

SVM solutions with Gaussian kernel



Closure properties of kernels

If $k_1, k_2 \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ are SPSD kernels, then

- $k_+(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$
- $k_\times(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') \cdot k_2(\mathbf{x}, \mathbf{x}')$

are SPSD kernels.

If $k_1, k_2, \ldots \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a pointwise convergent sequence of SPSD kernels, then $\lim_{i\to\infty} k_i(\mathbf{x}, \mathbf{x}')$ is a SPSD kernel.

If $k_1 \colon \mathcal{X}_1 \times \mathcal{X}_1 \to \mathbb{R}$ is an SPSD kernel on $\mathcal{X}_1$ and $k_2 \colon \mathcal{X}_2 \times \mathcal{X}_2 \to \mathbb{R}$ is a SPSD kernel on $\mathcal{X}_2$, then

$$k((x_1, x_2), (x_1', x_2')) = k_1(x_1, x_1') \cdot k_2(x_2, x_2')$$

is a PSD kernel on $\mathcal{X}_1 \times \mathcal{X}_2$.

Support vector machine example

Performance on classifying full MNIST dataset:

Classifier	Test Error
linear	8.4%
3-nearest-neighbor	2.4%
RBF-SVM	1.4 %
Tangent distance	1.1 %
LeNet	1.1 %
Boosted LeNet	0.7 %
Translation invariant SVM	0.56 %

Kernel mystique

- But what does the kernel really capture about the data?
- What is this magical Hilbert space $\mathcal{H}$?
- If $\mathcal{H}$ is so high dimensional, what stops the SVM from overfitting like crazy?
- How does the kernel SVM control the complexity of the returned hypothesis?

Reproducing kernel Hilbert spaces

Reproducing kernel Hilbert space

Let $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel.

1. Define $k_x\colon \mathcal{X} \to \mathbb{R}$ as the function $k_x(x') = k(x, x')$.
2. Define the function space $\mathcal{H}_k^{\text{pre}}$ as the space of linear combinations

$$\xi(x') = \sum_{i=1}^{m} \alpha_i k_{x_i}(x')$$

   for any $m \in \mathbb{N}$, any $x_1, \ldots, x_m \in \mathcal{X}$ and $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$.
3. Define $\langle k_x, k_{x'} \rangle = k(x, x')$ and extend it to the rest of $\mathcal{H}_k^{\text{pre}}$ by linearity.
4. Add to $\mathcal{H}_k^{\text{pre}}$ the limit points of all Cauchy sequences.

The resulting space $\mathcal{H}_k$ is the Reproducing Kernel Hilbert Space (RKHS) induced by k . Clearly, if $\phi\colon x \mapsto k_x$, then $k(x, x') = \langle \phi(x), \phi(x') \rangle$ proving that for any PSD kernel there is a $\mathcal{H}$ and an ϕ which realize it in this way.

The reproducing property

For any $f \in \mathcal{H}_k$ expressible as a finite linear combination

$$f(x) = \sum_{i=1}^{m} \alpha_i k_{x_i}(x),$$

function evaluation and the inner product are linked by the remarkable property

$$f(x) = \sum_{i=1}^{m} \alpha_i k(x_i, x) = \sum_{i=1}^{m} \alpha_i \langle k_{x_i}, k_x \rangle = \left\langle \sum_{i=1}^{m} \alpha_i k_{x_i}, k_x \right\rangle = \langle f, k_x \rangle .$$

Regularized Risk Minimization

RRM in RKHS

Recall that at the abstract level many ML algorithms just perform some form of regularized risk minimization:

$$\widehat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \left[\underbrace{\frac{1}{m} \sum_{i=1}^{m} \ell(f(x_i), y_i)}_{\text{training error}} + \underbrace{\Omega(f)}_{\text{regularizer}} \right].$$

In general, searching an infinite dimensional space of functions on a computer is pretty challenging. However, if $\mathcal{F} = \mathcal{H}_k$ and $\Omega(f) = \|f\|^2$, then

$$f(x_i) = \langle f, k_{x_i} \rangle, \qquad \Omega(f) = \langle f, f \rangle,$$

so it reduces to a problem in linear algebra!

