

How to learn from very few examples?

Dengyong Zhou

Department of Empirical Inference
Max Planck Institute for Biological Cybernetics
Spemannstr. 38, 72076 Tuebingen, Germany

MAX-PLANCK-GESELLSCHAFT

Outline

- Introduction
- Part A — Algorithms
- Part B — Theory
- Part C — Related work
- Conclusions

Problem setting

Consider an **input space** $\mathcal{X} = \{x_1, \dots, x_l, x_{l+1}, \dots, x_n\}$ and **output space** $\mathcal{Y} = \{-1, 1\}$.

- Observes the labels of the first l points: $(x_1, y_1), \dots, (x_l, y_l)$.
- Predicts the labels of the remaining points: x_{l+1}, \dots, x_n .

The elements of the input space can be everything of your interest, e.g., web pages, images, proteins and so on.

K-Nearest-Neighbor (*k*-NN) classification

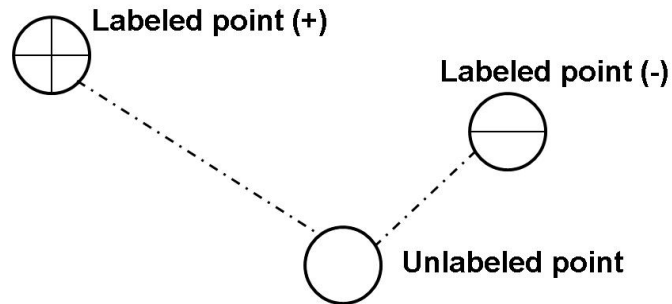
The predicted class is chosen by majority vote amongst the *k*-nearest labeled neighbors, i.e.,

$$y_u = \frac{1}{k} \sum_{i \in N_k(u)} y_i, \text{ for any } l < u \leq n.$$

K-Nearest-Neighbor (*k*-NN) classification

The predicted class is chosen by majority vote amongst the *k*-nearest labeled neighbors, i.e.,

$$y_u = \frac{1}{k} \sum_{i \in N_k(u)} y_i, \text{ for any } l < u \leq n.$$

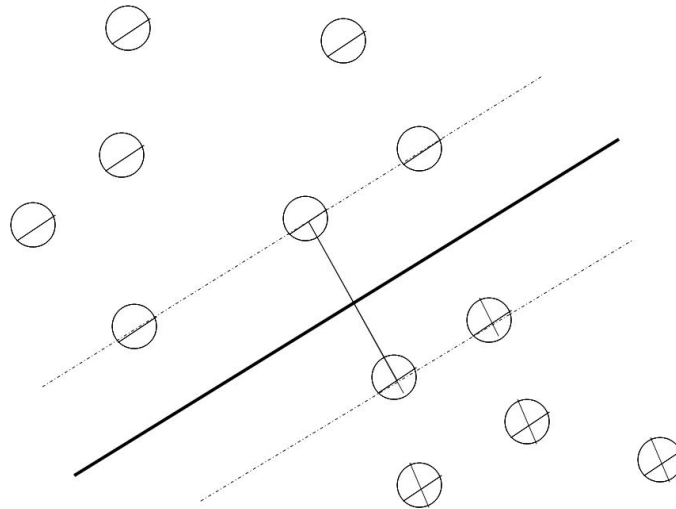


Support Vector Machines (SVMs)

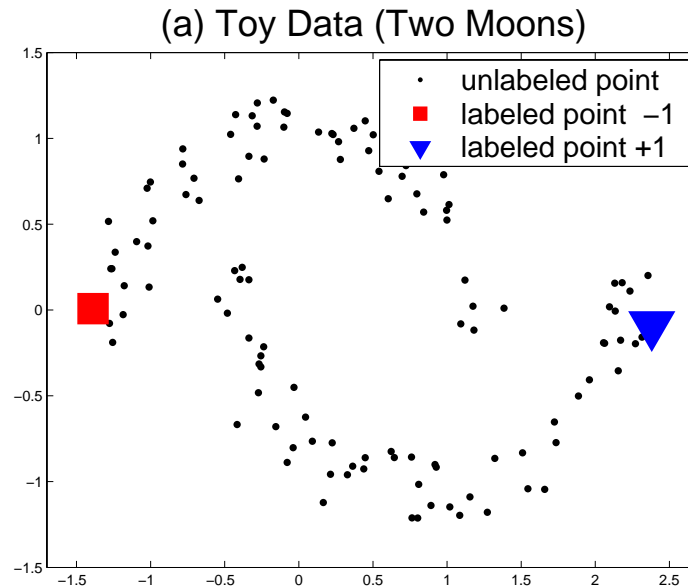
Using the **optimal separating hyperplane** to separate the two classes and maximize the distance to the closest point from either class (Boser et al., 1992; Cortes and Vapnik, 1995).

Support Vector Machines (SVMs)

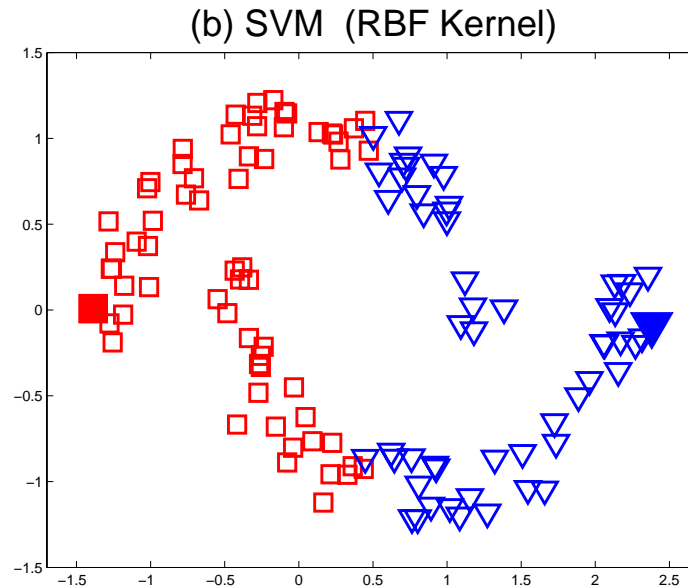
Using the **optimal separating hyperplane** to separate the two classes and maximize the distance to the closest point from either class (Boser et al., 1992; Cortes and Vapnik, 1995).



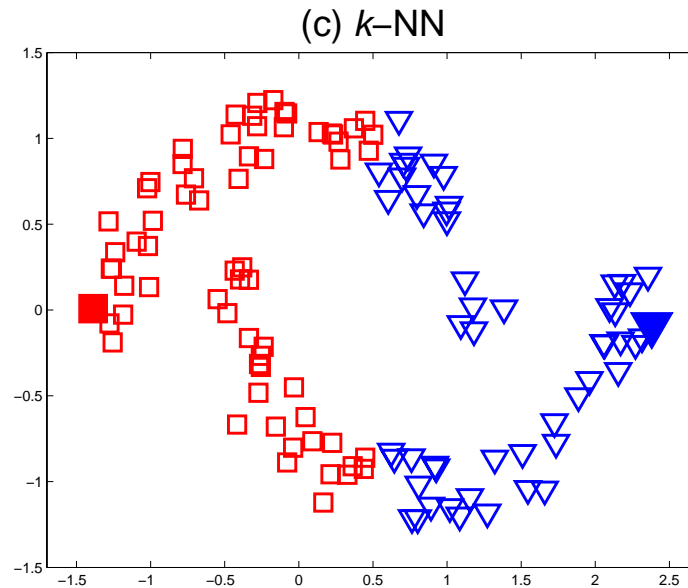
A toy classification problem: two moons



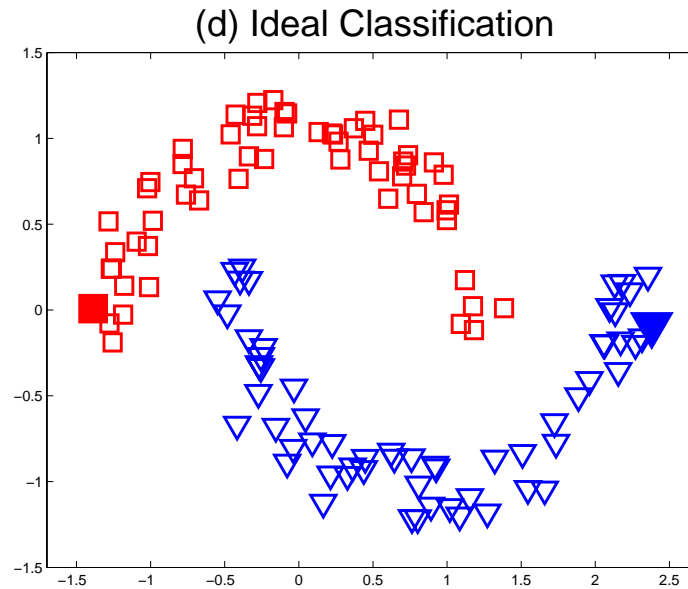
The classification result from SVMs



The classification result from k -NN



The ideal classification



Part A—Algorithms

A powerful but simple classification algorithm

1. Define a $n \times n$ affinity matrix W in which the elements are non-negative, symmetric, and furthermore the diagonal elements are zeros.

