

SDS 385: Stat Models for Big Data

Lecture 10: Pagerank and related methods

Purnamrita Sarkar
Department of Statistics and Data Science
The University of Texas at Austin
<https://psarkar.github.io/teaching>

Ranking and Pagerank

- Goal: obtain a ranking of webpages which are connected via hyperlinks
- Hope: webpages pointed to by other “important” webpages are also important.
- Developed by Brin and Page (1999)
- Many subsequent works:
 - HITS (Kleinberg, 1998)
 - Pagerank (Page and Brin, 1998)

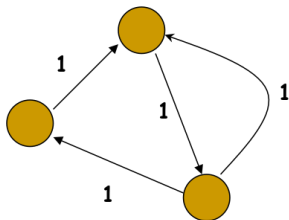
Definitions

- $n \times n$ **Adjacency matrix A**
 - A_{ij} = weight on an edge from i to j
 - If graph is undirected $A(i, j) = A(j, i)$
- $n \times n$ **Probability transition matrix P**
 - P has rows summing to one, i.e. **row stochastic**
 - $P(i, j)$ is the probability that a random walker will step on j from i .
 - $$P(i, j) = \frac{A(i, j)}{\sum_j A(i, j)}$$
- $n \times n$ **Laplacian matrix L**
 - $L = D - A$, where D is the diagonal matrix of degrees
 - It is symmetric positive semidefinite for undirected graphs.
 - Singular, i.e. has a zero eigenvalue

Definition

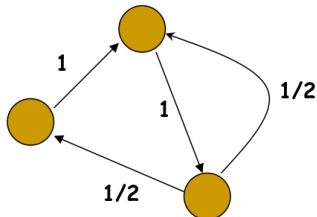
0	1	0
0	0	1
1	1	0

Adjacency matrix A

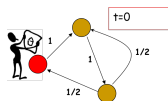


0	1	0
0	0	1
1/2	1/2	0

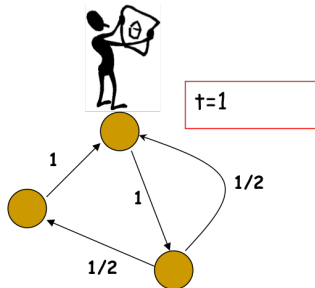
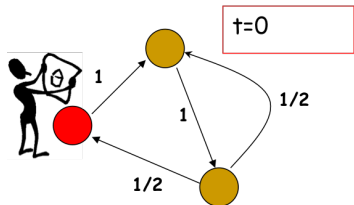
Transition matrix P



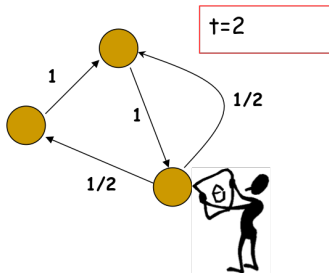
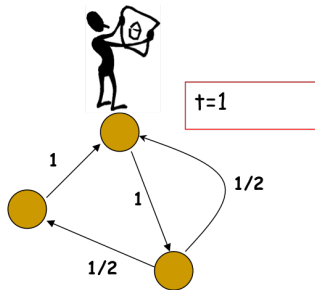
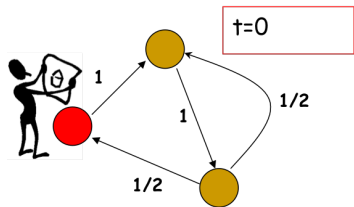
Random walks



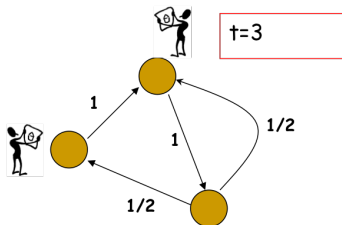
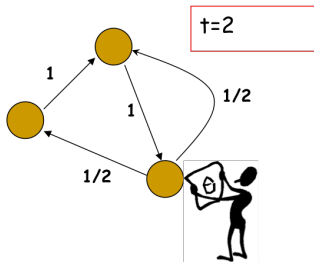
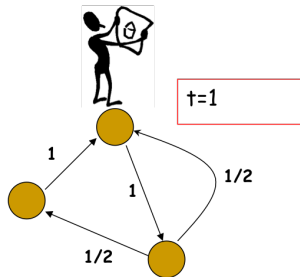
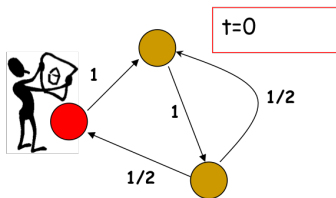
Random walks



Random walks



Random walks



- $x_t(i)$ denotes the probability that the surfer is at node i at time t .

$$x_{t+1}(i) = \sum_j x_t(j)P(j, i)$$

$$x_{t+1}^T = x_t^T P = x_{t-1}^T P^2 = \dots = x_0^T P^t$$

- What happens if the surfer keeps walking for a long time?

