Lecture 5: Feature Selection Filters

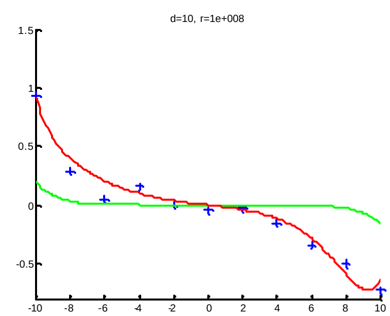
Isabelle Guyon
guyoni@inf.ethz.ch

Part I: Review of past lectures

We have learned about…

- How to jugulate overfitting by favoring simpler solutions
- The need to reduce dimensionality/select features when n>>m because even simple models can overfit (curse of dimensionality)
- Dot products are important in machine learning, they are the basis of several:
  - Machine architectures (linear models, kernel methods, neural networks),
  - Learning algorithms (Hebb's rule, gradient descent),
  - Preprocessing (filter banks and convolutional filters)

Polynomial Regression

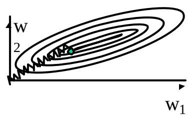


Curse of Dimensionality

- n>m, the linear set of equations

$$\underset{(m,n)}{X}\ \underset{(n,1)}{w^T} = \underset{(m,1)}{y}$$

  has an infinite number of solutions.
- The pseudo-inverse solution is the least-square solution of minimum norm $||\mathbf{w}||$.
- Better predictors can sometimes be achieved with larger penalties on $||\mathbf{w}||$.



All Purpose Dot Products

- We all know the "regular" dot product (or scalar product) in a Euclidean space $\mathbf{x} \bullet \mathbf{x'} = \Sigma_j x_j x'_j$
- More generally, a dot product on a vector space V is a **positive symmetric bilinear form**:

  $<.,.>: V \times V \rightarrow R$

  $(\mathbf{x}, \mathbf{x'}) \rightarrow <\mathbf{x}, \mathbf{x'}>$

  Symmetry: $<\mathbf{x}, \mathbf{x'}> = <\mathbf{x'}, \mathbf{x}>$

  Bilinearity: $<\lambda\mathbf{x}, \mathbf{x'}> = \lambda <\mathbf{x}, \mathbf{x'}>$

  $<\mathbf{x}, \lambda\mathbf{x'}> = \lambda <\mathbf{x}, \mathbf{x'}>$

  Positivity: $<\mathbf{x}, \mathbf{x}> \geq 0$ with equality only for $\mathbf{x}=0$

Examples of Dot Products

- $k(\mathbf{x}, \mathbf{x'}) = \mathbf{x} \bullet \mathbf{x'}$ Linear kernel
- $k(\mathbf{x}, \mathbf{x'}) = \exp(-\gamma||\mathbf{x}-\mathbf{x'}||^2)$ Gaussian kernel
- $k(\mathbf{x}, \mathbf{x'}) = 1/||\mathbf{x}-\mathbf{x'}||$ Potential function
- $k(\mathbf{x}, \mathbf{x'}) = (\mathbf{x} \bullet \mathbf{x'})^q$ Polynomial kernel

$$\underbrace{([x_1,x_2]\bullet[x'_1,x'_2])^2}_{k(\mathbf{x},\mathbf{x'})} = \underbrace{[x_1^2,x_2^2,\sqrt{2}x_1x_2]}_{\phi(\mathbf{x})} \bullet \underbrace{[x'^2_1,x'^2_2,\sqrt{2}x'_1x'_2]}_{\phi(\mathbf{x'})}$$

A kernel is a dot product in *some* feature space:

$$k(\mathbf{x}, \mathbf{x'}) = \phi(\mathbf{x}) \bullet \phi(\mathbf{x'})$$

Fancier Dot Products

- $\mathbf{x} \bullet \mathbf{x'} = \Sigma_{j=1}^{n} x_j x'_j$
- $(\mathbf{x} \bullet \mathbf{x'})^q = \Sigma_{j=1}^{N} \phi_j(\mathbf{x})\ \phi_j(\mathbf{x'})$
- $\exp(-\gamma||\mathbf{x}-\mathbf{x'}||^2) = \Sigma_{j=1}^{\infty} \phi_j(\mathbf{x})\ \phi_j(\mathbf{x'})$
- $k(x, x') = \int \phi(\mathbf{x}, t)\ \phi(\mathbf{x'}, t)\ dt$

