# Learning Good Representations for Learning, Part II

Summary:

1. Representing images: Edges (with Ben Allen)

2. Representing general features: incorporating prior knowledge (Charles DeLisi, Yue Fan)

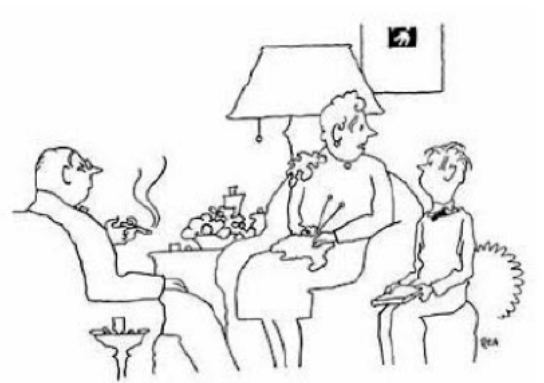
Invariance (e.g. translation-invariance) in representating images is good for many purposes, including

Image Compression: fewer numbers needed to store image

Image Classification: fewer numbers to confuse classifier (in identifying image)!

So good compression and good classification share an important common goal: fewer numbers to represent the image.

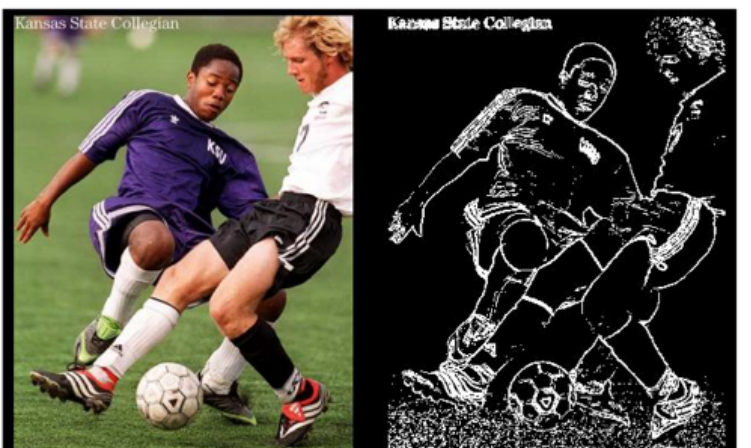
Widely studied way to store and analyze images with fewer numbers: keep only line/edge information:



Rea Gardner, New Yorker

## Now-classic method (1980's):  edge detection

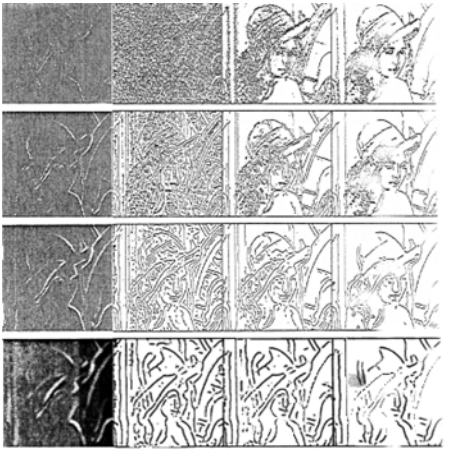
Image:



## Multiscale edges:

N. Saito, http://www.math.ucdavis.edu/~saito/talks/ucdmathbio.pdf

## Multiscale edges:



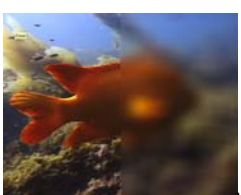
S. Mallat

Mallat's edge-based compression algorithm

Mathematically - can blur image by convolving with Gaussian:

$$f(x) \rightarrow f_\sigma(x) = f(x) * g_\sigma(x) = \int_{-\infty}^{\infty} f(x-y) g_\sigma(y) dy;$$

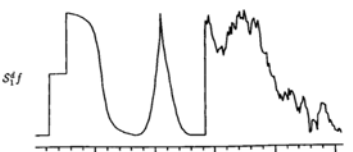
$$g_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}.$$



www.borisfx.com

Technically, find edges as zero crossings of

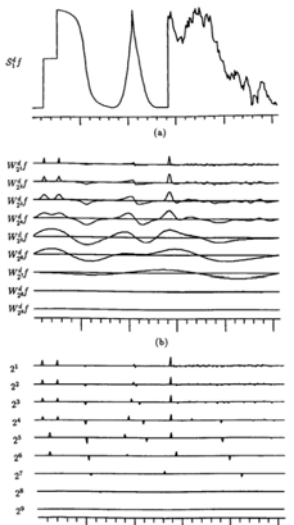
$$\Delta f_\sigma(x) = \frac{d^2}{dx^2} f_\sigma(x).$$



original image $f(x)$

$\rightarrow$


blurred image $f_\sigma(x)$




edge locations: multiscale zeroes of $\frac{d^2}{dx^2} f_\sigma(x)$ (various scales $\sigma$)

Upper left: original image; upper right reconstructed image (S. Mallat, 1992)