Stationary Distribution

- When the surfer keeps walking for a long time
- When the distribution does not change anymore i.e. $x_{T+1} = x_T$
- For well-behaved graphs this does not depend on the start distribution!!

Stationary distribution

- The stationary distribution at a node is related to the amount of time a random walker spends visiting that node.

Stationary distribution

- The stationary distribution at a node is related to the amount of time a random walker spends visiting that node.
- Remember that we can write the probability distribution at a node as

$$x_{t+1}^T = x_t^T P$$

Stationary distribution

- The stationary distribution at a node is related to the amount of time a random walker spends visiting that node.
- Remember that we can write the probability distribution at a node as

$$x_{t+1}^T = x_t^T P$$

- For the stationary distribution v_0 we have

$$v_0^T = v_0^T P$$

Stationary distribution

- The stationary distribution at a node is related to the amount of time a random walker spends visiting that node.
- Remember that we can write the probability distribution at a node as

$$x_{t+1}^T = x_t^T P$$

- For the stationary distribution v_0 we have

$$v_0^T = v_0^T P$$

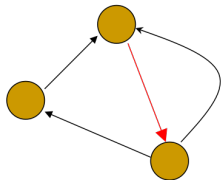
- Whoa! thats just the **left eigenvector** of the transition matrix !

Stationary distribution

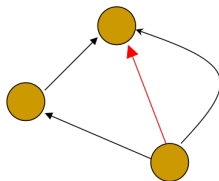
- Lot of theory hiding here.
- For example, what is the guarantee that there will be a unique left eigenvector, or the random walk will at all converge?
- Can't it just keep oscillating?

Well-behaved Markov chains

- **Irreducible:** There is a path from every node to every other node.



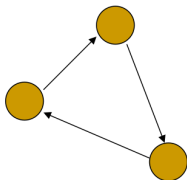
Irreducible



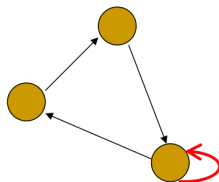
Not irreducible

Well-behaved Markov chains

- **Aperiodic:** The GCD of all cycle lengths is 1. The GCD is also called period.



Periodicity is 3



Aperiodic

Well-behaved Markov chains

- If a markov chain is irreducible and aperiodic then the largest eigenvalue of the transition matrix will be equal to 1 and all the other eigenvalues will be strictly less than 1.
- These results imply that for a well behaved graph there exists an unique stationary distribution.

Pagerank and Perron Frobenius

- Perron Frobenius only holds if the graph is irreducible and aperiodic.
- But how can we guarantee that for the web graph? Do it with a small restart probability c .
- At any time-step the random surfer
 - jumps (teleport) to any other node with probability c
 - jumps to its direct neighbors with total probability $1 - c$.

$$\tilde{P} = (1 - c)P + c\mathbf{1}\mathbf{1}^T/n$$

- Power Iteration is an algorithm for computing the stationary distribution.
 - Start with any distribution x_0
 - Keep computing $x_{t+1}^T = x_t^T P$
 - Stop when x_{t+1} and x_t are almost the same

Power Iteration

- Why should this work?
- Write x_0 as a linear combination of the left eigenvectors $\{v_0, v_1, \dots, v_{n-1}\}$ of P
- Remember that v_0 is the stationary distribution.

$$x_0 = c_0 v_0 + c_1 v_1 + c_2 v_2 + \dots + c_{n-1} v_{n-1}$$

Power Iteration

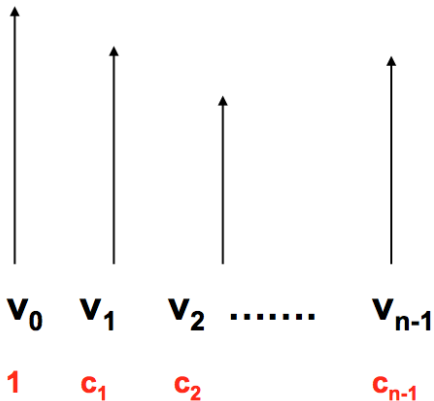
- Why should this work?
- Write x_0 as a linear combination of the left eigenvectors $\{v_0, v_1, \dots, v_{n-1}\}$ of P
- Remember that v_0 is the stationary distribution.

