Chapter 5: Microarray Techniques

5.3 Classification & Machine Learning Techniques

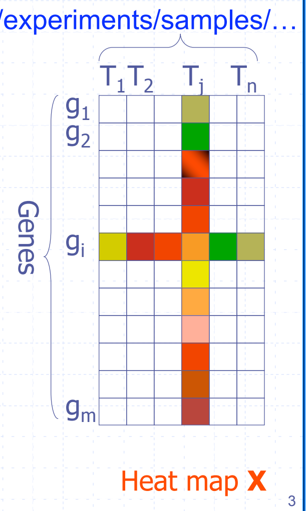
Prof. Yechiam Yemini (YY)
Computer Science Department
Columbia University

Overview

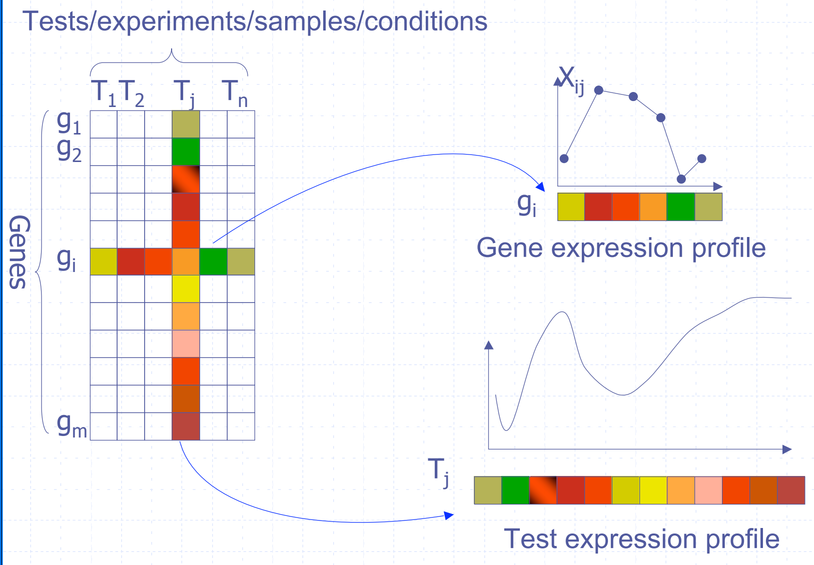
- Principal components analysis (PCA)
- Linear classifiers; perceptrons; neural nets..
- SVM Classifiers

Microarray Heat Map

- Microarray measurements may be organized in a heat-map matrix
- Row represent genes
- Columns represent tests
- X_{ij}=expression level of g_i under test Tj
- Expression level is visualized via colors
  - Green= under expressed (down regulated)
  - Red = over expressed (up regulated)



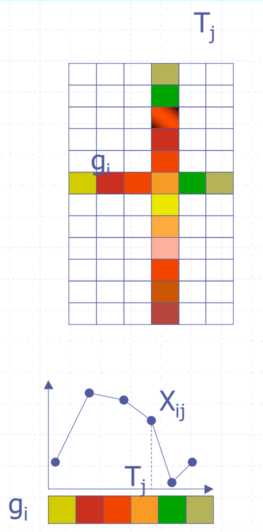
Heat Map Provides Expressions Profiles



Microarray Experiments

There are two typical experiments:

- Differentiation
  - Compare expression levels under different conditions
  - A test T_j represents expression levels of a condition
  - E.g., cancer or drug-treated cell vs. normal cell
- Temporal expression
  - Explore temporal evolution of expression levels
  - A test T_j represents expression levels at a given time
  - E.g., study cell response to heat-shock, starvation

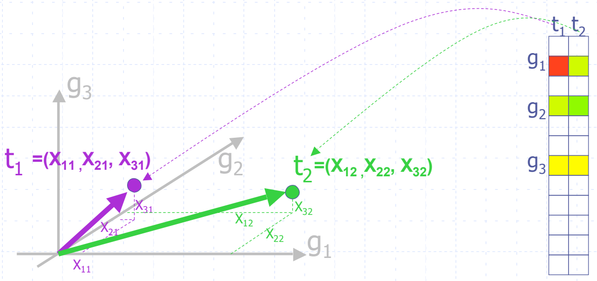


Gene expression profile

5

Some Basic Geometry

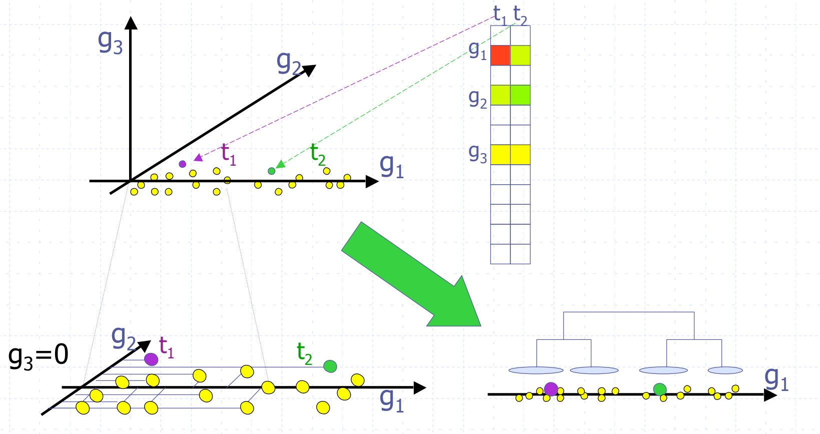
- Genes/tests may be modeled as n dimensional vectors
  - Define $t_j=(\underset{1}{0},0\ldots0,\underset{j}{1},0\ldots.\underset{n}{0})$ then $g_i=\Sigma X_{ij}t_i$
- Differentiation example:
  - t_1,t_2 have similar expression by g_2 and g_3 ($X_{31}=X_{32}$ $X_{21}\sim X_{22}$)
  - But g_1 provides good differentiation of t_1 and t_2



6

Principal Component Analysis

Goal: Reduce Dimensionality

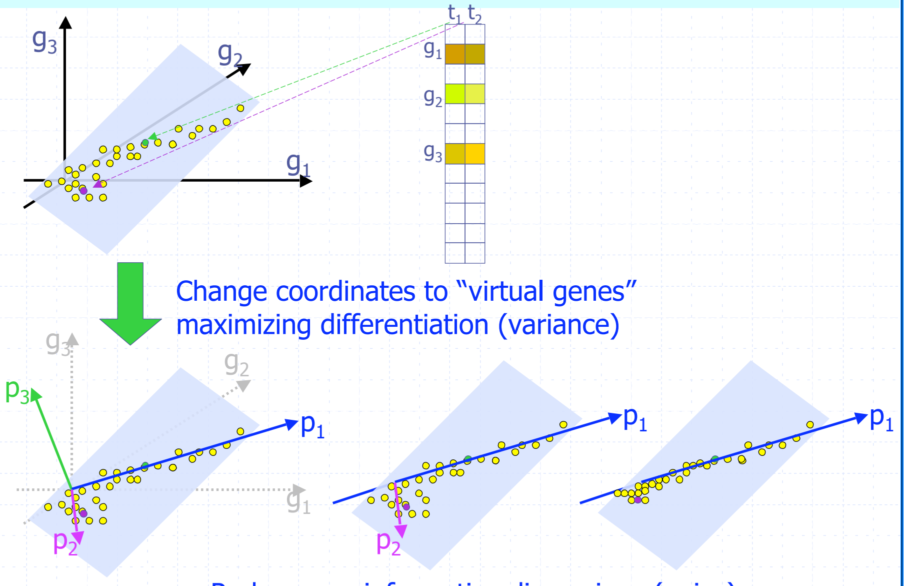


g_2 , g_3 do not differentiate

g_1 maximizes differentiation

differentiation ~ variance

Key Idea: Change Coordinates



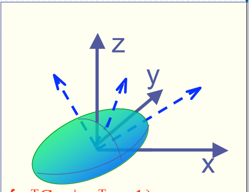
Change coordinates to “virtual genes” maximizing differentiation (variance)

Reduce non-informative dimensions (noise)