A powerful but simple classification algorithm

1. Define a $n \times n$ affinity matrix W in which the elements are non-negative, symmetric, and furthermore the diagonal elements are zeros.
2. Construct the matrix $S = D^{-1/2} W D^{-1/2}$ in which D is a diagonal matrix with its (i, i) -element equal to the sum of the i -th row of W .

A powerful but simple classification algorithm

1. Define a $n \times n$ affinity matrix W in which the elements are non-negative, symmetric, and furthermore the diagonal elements are zeros.
2. Construct the matrix $S = D^{-1/2}WD^{-1/2}$ in which D is a diagonal matrix with its (i, i) -element equal to the sum of the i -th row of W .
3. For each point i , iterate $f_i^{t+1} = \alpha \sum_{i,j} S_{ij} f_j^t + (1 - \alpha)y_i$ until convergence, where α is a parameter in $(0, 1)$.

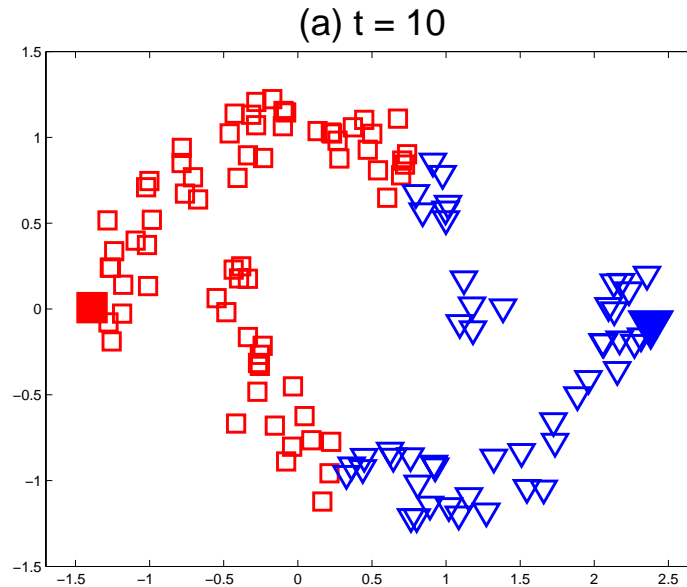
A powerful but simple classification algorithm

1. Define a $n \times n$ affinity matrix W in which the elements are non-negative, symmetric, and furthermore the diagonal elements are zeros.
2. Construct the matrix $S = D^{-1/2} W D^{-1/2}$ in which D is a diagonal matrix with its (i, i) -element equal to the sum of the i -th row of W .
3. For each point i , iterate $f_i^{t+1} = \alpha \sum_{i,j} S_{ij} f_j^t + (1 - \alpha) y_i$ until convergence, where α is a parameter in $(0, 1)$.
4. Let f_i^* denote the limit of the sequence $\{f_i^t\}$, and label point i as $\text{sgn} f_i^*$.

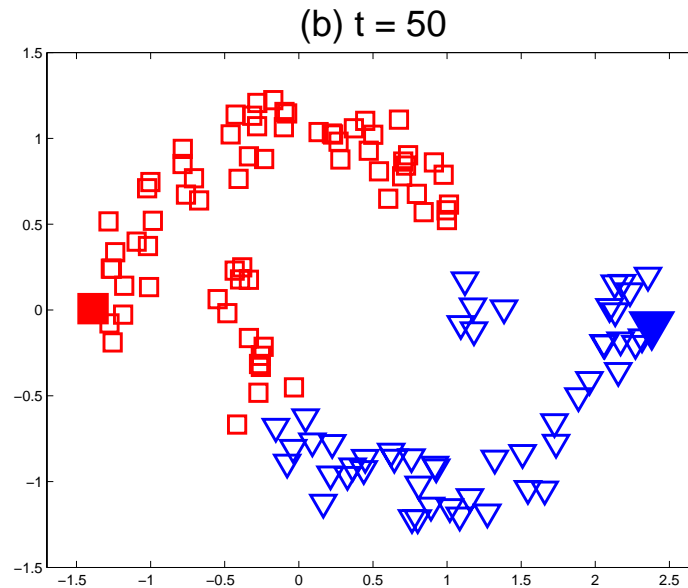
Define an affinity matrix

- For vectorial data, the affinity matrix can typically be defined by a Gaussian $W_{ij} = \exp(-\|x_i - x_j\|^2/2\sigma^2)$ except that $W_{ii} = 0$, where $\|\cdot\|$ represents Euclidean norm.
- For (undirected) graph data, the affinity matrix can be defined by $W_{ij} = 1$ if points i and j are connected by an edge, and 0 otherwise.

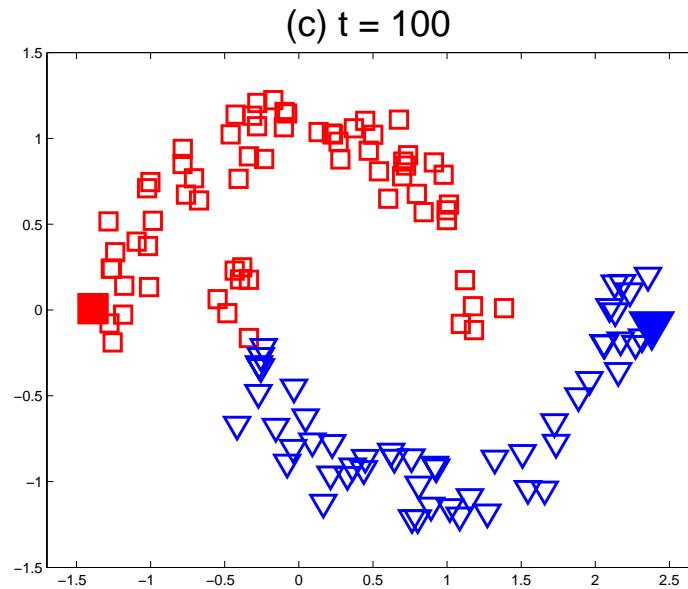
Solving the two-moon problem



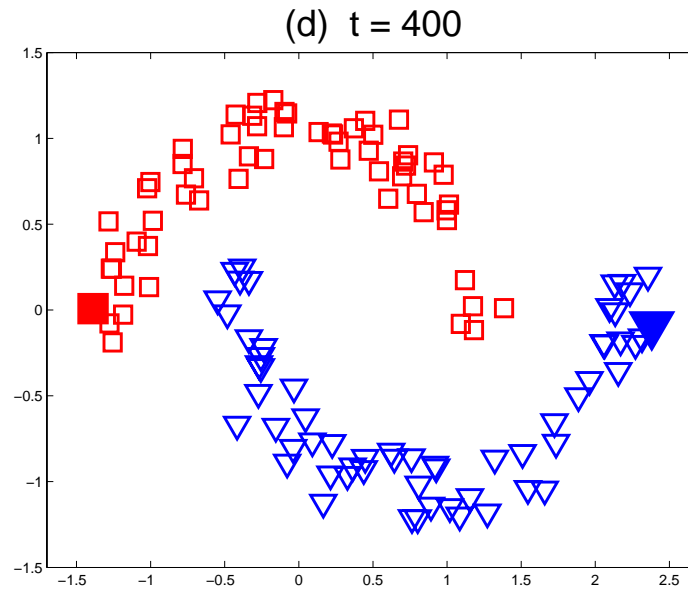
Solving the two-moon problem



Solving the two-moon problem



Solving the two-moon problem



Proof of the convergence

Theorem *Let y be a $n \times 1$ vector with $y_i = 1$ or -1 if point i is labeled as plus or minus, and 0 otherwise. Then $f^* = (1 - \alpha)(I - \alpha S)^{-1}y$.*

Proof. It is easy to see that

$$f^t = (\alpha S)^t f^0 + (1 - \alpha) \sum_{i=0}^{t-1} (\alpha S)^i y.$$

Since $0 < \alpha < 1$ and the eigenvalues $\lambda(S)$ of S satisfy $|\lambda(S)| \leq 1$,

$$\lim_{t \rightarrow \infty} (\alpha S)^t = 0, \text{ and } \lim_{t \rightarrow \infty} \sum_{i=0}^{t-1} (\alpha S)^i = (I - \alpha S)^{-1}.$$

Hence $f^* = (1 - \alpha)(I - \alpha S)^{-1}y$.