$$x_0 = c_0 v_0 + c_1 v_1 + c_2 v_2 + \dots + c_{n-1} v_{n-1}$$

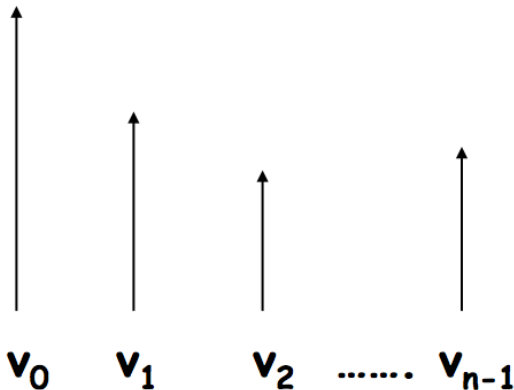
- $c_0 = 1$. Why?
 - First note that $1^T v_i = 0$ if $i \neq 0$
 - So $x_0^T 1 = c_0 = 1$, since both x_0 and v_0 are distributions.

Power Iteration

x_0

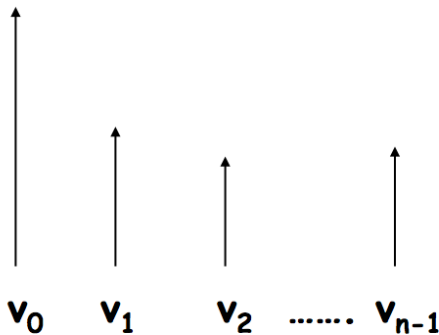


$$\mathbf{x}_1 = \mathbf{x}_0 \tilde{\mathbf{P}}$$



Power Iteration

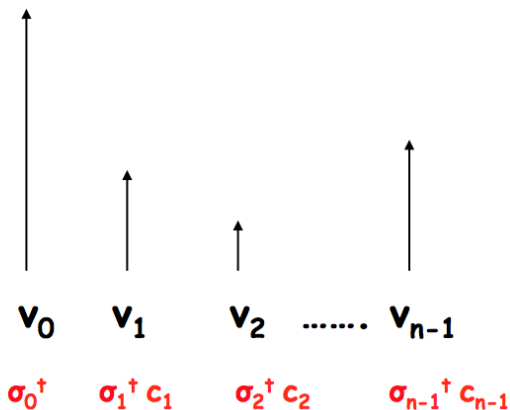
$$\mathbf{x}_2 = \mathbf{x}_1 \tilde{\mathbf{P}} = \mathbf{x}_0 \tilde{\mathbf{P}}^2$$



$$\sigma_0^2 \quad \sigma_1^2 c_1 \quad \sigma_2^2 c_2 \quad \dots \quad \sigma_{n-1}^2 c_{n-1}$$

Power Iteration

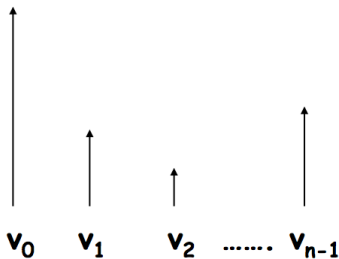
$$\mathbf{x}_t = \mathbf{x}_0 \tilde{\mathbf{P}}^t$$



Power Iteration

$$\mathbf{x}_t = \mathbf{x}_0 \tilde{\mathbf{P}}^t$$

$$\sigma_0 = 1 > \sigma_1 \geq \dots \geq \sigma_n$$



1

$\sigma_1^\dagger c_1$

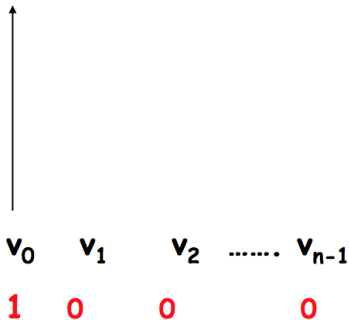
$\sigma_2^\dagger c_2$

$\sigma_{n-1}^\dagger c_{n-1}$

Power Iteration

\mathbf{x}_∞

$$\sigma_0 = 1 > \sigma_1 \geq \dots \geq \sigma_n$$



Second eigenvalue

- Smaller σ_2 is faster the chain mixes.
- For pagerank, we wonder what the second largest eigenvalue is of $\tilde{P} = (1 - c)P + cU$
- The largest eigenvalue is 1
- The second largest is less than $1 - c$ in magnitude.
- So pagerank computation converges fast.

- We are looking for the vector v s.t.

$$v^T = (1 - c)v^T P + cr^T$$

- r is a distribution over web-pages.
- If r is the uniform distribution we get pagerank.
- What happens if r is non-uniform?

- We are looking for the vector v s.t.

$$v^T = (1 - c)v^T P + cr^T$$

- r is a distribution over web-pages.
- If r is the uniform distribution we get pagerank.
- What happens if r is non-uniform?
- Personalization

Personalized Pagerank

- The only difference is that we use a non-uniform teleportation distribution, i.e. at any time step teleport to a set of webpages.
- In other words we are looking for the vector v s.t.

$$v^T = (1 - c)v^T P + cr^T$$

- r is a non-uniform preference vector specific to an user.
- v gives personalized views of the web.