9

Maximizing Variance

- Given n samples $\{\underline{x}_k\}$ of a random m-dimensional vector $\underline{\boldsymbol{x}}$
  - Assume, for simplicity, that $E[\underline{\boldsymbol{x}}]=\underline{\boldsymbol{0}}$
- Which direction $\underline{\mathbf{w}}$ ($||\underline{\mathbf{w}}||=1$) maximizes the variance $VAR[\underline{\mathbf{w}}^T\underline{\mathbf{x}}]$?
  - Define an nxm data matrix X whose columns are the samples $\{\underline{x}_k\}$
  - $VAR[\underline{\mathbf{w}}^T\underline{\boldsymbol{x}}]=E[(\underline{\mathbf{w}}^T\underline{\boldsymbol{x}})(\underline{\boldsymbol{x}}^T\underline{\mathbf{w}})]= E[\underline{\mathbf{w}}^T\underline{\boldsymbol{xx}}^T\underline{\mathbf{w}})]= \underline{\mathbf{w}}^T E[\underline{\boldsymbol{xx}}^T]\underline{\mathbf{w}}=$
    $=(1/n^2)\ \underline{\mathbf{w}}^T[\Sigma \underline{\mathbf{x}}_r\underline{\mathbf{x}}^T{}_s]\underline{\mathbf{w}} =(1/n^2)\underline{\mathbf{w}}^T XX^T\underline{\mathbf{w}}$
  - Therefore, maximizing variance is equivalent to:
    $Max\{\underline{\mathbf{w}}^T C\underline{\mathbf{w}} \mid \underline{\mathbf{w}}^T\underline{\mathbf{w}}=1\}$
    where $C=XX^T$ is the auto-covariance matrix of $\underline{\boldsymbol{x}}$



$\{w^TCw \mid w^Tw=1\}$ is an ellipsoid whose axes are the eigenvectors of C

- How do principal eigenvectors arise?
  - Use Lagrangian to solve the constrained quadratic optimization
  - $L(w,\lambda)=\mathbf{w}^T C\mathbf{w}-\lambda\mathbf{w}^T\mathbf{w}$; the solution must satisfy $0=grad_{\underline{w}}L= C\mathbf{w}-\lambda\mathbf{w}$
  - Therefore $C\mathbf{w}=\lambda\mathbf{w}$ ➔ $\mathbf{w}$ is an eigenvector.
  - Furthermore $\mathbf{w}^T C\mathbf{w} =\lambda\mathbf{w}^T\mathbf{w}=\lambda$
    ➔ the variance maxing direction is the eigenvector with largest eigenvalue

10

Principal Components Analysis

- Represent the data in the eigenvectors space
  - Compute autocovariance: $\mathbf{X}^T\mathbf{X}$
  - Eigenvectors of $\mathbf{X}^T\mathbf{X}$ are the principal coordinates
  - Principal coordinates maximize residual variance
  - Eigenvalues correspond to maximal residual variance
- Use Singular Value Decomposition (SVD) to compute PCA
  - Compute factorization: $\mathbf{X}^T\mathbf{X} = \mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^T$
  - The transformation to principal coordinates is: **y**=U**x**
  - This PCA coordinate change is also called: Karhunen-Loeve transform
- Eliminate eigenvectors with small eigenvalues
  - Project data unto a subspace with maximal residual variance
  - This reduces dimensionality while maxing discrimination

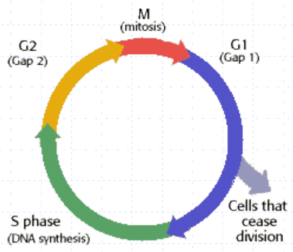
PCA Through Example

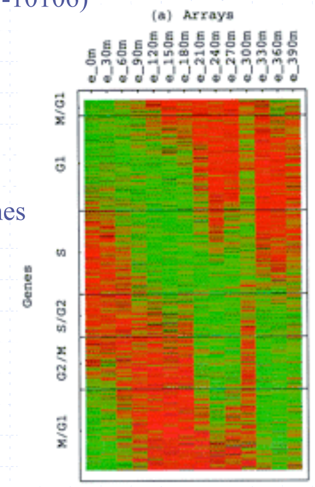
Cell cycle analysis

(Orly Alter *et al.*, *PNAS*, 2000, 97(18) 10101-10106)

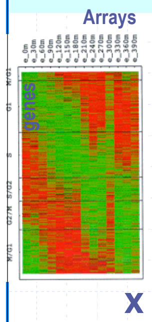
Time

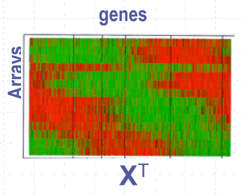
Genes





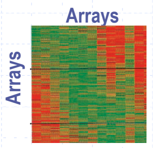
PCA Analysis





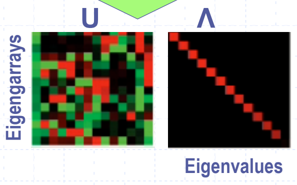
Covariance

X^TX



SVD Decomposition

$X^TX = U \Lambda U^T$



- Data: (5981 genes) x(14 arrays)
- PCA: reduce dimensionality
- SVD for array-space

Results

Alter, Brown & Botstein PNAS | August 29, 2000 | vol. 97 | no. 18 | 10101–10106 | AppendixA1–A6

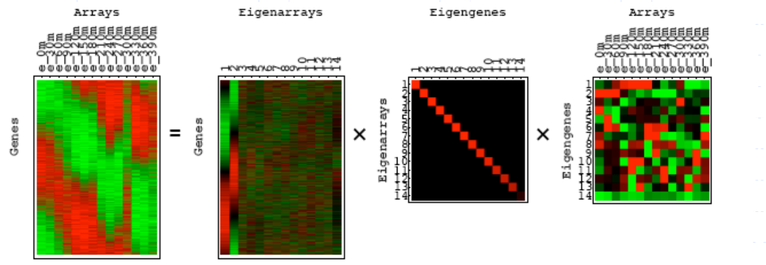
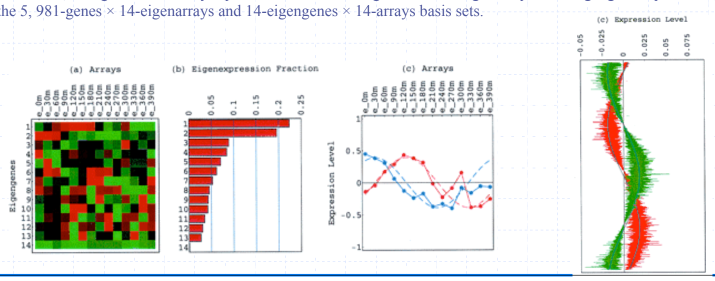
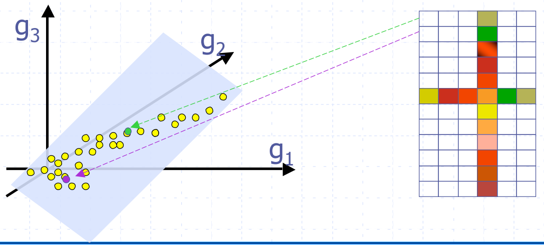


Fig. 7. SVD of the normalized and sorted elutriation data. Raster display of data with overexpression (red), no change in expression (black), and underexpression (green). Showing a linear transformation of the data from the 5, 981-genes × 14-arrays space to the reduced diagonalized 14-eigenarrays × 14-eigengenes space using the 5, 981-genes × 14-eigenarrays and 14-eigengenes × 14-arrays basis sets.



Notes On PCA

- Effective in reducing dimensionality
- More predictable and analyzable than clustering
- Intuitive interpretation
  - Eigengene = linear combination of gene profiles maxing variance
  - Let P^k be the projection on the subspace U^k=Span$\{u_1,u_2\ldots u_k\}$; u_{k+1} maximizes the residual variance of the projections $\{(I- P^k)g_i\}$
- SVD is often simpler to compute in array-space
  - Results may be applied and interpreted in gene-space

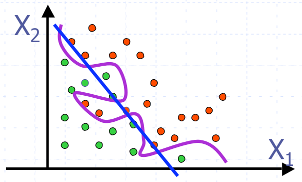


15

Linear Classifiers

Basic Classification Concepts

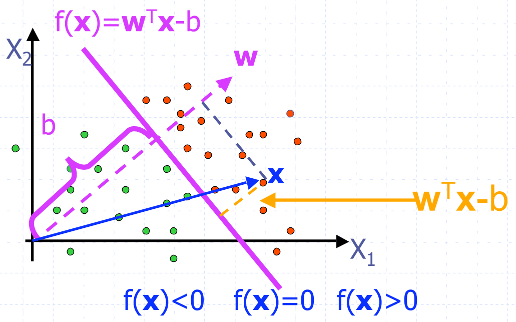
- Given: sample data $\{X^k\}$ and class association $Y^k \in \{-1,1\}$
- Goal: find a "good" function f(X) such that Y=sgn[f(X)]
  - There are numerous classification techniques
  - Classical statistics ➔ machine learning…
  - We consider only basics
- Supervised Learning
  - input $\{X^k, Y^k\}$; output f(X)
  - Avoid over-fitting