Architectures

- Linear model: $f(\mathbf{x}) = \mathbf{w} \bullet f(\mathbf{x})$ (or $\mathbf{w} \bullet \mathbf{x}$)
- Kernel method:

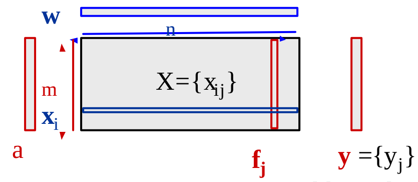
$$f(\mathbf{x}) = \Sigma_i \alpha_i k(\mathbf{x}_i, \mathbf{x})$$

$$k(\mathbf{x}_i, \mathbf{x}) = f(\mathbf{x}_i) \bullet f(\mathbf{x})$$

(kernel "trick" $\mathbf{w} = \Sigma_i \alpha_i f(\mathbf{x}_i)$)

- Neural nets: network of linear threshold units.

Learning Algorithms



- $w_j \leftarrow w_j + y_i x_{ij}$ Hebb's rule

  $w_j = \Sigma_i y_i x_{ij} = \mathbf{y} \bullet \mathbf{f}_j$ if $\mathbf{f}_j \leftarrow (\mathbf{f}_j - \mu_j)/\sigma_j$ Pearson correlation

- $w_j = \Sigma_i \alpha_i \phi_j(\mathbf{x}_i) = a \bullet f_j$ Other rules

Feature Construction

Example of one dimensional signal x(t) or x_j:

- Convolution:

  $\phi(s) = \int x(t) K(s-t) dt$

  $\phi_k = \Sigma_{j=0}^{p-1} x_j \; K_{k-j}$

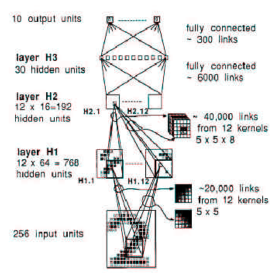
- Fourier and other filter bank transforms:

  $\phi(s) = \int x(t) K(s, t) dt$

  e.g. $K(s, t) = \exp(-ist)$

  Orthogonality: $\int K(s, t) K(s', t) dt = \delta_{ss'}$

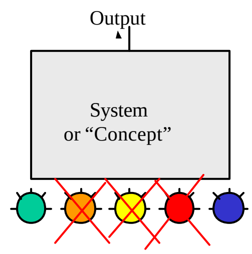
Convolutional Neural Nets



http://yann.lecun.com/exdb/lenet/

Part II:
Filters for feature selection

Relevance to the concept



Objectives

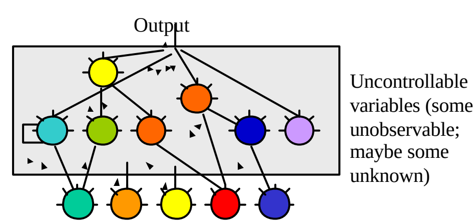
1- Eliminate distracters (irrelevant variables)

2 - Rank (combinations of) relevant variables

A big search problem

- Definition of distracter: if tweaked, no change in input/output relationship for **any position of all other knobs.**
- "Exhaustive search": Check all knob positions (see: factorial design). One knob at a time does not work if one variable alone does not control the output.
- Experimental design: In the continuous case we need efficient experimental design or "query" strategies.
- Sub-optimal/bogus designs: false positive relevance (e.g. confounded factors) and false negative relevance (e.g. joint effect unexplored.)

Reverse Engineering

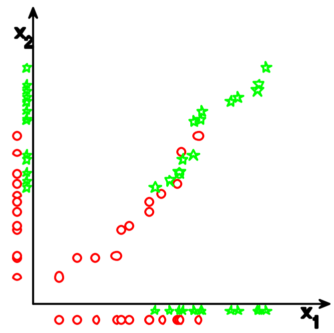


Making Predictions

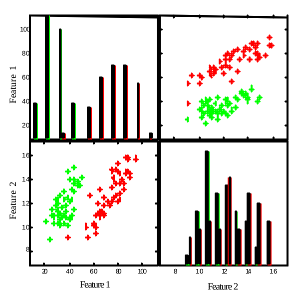
- Goal: find the smallest subset of variables, which provide at least as good predictions as all the variable.
- No uniqueness of the solution.