Personalized Pagerank

- Pre-computation: r is not known from before
- Computing during query time takes too long
- A crucial observation¹ is that the personalized pagerank vector is linear w.r.t r

$$r = \begin{pmatrix} \alpha \\ 0 \\ 1 - \alpha \end{pmatrix} \Rightarrow v(r) = \alpha v(r_0) + (1 - \alpha) v(r_2)$$
$$r_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, r_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

- Lots of literature for computing personalized pagerank fast, and on the go.

Rank Stability

- Pre-computation: r is not known from before
- Computing during query time takes too long
- A crucial observation¹ is that the personalized pagerank vector is linear w.r.t r

$$r = \begin{pmatrix} \alpha \\ 0 \\ 1 - \alpha \end{pmatrix} \Rightarrow v(r) = \alpha v(r_0) + (1 - \alpha) v(r_2)$$
$$r_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, r_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

- Lots of literature for computing personalized pagerank fast, and on the go.

Rank Stability

- How does the ranking change when the link structure changes?
- The web-graph is changing continuously.
- How does that affect page-rank?

Rank Stability

Rank on the
entire database.



Rank on 5 perturbed
datasets by deleting
30% of the papers



1	"Genetic Algorithms in Search, Optimization and...", Goldberg	1	1	1	1	1
2	"Learning internal representations by error...", Rumelhart+al	2	2	2	2	2
3	"Adaptation in Natural and Artificial Systems", Holland	3	5	6	4	5
4	"Classification and Regression Trees", Breiman+al	4	3	5	5	4
5	"Probabilistic Reasoning in Intelligent Systems", Pearl	5	6	3	6	3
6	"Genetic Programming: On the Programming of ...", Koza	6	4	4	3	6
7	"Learning to Predict by the Methods of Temporal ...", Sutton	7	7	7	7	7
8	"Pattern classification and scene analysis", Duda+Hart	8	8	8	8	9
9	"Maximum likelihood from incomplete data via...", Dempster+al	10	9	9	11	8
10	"UCI repository of machine learning databases", Murphy+Aha	9	11	10	9	10
11	"Parallel Distributed Processing", Rumelhart+McClelland	-	-	-	10	-
12	"Introduction to the Theory of Neural Computation", Hertz+al	-	10	-	-	-

1. Link analysis, eigenvectors, and stability, Andrew Y. Ng, Alice X. Zheng and Michael Jordan, IJCAI-01
2. Automating the construction of Internet portals with machine learning, A. Mc Callum, K. Nigam, J. Rennie, K. Seymore, In Information Retrieval Journal, 2000

Rank Stability

Rank on the
entire database.



Rank on 5 perturbed
datasets by deleting
30% of the papers



1	"Genetic Algorithms in Search, Optimization and...", Goldberg	1	1	1	1	1
2	"Learning internal representations by error...", Rumelhart+al	2	2	2	2	2
3	"Adaptation in Natural and Artificial Systems", Holland	3	5	6	4	5
4	"Classification and Regression Trees", Breiman+al	4	3	5	5	4
5	"Probabilistic Reasoning in Intelligent Systems", Pearl	5	6	3	6	3
6	"Genetic Programming: On the Programming of ...", Koza	6	4	4	3	6
7	"Learning to Predict by the Methods of Temporal ...", Sutton	7	7	7	7	7
8	"Pattern classification and scene analysis", Duda+Hart	8	8	8	8	9
9	"Maximum likelihood from incomplete data via...", Dempster+al	10	9	9	11	8
10	"UCI repository of machine learning databases", Murphy+Aha	9	11	10	9	10
11	"Parallel Distributed Processing", Rumelhart+McClelland	-	-	-	10	-
12	"Introduction to the Theory of Neural Computation", Hertz+al	-	10	-	-	-

1. Link analysis, eigenvectors, and stability, Andrew Y. Ng, Alice X. Zheng and Michael Jordan, IJCAI-01
2. Automating the construction of Internet portals with machine learning, A. Mc Callum, K. Nigam, J. Rennie, K. Seymore, In Information Retrieval Journal, 2000