17

Linear Classifier

- Classifier is a hyperplane $f(x)=\mathbf{w}^T\mathbf{x}-b=w_1x_1+w_2x_2-b$
- $y=\text{sgn}(\mathbf{w}^T\mathbf{x}-b)$ classifies $\mathbf{x}$
- Simplify this: $y=\text{sgn}(\mathbf{wx})$ $\mathbf{w}=(w_1,w_2,b)$, $\mathbf{x}=(x_1,x_2,-1)$



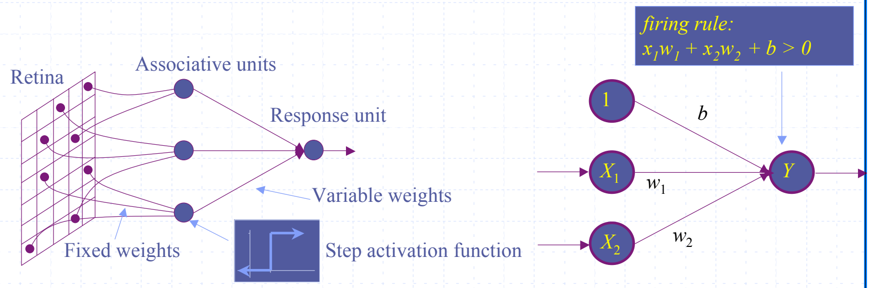
$f(\mathbf{x})=\mathbf{w}^T\mathbf{x}-b$
X_2
$\mathbf{w}$
b
$\mathbf{x}$
$\mathbf{w}^T\mathbf{x}-b$
X_1

$f(\mathbf{x})<0$ $f(\mathbf{x})=0$ $f(\mathbf{x})>0$

18

Neural Networks (NN) Background

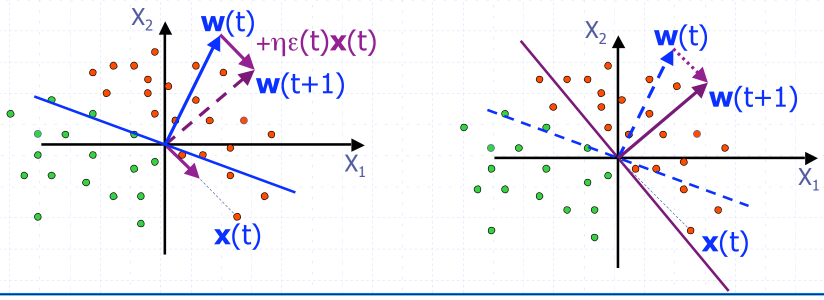
- History:
  - McCulloch-Pitt [1943]: synaptic connection as a linear classifier
  - Hebb [49]: reinforcement learning
  - Rosenblatt [57]: the perceptron training algorithm
  - Minsky & Papert [69]: what can a perceptron compute?
  - Starting mid 80's large explosion of NN models and apps…
- Rosenblatt's model:



19

The Perceptron Training Rule

- Training data {**x**(k),y(k)}
- Weight update rule: **w**(t+1)←**w**(t) +ηε(t)**x**(t)
  - ε(t)=y(t)-sgn[**w**(t)**x**(t)] is the classification error which is 0, -2, or 2
  - η is constant
  - (some variants use η=η(t), e.g., η(t)=1/t^2)

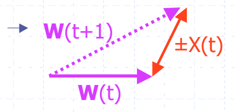


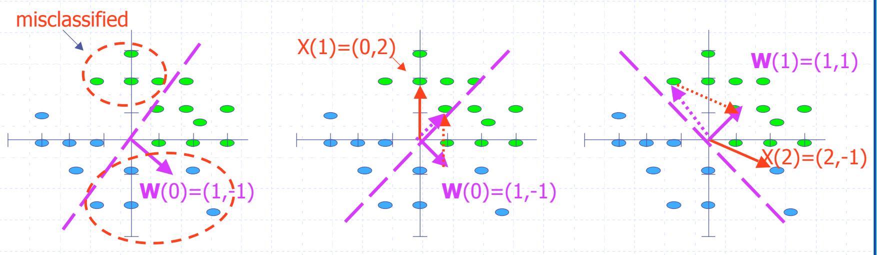
20

Training Geometry

- Initialize: w(0)=(1,-1) η=1/2
- Iterate: $\mathbf{w}(t+1) \leftarrow \mathbf{w}(t)+\eta\varepsilon(t)\mathbf{x}(t)$; $\varepsilon(t)=y(t)-\text{sgn}[\mathbf{w}(t)\mathbf{x}(t)]$

Add/subtract misclassified points

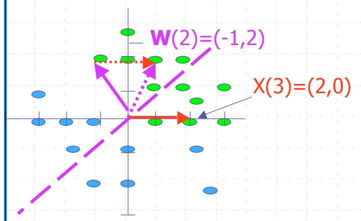




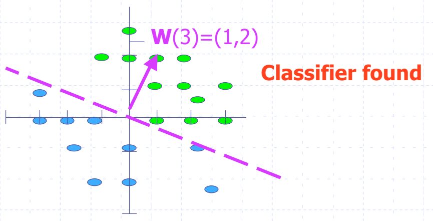
W(1)=(1,-1)+(1/2)*2*(0,2)=(1,1) W(2)=(1,1)-(2,-1)=(-1,2)

Training

$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t)+\eta\varepsilon(t)\mathbf{x}(t)$; $\varepsilon(t)=y(t)-\text{sgn}[\mathbf{w}(t)\mathbf{x}(t)]$



W(3)=(-1,2)+(2,0)=(1,2)



W(3)=(-1,2)+(2,0)=(1,2)

W(3)=(-1,1)+X(1)-x(2)+X(3)

More Generally

- A linear classifier y=sgn[f($\underline{\mathbf{x}}$)]=sgn[$\underline{\mathbf{w}}^T\underline{\mathbf{x}}$+$w_0$]
- The classifier may be represented as: f($\underline{\mathbf{x}}$)=($\underline{\mathbf{w}}^T$,w_0) $\begin{pmatrix}\underline{\mathbf{x}}\\1\end{pmatrix}$

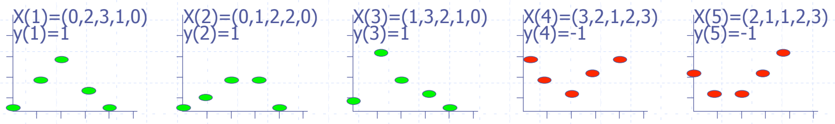
The perceptron training problem:

- Given: a training sample S={$\underline{\mathbf{x}}$(k),y(k)}
- Compute: $\underline{\mathbf{w}}$ such that y=sgn[$\underline{\mathbf{w}}^T\underline{\mathbf{x}}$] is consistent with S

23

Training To Classify Temporal Expression

- Classify temporal gene profile curves as convex~ 1 or concave ~ -1
  - Note: co-expression is revealed through montonicity & convexity
- Training patterns:



- Initialize weights w(0)=(1,1,1,1,1,0); η=1/2
  - It takes 3 iterations to converge (see table) to the classifier f(**X**)=sgn[(-2,1,3,0,-2)**X**]

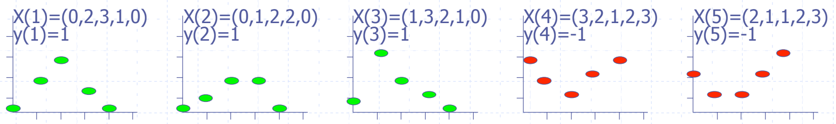
t	W(t)	x(t)	ε(t)
0	1,1,1,1,1,0	0,2,3,1,0,1	0
1	1,1,1,1,1,0	3,2,1,2,3,1	-2
2	-2,-1,0,-1,-2,-1	0,2,3,1,0,1	2
3	-2,1,3,0,-2,0	no errors (stop)	

w(t+1)$\Leftarrow$**w**(t)+ηε(t)**x**(t);
ε(t)=y(t)-sgn[**w**(t)**x**(t)]

24

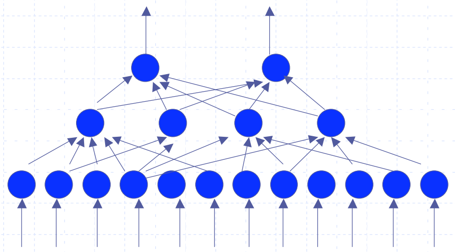
Example Notes

- Does f(**X**)=sgn[(-2,1,3,0,-2)**X**] discriminate convex from concave?
  - Try convex X=(2,4,2,1,0) f(X)=1; concave X=(2,0,1,3,4) f(X)=-1
  - But for X=(5,6,1,0,0) f(X)=-1; X=(0,3,4,2,0) f(X)=1; both are classification errors
- What did the perceptron learn from the training samples?
  - Assign heavily negative weights to the extremes; positive weights to the middle
  - Concave curves are higher at extremes; convex ones are higher at middle points
  - Training discovers weights distinguishing class features
  - Classification errors occur when patterns are mismatched with these features