- ***Relevance vs. usefulness:***
  - Relevance does not imply usefulness.
  - Usefulness does not imply relevance.

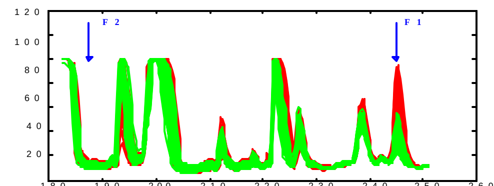
Redundant: Relevant but “Useless”



Real data: Mass spectrometry

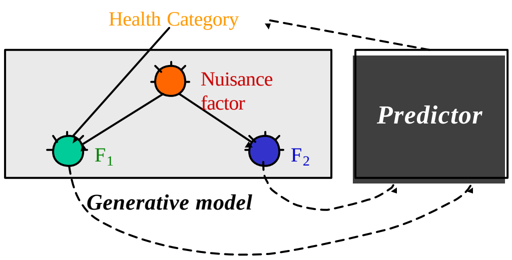


Explanation:



F1: The peak of interest
F2: The best local estimate of the baseline.

Useful but “Irrelevant”



Correlation and Causality

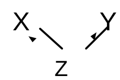
- **Correlation does not mean causality**.
- Direction:

$$X \rightarrow Y \text{ or } X \leftarrow Y$$

$$P(X, Y) = P(Y|X)P(X) = P(X|Y)P(Y)$$

Predictive model Generative model

- Hidden common cause:



~~Inference of Causality~~

- Need controllable variables and experimental design.
- Machine learning case:
  - “Canned data”, can only observe some variables, i.e. no controllable variables, some may be unobservable.
  - Finite sample size: no access to the “real” data distribution.

Defining “Relevance”

Variable Dependence

- Independence:

$$P(X, Y) = P(X)\,P(Y)$$

- Measure of dependence:

$$MI(X, Y) = \int P(X,Y) \log \frac{P(X,Y)}{P(X)P(Y)}\, dX\, dY$$

$$= KL\big(\, P(X,Y)\, ||\, P(X)P(Y)\, \big)$$

More than 2 variables...

- Surely irrelevant feature:

  $P(X_i, Y \,|\mathbf{X}^{-i}) = P(X_i \,|\mathbf{X}^{-i})P(Y \,|\mathbf{X}^{-i})$

  for all assignment of values to $\mathbf{X}^{-i}$
- Define conditional mutual information.
- Average over assignment of values to $\mathbf{X}^{-i}$:

$$EMI(X_i, Y) = \int_{\mathbf{X}^{-i}} P(\mathbf{X}^{-i})\, MI(X_i, Y \,|\mathbf{X}^{-i})\, d\mathbf{X}^{-i}$$

Elimination of "Distracters"

- Rank features X_i according to an empirical estimate of $EMI(X_i, Y)$.
- Eliminate all the features such that:

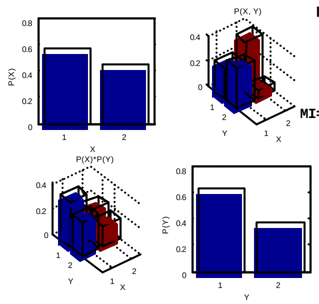
  $EMI(X_i, Y) \leq \varepsilon$

  for a chosen $\varepsilon \geq 0$.
- Next lecture: choose ε to have sufficient confidence that X_i is a distracter.

Are we done?

- $MI(X_i, Y)$ difficult to estimate:
  - we need to "regularize", relate on distribution first moments or smooth the distribution.
- $EMI(X_i, Y)$ even worse:
  - Super overfitting problem.
    - We may not be able to estimate the joint distribution of more than 3 variables.
    - We should anyways not consider all possible subsets.
- MI is NOT the best criterion:
  - If the goal is not density estimation but classification or regression: too many features will be retained.