- Ng et al 2001: $\tilde{P} = (1 - c)P + cU$
- Theorem: if v is the left eigenvector of P . Let the pages i_1, i_2, \dots, i_k be changed in any way, and let v' be the new pagerank. Then

$$\|v - v'\|_1 \leq \frac{\sum_{j=1}^k v(i_j)}{c}$$

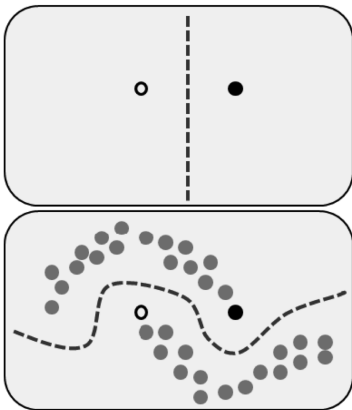
- Ng et al 2001: $\tilde{P} = (1 - c)P + cU$
- Theorem: if v is the left eigenvector of P . Let the pages i_1, i_2, \dots, i_k be changed in any way, and let v' be the new pagerank. Then

$$\|v - v'\|_1 \leq \frac{\sum_{j=1}^k v(i_j)}{c}$$

- So if c is not too close to 0, the system would be rank stable and also converge fast!

Semi-supervised learning

- You are given a lot of unlabeled data.
- Only a few points are labeled.
- Is this useful?



Semi-supervised learning

- Two broad ways
 - Label propagation:
 - Graph Based algorithm
 - Does not generalize to unseen data, i.e. **Transductive**
 - Manifold regularization
 - Graph Based regularization
 - Does generalize to unseen data, i.e. **Inductive**

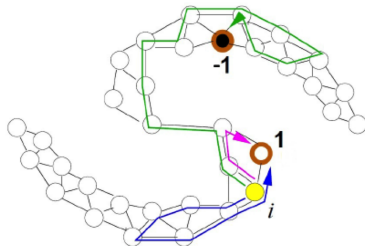
- Input n data points x_1, \dots, x_n
- Define similarity matrix $S \in \mathbb{R}^{n \times n}$

$$S_{ij} = \exp(-\|x_i - x_j\|^2 / 2\sigma^2)$$

- Since S is dense, often k nearest neighbor graphs are also used.
(Your homework!)

Label propagation [Zhu et al, 2003]

- Input:
 - ℓ labeled datapoints $(x_1, y_1), \dots, (x_\ell, y_\ell)$
 - u unlabeled datapoints $x_{\ell+1}, \dots, x_{\ell+u}$
- Predict: labels $y_{\ell+1}, \dots, y_{\ell+u}$



Label propagation algorithm

- Compute $P \in [0, 1]^{(\ell+u) \times (\ell+u)}$
- $P_{ij} = \frac{S_{ij}}{\sum_j S_{ij}}$
- Harmonic function:
 - Function value at an unlabeled node is an average of function values at its neighbors
 - For $j = \ell + 1 : \ell + u$,

$$f(j) = \frac{\sum_i S_{ij} f(i)}{\sum_i S_{ij}}$$

- In other words, for the unlabeled nodes, this fixed point equation is satisfied

$$f[U] = Pf[U] \quad f[L] = y[L]$$

- For a vector v and set S , we denote by $v[S]$ the subset of values in S
- U and L denote the set of unlabeled and labeled points respectively.

Closed form

- Assume there are just two classes. Set $y[L] \in \{0, 1\}^\ell$ accordingly.
- We have:

$$\begin{bmatrix} P_{LL} & P_{LU} \\ P_{UL} & P_{UU} \end{bmatrix} \begin{bmatrix} Y_L \\ Y_U \end{bmatrix} = \begin{bmatrix} Y_L \\ Y_U \end{bmatrix}$$

- Expanding, we get:

$$P_{UL} Y_L + P_{UU} Y_U = Y_U$$

- Moving things around:

$$Y_U = (I - P_{UU})^{-1} P_{UL} Y_L$$

- Can use a linear system solver

Label propagation - Random walk interpretation

- Think of the labeled nodes as absorbing states
- Use

$$\begin{aligned} Y_U &= \sum_{t=0}^{\infty} P_{UU}^t P_{UL} Y_L \\ &= \underbrace{P_{UL} Y_L}_{\text{Probability of reaching "1"s in one step}} + \underbrace{P_{UU} P_{UL} Y_L}_{\text{Probability of reaching in two steps}} + \dots \\ &= \text{Probability of reaching a label "1" in a long random walk} \end{aligned}$$

- Why is this useful?
 - If the labels are all reachable, a long walk must hit a "0" or a "1"
 - So if $Y_i > 1/2$, that means from i , it's more likely to reach a "1" than a "0"

Graph Laplacian interpretation

- Graph Laplacian: $L = D - S$, where $D_{ii} = \sum_j S_{ij}$ is a diagonal matrix
- L is positive semi-definite (we are assuming $S_{ij} > 0$)
- Why?

- For any vector v ,

$$v^T L v = \sum_{ij} S_{ij} (v_i - v_j)^2$$

- So this measures how unsmooth v is w.r.t S
- L is also singular, why?