Linear Classifiers May Be Generalized

- Linear classifiers are limited and sensitive to noise
- Would like to generalize them to admit non-linearity & noise
- Generalize to multilayer Neural Networks
  - Can handle non-linearity but
  - Have limited noise resiliency, are difficult to scale, train or interpret…
  - Convergence problems of gradient learning (e.g., local minima)..
  - Unclear how to avoid overfitting..
  - Have limited use in handling microarray data
- Generalize to Support Vector Machines (SVM)
  - Retain simplicity while offering new capabilities

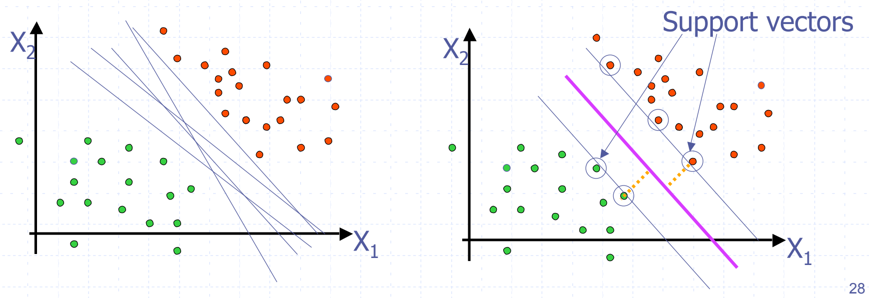


A multilayer NN

Support Vector Machines (SVM)

Generalizing The Perceptron

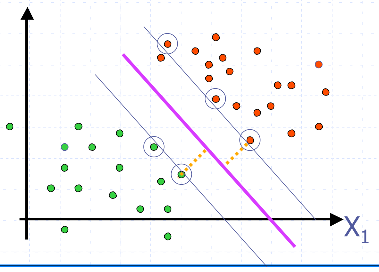
- The perceptron rule may be rewritten as:
  - if $y(t)\mathbf{w}(t)\mathbf{x}(t)<0$ then $\mathbf{w}(t+1) \Leftarrow \mathbf{w}(t)+\eta y(t)\mathbf{x}(t)$
- This means that $\mathbf{w}=\Sigma\alpha(t)y(t)\mathbf{x}(t)$ where $\alpha(t)\geq 0$
  - Learning: compute $\alpha(t)$ from sample data $\{y(t),\mathbf{x}(t)\}$
- There could be many separating hyperplanes
  - SVM: find the “best” hyperplane =max margins
  - Leads to a quadratic optimization problem



The Dual Learning Rule

- Rewrite the classifier: $f(x)=<\mathbf{wx}>+b=\Sigma\alpha(t)y(t)<\mathbf{x}(t)\mathbf{x}>+b$
- $f(x)=\Sigma\alpha(t)y(t)K(\mathbf{x}(t),\mathbf{x})+b$
  - $K(x,z)=\mathbf{x}\cdot\mathbf{z}$ -- the **kernel** --measures correlation between x(t) and x
- Dual learning rule:

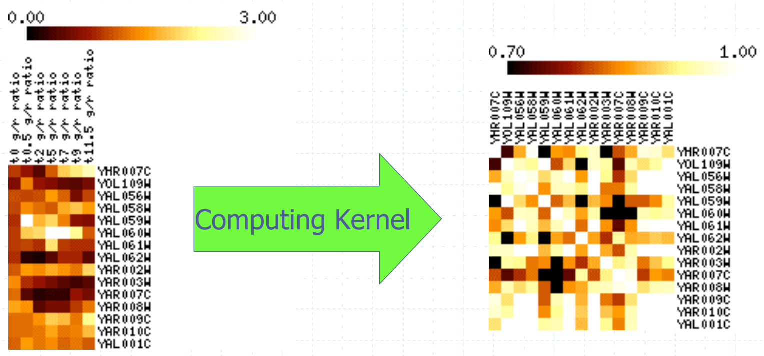
$$\text{if } \underset{\text{Real classification}}{y(i)}[\underbrace{\Sigma\alpha(t)y(t)K(\mathbf{x}(t)\mathbf{x}(i))+b}_{\text{Predicted classification}}]<0 \text{ then } \alpha(i)\leftarrow\alpha(i)+\eta$$



Training A Linear SVM

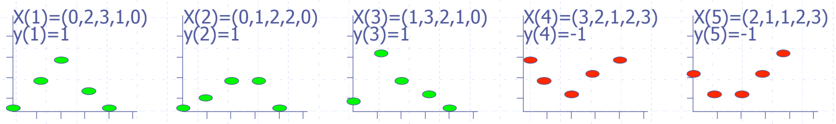
- Compute the kernel matrix
- Iterate the SVM learning rule

$$\text{if } y(i)[\Sigma\alpha(t)y(t)K(t,i)+b]\leq 0 \text{ then } \alpha(i)\leftarrow\alpha(i)+\eta$$



An SVM Training Example

- Consider again the convex/concave classification



- Compute a linear kernel

$$X=\underbrace{\begin{pmatrix}0,0,1,3,2\\2,1,3,2,1\\3,2,2,1,1\\1,2,1,2,2\\0,0,0,3,3\\1,1,1,1,1\end{pmatrix}}_{\mathbf{x}(1)\ldots\mathbf{x}(5)} \qquad K=X^TX=\begin{pmatrix}0,2,3,1,0,1\\0,1,2,2,0,1\\1,3,2,1,0,1\\3,2,1,2,3,1\\2,1,1,2,3,1\end{pmatrix}\begin{pmatrix}0,0,1,3,2\\2,1,3,2,1\\3,2,2,1,1\\1,2,1,2,2\\0,0,0,3,3\\1,1,1,1,1\end{pmatrix}=\begin{pmatrix}15,11,14,10,8\\11,10,10,9,8\\14,10,16,14,10\\10,9,14,28,23\\8,8,10,23,20\end{pmatrix}$$

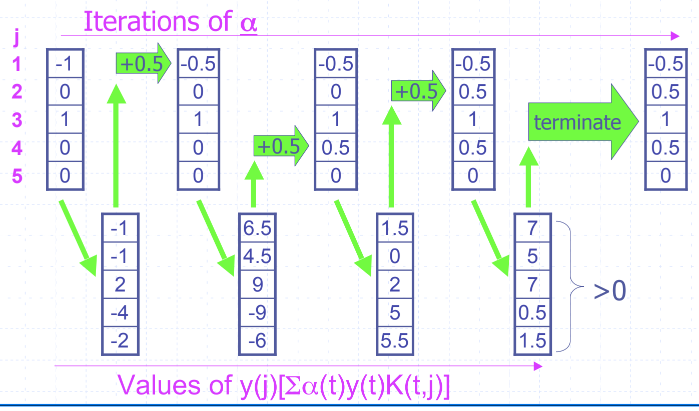
31

Training

- Initialize: η=0.5 $\underline{\alpha}$=(-1,0,1,0,0) **y**=(1,1,1,-1,-1)

$$K=\begin{pmatrix}15,11,14,10,8\\11,10,10,9,8\\14,10,16,14,10\\10,9,14,28,23\\8,8,10,23,20\end{pmatrix}$$

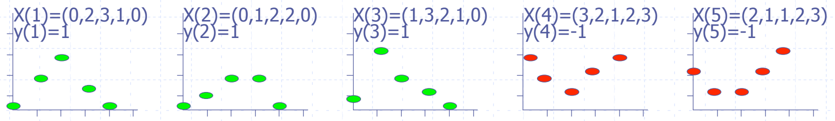
- Iterate:
  if $y(j)[\Sigma\alpha(t)y(t)K(t,j)]\leq 0$ then $\alpha(j)\leftarrow\alpha(j)+\eta$



32

SVM Example Continued

- Training result: $\underline{\alpha}$=(-0.5,0.5,1,0.5,0)=0.5(-1,1,2,1,0)
- Computing the SVM classifier:
  $f(x)=<[\Sigma\alpha(t)y(t)\mathbf{x}(t)],\mathbf{x}>=<\mathbf{w},\mathbf{x}>$
  $\mathbf{w} = \Sigma\alpha(t)y(t)\mathbf{x}(t)=0.5(-\mathbf{x}(1)+\mathbf{x}(2)+2\mathbf{x}(3)-\mathbf{x}(4))=$
  $=0.5(-1,3,2,1,-3,1)$
- Classifier: $f(x)= -x_1+3x_2+2x_3+x_4-3\ x_5+1$