Stephane Mallat (1992) reconstructed images from edge information using (instead of Gaussian function)

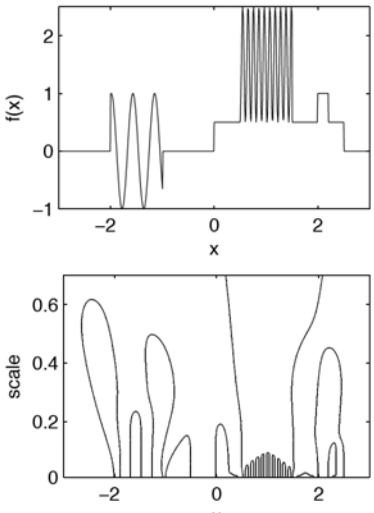
$$g(x) = \text{cubic B spline.}$$

Marr conjecture: an image is uniquely determined by its Gaussian multiscale edges.

1 dimensional version: slices of images (like functions above) are uniquely determined from their edge information.

Some significant work:

- Logan (1977) (band-limited images $f(x)$)
- Curtis, Shitz, Oppenheim (1987) (band-limited signals $f(x)$)
- Yuille, Poggio (1987); polynomial signals $f(x)$ in multiple dimensions; less edge information needed
- Hummel, Monoit (1989) (knowledge of gradient of $f(x)$)
- Meyer (1994) (counterexample -false if infinite extent)




A 1-D image intensity map and its Gaussian edges (of blurred function); scale = blurring level

**Theorem:** *Assume a signal $f(x)$ representing the intensity (brightness) of an image at point $x$. Assume we know only the edges of $f(x)$ for a fixed infinite sequence of blurring scales $\sigma_1, \sigma_2, \ldots$ (not converging to 0). Then this sequence of zeroes*

*(a) uniquely determines $f(x)$ if it decays exponentially at infinity, i.e., if $|f(x)| \leq Ce^{-k|x|}$.*

*(b) does not uniquely determine $f(x)$ generally if it decays at most algebraically, i.e., $|f(x)| \geq C|x|^{-l}$ for some $l > 0$.*

*(c) However, if the scale sequence $\sigma_i \to 0$ (i.e., $f_{\sigma_i} \to f$ !), the zeroes generally do not determine the image $f(x)$ uniquely for any decay rate of $f$.*

Conclusion: knowing edges of a blurred image $f(x)$ for a any sequence of blurring levels determines $f(x)$ uniquely, unless the blurrings $\sigma_1, \sigma_2, \ldots$ approach 0 (i.e. almost no blurring at all).

**Conclusion:** *An image is determined by its (multiscale) edges if the image intensity $f(x)$ decays exponentially (e.g. is bounded in extent!).*

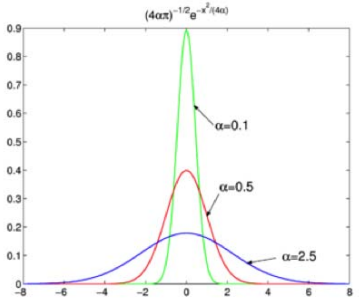
**Conclusion:** *There are images which decay algebraically arbitrarily fast (i.e., like $|f(x)| \leq C|x|^{-l}$ for any $l > 0$) but which are not determined by their edges.*

Extends Meyer result showing that non-decaying (infinite extent) functions are not determined by their edges.

**Proof of Theorem:** For our one dimensional case write

$$f(x) = \sum_{k=1}^{\infty} a_k \delta^{(k)}(x),$$

where $\delta(x)$ is the delta function:



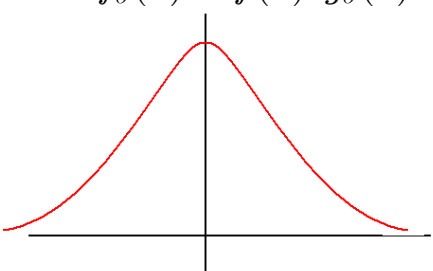

http://en.citizendium.org/wiki/Dirac_delta_function

$$\delta(x) = \lim_{\sigma \to 0} g_\sigma(x) = \lim_{\sigma \to 0} \frac{1}{(2\pi)^{1/2}\sigma} e^{-x^2/(2\sigma)}$$

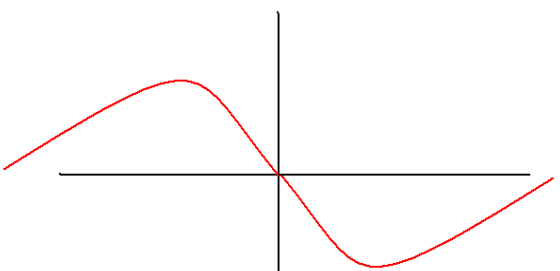
Idea: First term $a_0\delta(x)$ of series represents 'monopole moment' of $f$, i.e., for large blurring $\sigma$, to first order

$$a_0\delta(x)*g_\sigma(x) \approx f(x)*g_\sigma(x),$$

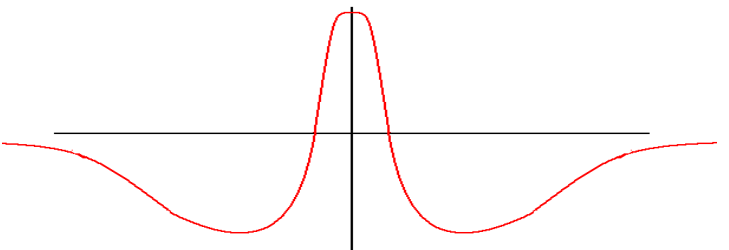
i.e., for very large blurrings $\sigma$, this approximates the Gaussian function $f_\sigma(x) = f(x)*g_\sigma(x)$ which looks like