Describe the algorithm again

1. Define a $n \times n$ affinity matrix W in which the elements are non-negative, symmetric, and furthermore the diagonal elements are zeros.
2. Construct the matrix $S = D^{-1/2}WD^{-1/2}$ in which D is a diagonal matrix with its (i, i) -element equal to the sum of the i -th row of W .
3. Compute $f = (I - \alpha S)^{-1}y$, where α is a parameter in $(0, 1)$, and assign a label $\text{sgn}(f_i)$ to point i .

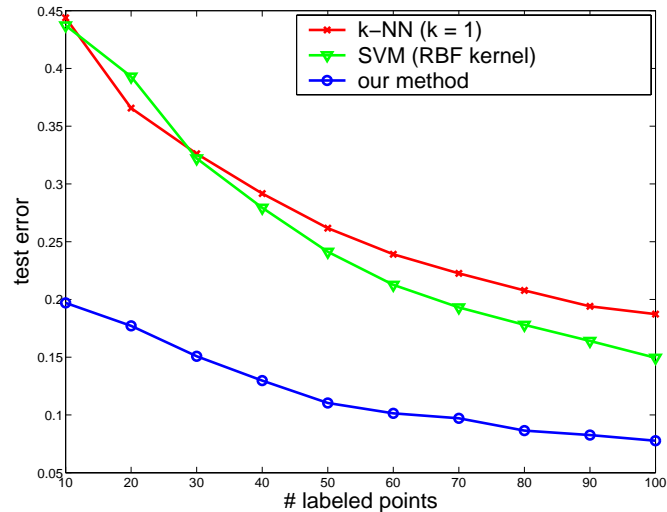
Multi-class classification

- Multi-class representation

$$Y = \begin{array}{ccc|l} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{array}{l} \text{Class A} \\ \text{Class B} \\ \text{Class C} \\ \hline \text{Unlabeled Data} \end{array} \end{array}$$

- Compute $F = (I - \alpha S)^{-1}Y$, and label each point i as $\arg \max_j F_{ij}$.

Hand-written digit recognition



Digit recognition with USPS handwritten 16x16 digits dataset for a total of 9298. The panel shows the test errors for the different algorithms with the number of labeled points increasing from 10 to 100.

An optimization framework

Theorem Define a cost function associated with function f to be

$$\Omega(f) = \frac{1}{2} \left\{ \sum_{i,j=1}^n W_{ij} \left(\frac{1}{\sqrt{D_{ii}}} f_i - \frac{1}{\sqrt{D_{jj}}} f_j \right)^2 + \mu \sum_{i=1}^n (f_i - y_i)^2 \right\},$$

where $\mu > 0$ is the regularization parameter. Then the closed form solution of the iterative algorithm is just

$$f^* = \arg \min_{f \in \mathbb{R}^n} \Omega(f).$$

Remark The first term in the cost function is called the **smoothness term**, which means that a good classifying function should not change too much between nearby points. The second term is the **fitting term**, which means a good classifying function should not change too much from the initial label assignment.

Proof. Differentiating $\Omega(f)$ with respect to f , we have

$$\left. \frac{\partial \Omega(f)}{\partial f} \right|_{f=f^*} = f^* - Sf^* + \mu(f^* - y) = 0,$$

