We have learned about...

- How to mitigate 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
  \[
  X \mathbf{w}^\top = \mathbf{y} 
  \]
  \((m,0) \times (n,1) = (m,1)\)

  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 \(\mathbf{x} \cdot \mathbf{x}' = \Sigma_j x_j x'_j\)
- More generally, a dot product on a vector space \(V\) is a **positive symmetric bilinear form**:
  \[
  \langle \cdot, \cdot \rangle : V \times V \rightarrow \mathbb{R} \quad (\mathbf{x}, \mathbf{x}') \rightarrow \langle \mathbf{x}, \mathbf{x}' \rangle 
  \]

  Symmetry: \(\langle \mathbf{x}, \mathbf{x}' \rangle = \langle \mathbf{x}', \mathbf{x} \rangle\)
- Bilinearity: \(\langle \lambda \mathbf{x}, \mathbf{x}' \rangle = \lambda \langle \mathbf{x}, \mathbf{x}' \rangle\)
  \(\langle \mathbf{x}, \lambda \mathbf{x}' \rangle = \lambda \langle \mathbf{x}, \mathbf{x}' \rangle\)
- Positivity: \(\langle \mathbf{x}, \mathbf{x} \rangle \geq 0\) with equality only for \(\mathbf{x} = 0\)

**Examples of Dot Products**

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

A kernel is a dot product in some feature space: \(k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}')\)

**Fancier Dot Products**

- \(\mathbf{x} \cdot \mathbf{x}' = \Sigma_j x_j x'_j\)
- \((\mathbf{x} \cdot \mathbf{x}')^q = \Sigma_j^N \phi_j(\mathbf{x}) \phi_j(\mathbf{x}')\)

\[
\exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2) = \Sigma_{j=1}^N \phi_j(\mathbf{x}) \phi_j(\mathbf{x}')
\]

- \(k(\mathbf{x}, \mathbf{x}') = \int \phi(\mathbf{x}, t) \phi(\mathbf{x}', t) \, dt\)
Architectures

• Linear model: \( f(x) = w \cdot \phi(x) \) (or \( w \cdot x \))

• Kernel method:
  \[
  f(x) = \sum_i \alpha_i k(x_i, x)
  \]
  \[k(x_i, x) = \phi(x_i) \cdot \phi(x)\]
  (kernel “trick” \( w = \sum_i \alpha_i \phi(x_i) \))

• Neural nets: network of linear threshold units.

Learning Algorithms

- Hebb’s rule
  \[ w_j \leftarrow w_j + y_i x_{ij} \]

- Pearson correlation
  \[ w_j = \sum_i y_i x_{ij} = y \cdot f_j \]

- Other rules
  \[ w_j = \sum_i \alpha_i \phi(x_i) = \alpha \cdot f_j \]

Feature Construction

Example of one dimensional signal \( x(t) \) or \( x_i \):

• Convolution:
  \[ \phi(s) = \int x(t) K(s-t) \, dt \]
  \[ \phi_k = \sum_{j=0}^{p-1} x_j K_{k-j} \]

• 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

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

Uncontrollable variables (some unobservable; maybe some unknown)
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:**
  \[ X \quad Y \quad Z \]

### 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:**
  \[
  \text{MI}(X, Y) = \int P(X, Y) \log \frac{P(X, Y)}{P(X)P(Y)} \, dX \, dY
  = \text{KL}(P(X,Y) \parallel P(X)P(Y))
  \]

### More than 2 variables...

- **Surely irrelevant feature:**
  \[
  P(X_i, Y | X^{-i}) = P(X_i | X^{-i})P(Y | X^{-i})
  \]
  for all assignment of values to \( X^{-i} \)
- Define conditional mutual information.
- Average over assignment of values to \( X^{-i} \):
  \[
  \text{EMI}(X_i, Y) = \int_{X^{-i}} P(X^{-i}) \text{MI}(X_i, Y | X^{-i}) \, dX^{-i}
  \]

### Elimination of “Distracters”

- Rank features \( X_i \) according to an empirical estimate of \( \text{EMI}(X_i, Y) \).
- Eliminate all the features such that:
  \[
  \text{EMI}(X_i, Y) \leq \epsilon
  \]
  for a chosen \( \epsilon \geq 0 \).
- Next lecture: choose \( \epsilon \) to have sufficient confidence that \( X_i \) is a distracter.