MI Estimation




```
pxy=[4/16, 6/16; 5/16, 1/16]
px=sum(pxy)
py=sum(pxy, 2)
pxpy=(py*px)
MI=sum(sum(pxy.*log2(pxy./pxpy)))
pxy =   0.2500   0.3750
        0.3125   0.0625
px =    0.5625   0.4375
py =    0.6250
        0.3750
pxpy =  0.3516   0.2734
        0.2109   0.1641
MI =    0.1381
```

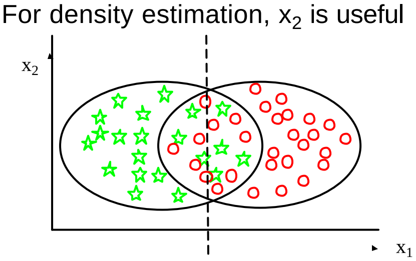
Possible normalization by (H(X)+H(Y))/2

Non-Binary Case

- Create histograms, but the number of counts in each bin may be too low to get accurate results: k variables examined together, v values per variable, v^k bins! … and only m examples to fill them.
- Estimate the densities with non-parametric methods (e.g. Parzen windows).
- Make simplifying assumptions about the distribution (e.g. Normal).

What Objective?

- For classification, x_2 is not useful
- For density estimation, x_2 is useful



No, we are not done…

We will:

1) Definine ranking criteria using second order moments (variance).
2) Search feature space with greedy strategies: creating nested subsets of features by forward selection.

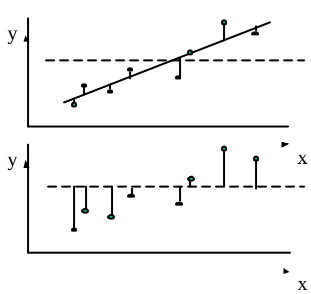
Single Feature Relevance: Simple Criteria

Pearson Correlation

- $R = \dfrac{\sigma_{xy}^2}{\sigma_x \, \sigma_y}$
- $R = (1/m) \dfrac{\Sigma_i (x_i-\mu_x)(y_i-\mu_y)}{\sigma_x \quad \sigma_y}$
- $R \sim \mathbf{x} \bullet \mathbf{y}$ after "standardization" $\mathbf{x} \leftarrow (\mathbf{x}-\mu_x)/\sigma_x$

Correlation and Linearity

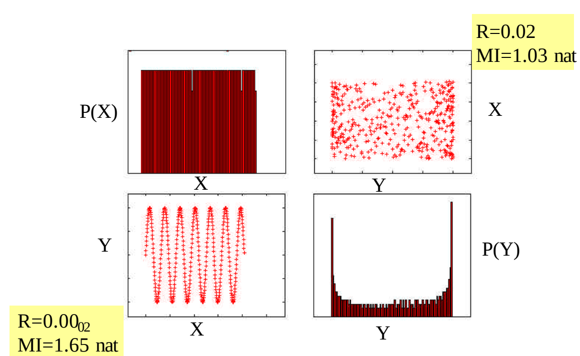
For the least-square linear regression, $R^2=1- \sigma_r^2 / \sigma_y^2$



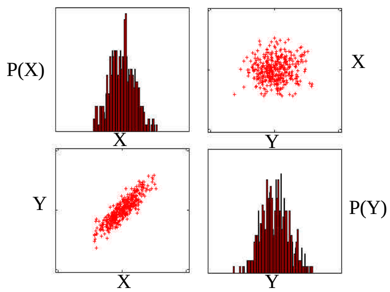
Residual variance: σ_r^2

Total variance: σ_y^2

Correlation and MI

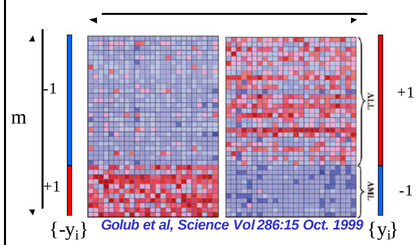


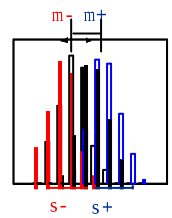
Gaussian Distribution



MI(X, Y) = -(1/2) log(1-R²)

S2N





$$\text{S2N} = \frac{|\mu_+ - \mu_-|}{\sigma_+ + \sigma_-}$$

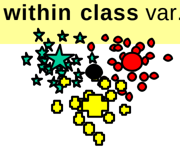
S2N $\cong$ R ~ **x • y**

after "standardization" **x** $\leftarrow$(**x**-μ_x)/σ_x

Fisher Score

- Multi-class classification of continuous variables (or regression of categorical variables):