which can be transformed into

$$f^* - \frac{1}{1 + \mu} Sf^* - \frac{\mu}{1 + \mu} y = 0.$$

Let us introduce a variable $\alpha = 1/(1 + \mu)$. Then

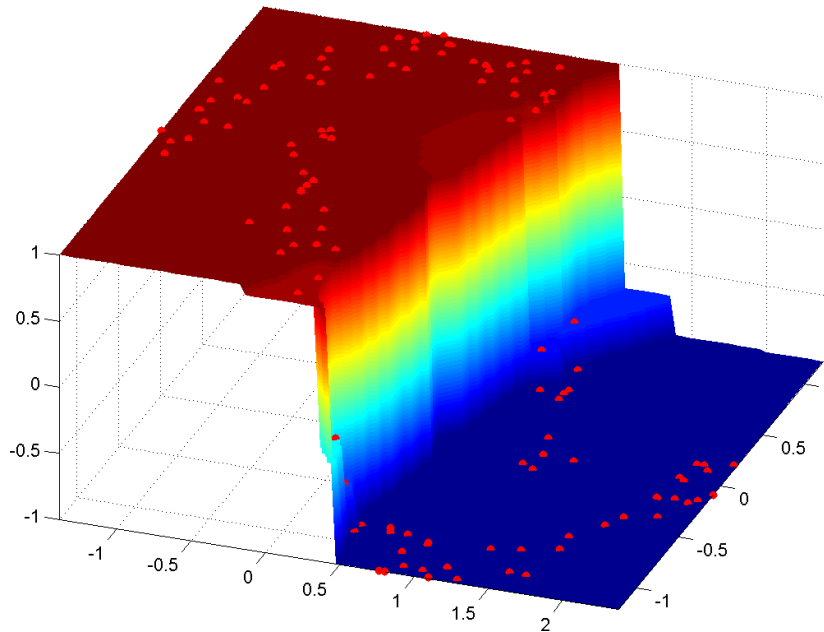
$$(I - \alpha S)f^* = (1 - \alpha)y,$$

Since $I - \alpha S$ is invertible, we have

$$f^* = (1 - \alpha)(I - \alpha S)^{-1}y.$$

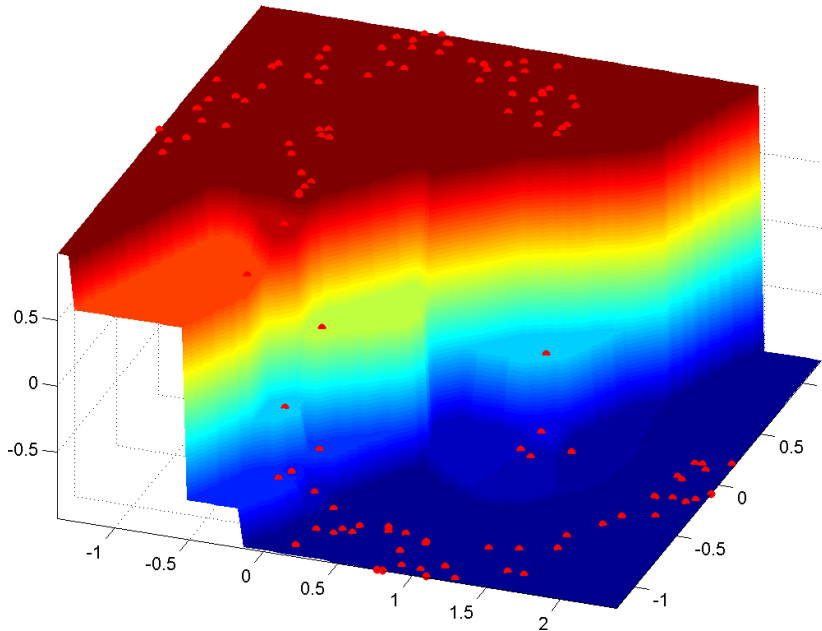
Illustrate the smooth function

(a) $t = 10$



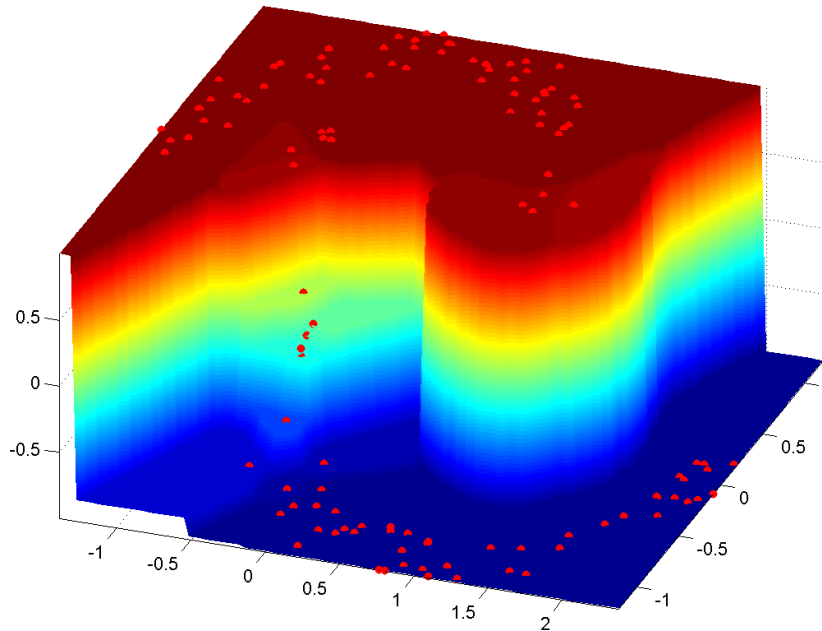
Illustrate the smooth function

(b) $t = 50$



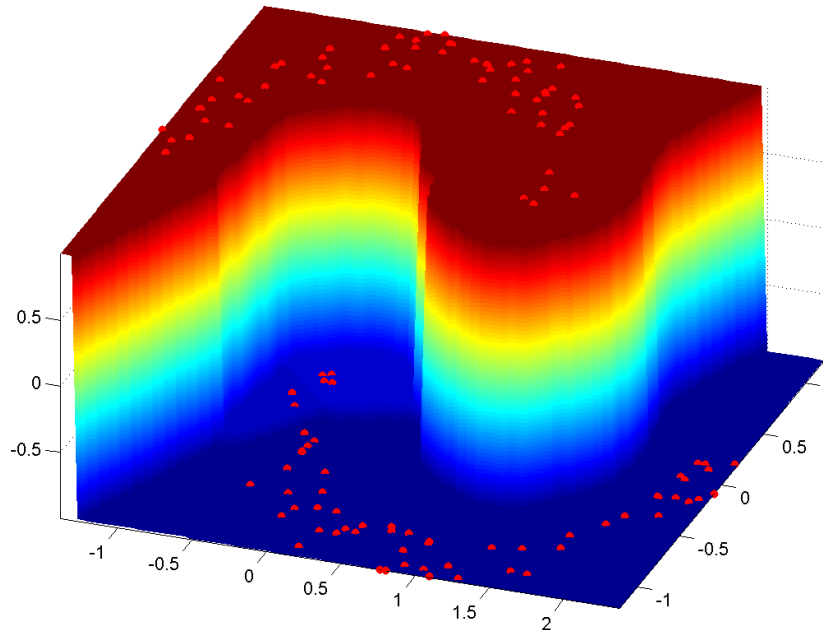
Illustrate the smooth function

(c) $t = 100$



Illustrate the smooth function

(d) $t = 400$



The ranking problem

Given an **input space** $\mathcal{X} = \{x_0, x_1, \dots, x_n\} \in \mathbb{R}^m$, the first point is the **query**. The goal is to rank the remaining points with respect to their **relevances** or **similarities** to the query.

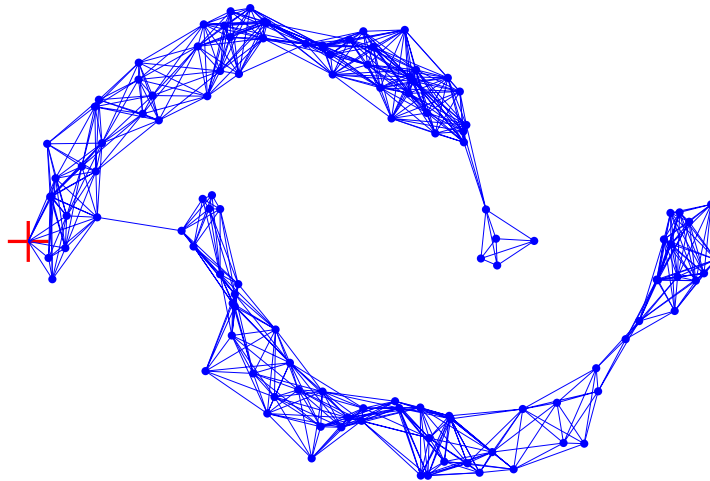
[See also (e.g., Crammer and Singer, 2001; Freund et al., 2004) for other ranking work in the machine learning community.]

A powerful but simple ranking algorithm

1. Sort the pairwise Euclidean distances between points in ascending order. Repeat connecting the two points with an edge according the order until a connected graph is obtained.
2. Form the affinity matrix W defined by $W_{ij} = \exp[-d^2(x_i, x_j)/2\sigma^2]$ if there is an edge linking x_i and x_j . Note that $W_{ii} = 0$ because there are no loops in the graph.
3. Symmetrically normalize W by $S = D^{-1/2}WD^{-1/2}$ in which D is the diagonal matrix with (i, i) -element equal to the sum of the i -th row of W .
4. Compute $f = (I - \alpha S)^{-1}y$, and rank each point i according the function value f_i^* on it (largest ranked first).