Second (first derivative) term $a_1\delta^{(1)}(x)$ of series gives next order of approximation - $a_1\delta^{(1)}(x)*g_\sigma(x)$ includes 'dipole moment' term to the blurred $f_\sigma(x)$, asymptotically of the form (derivative of Gaussian)



Quadrupole moment term:



If $\sigma \to \infty$,

- Each term $a_k \delta^{(k)} * g_\sigma(x)$ in series for $f_\sigma$ has a different rate of decay

- Each $a_k$ in turn corresponds to one term in an *asymptotic series* giving rate of convergence of the edges of $f_\sigma$) toward an asymptote as $\sigma_k \to \infty$.

- This convergence rate of the zeroes determines $a_k$ and thus $f(x)$ through solution of the moment problem.

Caveat: the edges of the 'sharp' blurrings $f_\sigma(x)$ as $\sigma \to 0$ are *not* sufficient to determine $f$; an infinite set of blurrings $\sigma_i$ away from 0 is needed!

# Learning Good Representations for Classifying Cancer Subtypes

(with C. DeLisi, Y. Fan, S. Kim, R. Raphael)

Below:

I. Machine learning approaches

II. Applications to Cancer

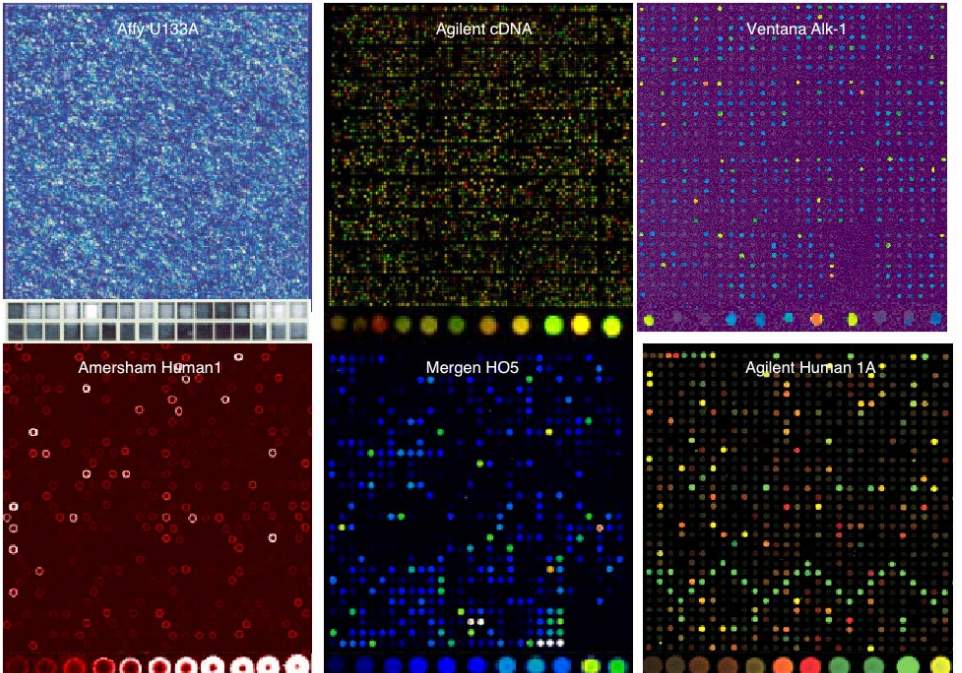
## 1. Machine learning methods in computational biology

Example: microarray-based inference

A microarray produces approximately 20k biomarkers describing a tissue sample:

# Machine learning methods

# Machine learning methods



## Machine learning methods

**Result:** for each subject tissue sample $s$, obtain feature vector

$$\Phi(s) = \mathbf{x} = (x_1, \ldots, x_{20{,}000})$$

$$= \text{vector of gene expression levels}$$

## Machine learning methods

If this is an ovarian cancer tissue sample:

**Questions:**

(a) What type of cancer is it?

(b) What is prognosis if untreated?

(c) What will be the reaction to standard chemotherapies?

## Machine learning methods

**Goals:**

1. Differentiate two different but similar cancers.

2. Determine the future course of the cancer

3. Determine what chemical agents the cancer will respond to

4. Understand genetic origins and pathways of cancer

## Machine learning methods

Basic difficulties: few samples (e.g., 30-200); high dimension (e.g., 5,000 - 100,000).