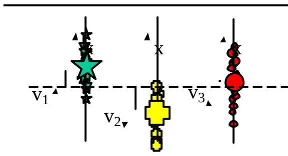
$$F = \frac{\textbf{between class variance}}{\text{pooled } \textbf{within class} \text{ var.}}$$



- Two-class case:

$$F = \frac{(\mu_+ - \mu_-)^2}{(m_+/m)\sigma_+^2 + (m_-/m)\sigma_-^2} \Leftrightarrow \text{S2N} = \frac{|\mu_+ - \mu_-|}{\sigma_+ + \sigma_-}$$

Fisher Score and ANOVA



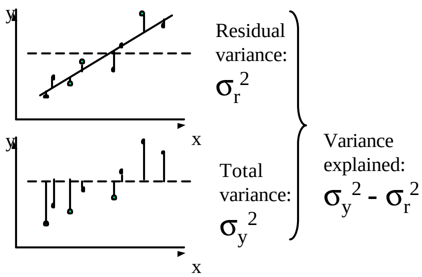
$$F = \frac{\text{variance explained}}{\text{residual variance}}$$

ANOVA model: $x_{ij} = \mu + v_j + \varepsilon_{ij}$

Reminder: model of the effect on observations x of a **systematic** factor of variability $v \in \{v_1, v_2, \ldots v_j, \ldots\}$ and **intrinsic** variability ε (random error, normally distributed).

Fisher Score and Regression

$$F = \frac{\text{variance explained}}{\text{residual variance}} = \frac{\sigma_y^2 - \sigma_r^2}{\sigma_r^2} = \frac{1}{1-R^2} - 1$$



Eliminating Redundancy: Conditional Relevance

Forward Selection with MI

Fleuret, 2004. *Practical only for binary features.*

- Select a first feature $X_{?(1)}$ with maximum MI with the target.
- For each remaining feature X_i and each previously selected feature $X_{?(j)}$, compute the conditional mutual information:

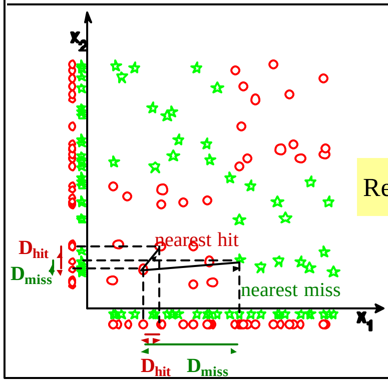
  $CMI(X_i, Y | X_{?(j)}) = \Sigma_{X?(j)} P(X_{?(j)}) \, MI(X_i, Y | X_{?(j)})$
- Select the feature with maximum CMI.

Forward Selection with GS

Stoppiglia, 2002. Gram-Schmidt orthogonalization.

- Select a first feature $X_{?(1)}$ with maximum cosine with the target $\cos(\mathbf{x}_i, \mathbf{y}) = \mathbf{x}.\mathbf{y}/||\mathbf{x}|| \, ||\mathbf{y}||$
- For each remaining feature X_i
  - Project X_i and the target Y on the null space of the features already selected
  - Compute the cosine of X_i with the target in the projection
- Select the feature $X_{?(k)}$ with maximum cosine with the target in the projection.

Relief



$Relief = \langle D_{miss} / D_{hit} \rangle$

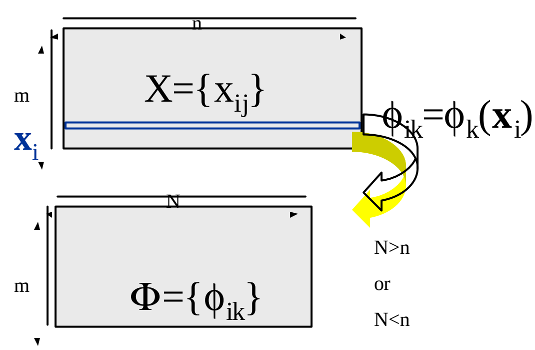
Other criteria: see chapter 3!