The Representer Theorem

Theorem (Wahba)

Let $k\colon \mathcal{X} \to \mathcal{X} \to \mathbb{R}$ be a PSD kernel and $\mathcal{H}_k$ be the corresponding RKHS. Then for any loss function ℓ and any monotonically increasing $\chi\colon \mathbb{R} \to \mathbb{R}$, the solution to

$$\underset{f \in \mathcal{H}_k}{\text{minimize}} \frac{1}{m} \sum_{i=1}^{m} \ell(f(x_i), y_i) + \chi(\|f\|_{\mathcal{H}_k})$$

is of the form

$$f(x) = \sum_{i=1}^{m} \alpha_i \, k(x_i, x).$$

This is the key to making kernel machines implementable on computers, since we only need to optimize for the m real numbers $\alpha_1, \ldots, \alpha_m$.

[Wahba, 198]

RRM form of the SVM

$$\underset{w\in\mathcal{H}_k,\,\xi_1,\ldots,\xi_m}{\text{minimize}} \left[\frac{1}{2}\|w\|^2+\frac{C}{m}\sum_i \xi_i\right] \quad \text{s.t.} \quad y_i\langle w,\phi(\mathbf{x}_i)\rangle \geq 1-\xi_i \quad \xi_i \geq 0$$

$$\Updownarrow$$

$$\underset{f\in\mathcal{H}_k,\,\xi_1,\ldots,\xi_m}{\text{minimize}} \left[\frac{1}{2}\|f\|^2+\frac{C}{m}\sum_i \xi_i\right] \quad \text{s.t.} \quad y_i f(x_i) \geq 1-\xi_i \quad \xi_i \geq 0.$$

$$\Updownarrow$$

$$\underset{f\in\mathcal{H}_k,\xi_1,\ldots,\xi_m}{\text{minimize}} \left[\frac{1}{2}\|f\|^2+\frac{C}{m}\sum_i (1-y_i f(x_i))_{\geq 0}\right]$$

So the soft margin SVM is a special case of RRM with $\mathcal{F}=\mathcal{H}_k$,

$$\ell(\widehat{y},y)=(1-y\,\widehat{y})_{\geq 0}, \qquad \Omega(f)=\frac{1}{2C}\|f\|^2, \qquad h(x)=\operatorname{sgn}(f(x)).$$

Question: But what does the regularizer $\|f\|^2$ express?

The RKHS of the Gaussian kernel

Theorem (Girosi et al., 1995)

For the Gaussian kernel $k(x, x') = \exp(-\| x - x' \|^2 / (2\sigma^2))$ on $\mathbb{R}^n$,

$$\| f \|^2_{\mathcal{H}_k} \propto \int e^{\sigma^2 \omega^2 / 2} \, | \tilde{f}(\omega) |^2 d\omega,$$

where

$$\tilde{f}(\omega) = \int f(x) \, e^{-2\pi i \omega \cdot x} \, dx$$

is the Fourier transform of f.

$\rightarrow$ support vector machines with the Gaussian kernel totally make sense!

Three faces of the kernel

1. Inner product in feature space — mostly magic.
2. Measure of similarity between data points — pragmatic.
3. Responsible for regularization — makes sense.

Examples of Kernels

Kernels

A kernel $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ may be specified either

- Explicitly, like the Gaussian RBF kernel

$$k(x, x') = \exp(-\|x - x'\|^2/(2\sigma^2)).$$

- Implicitly, via some algorithm that computes $k(x, x')$ for given x, x'.

Criteria for a good kernel:

- Positive semi-definiteness (and, of course, symmetry).
- Good notion of similarity between data / good regularizer.
- Efficiently computable.