33

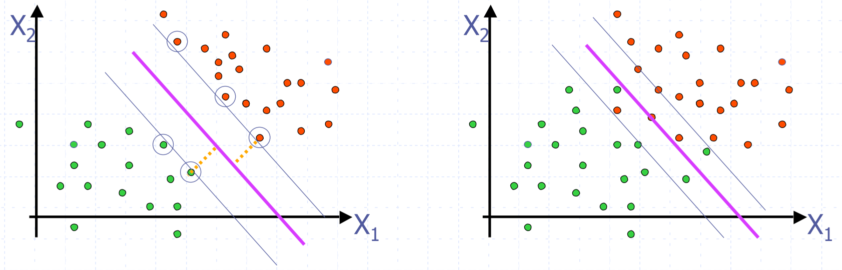
Notes On SVM Training

- What did the SVM classifier learn about convexity?
  - $f(x)= -x_1+3x_2+2x_3+x_4-3\ x_5+1$ much like perceptron, assigns negative weights to the extremes and positive to the middle
  - Consider the samples misclassified by the perceptron:
    X1=(5,6,1,0,0); X2=(0,3,4,2,0)
  - The SVM classifier classifies X1 correctly but errs in classifying X2
  - (What is the source of the error? What training samples can improve this?)

34

Handling Non-Separable Data

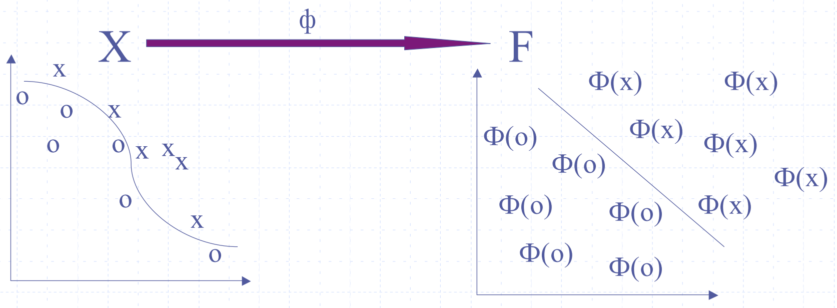
- When data is not separable use soft-margins
- Optimize margins given a relative cost of error



35

Generalization To Kernel Machines

- Simplified approach to non-linear classification
  - Map data to feature space via non-linear transformation
  - Use linear classification in feature space
  - F may have different dimension than X (e.g., reduce dimensionality)



36

Linear Classification in Feature Space

- Consider the classification in feature space: $f(x)=\Sigma\alpha(t)y(t)<\phi(\mathbf{x}(t))\phi(\mathbf{x})>+b$
- Define the Kernel of the transformation: $K(u,v)=<\phi(u),\phi(v)>$
- The kernel specifies the "feature space" classifier:

$$f(x)=\Sigma\alpha(t)y(t)K(\mathbf{x}(t),\mathbf{x})+b$$

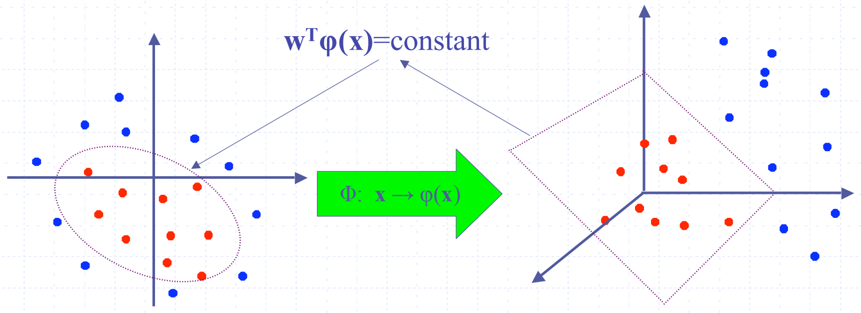
Example Kernel Functions:

1) Polynomial, $\Phi(x_i,x_j)=(x_i\ x_j+1)^d$
2) Gaussian, $\Phi(x_i,x_j)=e^{-\| x_i-x_j\|/\sigma 2}$

37

Example: Polynomial Kernel

- $K(\mathrm{x}_i,\mathrm{x}_j)=(1+\mathrm{x}_i^T\mathrm{x}_j)^2$
- $K(\mathrm{x}_i,\mathrm{x}_j)=1+x_{i1}^2x_{j1}^2+2\,x_{i1}x_{j1}\,x_{i2}x_{j2}+x_{i2}^2x_{j2}^2+2x_{i1}x_{j1}+2x_{i2}x_{j2}$
$=[1,x_{i1}^2,\sqrt{2}\,x_{i1}x_{i2},x_{i2}^2,\sqrt{2}x_{i1},\sqrt{2}x_{i2}]^T\,[1,x_{j1}^2,\sqrt{2}\,x_{j1}x_{j2},x_{j2}^2,\sqrt{2}x_{j1},\sqrt{2}x_{j2}]$
- $K(\mathrm{x}_i,\mathrm{x}_j)==\varphi(\mathrm{x}_i)^T\varphi(\mathrm{x}_j)$, where $\varphi(\mathrm{x})=[1,x_1^2,\sqrt{2}\,x_1x_2,x_2^2,\sqrt{2}x_1,\sqrt{2}x_2]$


$\mathbf{w}^T\varphi(\mathbf{x})$=constant
Φ: $\mathbf{x}\rightarrow\varphi(\mathbf{x})$

38

The Kernel "Trick"

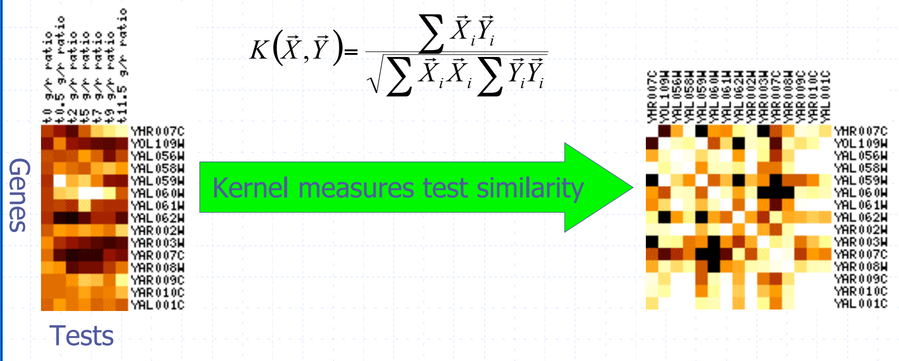
Consider the classifier: $f(x)=\Sigma\alpha(t)y(t)K(\mathbf{x}(t),\mathbf{x})+b$ and
training algorithm: $y(i)[\Sigma a(t)y(t)K(\mathbf{x}(t),x(i))+b]\leq 0$ then $\alpha(i)\leftarrow\alpha(i)+\eta$

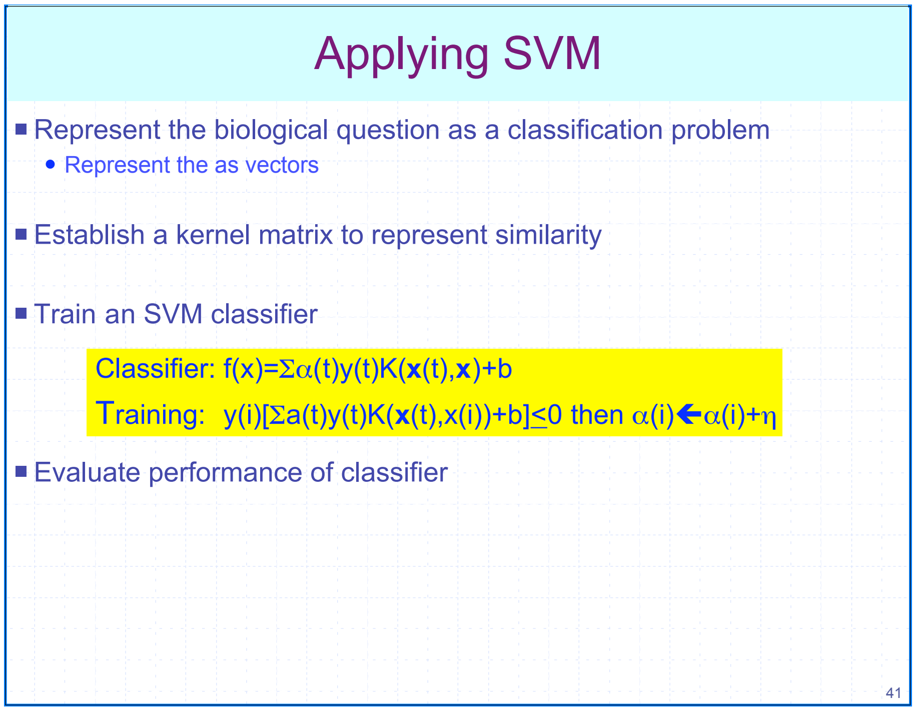
- An SVM classifier may be computed from the kernel alone
  - No need to know the underlying mapping $\varphi(x)$
- We just need to know that the kernel is appropriate
  - $K(u,v)=<\varphi(u),\varphi(v)>$ for some φ
- Mercer: any symmetric positive definite matrix is a kernel

