Summary
Modern Data Analysis

Barbara Hammer, CITEC Centre of Excellence, Bielefeld University
Thanks to Christina Naundorf for corrections and comments!

January 29, 2014

The lecture 'Modern data analysis' deals with a number of hot topics on how to process data, thereby addressing the questions how to represent data efficiently or how to learn a mapping on the data efficiently. Most topics are 'non-classical' in the sense that they rely on one cool idea which adds some aspect to 'classical' problems which allows a particularly fast or elegant solution, or, in the case of otherwise ill-posed problems, a solution at all. Thus the techniques carry the potential of becoming 'classics', some are already on its way towards that. Most of the topics are directly taken from the research literature, a few are covered in textbooks.

The following is a summary of the main ideas, highlighting the logic behind the argumentations and the main results, as well as references to the literature. Our apologies at this point: since we have taken the contents directly from diverse articles in most cases, the overall notation is not yet streamlined. Further, the presentation is in parts incomplete, partially not including proofs, for example. Obviously, the topics will be included one after the other, thereby allowing incremental printing because each topic has at least a new page. :-)

The following topics will be considered:

- Intelligent representation of data, in particular
  - extraction of semantically meaningful entities from time series without any knowledge about the semantics: slow feature analysis
  - adaptive efficient encoding of data by sparse coding and non negative matrix factorization
  - the one pixel camera: compressed sensing
- Efficient classification, in particular

  - how to know which features are relevant: support feature machine,
  - classification and co. in linear time based on core techniques,
  - prior over function instead of parameterization: Gaussian processes

- Time series processing, in particular
  - a classical approach for time series and its kernel: dynamic time warping

1 Slow Feature Analysis

Slow feature analysis (SFA) was pioneered by Laurens Wiskott, who is currently professor in Bochum. An excellent overview of the technique together with links to code and further readings can be found at scholarpedia:
http://www.scholarpedia.org/article/Slow_feature_analysis

1.1 Underlying problem and key idea

Assume a person in a lecture room sitting on a chair, falling asleep because of the boring lecture, and as a result falling from the chair. As a human observer of the scene, it is very easy to extract the relevant entities of the scene from the visual input, in this case observing the person as one entity which is slowly falling from the chair. The question is whether this capability of humans of recognizing a semantic entity (in this case the human) in a high dimensional time dependent signal (in this case the visual observation) can be mimicked by an algorithmic counterpart. The crucial part is that we want to do so without any prior knowledge or prior assumptions on the semantic entity! The problem is not to devise a classifier for persons based on examples, rather the question is to devise a method which can indicate an arbitrary semantic entity in an arbitrary temporal scene, if present, without relying on any examples or prior knowledge.

Slow feature analysis deals with the following problem:

Given a time series of high dimensional data, how can we algorithmically extract meaningful entities from the scene without posing any specific assumptions on the shape of the entities or specifying it by means of examples?

This problem constitutes an ill-posed problem: there is no way to infer meaningful entities from raw pixels since it is not clear what "meaningful" is referring to. Everything starting from single pixels to arbitrary groupings would be ok. Of course, as humans, we have some idea about what a meaningful entity would mean, clearly not aiming at single pixels only. Here, the main idea of SFA comes into the play:

Entities are characterized by the fact that they vary slowly over time.

This specifies entities as input components which are of a similar pairwise relation when considered over time: e.g. a person falling from a chair wears the same clothes in all images, resulting in the same colors and textures over time. This principle does not refer to any specifics of the domain, rather it refers to general characteristics of smooth processes in time, so it is a universal principle applicable to any high dimensional time series. The surprise is that, based on this assumption

alone and some reasonable additional technical formalizations, a proper solution of the problem can be found to some extent!

Besides this fundamental idea to rely solely on the slowness principle, the big achievement of SFA is to derive an explicit mathematical solution for the problem under this assumption, and a few typical application scenarios as well as theoretical insights have been obtained.

1.2 Formalization

Some basics on data processing:

First, we need to introduce a few classical pieces of data processing, which are of its own use if considered e.g. in the context of data normalization, dimensionality reduction, or data analysis.

The setting is a simple vectorial one, for the moment. Assume vectorial data $\vec{x}^1, \ldots, \vec{x}^N \in \mathbb{R}^n$ are given. Important statistical characteristics of these data are:

- *mean value/expectation:* $E\vec{x} = \sum_{i=1}^{N} \vec{x}^i/N$
- *covariance matrix*: $C = E(\vec{x}\vec{x}^t) = XX^t/N = \left[\sum_{i=1}^{N} x_j^i x_k^i/N\right]_{jk}$ where we assume that the data of the data matrix $X = (\vec{x}^1, \ldots, \vec{x}^N)$ has zero expectation: $E\vec{x} = \vec{0}$ (e.g. by data centering). The covariance matrix describes the scaling and pairwise correlation of the data dimensionalities: assume data are generated according to an underlying probability. Then, the resulting covariance matrix is diagonal if and only if data dimensions are uncorrelated. This displays in a data set which has a spherical shape. For correlated data dimensions, the information about the main axes along which data are located can be retrieved from the matrix. The covariance matrix C is always symmetric and positive semidefinite, which implies that it it diagonalizable.
- *principal components*: Since the covariance matrix C is symmetric and positive semidefinite, we know from linear algebra that it can be diagonalized: $C = VDV^t$ (or $CV = DV$) for an orthonormal matrix V and a diagonal matrix D. The eigenvectors of C, i.e. the columns of V, are called *principal components*. If sorted according to the size of the corresponding eigenvalue, they single out the directions of the data with largest variance (i.e. largest information content in the absence of auxiliary knowledge).

We can use the above statistical quantities for data transformation:

- *Data centering:* Subtract the mean of the data vectors from each data vector. This way, we get rid of potential offsets.

- *Dimensionality reduction / principal component analysis:* Project the data to the main principal components, i.e. substitute X by $V_{\text{main}}^t X$ with V_{main} consisting of the main principal components, a choice of only two yielding a standard linear visualization of the data.

  Attention: This approach might cause information to be lost. The dimensions are possibly no longer meaningful since they correspond to linear combinations of the original ones.

- *Whitening:* transform data linearly to zero expectation, and a correlation matrix given by the identity. The data vectors have the same covariance matrix as a set of white noise vectors after this transformation. This is achieved by the following steps:

  subtract expectation
  decompose covariance matrix $C = VDV^t$
  consider $\underbrace{D^{-1/2}}_{\text{scaling}} \underbrace{V^t X}_{\text{rotation}}$

  Attention: noise represented by eigenvectors with small eigenvalues is emphasized this way. Whitening preserves only non-Gaussian structure, linear transformations of Gaussians become the standard Gaussian distribution with zero mean and unit covariance.

- *Online major component analysis:* refers to PCA done by means of iterative updates according to single data points. Inspired by neural mechanisms (Hebbian learning of a single neuron results in PCA, this is the famous Oja learning rule and accompanying theory), this is also applicable for high dimensionality or very large/streaming data sets. The essential scheme is always as follows, based on the observation that eigenvalues are optima of the function $\vec{v}^t C \vec{v}$:

  init eigendirections $\vec{v}$
  iterate:
  update with major term proportional to $(\vec{v}^t \vec{x}^i) \cdot \vec{x}^i$
  (this term is the stochastic gradient of the quadratic function $\vec{v}^t C \vec{v}$)
  normalize vectors $\vec{v}$
  decorrelate vectors $\vec{v}$

  One concrete example of this procedure is Candid Covariance free incremental PCA:

initialize eigenvectors $\vec{v}^j$ randomly
repeat:
given data $\vec{x}$
$\vec{x}^1 := \vec{x}$,
for all i:
$\vec{v}^i := (1-\eta)\vec{v}^i + \eta \frac{(\vec{x}^i)^t \vec{v}^i}{|\vec{v}^i|} \vec{x}^i$
$\vec{x}^{i+1} := \vec{x}^i - \left((\vec{x}^i)^t \frac{\vec{v}^i}{|\vec{v}^i|}\right) \frac{\vec{v}^i}{|\vec{v}^i|}$

- *Online minor component analysis:* refers to an online extraction of the eigenvectors with smallest eigenvalues. Since these are minima of $\vec{v}^t C \vec{v}$, it could be done by gradient descent similar to online PCA; However, this is numerically unstable and not guaranteed to converge depending on the size of the eigenvalues; thus, corrections are usually necessary if anti-Hebbian learning takes place. There are different approaches, some of which are based on the following observation
  - $C' = -C + \gamma I$ where $\gamma >$ largest eigenvalue of C has the same eigenvectors as C but in opposite order. So one can do online estimation of an appropriate γ and then online PCA for the corrected values. Due to the inverse sign, this essentially corresponds to anti-Hebbian learning where a correction according to γ takes place.

Definition of slow features

Assume a time series of data $\vec{x}(t) \in \mathbb{R}^n$ are given with $t = 1, \ldots, T$.

Definition 1 Slow features (SF) *are projections*

$$g_j : \mathbb{R}^n \to \mathbb{R}$$

with the following constraints:

1. *Zero expectation over time:* $\sum_{t=1}^{T} g_j(\vec{x}(t)) = 0$ *for all* j. *This is just a normalization of the offset of the SF.*
2. *Normalized standard deviation:* $\sum_{t=1}^{T} g_j(\vec{x}(t))^2/T = 1$ *for all* j. *This normalizes the variance of* g_j *and, at the same time, together with 1, prevents trivial SF such as constants.*
3. *Orthogonality of features:*

$$\sum_{t=1}^{T} g_j(\vec{x}(t)) g_{j'}(\vec{x}(t)) = 0 \text{ for all } j \neq j'$$

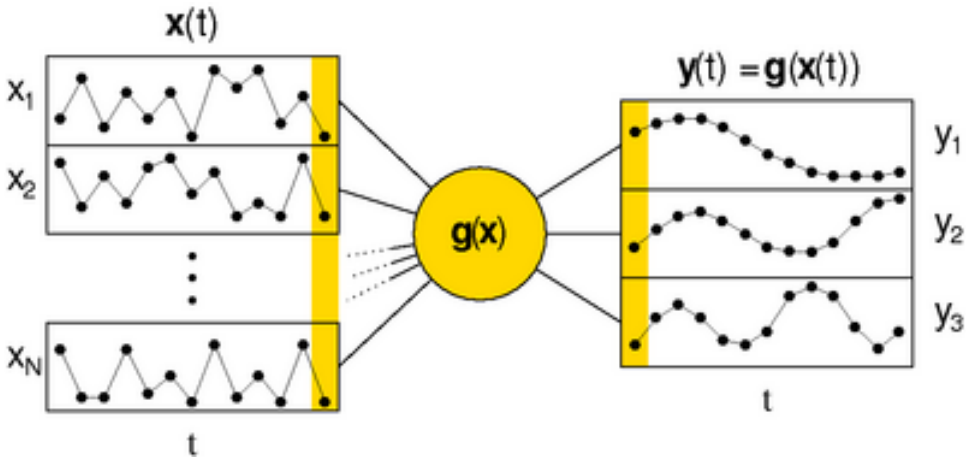


Figure 1: Functional principle of SFA: the slow features g extract instantanous characteristics from the data, where learning makes use of the temporal characteristics, but not the feature transformation itself. (Image taken from [1])

This prevents different features from being identical or capturing similar entities.