### Are we done?

- **MI(X, Y) difficult to estimate:**
  - We need to “regularize”, relate on distribution first moments or smooth the distribution.
- **EMI(X, 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**

\[ \text{pxy} = \begin{bmatrix} 0.2500 & 0.3750 \\ 0.3125 & 0.0625 \end{bmatrix} \]

\[ \text{px} = \begin{bmatrix} 0.5625 & 0.4375 \end{bmatrix} \]

\[ \text{py} = \begin{bmatrix} 0.6250 \\ 0.3750 \end{bmatrix} \]

\[ \text{pxpy} = \begin{bmatrix} 0.3516 & 0.2734 \\ 0.2109 & 0.1641 \end{bmatrix} \]

\[ \text{MI} = 0.1381 \]

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) Define 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**
  
  \[ R = \frac{\sigma_{xy}^2}{\sigma_x \sigma_y} \]
  
  \[ R = \frac{1}{m} \sum \left( \frac{x_i - \mu_x}{\sigma_x} \right) \left( \frac{y_i - \mu_y}{\sigma_y} \right) \]
  
  \[ R \sim x \cdot y \text{ after 'standardization' } x \leftarrow (x - \mu_x) / \sigma_x \]

---

**Correlation and Linearity**

For the least-square linear regression, \[ R^2 = 1 - \frac{\sigma_r^2}{\sigma_y^2} \]

**Residual variance:** \( \sigma_r^2 \)

**Total variance:** \( \sigma_y^2 \)

---

**Correlation and MI**

- \( R = 0.02 \)
  - \( \text{MI} = 1.03 \text{ nat} \)

- \( R = 0.00 \text{nat} \)
  - \( \text{MI} = 1.65 \text{ nat} \)
### Gaussian Distribution

\[
P(X) \quad P(Y) \\
X \quad Y \\
\text{MI}(X,Y) = -(1/2) \log(1-R^2)
\]

### S2N

\[
S2N = \frac{\|\mu_x - \mu_y\|}{\sigma_x + \sigma_y} \\
S2N \quad R = x \cdot y \\
\text{after "standardization" } x \leftarrow (x-\mu)/\sigma
\]

### Fisher Score

- Multi-class classification of continuous variables (or regression of categorical variables):
  \[
  F = \frac{\text{between class variance}}{\text{pooled within class var.}}
  \]

- Two-class case:
  \[
  F = \frac{(\mu_1 - \mu_2)^2}{(m_1/m)_\sigma^2 + (m_2/m)_\sigma^2} \Leftrightarrow S2N = \frac{|\mu_1 - \mu_2|}{\sigma + \sigma}
  \]

### Fisher Score and ANOVA

ANOVA model: \[x_{ij} = \mu + v_j + \epsilon_{ij}\]

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 \(\epsilon\) (random error, normally distributed).
**Fisher Score and Regression**

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

- Residual variance: \( \sigma_r^2 \)
- Total variance: \( \sigma_y^2 \)
- Variance explained: \( \sigma_y^2 - \sigma_r^2 \)

**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:

\[
\text{CMI}(X_i, Y \mid X_{(j)}) = \sum_{X_{(j)}} R(X_{(j)}) \text{MI}(X_i, Y \mid X_{(j)})
\]
- Select the feature with maximum CMI.

**Forward Selection with GS**


- Select a first feature \( X_{(1)} \) with maximum cosine with the target \( \cos(x_i, y) = x_i \cdot y / \| x_i \| \| y \| \)

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

\[ \text{Relief} = \frac{D_{\text{miss}}}{D_{\text{hit}}} \]

Other criteria: see chapter 3!

<table>
<thead>
<tr>
<th>Method</th>
<th>X</th>
<th>P</th>
<th>Formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hit measure</td>
<td></td>
<td></td>
<td>\text{Hit measure} \quad \text{Eq. 13.2}</td>
</tr>
<tr>
<td>balanced accuracy</td>
<td></td>
<td></td>
<td>\text{Balanced accuracy} \quad \text{Eq. 13.1}</td>
</tr>
<tr>
<td>Distance measure</td>
<td></td>
<td></td>
<td>\text{Distance measure} \quad \text{Eq. 13.5}</td>
</tr>
<tr>
<td>Mahalanobis distance</td>
<td></td>
<td></td>
<td>\text{Mahalanobis distance} \quad \text{Eq. 13.8}</td>
</tr>
<tr>
<td>Feature transform</td>
<td></td>
<td></td>
<td>\text{Feature transform} \quad \text{Eq. 13.11}</td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
<td>\text{Other} \quad \text{See chapter 3}</td>
</tr>
</tbody>
</table>

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/](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

\[ X = \{x_{ij}\} \]

\[ \Phi = \{\phi_{ik}\} \]
## 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.

```matlab
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);
```

## Hadamard Transform

Sample 8x8 filter bank

```matlab
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

```matlab
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

```matlab
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

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.

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

Convolution: smoothing

Best so far...

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

```
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'}))
```

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

```
my_ker=gauss_ker({'dim1=3','dim2=3','sigma1=1.8','sigma2=1.8'});
[d, my_ker]=train(convolve(my_ker), D.train);
browse_digit(d.X, d.Y);
```

```
pixelGisette_exp_conv_p4_s0.1
test_BER=0.91%
```
**Tips to outperform baselineGisette**

```matlab
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**

\[
y = x_1 + 2(x_2 - 1) \theta(x_1) \theta(x_2)
\]

**Theory and practice**