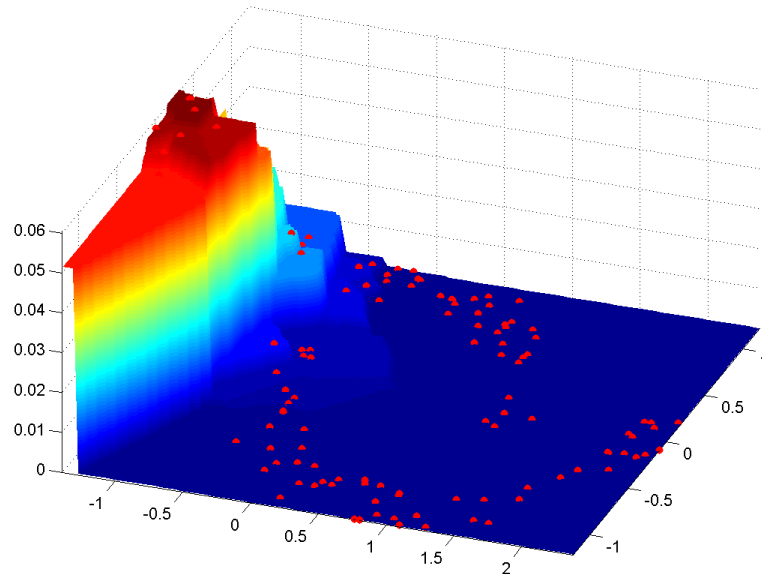
A toy ranking problem

(a) Connected graph



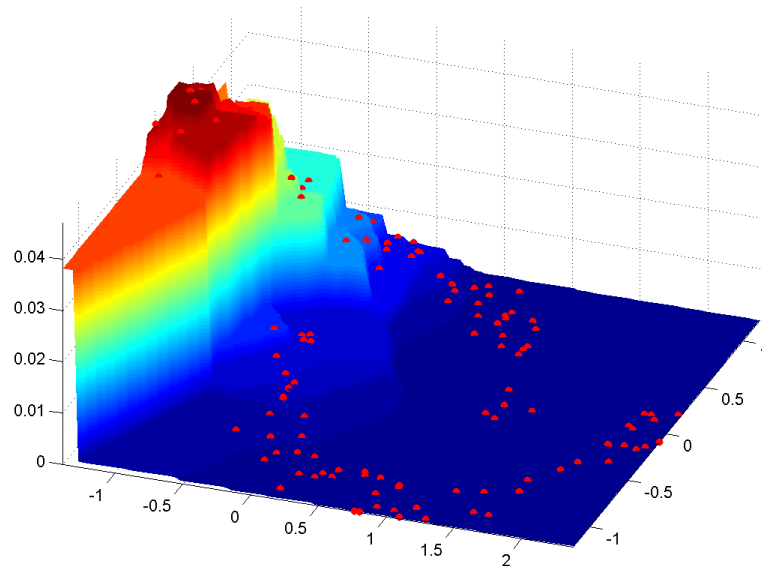
A toy ranking problem

(b) $t = 5$



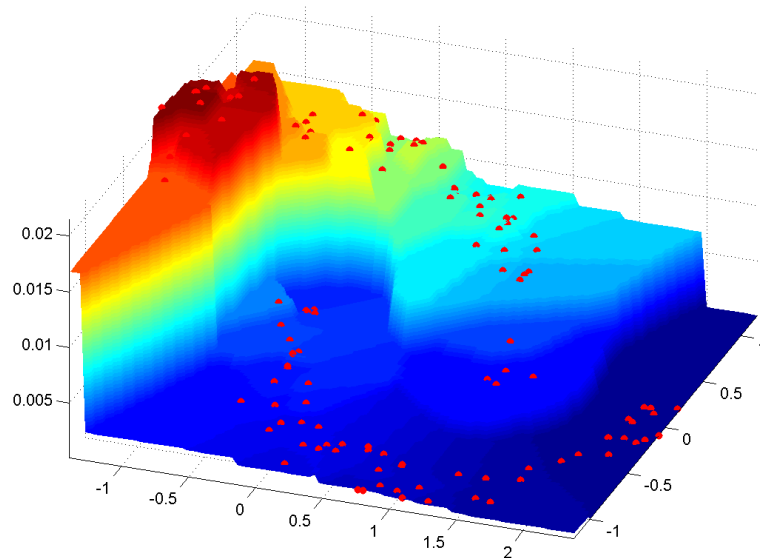
A toy ranking problem

(c) $t = 10$



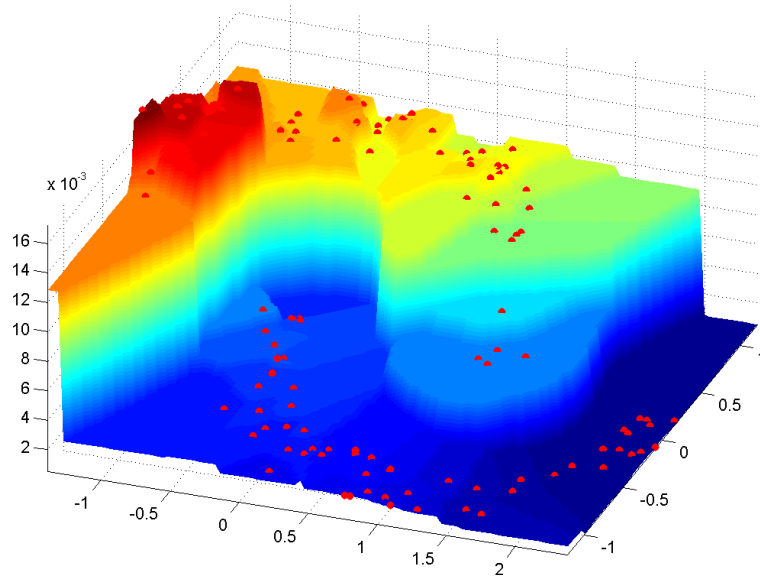
A toy ranking problem

(d) $t = 50$



A toy ranking problem

(e) $t = 100$



Euclidean distance based ranking

(f) Euclidean distance

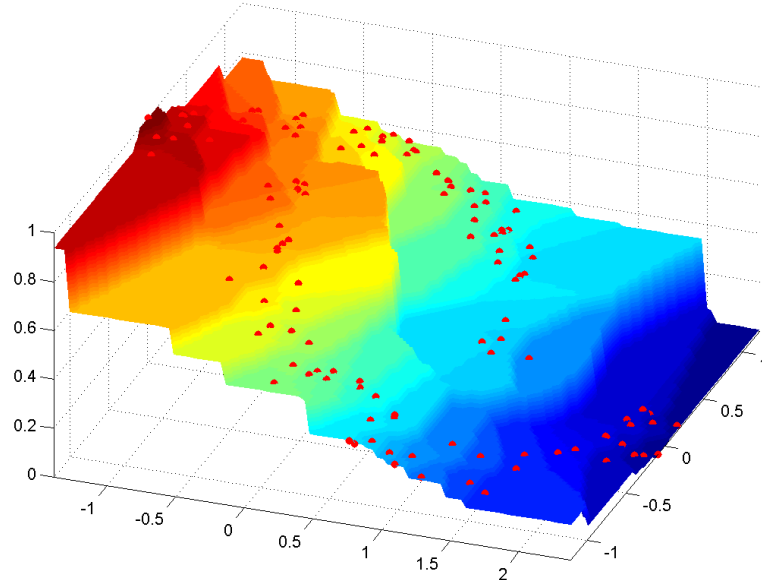


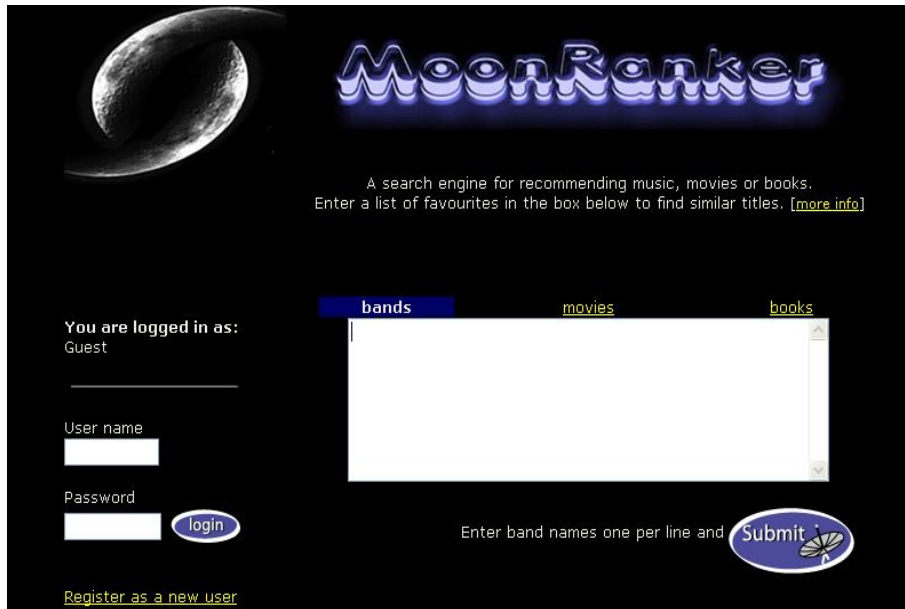
Image ranking



Ranking digits in USPS. The top-left digit in each panel is the query. The left panel shows the top 99 by our method; and the right panel shows the top 99 by the Euclidean distance based ranking. Note that in addition to 3s there are many more 2s with knots in the right panel.

MoonRanker: A recommendation system

<http://www.moonranker.com/>



The image shows the MoonRanker website interface. At the top left is a large image of the Moon. The title 'MoonRanker' is displayed in a stylized, glowing blue font. Below the title, a description reads: 'A search engine for recommending music, movies or books. Enter a list of favourites in the box below to find similar titles. [\[more info\]](#)'. On the left side, there is a login section with the text 'You are logged in as: Guest' and a horizontal line. Below this are input fields for 'User name' and 'Password', followed by a blue 'login' button. On the right side, there are three tabs: 'bands' (selected), 'movies', and 'books'. Below the 'bands' tab is a large white text input area. At the bottom right, there is a blue 'Submit' button with a paper plane icon. Below the input area, the text 'Enter band names one per line and' is visible. At the bottom left, there is a link 'Register as a new user'.