4. *Slow temporal variation:*

$$\langle \dot{g}_j \rangle := \sum_{t=1}^{T-1} \left(g_j(\vec{x}(t+1)) - g_j(\vec{x}(t)) \right)^2$$

is minimal. This prefers functions such that the function values change slowly over time. It constitutes the key property imposed on slow features.

In which sense are these potentially meaningful entities? Note that, while the characterization of the properties of g_j refers to the temporal axes, the features g_j are instantaneous. That means, given a signal $\vec{x}$ at one time point only, we can compute $g_j(\vec{x})$. The assumption is that $g_j(\vec{x})$ gives a large value if and only if some specific entity is present in the current signal. Due to condition 4, it is beneficial to build the output on subsets of the signals which change slowly over time (and to neglect the other input dimensions). Due to conditions 1 and 2, trivial features are thereby excluded. See Fig. 2 for the principled functional form and Fig. 3 for the algorithm derived thereof.

Note that SFA does not really extract features from the signals, rather it serves as an indicator whether some prominent feature is present as detected via g_j. So taking several such slow features g_j a more compact representation of the scene in terms of indicators of meaningful entities is found. The entities itself, however,

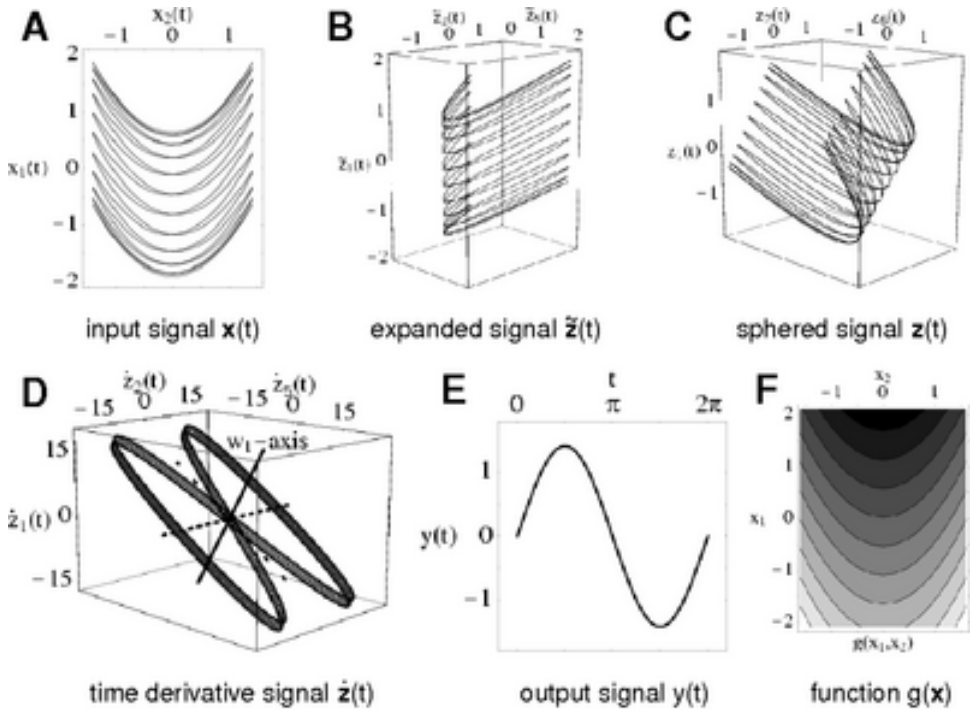


Figure 2: Demonstration of the extraction of slow features in one simple example. (Image taken from [1])

need to be investigated by referring to properties of the slow features such as the inverse images of g_j. An example application of SFA for video data is shown in Fig. 3.

Formalization of SFA:

Conditions 1-4 constitute a functional problem and are still ill-posed if the functions g_j can have an arbitrary shape. The problem can be rendered meaninful by imposing restrictive conditions for the g_j. In Slow Feature Analysis, a fixed parametric form is assumed for the slow features:

$$g_j(\vec{x}) = \sum_{k=1}^{K} a_k^j \Phi_k(\vec{x})$$

with fixed base functions $\Phi_k : \mathbb{R}^n \to \mathbb{R}$, $k = 1, \ldots, K$. Having chosen these base funtions, we can imagine each slow feature g_j as an element of the vector space spanned by $\Phi_1, \ldots, \Phi_K$, which is represented by the coefficient vector $\vec{a}^j = \left(a_1^j, \ldots, a_K^j\right)$. Typically, the base functions Φ_k are chosen as monomials up to a

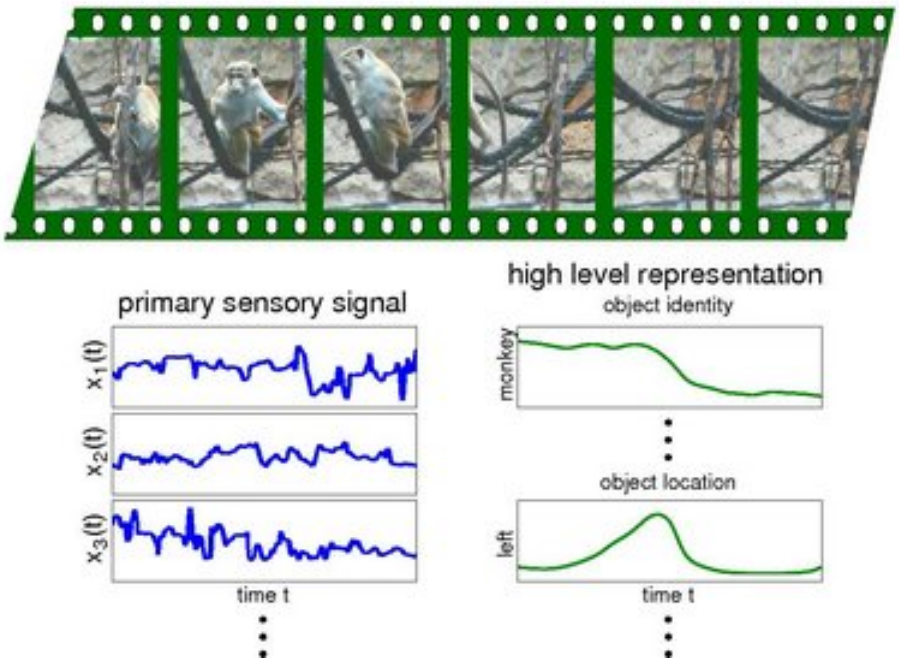


Figure 3: Example for slow features from image data. (Image taken from [1])

fixed degree d, i.e. $x_{i_1} \cdot \ldots \cdot x_{i_d}$. Typically d is small resulting in a linear ($d = 1$) or at most quadratic ($d = 2$) transformation to avoid high dimensionality for large n. Such functions are also referred to as general linear models, since the parameters of the function a_k^j are linear for inputs which undergo a fixed possibly nonlinear preprocessing. Under this assumption, one can rephrase the conditions 1-4 to conditions on the parameters a_k^j resulting in linear or quadratic equations which depend on the matrix

$$\Phi = [\Phi_k(\vec{x}(t)]_{t,k}$$

Here we explicitly compute the resulting conditions:

1. Zero expectation over time:

$$\sum_{t=1}^{T} g_j(\vec{x}(t)) = \sum_{t=1}^{T} \sum_{k=1}^{K} a_k^j \Phi(\vec{x}(t)) = \sum_{k=1}^{K} a_k^j \sum_{t=1}^{T} \Phi_k(\vec{x}(t)) \stackrel{!}{=} 0$$

   This property holds provided the latter sums, i.e. the expectations of the preprocessed vectors $\Phi_k(\vec{x}(\cdot))$ taken over time are zero. Thus, if we assume that the preprocessed vectors are *centered*, the condition is fulfilled.

2. Normalized standard deviation:

$$\begin{aligned}\sum_{t=1}^{T}(g_j(\vec{x}(t))^2/T &= \sum_{t=1}^{T}\left(\sum_{k=1}^{K}a_k^j\Phi_k(\vec{x}(t))\right)^2/T \\ &= \sum_{t=1}^{T}\left(\sum_{kk'}\left(a_k^j\Phi_k(\vec{x}(t))a_{k'}^j\Phi_{k'}(\vec{x}(t))\right)\right)/T \\ &= \sum_{kk'}\left(a_k^j a_{k'}^j\sum_{t=1}^{T}\left(\Phi_k(\vec{x}(t))\Phi_{k'}\left(\vec{x}(t)\right)\right)\right)/T \\ &\stackrel{!}{=} 1\end{aligned}$$

If we assume that the preprocessed vectors $\Phi_k(\vec{x}(\cdot))$ are *whitened* the latter summands vanish for $k \neq k'$, and they equal 1 otherwise. Hence, under this condition, this requirement boils down to the requirement $\sum_k (a_k^j)^2 = 1$, i.e. to have *normalized coefficient vectors* $a_\cdot^j$. (We can choose a constant different from 1, as long as it is the same for every j.)

3. Orthogonality of features:

$$\begin{aligned}\sum_{t=1}^{T} g_j(\vec{x}(t)) \cdot g_{j'}(\vec{x}(t)) &= \sum_{t=1}^{T}\left(\left(\sum_{k=1}^{K}a_k^j\Phi_k(\vec{x}(t))\right)\cdot\left(\sum_{k'=1}^{K}a_{k'}^{j'}\Phi_{k'}(\vec{x}(t))\right)\right) \\ &= \sum_{t=1}^{T}\left(\sum_{kk'}\left(a_k^j\Phi_k(\vec{x}(t))a_{k'}^{j'}\Phi_{k'}(\vec{x}(t))\right)\right) \\ &= \sum_{kk'}a_k^j a_{k'}^{j'}\sum_{t=1}^{T}\Phi_k(\vec{x}(t))\Phi_{k'}(\vec{x}(t)) \\ &\stackrel{!}{=} 0 \text{ for } j \neq j'\end{aligned}$$

Again, under the assumption of whitened vectors, this reduces to the requirement $\sum_k a_k^j a_k^{j'} = 0$ for $j \neq j'$, i.e. the requirement that the coefficient vectors $a_\cdot^j$ and $a_\cdot^{j'}$ are orthogonal.