Note: $\mathcal{X}$ can be almost anything, doesn't need to be $\mathbb{R}^d$.

Kernels between distributions

Let $\mathcal{X}$ be the space of distributions on some space $\mathcal{S}$. **Bhattacharyya kernel:**

$$k_{\mathrm{Bhatta}}(p, p') = \int \sqrt{p(x)}\sqrt{p'(x)}\, dx$$

Question: Is this a valid kernel (i.e., PSD)? In general, difficult to compute, however if $p = \mathcal{N}(\mu, \Sigma)$ and $p' = \mathcal{N}(\mu', \Sigma')$, then

$$\begin{aligned} k_{\mathrm{Bhatta}}(p, p') = &\, |\,\Sigma\,|^{-1/4}\,|\,\Sigma'\,|^{-1/4}\,|\,\Sigma^{\dagger}\,|^{1/2} \\ &\exp\left(-\frac{1}{4}\mu^{\top}\Sigma^{-1}\mu - \frac{1}{4}\mu'^{\top}\Sigma'^{-1}\mu' + \frac{1}{2}\mu^{\dagger\top}\Sigma^{\dagger}\mu^{\dagger}\right) \end{aligned} \quad (1)$$

where $\Sigma^{\dagger} = \left(\frac{1}{2}\Sigma^{-1} + \frac{1}{2}\Sigma'^{-1}\right)^{-1}$ and $\mu^{\dagger} = \frac{1}{2}\Sigma^{-1}\mu + \frac{1}{2}\Sigma'^{-1}\mu'$.

[K & Jebara, 2003]

Contiguous substring kernel

Let Σ be an alphabet and let $n_u(a)$ denote the number of places that the string $u \in \Sigma^\ell$ appears in the string $a \in \Sigma^*$ as a contiguous substring. Given two strings $a, b \in \Sigma^*$ the contiguous substring kernel is

$$k(a, b) = \sum_{u \in \Sigma^\ell} n_u(a)\, n_u(b).$$

- Question: Is this a valid kernel? Yes, because it corresponds to the feature map $\phi \colon \Sigma^* \to \mathbb{N}^{\Sigma^\ell}$ with $[\phi(a)]_u = n_u(a)$.
- Question: What is the complexity of computing it? $O(\ell\,|\,a\,|\,|\,b\,|)$

Gappy substring kernels

Given an index sequence $\boldsymbol{i} = (i_1, \ldots, i_\ell)$ with $1 \leq i_i < i_2 < \ldots < i_\ell$ let $\boldsymbol{a_i}$ denote the gappy substring $a_{i_1} a_{i_2} \ldots a_{i_\ell}$. Let $n_u^{\text{gap}}(\boldsymbol{a}) = |\{ \boldsymbol{i} \mid \boldsymbol{a_i} = u \}|$. Consider the kernel

$$k_\ell(\boldsymbol{a}, \boldsymbol{b}) = \sum_{u \in \Sigma^\ell} n_u^{\text{gap}}(\boldsymbol{a})\, n_u^{\text{gap}}(\boldsymbol{b}).$$

- Question: Is this a valid kernel? Yes.
- Question: What is the complexity of computing it? $O(\ell\,|\boldsymbol{a}|\,|\boldsymbol{b}|)$ using the dynamic programming recursion

$$k_\ell(\boldsymbol{a}_{1:i+1}, \boldsymbol{b}_{1:j}) = k_\ell(\boldsymbol{a}_{1:i}, \boldsymbol{b}_{1:j}) + \sum_{p=\ell}^{j} \mathbb{I}(a_{i+1} = b_p)\, k_{\ell-1}(\boldsymbol{a}_{1:i}, \boldsymbol{b}_{1:p-1})$$

Random walk kernel on graphs

Let $G = (V, E)$ be a graph with adjacency matrix A , and $\mathrm{path}_p(x, x')$ be the set of all path from x to x' in G of length p . Consider

$$k_{2\ell}(x, x') = |\,\mathrm{path}_{2\ell}(x, x')\,|\,.$$

- Question: Is this a valid kernel? Yes.
- Question: What is the complexity of computing it?

$$k_{2\ell}(x, x') = [A^{2\ell}]_{x,x'}$$

Question: What is the problem with the random walk kernel?