MoonRanker

A search engine for recommending music, movies or books.
Enter a list of favourites in the box below to find similar titles. [\[more info\]](#)

You are logged in as:
Guest

User name

Password
 [login](#)

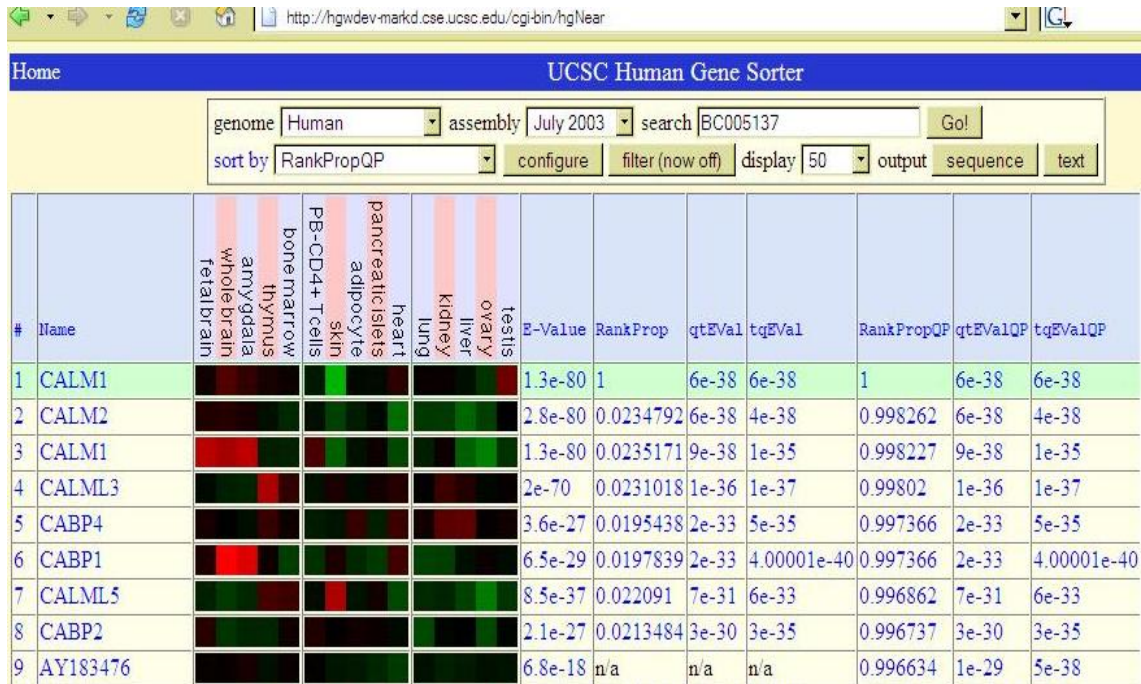
[bands](#) [movies](#) [books](#)

Enter band names one per line and [Submit](#)

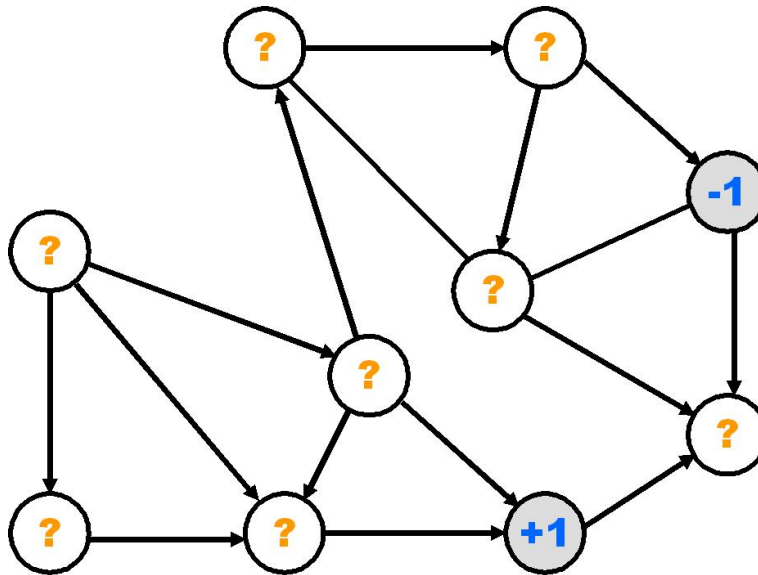
[Register as a new user](#)

Protein ranking

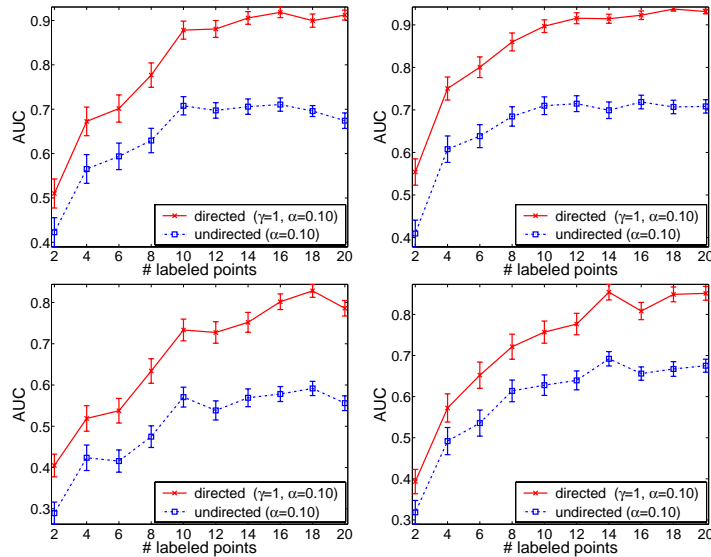
(With J. Weston, A. Elisseeff, C. Leslie and W.S. Noble) **Protein ranking: from local to global structure in the protein similarity network.** Proceedings of the National Academy of Sciences (PNAS) 101(17) (2004).



Classification and ranking on directed graphs



The importance of directionality



Classification on the WebKB dataset: student vs. the rest in each university. Taking the directionality of edges into account can yield substantial accuracy gains.

Part B—Theory

Analysis, geometry and regularization on discrete spaces

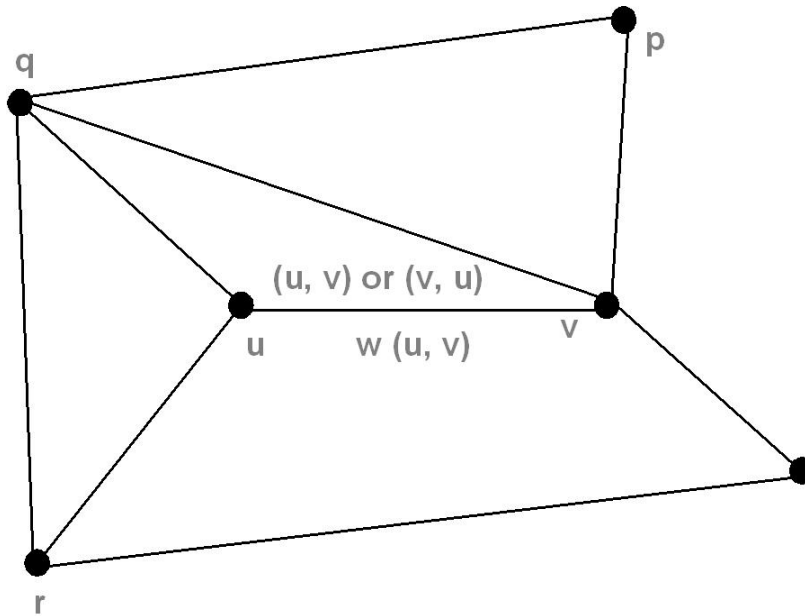
- Discrete analysis and differential geometry on discrete spaces
- Discrete regularization framework based on discrete differential operators
- Recover the algorithms presented before from the discrete framework, and derived new approaches as well

Warning The discrete analysis and geometry is developed by (Zhou and Schölkopf, 2004). It is NOT standard mathematics.

Some basic notions in graph theory

- A **graph** $\Gamma = (V, E)$ consists of a set V of **vertices** and a set of pairs of vertices $E \subseteq V \times V$ called **edges**.
- A graph is **undirected** if for each edge $(u, v) \in E$ we also have $(v, u) \in E$.
- A graph is **weighted** if it is associated with a function $w : E \rightarrow \mathbb{R}_+$ satisfying $w(u, v) = w(v, u)$.

Some basic notions in graph theory (cont.)