4. Slow temporal variation:

$$\begin{aligned}
\langle \dot{g}_j \rangle &= \sum_{t=1}^{T-1} \left(g_j(\vec{x}(t+1)) - g_j(\vec{x}(t))\right)^2 \\
&= \sum_{t=1}^{T-1} \left(\sum_{k=1}^{K} a_k^j \Phi_k(\vec{x}(t+1)) - \sum_{k'=1}^{K} a_{k'}^j \Phi_{k'}(\vec{x}(t)) \right)^2 \\
&= \sum_{t=1}^{T-1} \left(\sum_{k=1}^{K} a_k^j (\Phi_k(\vec{x}(t+1)) - \Phi_k(\vec{x}(t))) \right)^2 \\
&= \sum_{kk'} a_k^j a_{k'}^j \cdot \left[\sum_{t=1}^{T-1} [(\Phi_k(\vec{x}(t+1)) - \Phi_k(\vec{x}(t)))(\Phi_{k'}(\vec{x}(t+1)) - \Phi_{k'}(\vec{x}(t)))] \right] \\
&= (\vec{a}^j)^t \cdot C(\dot{\Phi})(\vec{a}^j) \stackrel{!}{=} \min
\end{aligned}$$

where $C(\dot{\Phi})$ is the (scaled) covariance matrix of the data vectors $\vec{\Phi}(\vec{x}(t+1)) - \vec{\Phi}(\vec{x}(t))$ where the dimensionality runs over the coefficients of the base functions. This is minimized by the eigenvector according to the smallest eigenvalue of the covariance matrix of the temporal differences: the minor component.

Due to these computations, it is obvious that these equations can now be explicitly solved in two steps:

- Whiten the data represented by the matrix Φ resulting in $\tilde{\Phi}$.
- Compute the temporal differences.
- The coefficients $\vec{a}^j$ are then characterized as minor components of the covariance matrix of these temporal differences.

Algorithm:

The algorithm becomes:

(A) Given data $\vec{x}(t) \in \mathbb{R}$, $t = 1, \ldots, T$.

(B) *Data expansion:*
Choose base functions $\Phi_k \colon \mathbb{R}^n \to \mathbb{R}$ e.g. given by monomials of degree at most d, typically, $d \in \{1, 2\}$
Transform the data using these base functions

$$\Phi = [\Phi_k(\vec{x}(t))]_{t,k}$$

(C) *Data whitening:*
Center data columns by subtracting the respective mean value from each column; this corresponds to a translation with vector $\vec{m}$.
Compute the principal components of the covariance matrix of these centered vectors and multiply matrix $D^{-1/2}V^t$ and data matrix, resulting in $\tilde{\Phi}$.

(D) *Time derivative signal:*
Compute the temporal difference vectors $\tilde{\Phi}_{t+1,-} - \tilde{\Phi}_{t,-}$.
Compute the minor components of the corresponding covariance matrix resulting in coefficient vectors $\vec{a}^1, \ldots, \vec{a}^K$. These vectors are normalized and orthogonal per definition. Further, they minimize the objective 4 with the first minor component corresponding to the slowest slow feature, the second minor component corresponding to the second slowest slow feature, etc.

(E) *Output:*
Then, slow features are given by

$$\begin{aligned} g_j : \mathbb{R}^n &\to \mathbb{R} \\ \vec{x} &\mapsto (\vec{a}^j)^t \cdot D^{-1/2} \cdot V^t \cdot (\Phi(\vec{x}) - \vec{m}) \end{aligned}$$

where the coefficient vectors $\vec{a}^j$ are taken according to the K smallest minor components and $\Phi(\vec{x})$ is given by the vector of base functions $\Phi_k(\vec{x})$.

Albeit SFA is usually computed in batch mode, a generalization to online scenarios, i.e. time series which might be subject to trend and non-stationarities are possible. This is obvious since the ingredients of the procedure which are the mean value of the data, its principal components for data whitening, and the minor components which actually yield the slow features can be computed in an online way. SFA itself is very costly if applied to high dimensional data, because of which property it is often applied hierarchically.

1.3 Further issues

Applications

- Representing time series data by its most prominent slow features, we arrive at a low dimensional representation of time series, which can be beneficial in areas such as robotics or image processing by enhancing e.g. robustness or generalization ability of classifications.

- SFA can extract invariant features for rotated or slowly transformed objects, see Fig. 4.

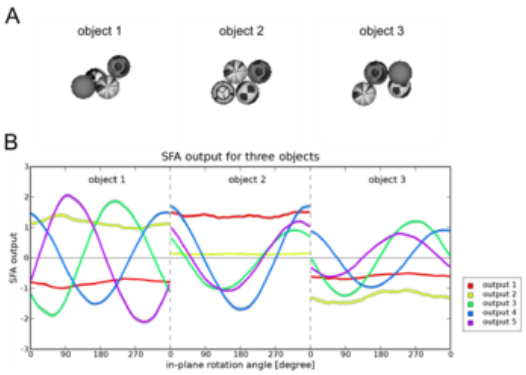


Figure 4: Slow features when training on different object constellations, entities such as e.g. object type can be extracted. (Image taken from [1])

- Preprocessing to predict a real valued function: labeled objects are sorted according to the size of the output. Then SFA prefers features which vary smoothly with the output one is interested in. Thus, it is likely that SFA extracts simple features which correlate with the desired output.
- Identification of driving forces in dynamic systems: if SFA is applied to time slices of the system, the slowly varying metaparameters of the dynamical system are likely detected, see Fig. 5.
- Independent source detection based on the observation that a mixture of two sources varies faster over time than the separate ones. So slow features are likely parts of these sources, see Fig. 6.
- SFA could be a principle used in biological systems: when applied to natural images, slow features resemble patterns found in cells of the primary visual cortex, see Fig. 7.

Theoretical guarantees

There exist quite a few investigations of SFA in idealized settings which show that, under some conditions, meaningful features are indeed obtained. It has been shown, for example, that the ideal solution for SFA results in standing waves on the data manifold, assumed the preprocessing Φ has sufficient flexibility. Hence restricting Φ to very smooth (linear or quadratic) functions implies that the method extracts simple linear combinations and correlations of input dimensions which vary smoothly over time, like entities moving through a scene.

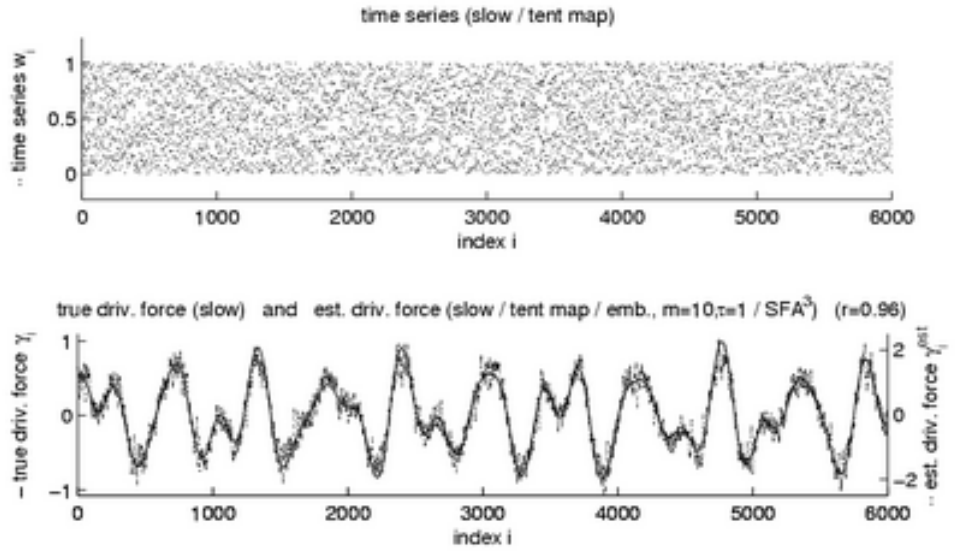


Figure 5: Time series generated with a slowly varying principle, the latter can be extracted using SFA. (Image taken from [1])

Literature

1. Slow feature analysis, Laurenz Wiskott et al. (2011), Scholarpedia, 6(4):5282 (recommended!)

2. Wiskott, L (2003). Slow feature analysis: A theoretical analysis of optimal free responses. Neural Computation 15(9): 2147-2177.

3. Wiskott, L and Sejnowski, T (2002). Slow feature analysis: Unsupervised learning of invariances. Neural Computation 14(4): 715-770.

4. Einhhäuser, W, and Hipp and J, and Eggert, J, and Körner E, and König P, (2005). Learning viewpoint invariant object representations using a temporal coherence principle. Biological Cybernetics 93: 7990.

5. V. Kompella, M. Luciw, J. Schmidhuber, Incremental Slow Feature Analysis, IJCAI2011.

6. Code: mdp-toolkit.sourceforge.net (modular toolkit for data processing, python)

7. Code: P.Berkes (2003) sfa-tk: Slow Feature Analysis Toolkit for Matlab (v.1.0.1). http://people.brandeis.edu/ berkes/software/sfa-tk/index.html

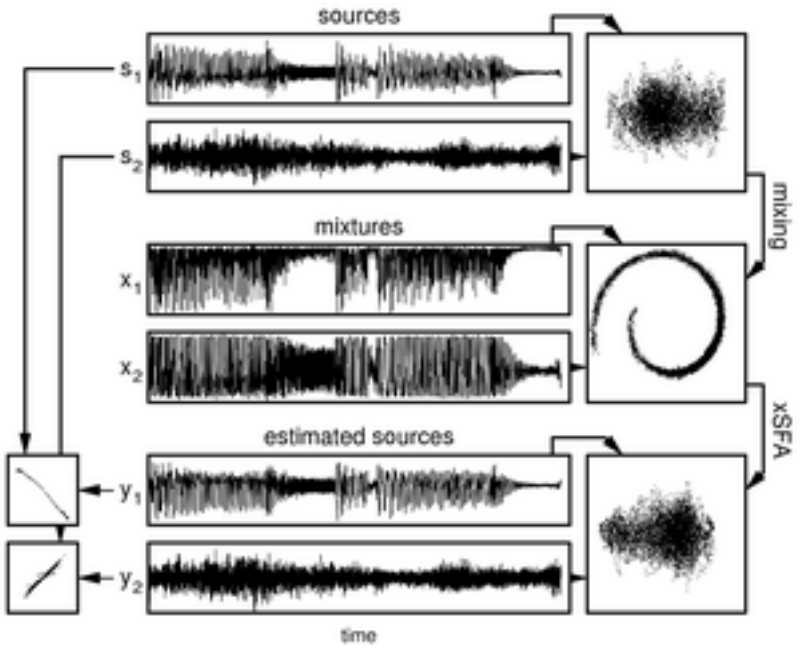


Figure 6: Slow feature analysis applied to a blind source separation problem. (Image taken from [1])

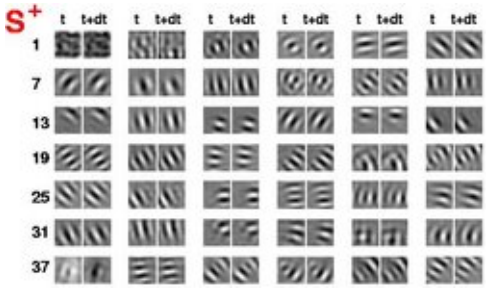


Figure 7: Slow features by training on natural imaged, which resemble cell activations as found in the primary visual cortex. (Image taken from [1])

2 Sparse Coding

Encoding of data constitutes a classical topic in computer science where the aim is to represent data in a more efficient way as regards some properties, in particular storage space. There exist well known data encoding formats such as e.g. Fourier- Wavelet- or Cosine-transformation of data, the latter being included in JPEG, or discretization of the data such as present in the GIF format. Unlike these classical formats, we are interested in data encoding schemes where the encoding is in some way adaptive to the data, and where it resembles some (biological) plausibility, which will mostly manifest in the observation that the entities used for encoding the data often have a semantic meaning due to the assumption of underlying universal principles, very similar to SFA. The key assumption lies in the sparsity of the observed code, as already indicated by the name *sparse coding*.