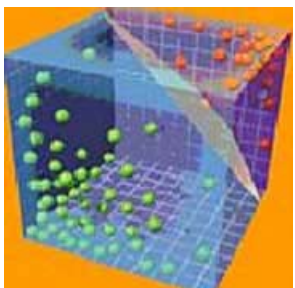
Curse of dimensionality - too few samples and too many parameters (dimensions) to fit them.

## 2. SVM as a tool

Method: Support vector machine (SVM)

Procedure: look at feature space $F$ in which $\Phi(s)$ lives, and differentiate examples of one and the other cancer with a hyperplane:

## SVM as a tool



Train machine: take $n = 50$ subjects with different responses to a particular therapy T, and locate their feature vectors in the space $F$, labeling them red (unresponsive) or green (responsive).

## SVM as a tool

Find separating hyperplane, and use this plane to separate feature vectors of future subjects into 'responsive' and 'unresponsive'.

There are a number of other *machine learning methods* which are able to discriminate feature vectors $\Phi(s)$ with respect to prognosis, response to therapies, etc.

Data obtained in collaboration with TCGA (the Cancer Genome Atlas).

**3. The principle: more is more**

## Machine learning

Past: too many variables spoil the statistics; < 50 variables was typical requirement

Present: more is better

Machine learning allows massive integration of relationship information:

## Machine learning

Relationship information on a gene level:

- protein-protein interactions
- co-expression
- gene ontology relationships
- pathway correlations
- epigenetic information (methylation, phosphorylation)

**Machine learning in study of cancer allows seamless combination of many different data types.**

**Method: kernel matrices**

## Machine learning

Kernel trick: incorporate relational information into a kernel matrix: for genes $g_i$ and $g_j$:

$$\mathbf{K}_{ij} = K(g_i, g_j) = \text{'closeness' of } g_i \text{ and } g_j$$

as measured by, e.g.,

- protein-protein interactions
- coexpression
- gene ontology
- pathway connections

Each type of gene relation gives a different kernel matrix.

## Machine learning

To integrate information in kernel matrices $\mathbf{K}^{(1)}$, $\mathbf{K}^{(2)}, \ldots, \mathbf{K}^{(n)}$, we add them:

$$\mathbf{K} = \mathbf{K}^{(1)} + \ldots + \mathbf{K}^{(n)}.$$

incorporates *all* of these measures into one.

No matter what the source of prior information in kernel matrix $\mathbf{K}^{(i)}$, it is automatically integrated with other sources by kernel addition.

Information types (for example, in The Cancer Genome

## Machine learning

Atlas, TCGA) contain:

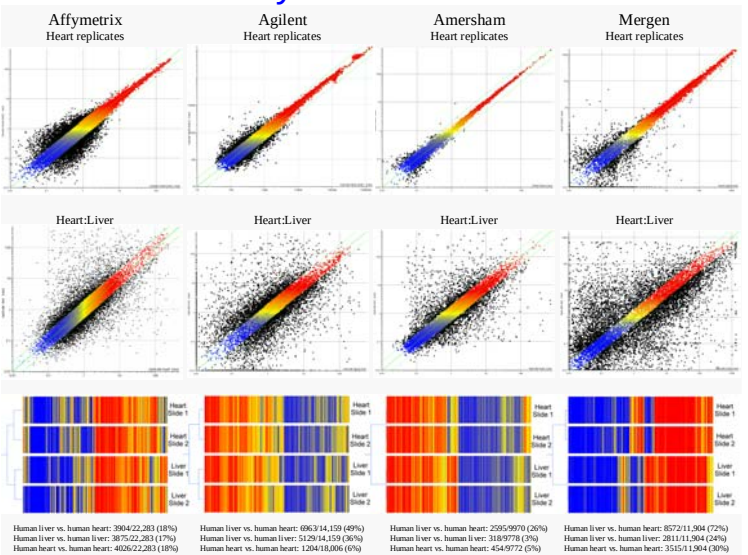
- Gene expression (microarray)
- Single nucleotide polymorphism (SNP) information
- Methylation, epigenetic information
- Gene copy numbers
- micro-RNA (miRNA) data

Noisy Biomarkers

## 4. Problem: biomarkers are noisy!

Microarray for example is non-self-replicating:

# Noisy Biomarkers



## Noisy Biomarkers

How to clean up the noise? Use the same methods as denoising functions in Euclidean space.

# Noisy Biomarkers

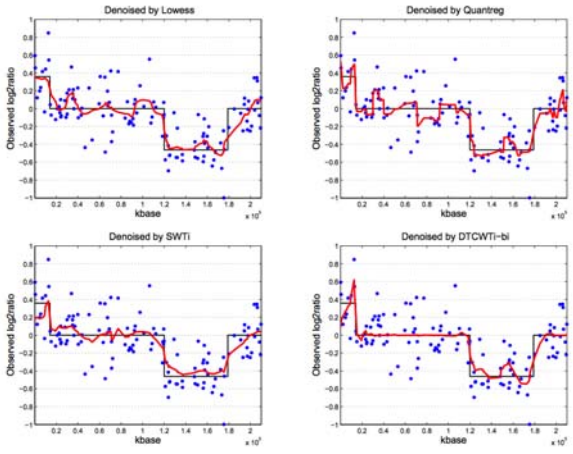



Figure: smoothing of gene copy number arrays using wavelet denoising.
Huang, et al. http://www.biomedcentral.com/content/pdf/1471-2164-9-S2-S17.pdf

## Noisy Biomarkers

What methods help in denoising functions on Euclidean spaces?