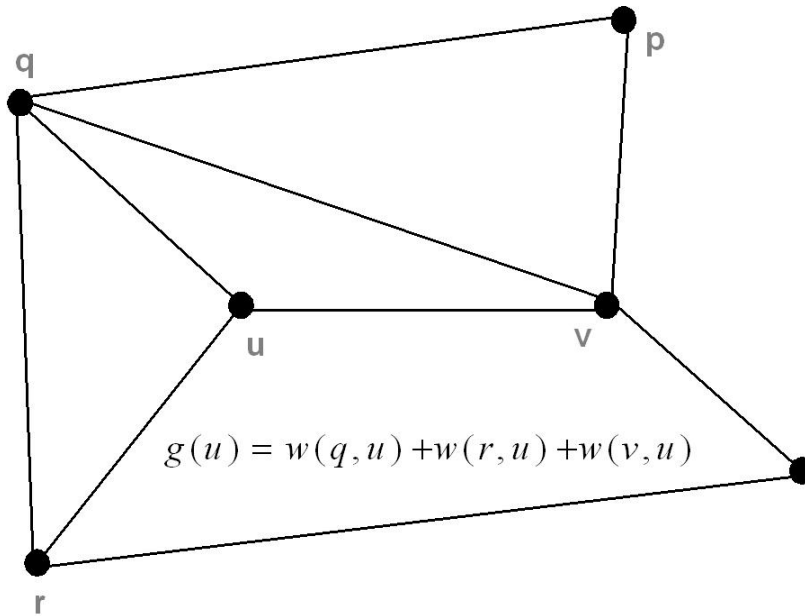
Some basic notions in graph theory (cont.)

- The **degree** function $g : V \rightarrow \mathbb{R}_+$ is defined to be

$$g(v) := \sum_{u \sim v} w(u, v),$$

where $u \sim v$ denote the set of vertices u connected to v via the edges (u, v) .

Some basic notions in graph theory (cont.)



The space of functions defined on graphs

- Let $\mathcal{H}(V)$ denote the Hilbert space of real-valued functions endowed with the usual inner product

$$\langle \varphi, \phi \rangle := \sum_v \varphi(v) \phi(v),$$

where φ and ϕ denote any two functions in $\mathcal{H}(V)$. Similarly define $\mathcal{H}(E)$. Note that function $\psi \in \mathcal{H}(E)$ need not be symmetric, i.e., we do not require $\psi(u, v) = \psi(v, u)$.

Gradient (or boundary) operator

- We define the **graph gradient** operator $d : \mathcal{H}(V) \rightarrow \mathcal{H}(E)$ to be

$$(d\varphi)(u, v) := \sqrt{\frac{w(u, v)}{g(u)}}\varphi(u) - \sqrt{\frac{w(u, v)}{g(v)}}\varphi(v),$$

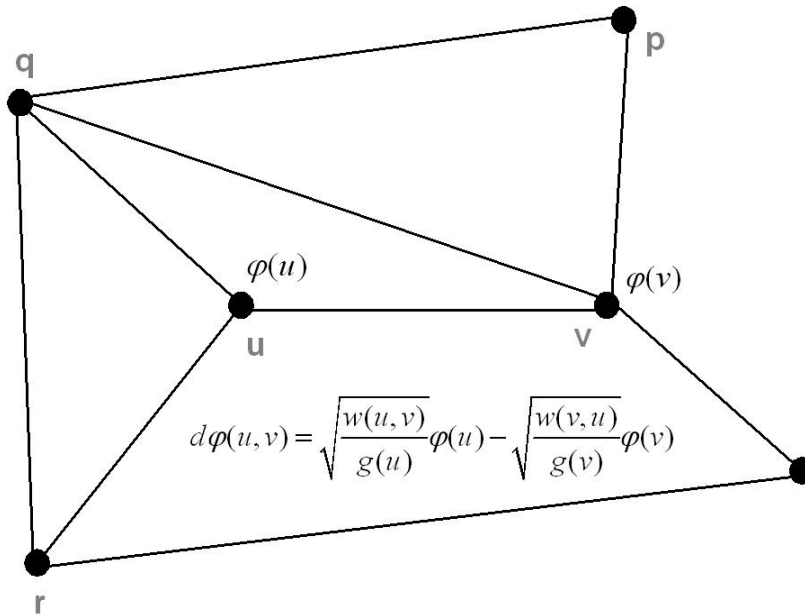
for all (u, v) in E .

Remark In the lattice case, the gradient degrades into

$$(d\varphi)(u, v) = \varphi(u) - \varphi(v),$$

which is the standard difference definition in numerical analysis.

Gradient (or boundary) operator



Edge derivative

- The **edge derivative** $\left. \frac{\partial}{\partial e} \right|_v : \mathcal{H}(V) \rightarrow \mathbb{R}$ along edge $e = (v, u)$ at vertex v is defined by

$$\left. \frac{\partial \varphi}{\partial e} \right|_v := (d\varphi)(v, u).$$

- Define the **local variation** of function φ in at v to be

$$\|\nabla_v \varphi\| := \left[\sum_{e \vdash v} \left(\left. \frac{\partial \varphi}{\partial e} \right|_v \right)^2 \right]^{1/2},$$

where $e \vdash v$ denotes the set of edges incident on v .

Divergence (or co-boundary) operator

- We define the adjoint $d^* : \mathcal{H}(E) \rightarrow \mathcal{H}(V)$ of d by

$$\langle d\varphi, \psi \rangle = \langle \varphi, d^*\psi \rangle, \text{ for all } \varphi \in \mathcal{H}(V), \psi \in \mathcal{H}(E).$$

We call d^* the **graph divergence** operator.

Note that the inner products are respectively in the space $\mathcal{H}(E)$ and $\mathcal{H}(V)$.

Laplacian operator

- We define the **graph Laplacian** $\Delta : \mathcal{H}(V) \rightarrow \mathcal{H}(V)$ by

$$\Delta := \frac{1}{2} d^* d.$$

- An **equivalent definition** is

$$(\Delta \varphi)(v) := \frac{1}{2} \sum_{e \vdash v} \frac{1}{\sqrt{g}} \left(\frac{\partial}{\partial e} \sqrt{g} \frac{\partial \varphi}{\partial e} \right) \Big|_v.$$

Remark See (cf. Jost, 2002) for the Laplace-Beltrami operator on Riemannian manifolds, and (cf. Chung, 2002) for the classical definition of the graph Laplacian.

Discrete regularization framework (I)

Theorem *The solution f of the optimization problem*

$$\operatorname{argmin}_{f \in \mathcal{H}(V)} \left\{ \frac{1}{2} \sum_v \|\nabla_v f\|^2 + \frac{\mu}{2} \|f - y\|^2 \right\}.$$

satisfies

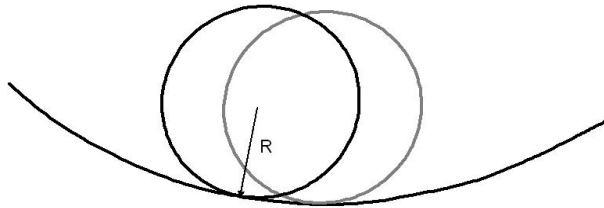
$$\Delta f + \mu(f - y) = 0.$$

Corollary $f = \mu(\mu I + \Delta)^{-1} y.$

Curvature operator

- By analogy with the curvature of a curve which is measured by the change in the unit normal, we define the **graph curvature** $\kappa : \mathcal{H}(V) \rightarrow \mathcal{H}(V)$ by

$$\kappa\varphi := d^*\left(\frac{d\varphi}{\|\nabla\varphi\|}\right).$$



Discrete regularization framework (II)

Theorem *The solution of the optimization problem*

$$\operatorname{argmin}_{f \in \mathcal{H}(V)} \left\{ \sum_v \|\nabla_v f\| + \frac{\mu}{2} \|f - y\|^2 \right\}$$

satisfies

$$\kappa f + \mu(f - y) = 0.$$

No closed form solution.

Discrete regularization framework (III)

Discrete large margin classification

$$\operatorname{argmin}_{f \in \mathcal{H}(V)} \left\{ \max_v \|\nabla_v f\| + \frac{\mu}{2} \|f - y\|^2 \right\}.$$

Only the **worst** case is considered!

Remark This is closely related to the classic **graph bandwidth** problem in combinatorial mathematics (cf. Linial, 2002), which is a NP-hard problem and has a polylogarithmic approximation.

Limitation: How to beat our method?