Method		X			Y			Comments
Name	Formula	B	M	C	B	M	C	
Bayesian accuracy	Eq. 3.1	+	s		+	s		Theoretically the golden standard, rescaled Bayesian relevance Eq. 3.2.
Balanced accuracy	Eq. 3.4	+	s		+	s		Average of sensitivity and specificity; used for unbalanced dataset, same as AUC for binary targets.
Bi-normal separation	Eq. 3.5	+	s		+	s		Used in information retrieval.
F-measure	Eq. 3.7	+	s		+	s		Harmonic of recall and precision, popular in information retrieval.
Odds ratio	Eq. 3.6	+	s		+	s		Popular in information retrieval.
Means separation	Eq. 3.10	+	i	+	+			Based on two class means, related to Fisher's criterion.
T-statistics	Eq. 3.11	+	i	+	+			Based also on the means separation.
Pearson correlation	Eq. 3.9	+	i	+	+	i	+	Linear correlation, significance test Eq. 3.12, or a permutation test.
Group correlation	Eq. 3.13	+	i	+	+	i	+	Pearson's coefficient for subset of features.
χ^2	Eq. 3.8	+	s		+	s		Results depend on the number of samples m.
Relief	Eq. 3.15	+	s	+	+	s	+	Family of methods, the formula is for a simplified version ReliefX, captures local correlations and feature interactions.
Separability Split Value	Eq. 3.41	+	s	+	+	s		Decision tree index.
Kolmogorov distance	Eq. 3.16	+	s	+	+	s	+	Difference between joint and product probabilities.
Bayesian measure	Eq. 3.16	+	s	+	+	s	+	Same as Vajda entropy Eq. 3.23 and Gini Eq. 3.39.
Kullback-Leibler divergence	Eq. 3.20	+	s	+	+	s	+	Equivalent to mutual information.
Jeffreys-Matusita distance	Eq. 3.22	+	s	+	+	s	+	Rarely used but worth trying.
Value Difference Metric	Eq. 3.22	+	s		+	s		Used for symbolic data in similarity-based methods, and symbolic feature-feature correlations.
Mutual Information	Eq. 3.29	+	s	+	+	s	+	Equivalent to information gain Eq. 3.30.
Information Gain Ratio	Eq. 3.32	+	s	+	+	s	+	Information gain divided by feature entropy, stable evaluation.
Symmetrical Uncertainty	Eq. 3.35	+	s	+	+	s	+	Low bias for multivalued features.
J-measure	Eq. 3.36	+	s	+	+	s	+	Measures information provided by a logical rule.
Weight of evidence	Eq. 3.37	+	s	+	+	s	+	So far rarely used.
MDL	Eq. 3.38	+	s		+	s		Low bias for multivalued features.

Homework 5

- Complete homework 4 and train a classifier using the new feature representation you chose or implemented.
- Make a submission to the website of the challenge to get your test set score: http://www.nipsfsc.ecs.soton.ac.uk/
- Email the result zip file of the results to guyoni@inf.ethz.ch with subject "Homework5" no later than:
  Tuesday November 29th.

Feature Transforms Implemented



Match Filters

Implementation:

One "match_filter" object that takes a " filter_bank" object as an argument.

Examples:

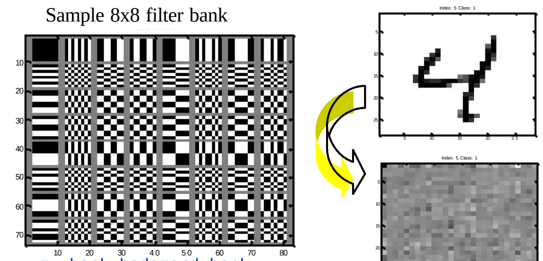
- hadamard_bank: Hadamard transform, similar to the Fourier transform, but has discrete valued orthogonal basis functions.

- pca_bank: uses the first " f_max" principal components as a filter bank.

- kmeans_bank: uses " f_max" cluster centers as a filter bank.

Hadamard Transform

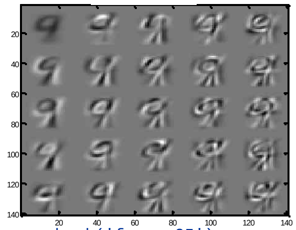
Sample 8x8 filter bank



```
my_bank=hadamard_bank;
show(my_bank);
my_prepro=match_filter(my_bank);
[d, my_prepro]=train(my_prepro, D.train);
browse_digit(d.X, d.Y);
```