1. Local averaging (Haar wavelet denoising) - above

2. Smoothing using convolutions $f(x) \rightarrow f * g(x)$, where $g(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$ is say a Gaussian kernel.

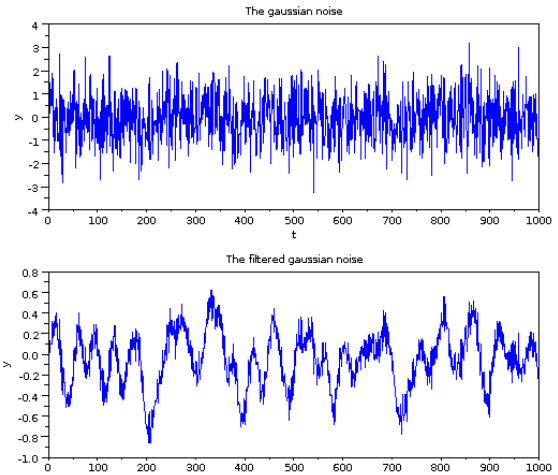
3. More generally, smoothing using kernel regression:

$$f(x) \rightarrow \sum_i \alpha_i K(x, x_i) + b$$

## Noisy Biomarkers

4. Spectral smoothing - filtering high spectral components of a function:

# Noisy Biomarkers




http://www.scilab.org/product/man/DesignEllipticFilter.html

## Noisy Biomarkers

and many other modes.

Rapaport, Vert, et al. (2007) have used spectral methods for denoising gene expression arrays.

## Euclidean denoising on gene space

**5. How to transfer Euclidean space methods to gene space?**

One can use similar methods for denoising gene expression arrays, and more generally machine learning (ML) feature vectors.

Gene expression arrays: Given a gene expression feature vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, we can *view it as a function on its indices* $G = \{1, 2, \ldots, n\}$ or equivalently the genes $g_1, \ldots, g_n$.

## Euclidean denoising on gene space

Purpose: if index set $G$ has a distance measure (e.g. a metric or network structure), and thus a notion of when two points $i, j$ in $G$ are 'close', then we will try to use this metric structure similarly to Euclidean metric to eliminate noise.

In Euclidean space denoising of a function $f(\mathbf{x})$ is done using continuity, i.e.,

$|f(\mathbf{x}) - f(\mathbf{y})|$ small when $d(\mathbf{x}, \mathbf{y})$ is small.

## Euclidean denoising on gene space

In ML denoising can be done when we expect

$$|f(i) - f(j)| \text{ small when } d(i,j) \text{ is small,}$$

where $d$ is a distance measure on indices $i, j$ (e.g. genes)

Genes in a network: if index $i$ represents gene $g_i$ and $j$ represents gene $g_j$, and if nodes $g_i$ and $g_j$ are close in the gene network, we believe their expressions $x_i$ and $x_j$ should be close to each other.

Euclidean denoising on gene space

Note this is an unsupervised method which can regularize feature vectors for any classifier (e.g., SVM, random forest, k-nearest neighbors, etc.)

## Formalities

**6. More formally:**

Given distance structure (e.g., metric or network) on the index set $G$ (e.g. genes) of a basis for a feature space $F$, so that **x** $\in F$ is

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

with index $G = \{1, 2, \ldots, n\}$.

## Formalities

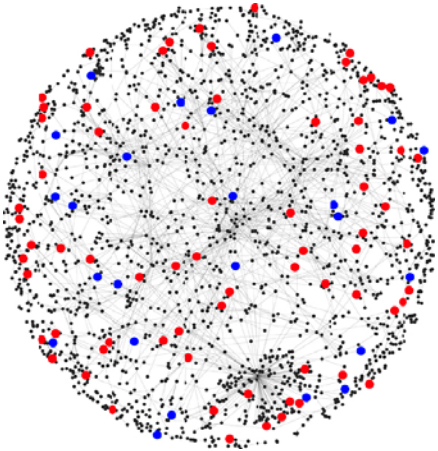
View features $x_i = f(i) = f(g_i)$ as a function on the indices $i$.

Model features (e.g. gene expressions)

$$x_i = f(g_i) = f_1(g_i) + \epsilon(g_i),$$

where $\epsilon_i$ represents noise, and $f_1(g_i)$ is the 'true' expression signal.