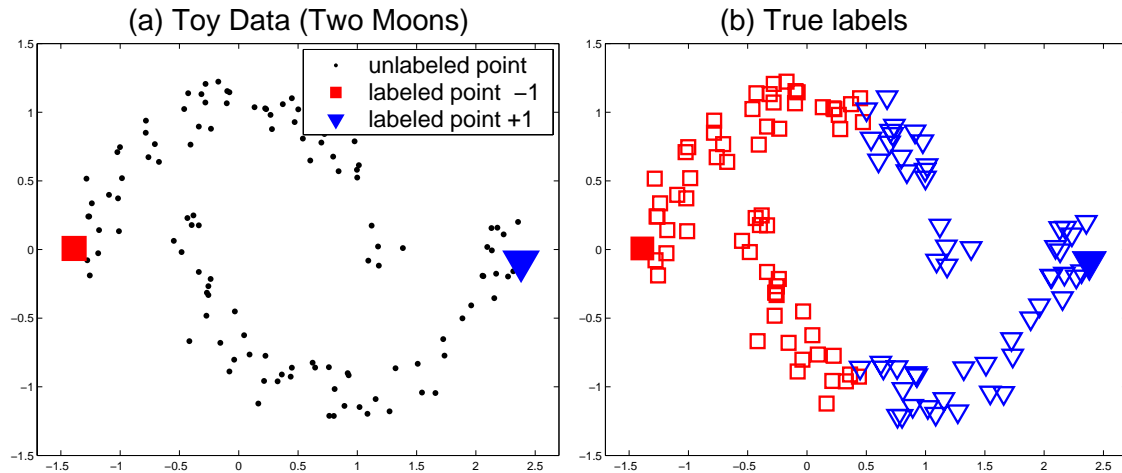
One can construct arbitrarily bad problems for a given algorithm:

Theorem [No Free Lunch, e.g., Devroye, 1996] *For any algorithm, any n and any $\epsilon > 0$, there exists a distribution P such that $R^* = 0$ and*

$$\mathbb{P}\left[R(g_n) \geq \frac{1}{2} - \epsilon\right] = 1,$$

where g_n is the function estimated by the algorithm based on the n training examples.

Limitation: How to beat our method? (cont.)



Part C—Related work

A closely related regularizer

- The regularizer we used is

$$\sum_{u,v} w(u, v) \left(\frac{f(u)}{\sqrt{g(u)}} - \frac{f(v)}{\sqrt{g(v)}} \right)^2$$

- A closely related one

$$\sum_{u,v} w(u, v) (f(u) - f(v))^2$$

is proposed by (Belkin and Niyogi, 2002, 2003; Zhu et al., 2003).
See also (Joachims, 2003) for a similar one.

Similarities between the two regularizers

- Both can be rewritten into the quadratic forms:

$$\sum_{u,v} w(u,v) \left(\frac{f(u)}{\sqrt{g(u)}} - \frac{f(v)}{\sqrt{g(v)}} \right)^2 = f^T D^{-\frac{1}{2}} (D - W) D^{-\frac{1}{2}} f$$

and

$$\sum_{u,v} w(u,v) (f(u) - f(v))^2 = f^T (D - W) f.$$

- Both $D^{-\frac{1}{2}} (D - W) D^{-\frac{1}{2}}$ and $D - W$ are called the graph Laplacian in Machine Learning Community (**unfortunate truth**).

Differences between the two regularizers: limit cases

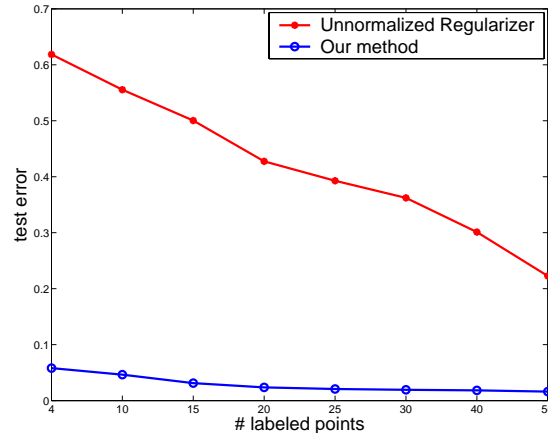
- (Bousquet et al., 2003) showed the following limit consequence:

$$\sum_{u,v} w(u,v)(f(u) - f(v))^2 \rightarrow \int \|\nabla f(x)\|^2 p^2(x) dx.$$

- A conjecture:

$$\sum_{u,v} w(u,v) \left(\frac{f(u)}{\sqrt{g(u)}} - \frac{f(v)}{\sqrt{g(v)}} \right)^2 \rightarrow \int \|\nabla f(x)\|^2 p(x) dx.$$

Difference between the two regularizers: experiments



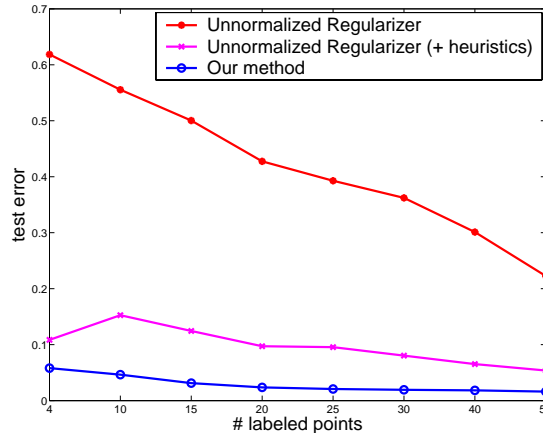
[Note: A subset of USPS containing the digits from 1 to 4; the same RBF kernel for all methods.]

Improve the unnormalized regularizer by heuristics

- (Belkin and Niyogi, 2002, 2003) Choose a number k and **construct a k -NN graph with 0/1 weights** over points. Using the weight matrix as the affinity among points.
- (Zhu et al., 2003) **Estimate the proportion of different classes** based on the labeled points, and then rescale the function based on the estimated proportion.

Both of them just empirically approximate to the normalization in our method.

Improve the unnormalized regularizer by heuristics: experiments



[Note: A subset of USPS containing the digits from 1 to 4; the same RBF kernel for all methods.]

Another related work: graph/cluster kernels

- **Graph or cluster kernels** (Smola and Kondor, 2003; Chapelle et al., 2002): Decompose the (normalized) graph Laplacian $K = U^T \Lambda U$ and then replace the eigenvalues λ with $\varphi(\lambda)$, where φ is a decreasing function, to obtain the so-called graph kernel:

$$\tilde{K} = U^T \text{diag}[\varphi(\lambda_1), \dots, \varphi(\lambda_n)] U.$$

- The matrix $(\mu I + \Delta)^{-1}$ contained in our closed form express can be viewed as a graph kernel with $\varphi(\lambda) = 1/(\mu + \lambda)$.

Difference from graph/cluster kernels

- The matrix $(\mu I + \Delta)^{-1}$ is naturally derived from our regularization framework for transductive inference. In contrast, graph/cluster kernels are obtained by **manipulating the eigenvalues**.
- SVM combined with the kernel matrix $(\mu I + \Delta)^{-1}$ does **not** work well in our transductive experiments.
- When we take other nonlinear regularizers, e.g., total variation, **no corresponding kernel** exists any more.

This talk is based on our following work:

- [Differential Geometry](#) *D. Zhou and B. Schölkopf*. Transductive Inference with Graphs. Technical Report, Max Planck Institute for Biological Cybernetics, August, 2004.
- [Directed Graphs](#) *D. Zhou, B. Schölkopf and T. Hofmann*. Semi-supervised Learning on Directed Graphs. **NIPS** 2004.
- [Undirected Graphs](#) *D. Zhou, O. Bousquet, T.N. Lal, J. Weston and B. Schölkopf*. Learning with Local and Global Consistency. **NIPS** 2003.
- [Ranking](#) *D. Zhou, J. Weston, A. Gretton, O. Bousquet and B. Schölkopf*. Ranking on Data Manifolds. **NIPS** 2003.
- [Bioinformatics](#) *J. Weston, A. Elisseeff, D. Zhou, C. Leslie and W.S. Noble*. Protein ranking: from local to global structure in the protein similarity network. **PNAS** 101(17) (2004).
- [Bioinformatics](#) *J. Weston, C. Leslie, D. Zhou, A. Elisseeff and W. S. Noble*. Semi-Supervised Protein Classification using Cluster Kernels. **NIPS** 2003.

Conclusions

- Proposed the new classification and ranking algorithms for vectorial and (directed) graph data
- Developed the discrete analysis and differential geometry, and constructed the discrete regularization frameworks
- Validated the approaches on real-world problems