Example: SVM Classification of Microarrays

- Kernel provides a measure of similarity

$$K(\vec{X},\vec{Y})=\frac{\sum \vec{X}_i\vec{Y}_i}{\sqrt{\sum \vec{X}_i\vec{X}_i\sum \vec{Y}_i\vec{Y}_i}}$$





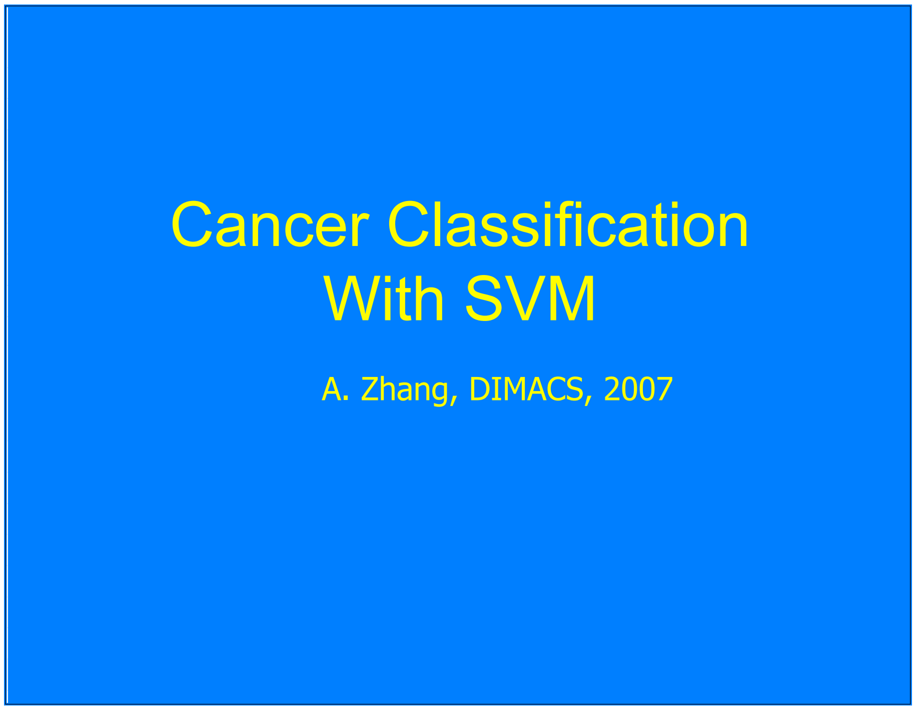
Applying SVM

- Represent the biological question as a classification problem
  - Represent the as vectors
- Establish a kernel matrix to represent similarity
- Train an SVM classifier

  Classifier: $f(x)=\Sigma\alpha(t)y(t)K(\mathbf{x}(t),\mathbf{x})+b$

  Training: $y(i)[\Sigma a(t)y(t)K(\mathbf{x}(t),x(i))+b]\leq 0$ then $\alpha(i)\leftarrow\alpha(i)+\eta$
- Evaluate performance of classifier

41



Cancer Classification With SVM

A. Zhang, DIMACS, 2007

Cancer Classification Study

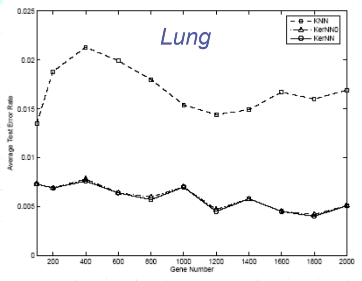
A. Zhang, DIMACS, 2007

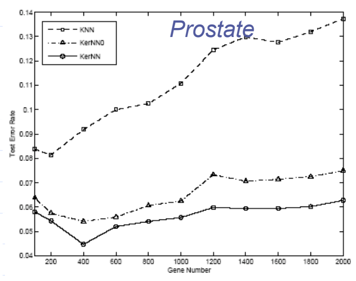
- Microarrays provide a small sample of high-dimensional data
  - Key challenge: overfitting
- Comparative study of classifiers over Microarray DBs
- Use SVM with improved kernel (max discrimination)
- Compare with kNN and linear discriminant classifiers

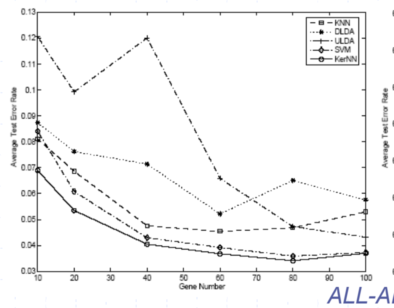
	sample size	number of genes
ALL-MAL	72	7129
Breast-ER	49	7129
Breast-LN	49	7129
CNS	60	7129
Colon	62	2000
Lung	181	12533
Lymphoma	77	7129
Ovarian	253	15154
Prostate	102	12600

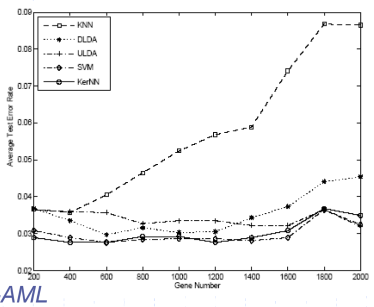
http://dimacs.rutgers.edu/Workshops/MLTechniques/slides/z 43

Cancer Classification









ALL-AML

44

But….

www.sci.usq.edu.au/research/seminars/files//seminar135/ausdm1.ppt

Decision-trees Boosting

Data set	C4.5	Random Forests	AdaBoostC4.5	BaggingC4.5	LibSVMs
Breast Cancer	84.5	88.7	90.7	85.6	72.2
Lung Cancer	98.3	99.5	98.3	97.8	100.0
Lymphoma	74.5	93.6	89.4	89.4	55.3
Leukemia	88.9	98.6	95.8	95.8	100.0
Colon	88.7	83.9	90.3	90.3	90.3
Ovarian	96.8	99.2	98.8	98.0	100.0
Prostate	95.2	100	95.2	95.2	100.0
Average	89.6	94.8	94.1	93.2	88.3

45

Cancer Studies

A. Statnikov, C. F. Aliferis,I. Tsamardinos.

Vanderbilt University, MEDINFO 2004

Microarray Datasets

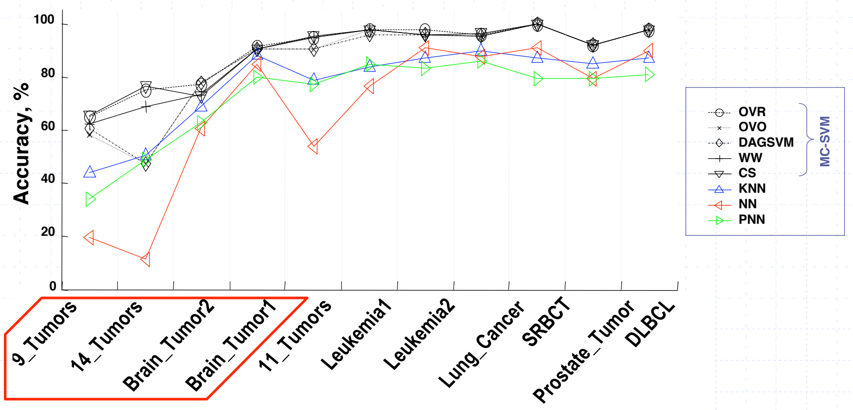
Dataset name	Number of Sam-ples	Number of Variables (genes)	Number of Cate-gories	Reference
11_Tumors	174	12533	11	Su, 2001
14_Tumors	308	15009	26	Ramaswamy, 2001
9_Tumors	60	5726	9	Staunton, 2001
Brain_Tumor1	90	5920	5	Pomeroy, 2002
Brain_Tumor2	50	10367	4	Nutt, 2003
Leukemia1	72	5327	3	Golub, 1999
Leukemia2	72	11225	3	Armstrong, 2002
Lung_Cancer	203	12600	5	Bhattacherjee, 2001
SRBCT	83	2308	4	Khan, 2001
Prostate_Tumor	102	10509	2	Singh, 2002
DLBCL	77	5469	2	Shipp, 2002