# Formalities



Base space $G$ for feature vector $\mathbf{x} = f(g)$ (gene network)

Lu, et al. http://www.nature.com/msb/journal/v3/n1/full/msb4100138.html

## 7. Example: local averaging noise reduction

Using the protein-protein interaction (PPI) gene network as an example: consider differentiation of metastatic and non-metastatic breast cancer (Wang; van de Vijver).

## Example: Local averaging

Wang data set:

- 93 metastatic
- 183 non-metastatic

van de Vijver data set:

- 79 metastatic
- 216 non-metastatic

How to predict metastasis?

## Example: Local averaging

Strategy - regularize the feature vectors before the classification begins.

Regularizer for feature vectors: clustering using PPI network and then averaging over clusters

Classifier: SVM

**Results:**

Area under ROC curve improved by 5% to 20%

# Example: Local averaging

| No. Clusters | Wang | | | van de Vijver | | |
|---|---|---|---|---|---|---|
| | AUROC | AUPRC | ACC90 | AUROC | AUPRC | ACC90 |
| 64 | 0.658 (0.014) | 0.450 (0.019) | 0.470 (0.027) | 0.687 (0.014) | 0.371 (0.015) | 0.472 (0.024) |
| 128 | 0.680 (0.015) | 0.462 (0.021) | 0.477 (0.023) | 0.705 (0.013) | 0.399 (0.019) | 0.520 (0.023) |
| 256 | 0.692 (0.019) | 0.475 (0.026) | 0.526 (0.029) | 0.689 (0.016) | 0.398 (0.022) | 0.490 (0.030) |
| 512 | 0.684 (0.019) | 0.487 (0.031) | 0.502 (0.029) | 0.686 (0.021) | 0.375 (0.023) | 0.489 (0.038) |
| 1024 | 0.708 (0.019) | 0.500 (0.032) | 0.527 (0.029) | 0.712 (0.019) | 0.403 (0.026) | 0.520 (0.026) |
| 2048 | 0.730 (0.017) | 0.522 (0.029) | 0.567 (0.024) | 0.500 (0.038) | 0.270 (0.026) | 0.311 (0.026) |
| RAW | 0.534 (0.044) | 0.362 (0.032) | 0.430 (0.035) | 0.660 (0.027) | 0.346 (0.028) | 0.535 (0.020) |

Performance of local averaging of microarray data locally averaged in PPI network

## Example: Local averaging

## 8. Performance using support vector regression:

In Euclidean space: replace noisy gene expression function by a regularized one based on support vector regression (here $x = g$ represents a variable gene in base space gene network)

$$f(x) \rightarrow f(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i) + b$$

for selected points (centers) $\{x_i\}_i$, where $K(x, y)$ is a kernel which gives a metric between genes $x$ and $y$.

## Example: Local averaging

Example: in our gene space kernel

$$K(x, y) \;=\; \text{graph diffusion kernel}$$

(heat kernel on gene network graph).

## Example: Local averaging

Regularized function $f(x) = f(g)$ is optimizer of objective fn.

$$f = \underset{f(g)}{\text{argmin}} \sum_j L(f(g_j), z_j) + \lambda \|f\|_K^2,$$

where $g_j = j^{th}$ gene

$f(g) = f(x) =$ fn. on genes (regularized expressions),

$z_j =$ original measured expression on gene $j$

$L(f(g_j), z_j) = (|f(g_j) - z_j| - \epsilon)^+$

$=$ loss function (difference between measured and regularized gene expression)

$\lambda =$ regularization parameter

## Example: Local averaging

$\|f\|_K =$ norm of $f$ with respect to kernel $K$

Regularization done within clusters of genes, grouped by similar expressions in the training set

# Example: Local averaging

| No. Clusters | Wang | | | van de Vijver | | |
|---|---|---|---|---|---|---|
| | AUROC | AUPRC | ACC90 | AUROC | AUPRC | ACC90 |
| 1 | 0.618 (0.013) | 0.405 (0.014) | 0.456 (0.024) | | | |
| 64 | 0.672 (0.018) | 0.503 (0.025) | 0.476 (0.033) | 0.706 (0.017) | 0.441 (0.025) | 0.468 (0.032) |
| 128 | 0.698 (0.018) | 0.519 (0.026) | 0.52 (0.032) | 0.738 (0.017) | 0.456 (0.024) | 0.527 (0.035) |
| 256 | 0.716 (0.017) | 0.526 (0.024) | 0.565 (0.030) | 0.741 (0.017) | 0.465 (0.025) | 0.536 (0.033) |
| 512 | 0.71 (0.016) | 0.515 (0.023) | 0.567 (0.027) | 0.746 (0.015) | 0.478 (0.026) | 0.552 (0.030) |
| 1024 | 0.701 (0.015) | 0.494 (0.022) | 0.558 (0.027) | 0.74 (0.013) | 0.48 (0.025) | 0.536 (0.022) |
| 2048 | 0.676 (0.019) | 0.47 (0.026) | 0.521 (0.031) | 0.718 (0.015) | 0.441 (0.026) | 0.532 (0.018) |
| RAW | 0.54 (0.040) | 0.364 (0.030) | 0.434 (0.035) | 0.661 (0.023) | 0.351 (0.026) | 0.535 (0.020) |

Support vector regression performance (expression clustering followed by regression in each cluster)

## Example: Local averaging

There are other potential gene metrics based on gene networks derived gene ontology (GO), gene copy number information (in cancer), etc.