2.1 Linear encoding

First, let us consider encoding of data as a general principle. Encoding and decoding of data is concerned with the following setting:

Definition 2 Encoding *of data: Given data $\vec{x}^i \in \mathbb{R}^n$ for some high dimensionality n, an encoding of the data is a mapping*

$$\vec{x}^i \in \mathbb{R}^n \mapsto \vec{a}^i \in \mathbb{R}^k$$

where, typically, $k \ll n$ and decoding *is possible, i.e. we can reconstruct the vectors $\vec{x}^i$ by an inverse mapping:*

$$\vec{a}^i \mapsto \tilde{x}^i \approx \vec{x}^i$$

So far, we could use a table lookup as long as we have a finite number of data to encode only, therefore, we need to put some constraints on the codes which are acceptable. We will restrict to a specific type of encoding which we refer to as linear encoding by means of base vectors:

Definition 3 *A linear encoding specifies a set of base vectors also called a dictionary $\vec{d^1}, \ldots, \vec{d^k}$ in $\mathbb{R}^n$, which is the same for the encoding of all data vectors, and a coefficient vector (encoding vector) $\vec{a}^i \in \mathbb{R}^k$ for every $\vec{x}^i$ such that*

$$\vec{x}^i \approx \sum_{j=1}^{k} a_j^i \vec{d^j}$$

holds. In matrix notation, writing $D = (\vec{d^1}, \ldots, \vec{d^k})$ this is

$$\vec{x}^i \approx D\vec{a}^i$$

Note that the dictionary is fixed throughout the encoding, while the coefficients constitute the actual encoding of a given data point. In the following, we will mainly be concerned with the following two questions:

- Given a fixed dictionary D, how to compute a code $\vec{a}^i$ for a given point $\vec{x}^i$?
- Given an application domain represented by a set of data points $\vec{x}^1$, ..., $\vec{x}^N \in \mathbb{R}^n$, what is a good dictionary D for this data?

Often, when addressing the second problem, the first one is also solved as a subproblem. Before deriving mechanisms that can help to infer good dictionaries, we consider a few classical examples, which are extremal as concerns some parameters of the setting. Further, we specify relevant properties of such an encoding to emphasize the benefits of sparse coding.

So let's start with important properties of a linear encoding scheme:

- *lossy/loss free encoding* concerns the question whether exact identity $\vec{x}^i = D\vec{a}^i$ holds. In general, this is possible only if the dictionary D constitutes a base for the considered vector space.
- *adaptive/static dictionary* concerns the question whether the dictionary D is tailored to a given data set or whether it is universal, such as the unit vectors (which usually do not allow a compression of data at all, so it is usually not referred to as an encoding, although, strictly speaking, it is ;-)) or a tailored base for certain application domains such as a Fourier base.
- *local/distributed encoding* considers the number of non-zero coefficients of $\vec{a}^i$, which we can count by the 0-norm $|\vec{a}^i|_0$. Sparsity directly relates to the efficiency of decoding and the amortized storage space for the data; for sparse codes decoding is faster since it is a sum over a small number of base functions only.
- *overcomplete/non-redundant dictionary* considers the question whether elements of the dictionary can be expressed as linear combinations of each other, and, hence, are redundant, since they could be substituted by the others. On the other hand, orthogonal base functions lead to non-redundant representations. For the latter, it is very easy to obtain an optimal encoding as concerns the least squares reconstruction error $\|\vec{x}^i - D^t\vec{a}^i\|_2^2$: the coefficients are given as projections to the base functions: $a^i_j = (\vec{d}^j)^t\vec{x}^i / \|\vec{d}^j\|_2^2$. For overcomplete dictionaries, this may constitute a problem, as we will see later. Closely connected to this property is the size of the dictionary, which can be reduced if the dictionary is non redundant. However, overcomplete dictionaries may allow a more local encoding.

Further interesting properties concern e.g. the robustness of the encoding, that is, if the codes $\vec{a}^i$ change by some noise, how much does this affect the resulting vector $D\vec{a}^i$.

Classical examples of coding schemes include the following:

- PCA: the main principal components of given data constitute an optimum dictionary for linear encoding in the sense that its squared reconstruction error is minimum if these base vectors are chosen. It has the following properties:
  - It is lossy unless the full space is taken or eigenvalues are exactly zero.
  - It is adapted according to the given data.
  - It is a distributed encoding with typical codes $|\vec{a}^i|_0 = k$. In consequence, decoding is costly unless the dictionary is small, i.e. only few principal components are taken.
  - The vectors from the dictionary form an orthonormal system such that encoding consists of a linear projection only.

  PCA is one extreme in the sense that the codes $\vec{a}^j$ are fully distributed and, in practice, neither these codes nor the dictionary typically provide semantically meaningful entities on their own – the principal component directions are just some directions in the data space which do not have the same properties as the data or characteristics which can directly be interpreted (besides their statistical meaning as directions of largest variance).

- Vector quantization (VQ): vector quantization represents the data in terms of typical prototypes which form the vectors of the dictionary $\vec{d^1}, \ldots, \vec{d^k}$. It can be interpreted as an encoding scheme in the sense that every data point $\vec{x}^i$ is represented by a coefficient vector $\vec{a}^i$ which is constrained as a unary vector: it is 1 at exactly one position, the rest is zero. In consequence, a data point $\vec{x}^i$ is represented by one fixed prototype $\vec{d^j}$ only according to $a^i_l = \delta_{lj}$.

  The dictionary is adapted according to given data such that the squared reconstruction error is minimized:

  $$\sum_{i=1}^{N} (\vec{x}^i - \sum_{j=1}^{k} a^i_j \vec{d^j})^2$$

  The squared reconstruction error is also often referred to as quantization error in this context. In the Vector Quantization case, $|\vec{a}^i|_0 = 1$, and the latter sum singles out the base vector $\vec{d^j}$ according to the nonzero entry of $\vec{a}^i$. While it is clear how to encode a data point given a fixed dictionary (the

base function closest to the data point is its best representation), it is not easy to find an optimal dictionary. Minimizing the quantization error is actually an NP-hard problem, such that heuristics are used in practice. One very popular heuristic is offered by the popular *k-means clustering algorithms* which iteratively determines optimum codes $\vec{a}^i$ assuming a fixed dictionary and optimum dictionary vectors assuming fixed codes as follows:

init dictionary (e.g. setting every $\vec{d^j}$ to some data point $\vec{x}^i$)
repeat:
set $a^i_j = 1$ if $\vec{d^j}$ is the dictionary vector closest to $\vec{x}^i$, 0 otherwise
set $\vec{d^j} = \sum_{a^i_j=1} \vec{x}^i / \sum_{a^i_j=1} 1$ as center of gravity of the data it encodes

One can show that this scheme converges in a finite number of steps to a local minimum of the cost function with respect to the dictionary. Global minima are usually not obtained, and multiple restarts are advisable since the algorithm is very sensitive to noise. (If you are planning to use k-means as a clustering algorithm, please notice that there are much better algorithms than k-means with the same complexity, such as Neural Gas ...). Interestingly, k-means serves as motivation for a very popular sparse encoding scheme which we will consider in the following, so-called k-SVD (which has been extended to neural gas schemes to avoid the problems of k-means as regards sensitivity to initialization).

The properties of vector quantization as an encoding scheme are as follows:

- It is a lossy encoding scheme, where the number of code vectors directly determines the quality. Usually, a high number is necessary for a good representation of data.
- It is adaptive, using e.g. k-means to find a good dictionary.
- It is an extremely local scheme with $|\vec{a}^i|_0 = 1$ taking the minimum possible value (usually we do not want to encode 0 which would allow an even sparserr scheme ...).
- The basis is overcomplete; still due to the extreme sparseness of the encoding, the computation of optimum codes is efficient (linear in k), and there are no direct redundancies.

Vector quantization is extreme in the sense that it considers the most local encoding scheme possible. In consequence, the resulting encoding is possibly not very space efficient in the sense that a large dictionary is required, but encoding and decoding are time efficient, and the resulting dictionary

consists of representative places in the space of the data itself. Hence the dictionary D is interpretable!

- Fourier transform: Just to consider one classical encoding scheme where the dictionary is fixed, we have a short glimpse at a discrete Fourier representation of vectors. Fixed base functions are used which stem from harmonic oscillators: $\vec{b}^l = (\exp(2\pi i l j/n))_{j=0,\dots,n-1}$ where $i = \sqrt{-1}$ and $l \in \{0, \dots, k-1\}$ refers to the index of the base vector. E.g. for $n = 4$ we have the complex vectors $(1,1,1,1)$, $(1,i,-1,-i)$, $(1,-1,1,-1)$, $(1,-i,-1,i)$ corresponding to vectors with increasing frequency of the oscillation. These vectors are orthonormal, and an efficient computation of optimum coefficients $\vec{a}$, given a signal $\vec{x}$ as well as its inverse is possible, commonly known as discrete Fourier transform and its inverse requiring time $\mathcal{O}(n \log n)$ only. Thereby, obviously, coefficients and base functions are complex numbers rather than restricted to the reals due to the connection to oscillations.

  Discrete Fourier transform constitutes a standard procedure to efficiently represent and process data with oscillatory characteristics such as audio signals, but also image data. Often, not the full base but only the slowest oscillations are taken, so that a compression takes place. The properties of a Fourier representation are:

  - It is a lossless encoding scheme, provided the full base is taken. If some frequencies are dropped, the representation is lossy. Often, high frequencies are dropped suppressing noise in the data.
  - The representation is static and, thus, it is suited only for domains where oscillations can be observed. This is often the case in practice, because of which discrete Fourier transform is one of the standard encoding schemes for real life data.
  - It is a distributed scheme where $|\vec{a}^i|_0 = k$ for general signals, unless the data decompose into few harmonic oscillatory signals (we will use this principle for the next chapter, compressed sensing).
  - The basis is an orthonormal basis.