Total:
- ~1300 samples
- 74 diagnostic categories
- 41 cancer types and 12 normal tissue types

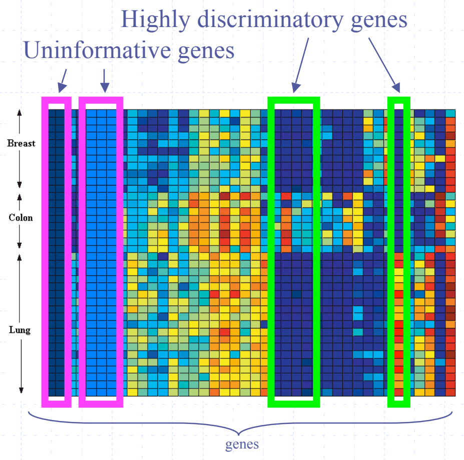
Classifiers

• K-Nearest Neighbors (KNN)	instance-based
• Backpropagation Neural Networks (NN)	neural networks
• Probabilistic Neural Networks (PNN)	
• Multi-Class SVM: One-Versus-Rest (OVR)	kernel-based
• Multi-Class SVM: One-Versus-One (OVO)	
• Multi-Class SVM: DAGSVM	
• Multi-Class SVM by Weston & Watkins (WW)	
• Multi-Class SVM by Crammer & Singer (CS)	
• Weighted Voting: One-Versus-Rest	voting
• Weighted Voting: One-Versus-One	
• Decision Trees: CART	decision trees

Without Gene Selection



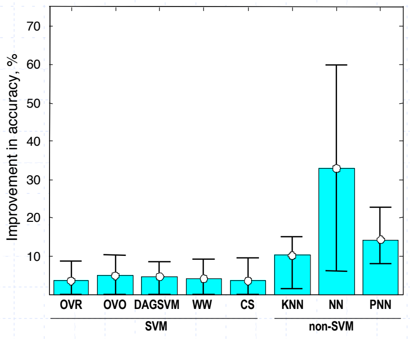
Gene Selection



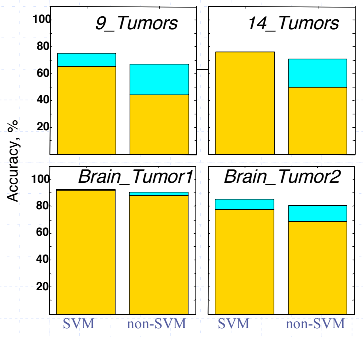
1. Signal-to-noise (S2N) ratio in one-versus-rest (OVR) fashion;
2. Signal-to-noise (S2N) ratio in one-versus-one (OVO) fashion;
3. Kruskal-Wallis nonparametric one-way ANOVA (KW);
4. Ratio of genes between-categories to within-category sum of squares (BW).

With Gene Selction

Improvement of diagnostic performance by gene selection (*averages for the four datasets*)



Diagnostic performance *before* and *after* gene selection



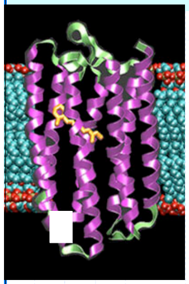
Average reduction of genes is 10-30 times

51

Protein Classification

(Based on W. S. Noble U. Washington)

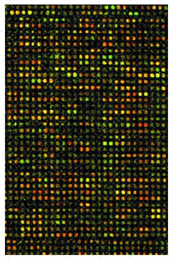
Classifying Transmembrane proteins



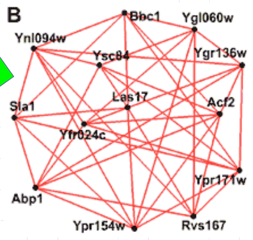
Challenge: build a classification model for diverse data

```
QFDACCFIDDVSKIYG-DYGPI
QFDACCFIDDVSKIYG-DHGPI
QFGACCFIDDVSKTFRLHDGPL
QFDAC-FIDDVSKIFRLHDGPL
RFDASCFIDDVSKIFRLHDGPI
QFSVYCLIDDVSKIYR-HDGPM
QFPVCSIIDDLSKMYR-HDSPV
QFPVFCLIDDLSKIYR-DDGLI
QFDARCFIDDLSKIYR-HDGQV
QFDARCFIDDLSKIYR-HDGQV
QFDARCFIDDLSKIYR-HDGPI
RFDACCFIDDVSKICK-HDGPV
QFDACCFIDDVSKICK-HDGPV
```

sequence data



mRNA expression data

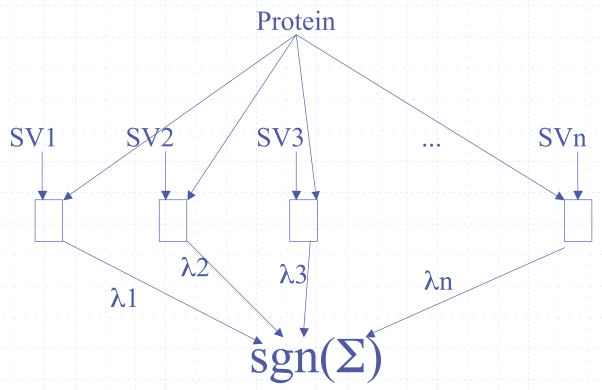


protein-protein interaction data

53

Key Idea

- Represent classification data in terms of SVM kernels
- Combine kernels to best apply all discriminating data



54

Classification Based on Sequence

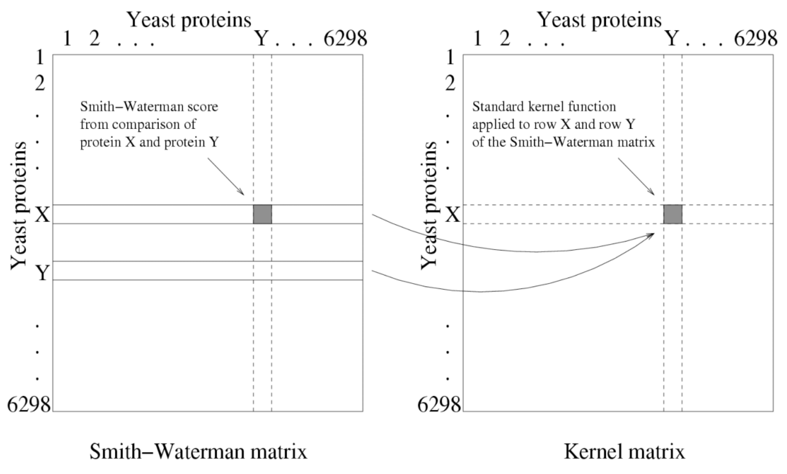
```
>ICYA_MANSE
GDIFYPGYCPDVKPVNDFDLSAFAGAWHEIAKLPLENENQGKCTIAEYKY
DGKKASVYNSFVSNGVKEYMEGDLEIAPDAKYTKQGKYVMTFKFGQRVVN
LVPWVLATDYKNYAINYNCDYHPDKKAHSIHAWILSKSKVLEGNTKEVVD
NVLKTFSHLIDASKFISNDFSEAACQYSTTYSLTGPDRH

>LACB_BOVIN
MKCLLLALALTCGAQALIVTQTMKGLDIQKVAGTWYSLAMAASDISLLDA
QSAPLRVYVEELKPTPEGDLEILLQKWENGECAQKKIIAEKTKIPAVFKI
DALNENKVLVLDTDYKKYLLFCMENSAEPEQSLACQCLVRTPEVDDEALE
KFDKALKALPMHIRLSFNPTQLEEQCHI
```

- How do we model strings-similarity in terms of a kernel matrix?
- Define sequence kernel in terms of similarity score

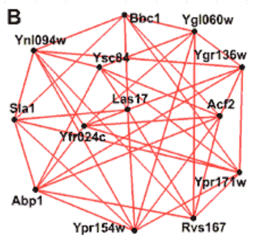
55

Pairwise comparison kernel



56

Classification By Interaction Profile



- How do we build classification model from protein interactions graph?
- Interaction Kernel: # of common neighbors

protein

```
           1 0 0 1 0 1 0 1
           1 0 1 0 1 1 0 1
           0 0 0 0 1 1 0 0
           0 0 1 0 1 1 0 1
protein →  0 0 1 0 1 0 0 1
           1 0 0 0 0 0 0 1
           0 0 1 0 1 0 0 0
```