## Example: Local averaging

Now consider a smoothing transformation $T$ on $f(g)$ to smooth out noise:

Mapping $f(g) \rightarrow T(f(g))$ gives

$$T(f(g)) = T(f_1(g)) + T(\epsilon(g)).$$

Transformation $T(f_1(g))$ will differ from the true expression $f_1(g)$, so we have introduced bias

If $k$ regularization parameter (cluster size)

## Example: Local averaging

loss of signal through bias increase:

(*) $f_1 - T(f_1)$ (increases with $k$)

However, the smoothing $T(\epsilon(g))$ of noise $\epsilon$ will quench it:
averaging over $k$ genes will reduce $\epsilon \longrightarrow \frac{1}{\sqrt{k}}\epsilon$

Gain in signal through variance decrease

(**) $\epsilon - \frac{1}{\sqrt{k}}\epsilon$ (decreases with $k$)

For some value of regularization parameter $k$ the bias loss in (*) is balanced by the variance gain in (**).

## Example: Local averaging

This is the usual bias-variance dilemma - when do we quench so much noise (**) that the increase (*) in bias is overcome?

Principle: local averaging eliminates noise.

## 9. Use of the Copula

Above in a feature space $F$, we want to discriminate between feature vectors **x** which represent metastatic vs. non-metastatic cancers.

## The Gaussian Copula

Use of SVM involves finding a *discriminant function*

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

which maps feature vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ into

positive/negative number depending on whether cancer is metastatic/nonmetastatic.

Denote such an $f(\mathbf{x})$ to be a *discriminant.*

## The Gaussian Copula

Consider microarray $\mathbf{x} = \mathbf{X}$ to be a random vector, randomly selected from metastatic patients.

Let $\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}$ be a random variable representing a random metastatic microarray,

Let $\mathbf{X}' = \begin{bmatrix} X_1' \\ X_2' \\ \vdots \\ X_n' \end{bmatrix}$

## The Gaussian Copula

be a random variable representing a random non-metastatic array.

For each $X_i$, let $F_i(x) = P(X_i \leq x)$ be the *empirical* distribution function of $X_i$, obtained from the *non-metastatic* data set.

Let $F_i'(x)$ [NOT a derivative] represent the same for the vector **X**′. Define the vector function

## The Gaussian Copula

$$\mathbf{F}(\mathbf{X}) = \begin{bmatrix} F_1(X_1) \\ F_2(X_2) \\ \vdots \\ F_n(X_n) \end{bmatrix}$$

which composes each random variable $X_i$ with its own disribution function $F_i$.

Then it is known that the feature map **F** maps **X** into an array of standard uniform random variables.

## The Gaussian Copula

Similarly define $\mathbf{F}'$ so that $\mathbf{F}'(\mathbf{X}')$ has uniform components if $\mathbf{X}'$ represents a random metastatic microarray.

The maps $\mathbf{F}, \mathbf{F}'$ are known as *uniform copulas.*

## The Gaussian Copula

Letting $\phi(x)$ be the cumulative distribution function of the standard Gaussian, for any vector $\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$, define

$$\Psi(\mathbf{X}) = \phi^{-1}(\mathbf{F}(\mathbf{X})),$$

where $\phi$ acts componentwise. Similarly define

$$\Psi'(\mathbf{X}') = \phi^{-1}(\mathbf{F}'(\mathbf{X}')).$$

## The Gaussian Copula

The feature maps $\Psi, \Psi'$ therefore map each component of $\mathbf{X}, \mathbf{X}'$, respectively into standard Gaussian distributions.

The maps $\Psi, \Psi'$ are know as *normal copulas*.

These maps take arbitrarily distributed features into features all of which have standard normal distributions.

## The Gaussian Copula

These can be used to find a likelihood $L(\mathbf{x})$ and $L'(\mathbf{x})$ representing the respective probabilities that a given microarray **x** comes from

(a) the non-metastatic population of microarrays, or
(b) the metastatic population of microarrays

**Notes:**

**1.** Conventional linear discriminant analysis (LDA) or quadratic discriminant analysis (QDA) for binary classification often assume features have approximately multivariate normal distributions.

## The Gaussian Copula

**2.** Hence such methods are sub-optimal when the data in fact are not normal and may be heavy-tailed.

**3.** This method (after application of the feature maps $\Psi, \Psi'$) guarantees that at least the marginals of the feature vectors (e.g. $\Psi(\mathbf{x})$) are in fact normal (e.g. if **x** is from the non-metastatic population).