These examples shed some light on extremes ranging from local schemes (like vector quantization) to distributed encoding (like PCA). These two extremes have complementary weaknesses and benefits: while vector quantization provides an efficient encoding and the possibility of interpreting the base vectors, it requires a large number of base vectors for an accurate representation. On the other hand, PCA offers an optimum representation of data with respect to the required number of base functions, the latter, however, not being interpretable or efficient for decoding, requiring a full linear combination for almost all data.

The idea of sparse coding is to combine the benefits of these two extremes by referring to a sparse encoding where a few coefficients of the vector $\vec{a}^i$ (rather than only one) are non-zero.

2.2 Sparse coding by probabilistic modeling

In the classical approach of Olshausen/Field, sparse coding is derived by means of probabilistic modeling, and the intuitivity of the obtained results is demonstrated for the first time in the application area of vision. Due to these fundamental findings, it is speculated that sparse coding can play an important role as concerns neural encoding in the brain.

The general setting is as follows: data vectors $\vec{x}^1, \ldots, \vec{x}^N \in \mathbb{R}^n$ are given. We search for a set of base vectors $\vec{d^1}, \ldots, \vec{d^k} \in \mathbb{R}^n$, resulting in a matrix $D \in \mathbb{R}^{n \times k}$, which best explains the observed data under the assumption of a sparse representation. Hence we have two constraints:

- Reconstruction constraint: data can be reconstructed, i.e. for every $\vec{x}^i$ there is an $\vec{a}^i \in \mathbb{R}^k$ with $\vec{x}^i \approx \sum_j a^i_j \vec{d^j}$
- Sparsity constraint: $|\vec{a}^i|_0$ is small

How can we algorithmically find appropriate $\vec{d^j}$ which best encode the data observed in this situation? We consider a probabilistic setting; we assume that data are generated from sparse codes, but we do not know the parameters and the dictionary. Rather, we want to retrieve the most reasonable dictionary and coefficients from given data. So we have

$$p(\vec{x}|D, \vec{a}) \sim \exp\left(-\|\vec{x} - D\vec{a}\|_2^2/(2\sigma^2)\right)$$

assuming a reconstruction of data which is affected by Gaussian noise. We assume a prior on the coefficients which treats these coefficient as independent over the base vectors

$$p(\vec{a}) = \prod_{l=1}^{k} p(a_l)$$

and which emphasizes sparseness

$$p(a_l) \sim \exp(-\beta S(a_l))$$

where S is some function such as

- $S(a) = \log(1 + a^2)$
- $S(a) = |a|$

- $S(a) = -\exp(-a^2)$

which is peaked at $S = 0$ such that sparse solutions have a higher density. These probabilities define the probability of an observed data point given a dictionary

$$p(\vec{x}|D) = \int p(\vec{x}|D, \vec{a})p(\vec{a})d\vec{a}$$

In practice, a number of observations are made: $\vec{x}^1, \ldots, \vec{x}^N$ which give rise to an empirical distribution $p^*(\vec{x})$ which is peaked at the observed vectors.

The goal is to determine a dictionary D which best explains p^* by the corresponding p as measured by the Kullback Leibler divergence as a standard error function for probability measures:

$$\min_D \int p^*(\vec{x}) \log \frac{p^*(\vec{x})}{p(\vec{x}|D)} d(\vec{x})$$

The minimization of this cost function is usually not possible because we cannot analytically compute the integral $p(\vec{x}|D)$. Due to this fact, an approximation is made which is typically referred to by the most likely coefficient vector

$$p(\vec{x}|D) \approx \max_{\vec{a}} p(\vec{x}|D, \vec{a})p(\vec{a}) \cdot \text{const}$$

This approximation is, of course, possibly a crude one. Further, it includes an immediate numerical consequence which we have to take into account: the term $\sum_{l=1}^{k} a_l \vec{d^l}$ allows a scaling of the terms which does not affect the output, by multiplying $\vec{d^l}$ with a constant and dividing a_l by the same constant. The latter has a positive effect on the prior probability $p(\vec{a})$ without increasing the sparseness of the solution. Due to this fact, it is crucial to restrict the length of the base functions $\vec{d^l}$.

Taking into account this approximation, we find that the optimization problem

$$\min_D \int p^*(\vec{x}) \log \frac{p^*(\vec{x})}{p(\vec{x}|D)} d(\vec{x})$$

is the same as

$$\max_D \sum_{i=1}^{N} \log p(\vec{x^i}|D)$$

since p^* is peaked and the denominator is constant, which is the same as

$$\max_D \max_{\vec{a}} \sum_{i=1}^{N} \log p(\vec{x^i}|D, \vec{a})p(\vec{a})$$

under the considered approximation of the integral. If we take into account the Gaussian form of the probability, this reduces to the problem

$$\min_D \min_{\vec{a}} \left(\sum_{i=1}^{N} \|\vec{x}^i - D\vec{a}\|_2^2 + \lambda \sum_{ij} S(a_j^i) \right)$$

where λ is a constant depending on β, σ, and the function S. Hence sparse coding can be solved by minimizing the following two criteria with respect to the dictionary D and the coefficients $\vec{a}$:

$$\min_D \min_{\vec{a}^i} \sum_{i=1}^{N} \underbrace{\|\vec{x}^i - D\vec{a}^i\|_2^2}_{\text{reconstruction error}} + \lambda \underbrace{\sum_{ij} S(a_j^i)}_{\text{sparsity constraint}}$$

where the base vectors $\vec{d^j}$ are constrained in length to prevent degeneration.

In the original work by Olshausen and Field, an iterative optimization scheme is proposed which in the inner cycle finds optimum coefficients for the representation $\vec{a}^i$ and, in the outer cycle, optimizes D. Both optimization strategies are based on gradient techniques respecting the following equations

$$\dot{a}_j^i = \sum_{l=1}^{n} \underbrace{(x_l^i - \sum_{j'=1}^{k} a_{j'}^i d_l^{j'})}_{\text{error per coefficient}} \cdot \underbrace{d_l^j}_{\text{base function}} - \lambda \cdot \underbrace{S'(a_j^i)}_{\text{sparsity}}$$

and

$$\Delta\vec{d^j} \sim \sum_{i=1}^{N} \underbrace{a_j^i}_{\text{responsibility}} \underbrace{(\vec{x}^i - \sum_{j'=1}^{k} a_{j'}^i \vec{d^{j'}})}_{\text{error}}$$

accompanied by a normalization of $\vec{d^j}$. These adaptation rules can also be put into neural network like architectures, since they mainly accumulate weighted sums and some nonlinear activation functions, dubbed *sparsenet* in the articles. The Matlab code for sparsenet can be retrieved from [3]. These simple techniques, however, can be prone to numerical difficulties due to the complex optimization problem behind. Interestingly, by applying sparsenet to whitened natural images (which can also be found at [3]), base functions very similar to the receptive fields in the visual cortex can be obtained, as shown in Fig. 8.

Many extensions of sparse coding are essentially concerned with similar models, but address the numerical problem of finding adequate solutions for this or related formalizations.

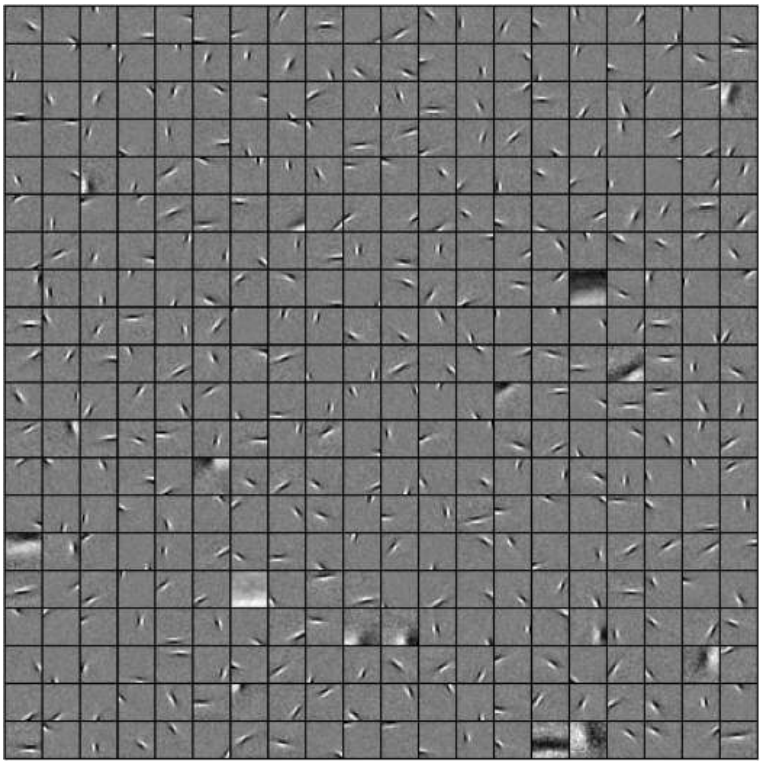


Figure 8: Base functions obtained by sparsenet when trained on natural image scenes, the image is taken from [2]

2.3 Non negative matrix factorization

The basic assumption of non-negative matrix factorization (NMF) is that observed signals $\vec{x}^i$ often have non-negative components only. For practical applications, this is the case if referring to grey values of images, intensities accumulated in spectral data, or word frequencies in documents. So we assume that all entries in the data $\vec{x}^1, \ldots, \vec{x}^N$ are non negative, accumulating all vectors in a matrix $X \in \mathbb{R}^{n \times N}$ with non negative coefficients only. In matrix notation, linear encoding of the data amounts to finding matrices $D \in \mathbb{R}^{n \times k}$ and a matrix of coefficient vectors $A \in \mathbb{R}^{k \times N}$ with

$$X \approx DA$$

For sparse coding, we would like to restrict the number of non-zero coefficients in A. In non negative matrix factorization, this is not explicitly required in the first approaches proposed in the literature. Rather, a different very natural constraint is taken:

The coefficients of the dictionary D as well as of the encoding A

should be *non negative*.

Obviously, this principle is also emphasized in the name of the techniques. Non negative matrix factorization has been pioneered by Lee and Seung [4]. Interestingly, it turns out that non negativity is sufficient to also obtain a sparse encoding and an interpretable dictionary in some (most ??) cases.