Incidence matrix

Interaction Kernel →

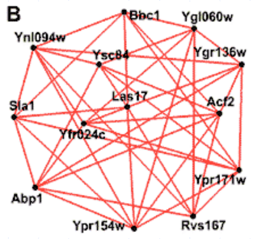
```
1 0 0 1 0 1 0 1
1 0 1 0 1 1 0 1
0 0 0 0 1 1 0 0
0 0 1 0 1 1 0 1
0 0 1 0 1 0 0 1
1 0 0 0 0 0 0 1
0 0 1 0 1 0 0 0
```

3

57

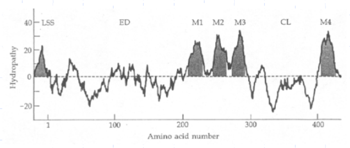
Diffusion kernel

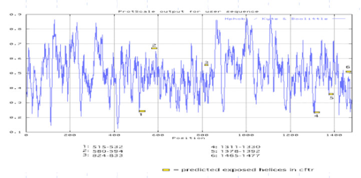
- General metric of similarity between graph nodes
- Based upon a random walk
- Kernel ~average time for random walk starting at x to first visit y
  - (# paths connecting two nodes, weighted by path lengths)



58

Hydrophobicity Kernel



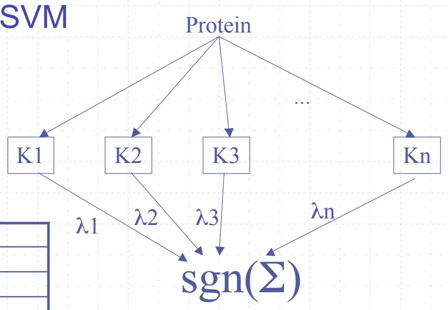


- Transmembrane regions are typically hydrophobic
- The hydrophobicity profile is conserved
- Represent data in terms of a kernel

59

Combining Kernel Machines

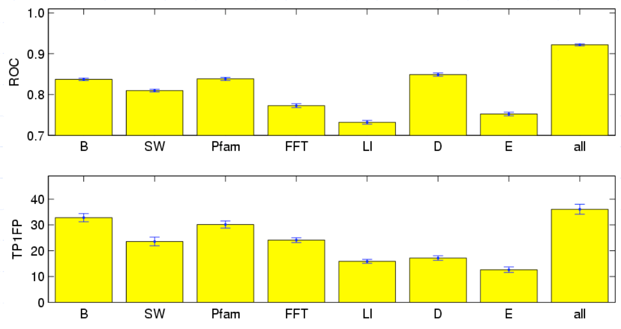
- Given kernels $K_i(u,v)= <\varphi_i(u),\varphi_i(v)>$
- Define a combined kernel $K(u,v)=\Sigma\lambda_i K_i(u,v)$ $(1=\Sigma\lambda_i)$
  - Corresponds to the mapping $\varphi(u)=(\varphi_1(u), \varphi_2(u),\ldots\varphi_n(u))$
  - And weighted inner product $<\varphi(u),\varphi(v)>=\Sigma\lambda_i<\varphi_i(u),\varphi_i(v)>$
- Extend training to combined SVM



Kernel	Data	Similarity measure
K_{SW}	protein sequence	Smith-Waterman
K_B	protein sequence	BLAST
K_{HMM}	protein sequence	Pfam HMM
K_{FFT}	hydropathy profile	FFT
K_{LI}	protein interactions	linear kernel
K_D	protein interactions	diffusion kernel
K_E	gene expression	radial basis kernel

60

Membrane Proteins



%TP at 1%FP

Simple rules from hydrophobicity profile

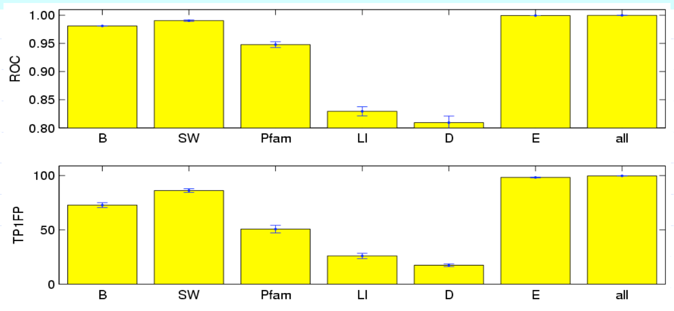
TMHMM

Combined SVM

Method	ROC	TP1FP
f_1	0.7345	16.70%
f_2	0.7504	13.48%
f_3	0.7879	21.93%
T_{PH}	0.7362	30.02%
T_{ENR}	0.8018	31.38%
SDP/SVM	0.9219	36.06%

61

Cytoplasmatic Ribosomal Proteins



What Can Errors Teach ?

Table 3: **Consistently misclassified proteins: cytoplasmic ribosome.** The table lists proteins that are consistently misclassified by SDP/SVM. The score column lists the mean SVM discriminant across multiple splits.

ORF	Gene	Error	Score	Description
YLR287C-A	RPS30A	FN	-0.097	40S small subunit ribosomal protein
YPL131W	RPL5	FN	-0.162	60S large subunit ribosomal protein L5.e
YGL189C	RPS26A	FN	-0.272	40S small subunit ribosomal protein S26e.c7
YFL034C-A	RPL22B	FN	-0.286	ribosomal protein
YLR406C	RPL31B	FN	-0.313	60S large subunit ribosomal protein L31.e.c12
YIL069C	RPS24B	FN	-0.510	40S small subunit ribosomal protein S24.e
YDL130W	RPP1B	FN	-0.524	60S large subunit acidic ribosomal protein L44prime

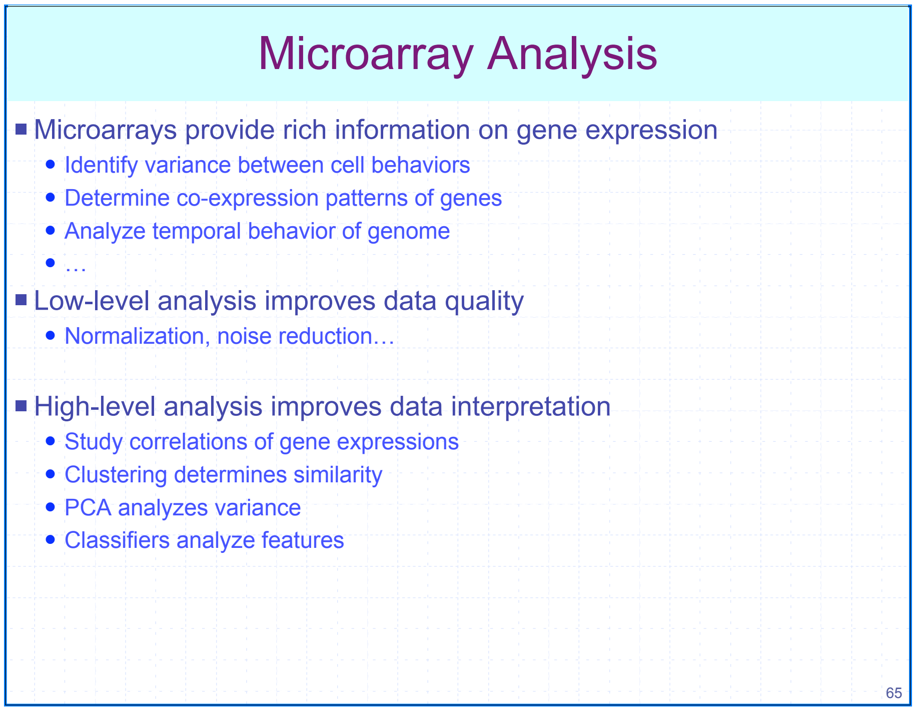
62

SVM Final Notes

- Kernel machines provide powerful classifiers
  - Kernels admit flexible modeling of similarity
  - Simple and general training procedure
  - Multiple classifiers may be combined to improve results
  - …
- But..
  - Choosing a good kernel is an art
  - Training results may be sensitive to training sample
- Other classification ideas
  - Boosting
  - Decision trees
  - ….

63

Conclusions



Microarray Analysis

- Microarrays provide rich information on gene expression
  - Identify variance between cell behaviors
  - Determine co-expression patterns of genes
  - Analyze temporal behavior of genome
  - …
- Low-level analysis improves data quality
  - Normalization, noise reduction…
- High-level analysis improves data interpretation
  - Study correlations of gene expressions
  - Clustering determines similarity
  - PCA analyzes variance
  - Classifiers analyze features

65