Label propagation algorithm

- We can also frame label propagation as

$$\arg \min_v v^T L v \quad s.t. v[L] = y_L$$

- Why?

Label propagation algorithm

- We can also frame label propagation as

$$\arg \min_v v^T L v \quad s.t. v[L] = y_L$$

- Why?
- Write $v = \begin{bmatrix} y_L & v_U \end{bmatrix}$

Label propagation algorithm

- We can also frame label propagation as

$$\arg \min_v v^T L v \quad \text{s.t. } v[L] = y_L$$

- Why?
- Write $v = \begin{bmatrix} y_L & v_U \end{bmatrix}$
- Now set the derivative to zero.
 - $Lv = 0$ such that $v[L] = y_L$

-

$$\begin{bmatrix} D_{LL} - S_{LL} & -S_{LU} \\ -S_{UL} & D_{UU} - S_{UU} \end{bmatrix} \begin{bmatrix} y_L \\ v_U \end{bmatrix} = 0$$

- Solving:

$$-S_{UL}y_L + (D_{UU} - S_{UU})v_U = 0$$

Label propagation algorithm

- We can also frame label propagation as

$$\arg \min_v v^T L v \quad \text{s.t. } v[L] = y_L$$

- Why?
- Write $v = \begin{bmatrix} y_L & v_U \end{bmatrix}$
- Now set the derivative to zero.
 - $Lv = 0$ such that $v[L] = y_L$

-

$$\begin{bmatrix} D_{LL} - S_{LL} & -S_{LU} \\ -S_{UL} & D_{UU} - S_{UU} \end{bmatrix} \begin{bmatrix} y_L \\ v_U \end{bmatrix} = 0$$

- Solving:

$$-S_{UL}y_L + (D_{UU} - S_{UU})v_U = 0$$

- rearranging:

$$v_U = (D_{UU} - S_{UU})^{-1} S_{UL} y_L = (I - P_{UU})^{-1} P_{UL} y_L$$

Experimentally?

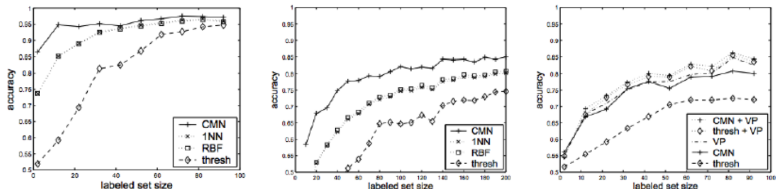


Figure 3. Harmonic energy minimization on digits "1" vs. "2" (left) and on all 10 digits (middle) and combining voted-perceptron with harmonic energy minimization on odd vs. even digits (right)

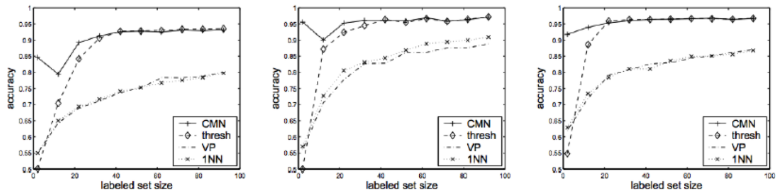


Figure 4. Harmonic energy minimization on PC vs. MAC (left), baseball vs. hockey (middle), and MS-Windows vs. MAC (right)

Manifold regularization

- Input:
 - ℓ labeled datapoints $(x_1, y_1), \dots, (x_\ell, y_\ell)$
 - u unlabeled datapoints $x_{\ell+1}, \dots, x_{\ell+u}$
- Predict: labels $y_{\ell+1}, \dots, y_{\ell+u}$

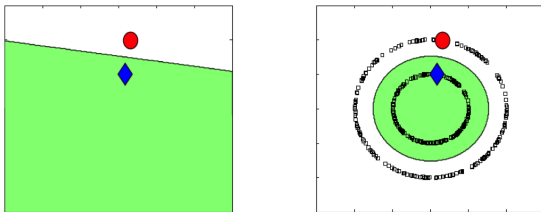


Figure 1: Unlabeled data and prior beliefs

Manifold regularization

- Input:
 - ℓ labeled datapoints $(x_1, y_1), \dots, (x_\ell, y_\ell)$
 - u unlabeled datapoints $x_{\ell+1}, \dots, x_{\ell+u}$
- Predict: labels $y_{\ell+1}, \dots, y_{\ell+u}$