Modelling NMF

Now we are facing the question how to solve the problem

$$\text{decompose } X \approx DA \text{ where } D, A \geq 0$$

For a formalization, the notion of '$\approx$' needs to be specified. Typical cost functions refer to the least squares error

$$\min \|X - DA\|_2^2 \text{ such that } D, A \geq 0$$

(referring to the Frobenius norm of a matrix) or the generalized Kullback Leibler divergence

$$\min D_{\mathrm{KL}}(X, DA) \text{ such that } D, A \geq 0$$

where

$$D_{\mathrm{KL}}(M^1, M^2) = \sum_{ij} M_{ij}^1 \log \frac{M_{ij}^1}{M_{ij}^2} - M_{ij}^1 + M_{ij}^2$$

This generalized Kullback Leibler divergence measures differences of non-negative vectors (possibility vectors), without requiring these vectors to be normalized, i.e. to represent probabilities. Because of this fact, offsets are added which vanish if probabilities are dealt with, yielding the standard Kullback Leibler divergence for the latter case. These problems are NP hard in general, such that we have to rely on heuristic solutions and we have to accept local optima of the problem. In the literature, gradient based optimization has been proposed.

First, we consider the squared error. Taking the derivative yields the update

$$D_{ij} := D_{ij} + 2\eta(XA^t - DAA^t)_{ij}$$

and

$$A_{ij} := A_{ij} + 2\eta(D^tX - D^tDA)_{ij}$$

with step size η and the additional condition $D, A \geq 0$. If we set the step size individually to $\eta := D_{ij}/(2DAA^t)_{ij})$ for D_{ij} and $\eta := A_{ij}/(2D^tDA)_{ij}$ for A_{ij},

we obtain

$$\begin{aligned} & D_{ij} + 2 \cdot D_{ij}/(2DAA^t)_{ij})(XA^t - DAA^t)_{ij} \\ = \quad & \frac{D_{ij}(DAA^t)_{ij} + D_{ij}(XA^t)_{ij} - D_{ij}(DAA^t)_{ij}}{(DAA^t)_{ij}} \\ = \quad & D_{ij} \cdot \frac{(XA^t)_{ij}}{(DAA^t)_{ij}} \end{aligned}$$

A similar computation for A_{ij} leads to the updates

$$\begin{aligned} D_{ij} \quad &:= \quad D_{ij} \cdot \frac{(XA^t)_{ij}}{(DAA^t)_{ij}} \\ A_{ij} \quad &:= \quad A_{ij} \cdot \frac{(D^tX)_{ij}}{(D^tDA)_{ij}} \end{aligned}$$

where the non negativity is automatically fulfilled provided the matrices D and A are initialized with non negative values.

For the generalized Kullback Leibler divergence, we obtain the gradients

$$D_{ij} := D_{ij} + \eta \left(\sum_{l=1}^{N} A_{jl} \frac{X_{il}}{(DA)_{il}} - \sum_{l=1}^{N} A_{jl} \right)$$

and

$$A_{ij} := D_{ij} + \eta \left(\sum_{l=1}^{n} D_{li} \frac{X_{lj}}{(DA)_{jl}} - \sum_{l=1}^{n} D_{li} \right)$$

which, setting $\eta := D_{ij}/\sum_l A_{jl}$ or $\eta := A_{ij}/\sum_l D_{li}$ yields the multiplicative updates

$$\begin{aligned} D_{ij} \quad &:= \quad D_{ij} \cdot \frac{\sum_l A_{jl} X_{il}/(DA)_{il}}{\sum_l A_{jl}} \\ A_{ij} \quad &:= \quad A_{ij} \cdot \frac{\sum_l D_{li} X_{lj}/(DA)_{lj}}{\sum_l D_{li}} \end{aligned}$$

where non-negativity is automatically fulfilled.

While a gradient descent converges to local optima under mild conditions on the cost function and the step size, this behavior is not clear if non-negativity is enforced, or if the step-sizes are chosen individually. Indeed, it can be shown that the multiplicative update decreases the costs in every step, but it is not guaranteed to converge to a local optimum. Due to this circumstance, an alternative optimization scheme has been proposed based on an alternating least squares scheme. The squared error constitutes a convex cost function for D or A, respectively, such

that an explicit analytic solution becomes possible, neglecting the non-negativity for the moment. Taking the derivative with respect to A, we get the condition $D^tX - D^tDA = 0$ or $A = (D^tD)^{-1}D^tX$ where the pseudo-inverse is referred to provided D^tD is not of full rank. A similar argument for D yields the following loop to optimize A and D:

repeat:
solve for A: $D^tDA = D^tX$, set negative entries to zero,
solve for D: $AA^tD^t = AX^t$, set negative entries to zero

This algorithm can easily yield to poor local optima because of the fact that entries which are set to 0 remain zero. Hence alternatives have been proposed in the literature where the non-negativity is integrated in the cost function by means of a penalty constraint.

Applications of NMF

Fig. 9 shows an example application of NMF for images of faces in comparison to a PCA or VQ encoding. In this setting, the resulting codes are sparse (albeit this is not explicitly modeled in NMF), and the base vectors have an interesting semantic interpretation: they represent important (sparse) parts of a face such as eyes, mouth, etc.

There are a couple of situations where the non-negativity of the observations X is given and a decomposition of the data into non-negative components can give some insight into the data:

- *Text mining*: Entities $\vec{x}^i$ are text documents represented by the frequency of the words which occur in those documents (typically, these are counted and weighted with the so-called term frequency times inverse document frequency, i.e. tf-idf value, but the precise form is not so important at this place as long as the numbers are non negative). Then X is the word-document matrix of the data set. A non-negative matrix factorization provided the number of base vectors k is small gives insights into the k hidden factors of these documents (such as the overall content which manifests itself in typical words used in this context) and a sparse representation which indicates which hidden factors are relevant for a document. In [7] this principle has been applied to e-mails exchanged in the Enron case. In this scandal, the energy company Enron Corporation went bankrupt in 2001, attributed as the biggest audit failure in American history. Fig. 10 gives the results of example base vectors found when analyzing the e-mails.

- *Coclustering for the analysis of Microarrays:* Microarrays constitute a standard technology in bioinformatics where the presence of thousands of genes

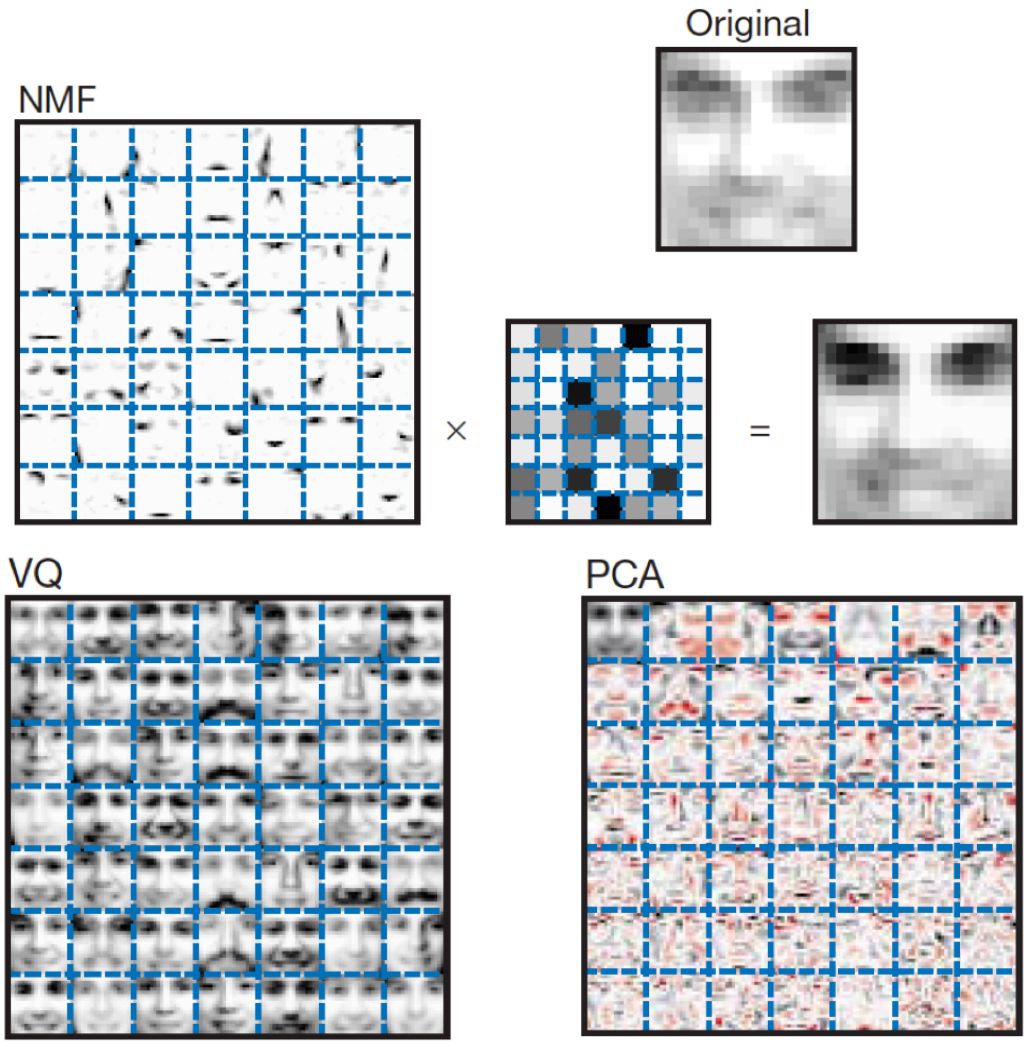


Figure 9: Result of NMF when applied to faces (top) in comparison to an encoding by means of vector quantization (bottom left) or PCA (bottom right). The images are taken from [4]

in a solution can be measured in one experiment. A typical biological problem is then to simultaneously cluster the solutions (i.e. underlying organisms, experimental conditions, or whatever has been analyzed and varied in the experimental series) according to the genes present, and to cluster the genes according to them being present in similar circumstances. This is typically referred to as co-clustering, with NMF offering one possible algorithmic approach. X corresponds to the matrix of microarray data, its

Feature Index (i)	Cluster Size	Topic Description	Dominant Nouns
2	3,970	Professional Football	Bettis, McNair, McNabb, stats, Faulk, rushing
10	6,190	California Blackout	Fichera, Escondido, biomass
13	9,234	Enron Downfall	Vinson, Elkins, destruction, scandal, auditing, Lieberman
18	3,583	Business Ventures	Teijin, Janus, Coale, Tata, BSES
21	4,011	India	Prabhu, Cline, Suresh, rupees, Vinay, renegotiation
23	8,526	World Energy/ Scotland	scottish, power, ENEL, Mitsui, vessel

Figure 10: Typical base functions found when analyzing e-mails in the Enron case. Topic descriptions are manually assigned, the most dominant nouns according to the weights are shown. The cluster size refers to the number of e-mails which have a high coefficient as regards this base function. The image is taken from [7].

decomposition to structures found in the probes on the one hand and basis functions of similar genes on the other hand. See Fig. 11 for an example application in the context of the analysis of Leukemia patients.