## The Gaussian Copula

The form of the discriminant is

$$f(\mathbf{x}) = \frac{L'(\mathbf{x})}{L(\mathbf{x})} =$$

$$J_1(x_1)\ldots J_d(x_d)e^{\mathbf{z}^T(\Sigma^{-1}-1)\mathbf{z}-\mathbf{z}'^T(\Sigma'^{-1}-1)\mathbf{z}'/2}\left(\frac{\det\Sigma}{\det\Sigma'}\right)^{1/2}$$

where

$$J_i(x) = \frac{\text{empirical density of } X_i \text{ at location } x}{\text{empirical density of } X_i' \text{ at location } x}$$

$=$ Jacobian of $i^{th}$ coordiante of map from $\Psi$ to $\Psi'$ feature space.

## The Gaussian Copula

$$\mathbf{z} = \Psi(\mathbf{x}); \;\; \mathbf{z}' = \Psi'(\mathbf{x}'),$$

and

$$\Sigma = \text{empirical covariance matrix of } \mathbf{X}$$

$$\Sigma' = \text{empirical covariance matrix of } \mathbf{X}'.$$

**Results:**

# The Gaussian Copula

## Accuracies: Normally distributed data:

| Case | Size\ Method | 2-D | | | | | 10-D | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | E | J | Q | S | T | E | J | Q | S | T |
| I | 100 | 71 | 71 | 75 | 72 | 75 | 69 | 69 | 72 | 65 | 82 |
| | 1000 | 75 | 75 | 75 | 75 | 75 | 79 | 79 | 81 | 79 | 82 |
| | 10000 | 76 | 76 | 76 | 76 | 76 | 80 | 82 | 82 | 81 | 82 |
| II | 100 | 76 | 76 | 77 | 75 | 79 | 80 | 82 | 83 | 81 | 93 |
| | 1000 | 79 | 79 | 79 | 79 | 79 | 91 | 90 | 92 | 91 | 92 |
| | 10000 | 79 | 80 | 79 | 80 | 79 | 92 | 93 | 93 | 92 | 93 |
| III | 100 | 83 | 83 | 85 | 84 | 85 | 91 | 91 | 94 | 90 | 98 |
| | 1000 | 85 | 85 | 85 | 85 | 85 | 96 | 96 | 97 | 95 | 97 |
| | 10000 | 85 | 85 | 85 | 85 | 85 | 97 | 98 | 98 | 98 | 98 |
| IV | 100 | 89 | 89 | 90 | 89 | 91 | 97 | 96 | 97 | 96 | 99 |
| | 1000 | 90 | 90 | 90 | 90 | 90 | 98 | 98 | 99 | 98 | 99 |
| | 10000 | 90 | 90 | 90 | 90 | 90 | 99 | 99 | 100 | 100 | 100 |

E = empirical normalization method
J = empirical normalization using underlying Jacobian
Q = quadratic discriminant (assuming normality)
S = SVM
T = Optimal Bayes method based on knowledge of underlying densities

## The Gaussian Copula

Accuracies: (Heavy-tailed) $t_1$-distributed data

| Case | Size\ Method | 2-D | | | | | 10-D | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | E | J | Q | S | T | E | J | Q | S | T |
| I | 100 | 56 | 59 | 52 | 53 | 64 | 58 | 57 | 52 | 60 | 71.0 |
| | 1000 | 61 | 62 | 50 | 51 | 64 | 65 | 60 | 51 | 59 | 71 |
| | 10000 | 63 | 63 | 50 | 48 | 64 | 66 | 59 | 50 | 66 | 71 |
| II | 100 | 74 | 77 | 54 | 66 | 80 | 81 | 80 | 68 | 83 | 89 |
| | 1000 | 77 | 79 | 51 | 67 | 80 | 85 | 84 | 51 | 87 | 89 |
| | 10000 | 79 | 79 | 50 | 59 | 80 | 86 | 84 | 50 | 89 | 89 |
| III | 100 | 80 | 82 | 58 | 79 | 86 | 85 | 85 | 81 | 90 | 93 |
| | 1000 | 84 | 84 | 52 | 77 | 85 | 89 | 88 | 55 | 91 | 93 |
| | 10000 | 85 | 85 | 49 | 84 | 86 | 90 | 89 | 49 | 92 | 93 |
| IV | 100 | 84 | 86 | 64 | 83 | 89 | 88 | 87 | 88 | 91 | 94 |
| | 1000 | 87 | 88 | 52 | 82 | 89 | 91 | 91 | 62 | 94 | 94 |
| | 10000 | 87 | 87 | 51 | 87 | 89 | 92 | 90 | 50 | 93 | 95 |

E = empirical normalization method
J = empirical normalization using underlying Jacobian
Q = quadratic discriminant (assuming normality)
S = SVM
T = Optimal Bayes method based on knowledge of underlying densities

## The Gaussian Copula

Accuracies: Benchmark Breast and Prostate cancer data (Prostate is cancer/no cancer only); CRF denotes Convergent Random Forest algorithm (Bienkowska)

| | QDA (Q) | CRF | SVM (S) | Emp (E) |
|---|---|---|---|---|
| **Prostate** | 94 | 95 | 93 | 94 |
| **Breast** | 79 | 84 | 68 | 89 |