Manifold regularization

- Input:
 - ℓ labeled datapoints $(x_1, y_1), \dots, (x_\ell, y_\ell)$
 - u unlabeled datapoints $x_{\ell+1}, \dots, x_{\ell+u}$
- Predict: labels $y_{\ell+1}, \dots, y_{\ell+u}$
- Before the constraint was $v[L] = y_L$, now instead we will use a loss function and learn a classifier on the labeled data

$$\min_w \underbrace{\sum_{i=1}^{\ell} \text{loss}(y_i, w^T x_i)}_{\text{loss}} + \lambda \underbrace{R(w)}_{\text{regularization}}$$

Manifold regularization

- Input:
 - ℓ labeled datapoints $(x_1, y_1), \dots, (x_\ell, y_\ell)$
 - u unlabeled datapoints $x_{\ell+1}, \dots, x_{\ell+u}$
- Predict: labels $y_{\ell+1}, \dots, y_{\ell+u}$
- Before the constraint was $v[L] = y_L$, now instead we will use a loss function and learn a classifier on the labeled data

$$\min_w \underbrace{\sum_{i=1}^{\ell} \text{loss}(y_i, w^T x_i)}_{\text{loss}} + \lambda \underbrace{R(w)}_{\text{regularization}}$$

- How about the unlabeled data?

Manifold regularization: Belkin et al 2006

$$\min_w \underbrace{\sum_{i=1}^{\ell} \text{loss}(y_i, w^T x_i)}_{\text{loss}} + \lambda \underbrace{R(w)}_{\text{regularization}} + \beta (Xw)^T L (Xw)$$

- Assume a linear predictor $w^T x$
- Idea: close/similar points have similar predicted labels.
- LapSVM:

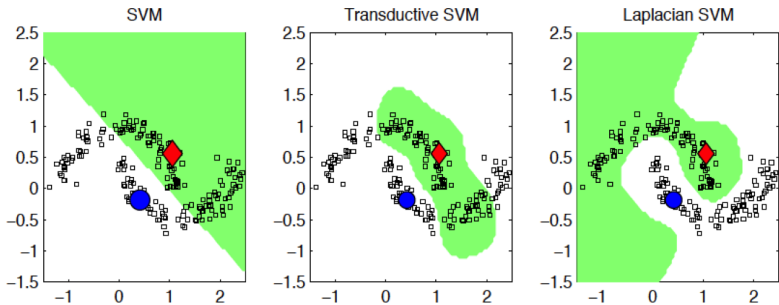
$$\min_w \sum_{i=1}^{\ell} (1 - y_i f(x_i))_+ + \lambda \underbrace{\|f\|_K^2}_{\text{regularization}} + \beta f^T L f$$

Transductive SVM: Joachims et al 1999

$$\min_{w, y_{\ell+1}, \dots, y_{\ell+u}} \sum_{i=1}^{\ell} (1 - y_i f(x_i))_+ + C' \sum_{i=\ell+1}^u (1 - y_i f(x_i))_+ + \lambda \underbrace{\|f\|_K^2}_{\text{regularization}}$$

- Iteratively solves SVM quadratic programs
- Switches labels to improve objective function
- Suffers from local optima, inherently combinatorial problem

Transductive SVM VS LapSVM



Summary

- Three main parts $+\epsilon$
- Large scale optimization:
 - Gradient descent, Newton Raphson
 - Stochastic gradient descent, proximal methods, subgradients, dual coordinate ascent, etc.
 - Le Cun, Leon Bottou Yann. "Large Scale Online Learning." NIPS, 2004.
 - Main point: Given the same time of training, SGD can allow more datapoints, and hence may have faster convergence

Summary

- Three main parts + ϵ
- Large scale optimization:
 - Gradient descent, Newton Raphson
 - Stochastic gradient descent, proximal methods, subgradients, dual coordinate ascent, etc.
 - Le Cun, Leon Bottou Yann. "Large Scale Online Learning." NIPS, 2004.
 - Main point: Given the same time of training, SGD can allow more datapoints, and hence may have faster convergence
 - John Duchi, Elad Hazan, Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." Journal of Machine Learning Research 12 (2011): 2121-2059.
 - Use the geometry of the data to choose a better loss function.
Bottomline - use diagonal approximation of the Hessian instead of full Hessian.

- Large scale optimization:
 - Stochastic gradient descent
 - Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In Advances in neural information processing systems, pages 315323, 2013
 - Main point: Talks about dual coordinate ascent and shows how this leads to variance reduction

- Large scale optimization:
 - Stochastic gradient descent
 - Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In Advances in neural information processing systems, pages 315323, 2013
 - Main point: Talks about dual coordinate ascent and shows how this leads to variance reduction
 - Yu et al., "Large Linear Classification when Data Cannot Fit In Memory", in KDD 2010) Oct 10
 - Block updates by dividing data into chunks that fit into main memory.