- *Unmixing of spectral signals*: Spectral data occurs in diverse contexts e.g. when analyzing radar signals, spectral imaging, liquid spectroscopy for the analysis of fluids, mass spectometry, etc. These areas have in common that spectral signals which characterize one data point $\vec{x}^i$ constitute a mixture of basis elements such as characteristic spectra of basis materials. When facing real data, it is thus interesting to analyze which components are contained in a given spectrum. In [7], NMF has been used for this purpose, with X

Fig. 2. W (basis matrix) and H (coefficient matrix) obtained from SNMF/R with $\beta = 0.01$ for the ALLAML leukemia dataset (38 samples: 19 ALL-B, 8 ALL-T, 11 AML) with the 5,000 most highly varying genes.

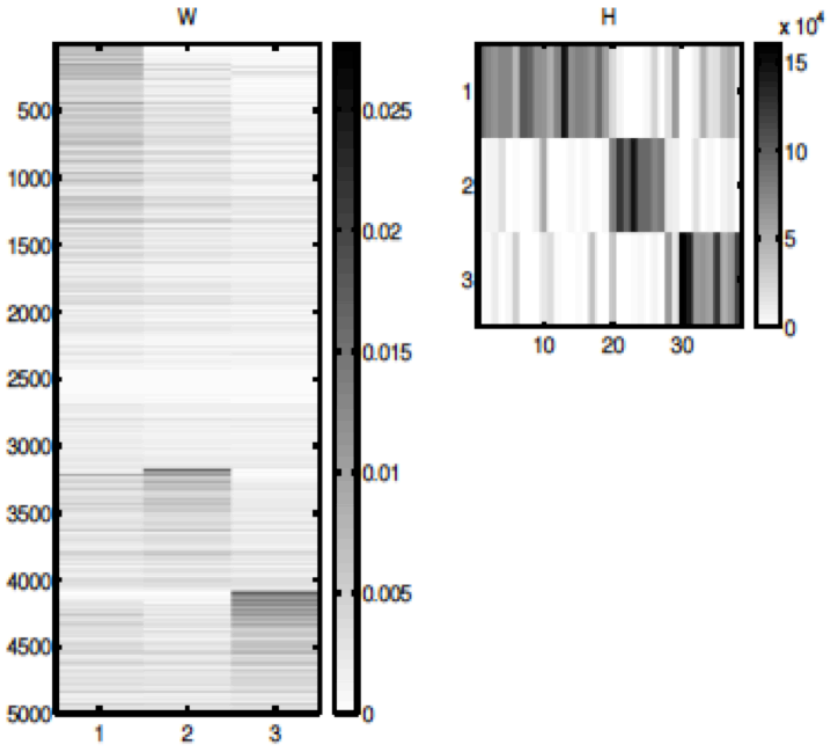


Figure 11: Cluster structure found when applying NMF to micorarray data, the image is taken from [8].

being a matrix of spectra, and the resulting dictionary D corresponding to characteristic spectra of basic elements (in this case materials found in a mixture of components).

Sparse NMF

While NMF often yields sparse representations of data, and also sparse base vectors, this happens only by accident, the main reason being that the constraint of non-negativity often favours sparse, meaningful basic constituents in real data. A few approaches deal with possibilities to explicitly incorporate a sparsity constraint into NMF.

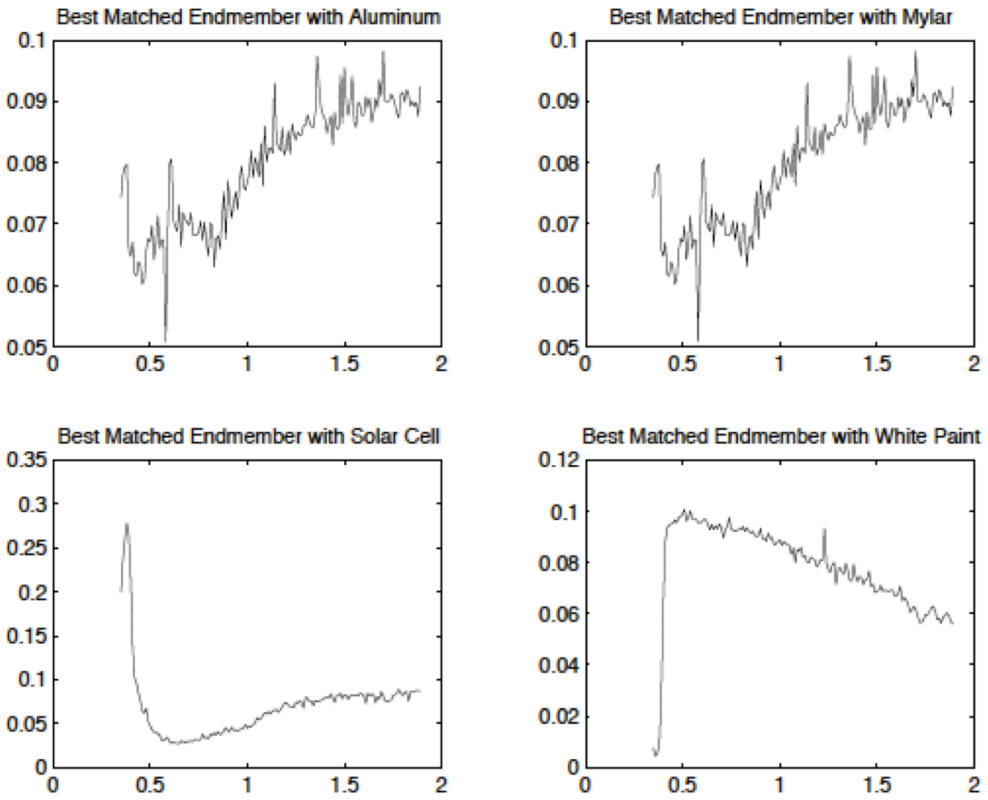


Figure 12: Unmixing of spectra by means of NMF, the image is taken from [7].

The approach of Hoyer relies on a sparsity constraint which is measured as

$$\text{sparseness}(\vec{x}) = \frac{\sqrt{n} - \|x_i\|_1/\|\vec{x}\|_2}{\sqrt{n} - 1}$$

for a given vector $\vec{x} \in \mathbb{R}^n$. We find the following relations:

$$\begin{aligned} \text{sparseness}(\vec{x}) = 1 \quad &\Longleftrightarrow \quad |\vec{x}|_0 = 1 \\ \text{sparseness}(\vec{x}) = 0 \quad &\Longleftrightarrow \quad |x_i| = \text{constant} \end{aligned}$$

Thus, the measure yields larger values the sparser the vector. This measure can be used to formalize the goal of *sparse NMF*:

$$\begin{aligned} \min \|X - DA\|_2^2 \quad \text{where} \quad & D, A \geq 0 \\ \text{and} \quad & \text{sparseness}(\vec{d^j}) = S_d \quad \forall\, j \\ & \text{sparseness}(\vec{a^i}) = S_a \quad \forall\, i \end{aligned}$$

for fixed constants S_d and S_a quantifying the sparsity.

The algorithm as proposed by Hoyer enlarges the gradient technique by an explicit step which enforces sparsity, resulting in the following iterative update:

init D, A,
repeat:
$D := D - \eta(DA - X)A^t$
enforce non negativity and sparsity of D
$A := A - \eta D^t(DA - X)$
enforce non negativity and sparsity of A

Here, $\eta > 0$ denotes the learning rate. While we already know how to enforce non negativity (simply setting negative values to 0), it is not clear how to enforce sparsity such that a fixed value S which equals S_a or S_d, respectively, is obtained. This problem is formalized in the following way:

- given non negative $\vec{x} \in \mathbb{R}^n$, find the closest vector $\vec{S} \in \mathbb{R}^n$ as measured in the L_2 norm with
  - only non negative coefficients $S_i \geq 0$,
  - a fixed L_2 norm: $\|\vec{S}\|_2 = L_2$ (which is usually chosen as the length of $\vec{x}$),
  - and a fixed L_1 norm $\|\vec{S}\|_1 = L_1$

Before spelling out an algorithm to solve this task, let's see why this actually enforces sparsity, i.e. small L_0 norm. For this fact, let's have a look at the definition of the sparsity measure which is equivalent to the form

$$\text{sparseness}(\vec{S}) = \frac{\sqrt{n} - \|\vec{S}\|_1/\|\vec{S}\|_2}{\sqrt{n} - 1}$$

Thus, keeping the L_2 norm fixed and minimizing the L_1 norm also emphasizes the sparsity. Albeit it is not always possible to arrive at exactly a desired sparsity, this can serve as a motivation to arrive at a given value for the related measure sparseness. Actually, since the L_0-norm is usually much more complex to deal with compared to the L_1-norm, this trick, substituting the former by the latter under constrained length, is a trick which is often encountered in this context. The following geometric routine can be used to arrive at an appropriate sparse vector $\vec{S}$:

project vector $\vec{x}$ onto the hyperplane with L_1-norm L_1:
$S_i := x_i + (L_1 - \sum_i x_i)/n$
$Z := \emptyset$ set of otherwise negative coefficients to be set to 0
repeat:
find closest point on the hyperplane with L_2-norm L_2 respecting Z:
$m_i := \begin{cases} 0 & \text{if } i \in Z \\ L_1/(n - |Z|) & \text{if } i \notin Z \end{cases}$
$\vec{S} := \vec{m} + \alpha(\vec{S} - \vec{m})$ where $\|\vec{S}\|_2 = L_2$
(α can be determined by solving the corresponding quadratic equation)
if $\vec{S} \geq 0$ return $\vec{S}$
else remove negative coefficients and project to hyperplane:
$Z := Z \cup \{i \mid S_i < 0\}$
$S_i := 0 \quad \forall\, i \in Z$
$c := (\sum S_i - L_1)/(n - |Z|)$
$S_i := \begin{cases} 0 & \text{if } i \in Z \\ S_i - c & \text{if } i \notin Z \end{cases}$

Note that it is sufficient to consider heuristics only for all steps, the NMF problem being bound to heuristics in the first place.

The effect of different degrees of sparsity is demonstrated in Fig.13 for images of faces, which seem to have a different internal characteristics as compared to the images used in the original approach from Lee and Seung, the latter yielding sparse representations without an explicit constraint.