Principal Components

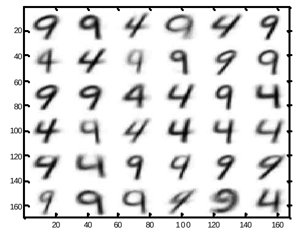
Filter bank obtained for 25 components



```
my_bank=pca_bank('f_max=25');
my_prepro=match_filter(my_bank);
[d, my_prepro]=train(my_prepro, D.train);
show(my_prepro);
```

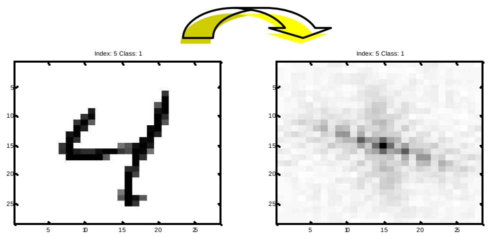
Kmeans Clustering

Filter bank obtained for 36 clusters



```
my_bank=kmeans_bank('f_max=36');
my_prepro=match_filter(my_bank);
[d, my_prepro]=train(my_prepro, D.train);
show(my_prepro);
```

Fourier Transform




```
my_prepro=fourier;
[d, my_prepro]=train(my_prepro, D.train);
browse_digit(d.X, d.Y);
```

Convolutions

Implementation:

One “convolve” object that takes a “xxx_ker” object as arg.

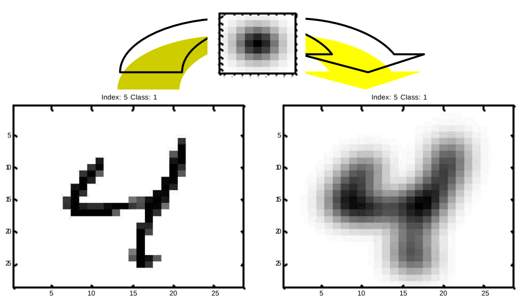
Examples:

- gauss_ker: Gaussian kernel. Four parameters: dim1, dim2 (kernel size) and sigma1, sigma2 (Gaussian width). The sigmas are scaled automatically to 0.2*dim if only the dimensions are given. It is better to chosen odd numbers for the kernel dimension.

- exp_ker: Exponential kernel. Same parameters.

- `chain({convolve(gauss_ker({‘dim1=5’, ‘dim2=1’})), convolve(gauss_ker({‘dim1=1’, ‘dim2=5’}))})` equivalent but faster than `convolve(gauss_ker({‘dim1=5’, ‘dim2=5’}))`

Convolution: smoothing




```
my_ker=gauss_ker({'dim1=9','dim2=9','sigma1=1.8','sigma2=1.8'});
d=train(convolve(my_ker), D.train); browse_digit(d.X, d.Y);
```

Best so far...

pixelGisette_exp_conv_p4_s0.1
test_BER=0.91%

Tips to outperfom baselineGisette

baselineGisette (testBER=1.8%, feat=20%)

```
my_classif=svc({'coef0=1', 'degree=3',
  'gamma=0', 'shrinkage=1'});
my_model=chain({normalize, s2n('f_max=1000'),
  my_classif});

D.alltrain=data([D.train.X;D.valid.X],
  [D.train.Y;D.valid.Y]);
cv_model=cv(my_model, {'folds=5',
  'store_all=0'});
Result=train(cv_model, D.alltrain);
OutX=[]; OutY=[]; for k=1:5, OutX=[OutX;
  Result.child{k}.X]; OutY=[OutY;
  Result.child{k}.Y]; end
CV_BER=balanced_errate(OutX, OutY);
```

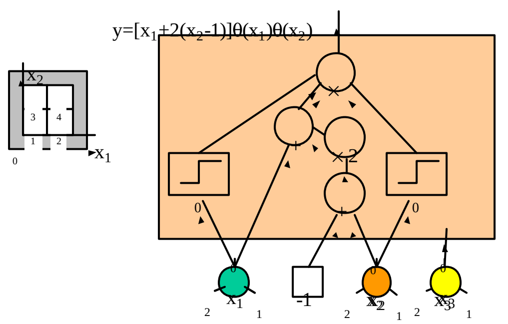
Keep good records!

- Keep your latest and greatest model and results (the zip file).
- Document what you did.

Class requirements:

- One complete entry (5 datasets) on the challenge website.
- A poster explaining what you did.

Tiny example



Theory and practice