Summary

- Nearest neighbor methods: locality sensitive hashing, random projections and Johnson-Lindenstrauss, tree structures
 - Weinberger, Kilian, et al. "Feature hashing for large scale multitask learning." ICML, 2009. Oct 17
 - Random projection type hash functions to bring high dimensional data down to lower dimensional space while not affecting the dot products (which are important for a various number of tasks).

Summary

- Nearest neighbor methods: locality sensitive hashing, random projections and Johnson-Lindenstrauss, tree structures
 - Weinberger, Kilian, et al. "Feature hashing for large scale multitask learning." ICML, 2009. Oct 17
 - Random projection type hash functions to bring high dimensional data down to lower dimensional space while not affecting the dot products (which are important for a various number of tasks).
 - Ping Li, Trevor Hastie, Kenneth Church. "Very sparse random projections",
 - Doing Johnson Lindenstrauss with very sparse projection matrices

Summary

- Nearest neighbor methods: locality sensitive hashing, random projections and Johnson-Lindenstrauss, tree structures
 - Weinberger, Kilian, et al. "Feature hashing for large scale multitask learning." ICML, 2009. Oct 17
 - Random projection type hash functions to bring high dimensional data down to lower dimensional space while not affecting the dot products (which are important for a various number of tasks).
 - Ping Li, Trevor Hastie, Kenneth Church. "Very sparse random projections",
 - Doing Johnson Lindenstrauss with very sparse projection matrices
 - Gray, Alexander, Ting Liu and Andrew W. Moore. "New Algorithms for Efficient High-Dimensional Nonparametric Classification." Journal of Machine Learning Research 7 (2006): 1135-1158.
 - Fast tree based methods for nearest neighbor search when data has imbalanced classes

Summary

- Parallel methods and Divide and Conquer: better compression for parallel SGD. Divide and conquer methods for community detection in very large graphs. Bootstrap and subsampling- Bag of little bootstraps.
 - Slow learners are fast". In NIPS 2009. Langford et al.
 - Effect of delayed updates in SGD for sharer memory systems

Summary

- Parallel methods and Divide and Conquer: better compression for parallel SGD. Divide and conquer methods for community detection in very large graphs. Bootstrap and subsampling- Bag of little bootstraps.
 - Slow learners are fast". In NIPS 2009. Langford et al.
 - Effect of delayed updates in SGD for sharer memory systems
 - Bradley, Joseph, et al. "Parallel Coordinate Descent for L1-Regularized Loss Minimization." International Conference on Machine Learning (2011).
 - Shotgun. Coordinate ascent without locks and the effect of correlations of the features on the final error

Summary

- Parallel methods and Divide and Conquer: better compression for parallel SGD. Divide and conquer methods for community detection in very large graphs. Bootstrap and subsampling- Bag of little bootstraps.
 - Slow learners are fast". In NIPS 2009. Langford et al.
 - Effect of delayed updates in SGD for sharer memory systems
 - Bradley, Joseph, et al. "Parallel Coordinate Descent for L1-Regularized Loss Minimization." International Conference on Machine Learning (2011).
 - Shotgun. Coordinate ascent without locks and the effect of correlations of the features on the final error
 - Feng Niu, Benjamin Recht, Christopher Re, Stephen J. Wright, Hogwild!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent", NIPS 2011.
 - Asynchronous SGD without locks—use the sparsity in data

Summary

- Parallel methods and Divide and Conquer: better compression for parallel SGD. Divide and conquer methods for community detection in very large graphs. Bootstrap and subsampling- Bag of little bootstraps.
 - Slow learners are fast". In NIPS 2009. Langford et al.
 - Effect of delayed updates in SGD for sharer memory systems
 - Bradley, Joseph, et al. "Parallel Coordinate Descent for L1-Regularized Loss Minimization." International Conference on Machine Learning (2011).
 - Shotgun. Coordinate ascent without locks and the effect of correlations of the features on the final error
 - Feng Niu, Benjamin Recht, Christopher Re, Stephen J. Wright, Hogwild!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent", NIPS 2011.
 - Asynchronous SGD without locks—use the sparsity in data
 - Mackey, Lester, Ameet Talwalkar, Michael I. Jordan. "Divide-and-Conquer Matrix Factorization." arXiv:1107.0789v6 (2012)
 - Divide up the data, do matrix factorization on each part and do a

- The ϵ - Semisupervised learning and Pagerank. Some Bayesian methods.
 - Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical bayesian optimization of machine learning algorithms"

Acknowledgments

- Cho-Jui Hsieh's lecture notes from UC Davis
- Zhu et al's paper in ICML "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions"
- Ng et al's paper on ranking Stability "Link Analysis, Eigenvectors and Stability", IJCAI 2001
- Belkin, Niyogi and Sindhwani's paper "Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples" in JMLR 2006
- My old talk on Random walks.