[Gärtner]

Diffusion kernel on graphs

Imagine a lazy random walker, which, when he is at a vertex of degree d

- with probability dp moves to one of the neighbors (selected randomly)
- with probability $1 - dp$ stays in place.

The transition matrix of this process is $T = I + pL$, where L is the **graph Laplacian**

$$[L]_{i,j} = \begin{cases} 1 & i \sim j \\ -d_i & i = j \\ 0 & \text{otherwise.} \end{cases}$$

After n timesteps the distribution of the random walker is given by T^n. Now take the continuous limit where $n \to \infty$ and simultaneously $p = 1/n$.

Diffusion kernel on graphs

After time t the distribution is given by

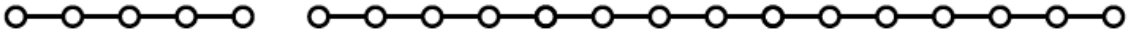
$$K_t = e^{tL} = \lim_{n\to\infty}\left(I + \frac{tL}{n}\right)^n = I + tL + \frac{t^2}{2}L^2 + \frac{t^3}{6}L^3 + \ldots .$$

The **diffusion kernel** on G with parameter t is $k_t(x, x') = [K_t]_{x,x'}$.

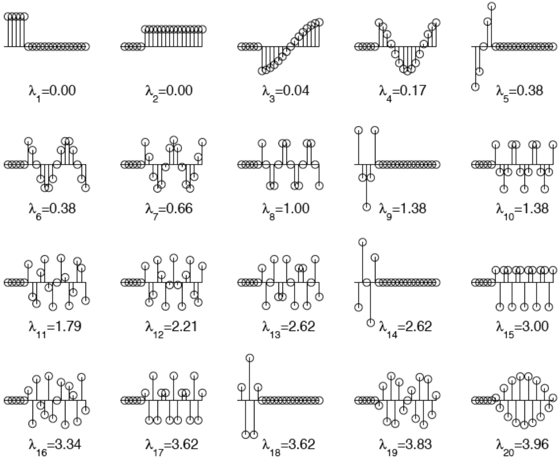
- Question: Is this a valid kernel? Yes, the exponential of a symmetric matrix is always PSD.
- Question: What is its computational complexity? Typically $O(|V|^3)$.

[K. & Lafferty, 2002]

Eigenvectors of the graph Laplacian



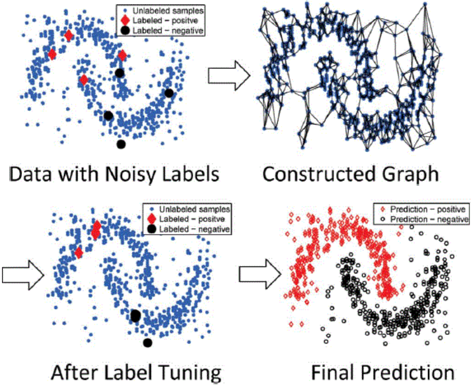
(a) a linear unweighted graph with two segments



(b) the eigenvectors and eigenvalues of the Laplacian L

[Zhu, Ghahramani & Lafferty, 2003]

Graph kernels for semi-supervised ML



- Often labeled data is expensive but unlabled data is abundant.
- Use the unlabeled data to construct a graph (mesh) $\rightarrow$ graph kernel k.
- Supervised learning with k enforces that f be smooth wrt. this graph.

FURTHER READING

- B. Schölkopf and A. J. Smola: **Learning with Kernels**
- N. Christianini and J–S Taylor: **An Introduction to Support Vector Machines...**
- I. Steinwart: **Support Vector Machines**