An alternative scheme has been proposed by Park which is based on an extension of the alternating least squares algorithm to incorporate a sparsity constraint. The NMF factorization problem is reformulated in the form

$$\min_{A,D \geq 0} \|X - DA\|_2^2 + \eta \|D\|_2^2 + \beta (\sum_i |\vec{a}^i|_1)^2$$

with constant η and β enforcing limited norm of the base vectors (measured in Frobenius norm) and the sparsity of the coefficients. The optima with respect to A and D, respectively, can be reformulated in matrix form as

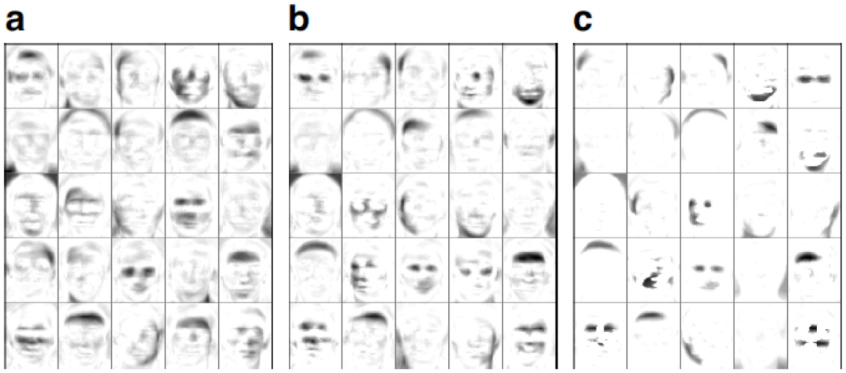
$$\min_{A \geq 0} \left\| \begin{pmatrix} D \\ \sqrt{\beta}\mathbf{1}_{1\times k} \end{pmatrix} A - \begin{pmatrix} X \\ \mathbf{0}_{1\times n} \end{pmatrix} \right\|_2^2$$

and

$$\min_{D \geq 0} \left\| \begin{pmatrix} A^t \\ \sqrt{\eta} I_k \end{pmatrix} D^t - \begin{pmatrix} X^t \\ \mathbf{0}_{k\times m} \end{pmatrix} \right\|_2^2$$

which can explicitly be solved as quadratic problem. The results shown in Fig. 11 are obtained this way.

NMF constitutes an active field of research. Hot topics are concerned with its applicability to big or streaming data sets, for example.



Features learned from the ORL face image database using NMF with sparseness constraints. When increasing the sparseness of the basis images, the representation switches from a global one (like the one given by standard NMF, cf Figure 1b) to a local one. Sparseness levels were set to **(a)** 0.5 **(b)** 0.6 **(c)** 0.75.

Figure 13: Effect of sparsity constraint on the found dictionary when applied to faces, the figure is taken from [6].

2.4 K-SVD

Unlike NMF, sparse coding in general deals with a data matrix X which also can have negative entries. Arbitrary possibly negative real number are acceptable in the decomposition $X \approx DA$. Hence, in this case, the requirement of sparsity is crucial to make the problem well posed and to emphasize semantically meaningful base vectors. K-SVD, which is dubbed k-singular value decomposition (kSVD) because of its similarity to k-means clustering, deals with the two problems to find optimum codes, and to find an an optimum dictionary.

OMP

We address the first problem with an algorithms typically referred to as *orthogonal matching pursuit* (OMP). OMP will be used as a subroutine for sparse coding later on. The task is to find optimum sparse coefficients given a fixed dictionary for all data $\vec{x}^1, \ldots, \vec{x}_N$. Actually, this can be solved independently for every vector $\vec{x} = \vec{x}^i$. The task is as follows

- find $\vec{a}$ such that $\|\vec{x} - \sum_j a_j \vec{d^j}\|_2^2 \leq \epsilon$ and $|\vec{a}|_0$ is minimum

The problem is NP hard, so approximations or heuristics are usually considered. We assume that the primitives of the dictionary are normalized

$$\|\vec{d^j}\|_2 = 1 \quad \forall\, j$$

If this is is not the case, normalization is easily possible.

OMP selects the entries a_j which are nonzero in a greedy approach: in every step, those coefficients are added which have maximum correlation to the remaining error and, hence, can reduce the error most provided everything else remains constant.

given a data vector $\vec{x}$, dictionary D with $\|\vec{d^j}\|_2 = 1$
init:
$I := \emptyset$, (set of nonzero indices)
$\vec{r} := \vec{x}$ (residual error)
repeat:
find $\arg\max_l |\vec{r}^t \vec{d^j}|$ (1)
set $I := I \cup \{l\}$;
compute $P := D_I (D_I^t D_I)^{-1} D_I^t$; (2)
update $\vec{r} = (\mathrm{Id} - P) \cdot \vec{x}$
until $\|\vec{r}\|_2 < \epsilon$ or other stopping criterion is fulfilled

What is the idea behind these steps?

(1) In this step, the base function is added which, keeping the old coefficients fixed, allows the largest decrease in the residual error. This can be seen as follows: Adding the base function $\vec{d^l}$ yields the remaining error $\|\vec{r} - a\vec{d^l}\|_2^2$. An optimum choice of a results from the derivative $2(\vec{d^l})^t(\vec{r} - a\vec{d^l}) = 0$ hence $a = \vec{r}^t\vec{d^j}$ respecting the normalization of $\vec{d^l}$. Then,

$$\|\vec{r} - a\vec{d^l}\|_2^2 = \|\vec{r} - (\vec{r}^t\vec{d^l})\vec{d^l}\|_2^2 = \|\vec{r}\|_2^2 - (\vec{r}^t\vec{d^l})^2$$

Thus, the largest decrease can be expected where the latter term $|\vec{r}^t\vec{d^l}|$ is maximum.

Note that this optimality constraint constitutes a compromise of efficiency and relevance since all coefficients are recomputed after adding $\vec{d^l}$. Hence, in theory, an alternative base function could yield to a smaller error if taking into account that all coefficients could be changed.

(2) In this step, optimum coefficients are computed if $\vec{d^l}$ is added to the set of base functions. I denotes all nonzero coefficients (including l), and D_I denotes the reduction of base functions to those with nonzero coefficients. $\vec{a}_I$ refers to the corresponding coefficient values. Then, taking the derivative of the costs $\|\vec{x} - D_I\vec{a}_I\|_2^2$ to zero yields $2D_I^t(\vec{x} - D_I\vec{a}_I) = 0$ hence $\vec{a}_I = (D_I^t D_I)^{-1} D_I^t \vec{x} =: P\vec{x}$ constitutes the standard solution by means of the pseudoinverse. The remaining residual error, after updating the coefficients accordingly, is $\vec{x} - (D_I^t D_I)^{-1} D_I^t \vec{x} = (\mathrm{Id} - P)\vec{x}$ with identity matrix Id.

The algorithm outputs the set of indices for nonzero coefficients I and the nonzero coefficients P. Due to the greedy approach, suboptimal sets can be obtained, but the algorithm is reasonably fast ($\mathcal{O}(k)$ provided the sparsity is fixed) with often good results.

Note that alternative stopping criteria are possible such as e.g. explicitly restricting the sparsity of $\vec{a}$ which results in a limit of the number of iterations.

One can show that an exact recovery of the coefficients is possible provided the correlation of the base vectors $\max_{i \neq j} \langle \vec{d^i}, \vec{d^j} \rangle$ is limited by $1/(2k-1)$ and the vector is sparse. This is also possible if noise is present.

K-SVD

This OMP step or any other subroutine addressing the same problem can be used as a subroutine in algorithms which retrieve an optimum dictionary D from a set of given data $\vec{x}^1, \ldots, \vec{x}^N$. The idea of k-SVD is very similar to k-means. In turn, optimum coefficients $\vec{a}^i$ are computed given a fixed dictionary (using OMP), and an optimum dictionary is computed given fixed coefficients $\vec{a}^i$, in particular given fixed choices of nonzero coefficients. The algorithm is as follows:

init D randomly
repeat until convergence:
    find optimum coefficients $\vec{a}^1, \ldots, \vec{a}^N$ using OMP for the problem:
        $\min_{\vec{a}^i} \|\vec{x}^i - D\vec{a}^i\|_2^2$ where $|\vec{a}^i|_0 \leq \text{const}$
    improve the base functions for these coefficients as follows:
        for $j = 1, \ldots, k$ repeat:
        determine $R_j := \{i \mid a_j^i \neq 0\}$
        compute residuals $E_j = \left(\vec{x}^i - \sum_{k \neq j} a_k^i \vec{d^k}\right)_{i \in R_j}$
        perform singular value decomposition $E_j = U \Delta V^t$
        adapt $\vec{d^j}$ = first column of U
        adapt a_j^i = first column of $V \cdot \triangle_{1,1}$

Let's recall: a singular value decomposition (SVD) decomposes a rectangular matrix M in a similar way as a PCA into three parts $M = U \Delta V^t$ with diagonal matrix Δ, matrix of left singular values U which are eigenvectors of $M^t M$ and matrix of right singular values V which are eigenvectors of $M M^t$. According to the Eckart-Young theorem, an SVD can be used to obtain the best low rank approximation of a matrix M in a least squares sense: $U \Delta_K V^t$ is the best rank K approximation of M, where Δ_K reduces to the first (largest) K coefficients and sets the other to zero.

As a consequence, we can easily interpret the rational behind the optimization step for the base functions of k-SVD: for all vectors $\vec{x}^i$ which actually use the base

function $\vec{d}^l$ the remaining error is computed which results from a representation that doesn't take $\vec{d}^l$ into account. Then, the best vector is chosen which reduces these remaining residual errors in a least squares sense as new vector $\vec{d}^l$. This corresponds to finding the best one-dimensional approximation of E_i. The resulting SVD yields the first column of V multiplied by $\Delta_{1,1}$ as coefficient vector and the first column of U as dictionary vector.

2.5 Applications

Due to the problem being NP hard, a variety of a algorithms have been proposed as alternatives which can give better results depending on the situation at hand. Before addressing alternative schemes we have a look at possible applications of sparse coding. Sparse coding constitutes a rapidly increasing research field in the last years, such that this short glimpse is necessary very rudimentary.

- *Image denoising:* The idea is that a given (noisy) image consists of the original data (one is interested in) overlaid with noise. The (preprocessed) image is decomposed into image patches of fixed size. These are then exchanged by their closest sparse representation with respect to an appropriate dictionary, which has been trained using patches from natural images. Due to this procedure, noise or artifacts which are not encoded in the dictionary are cancelled.

  Examples for this behavior can be seen in Figs. 14. In these cases, noise, scratches, and even image impainting can be treated based on sparse coding approches.

- *Discriminative dictionaries for learning specific edges:* In discriminative dictionaries, a number of different classes and representative vectors $\vec{x}$ for every class are given, and the task is to find a dictionary for every class which enables sparse encoding of data of its class as well as a good separation of the classes in the sense that a dictionary encodes only members of its own class in a sufficient way. The learning algorithms can be extended accordingly by specifying an appropriate cost function. This principle has been used to enhance classical edge detection mechanisms: based on the Canny filter, discriminative dictionaries based on correct versus incorrect edges allow a reduction of the potential edges as shown in Fig. 15 (top). Similarly, it is possible to train dictionaries on edges from typical objects, enabling an object specific filter as shown in Fig. 15 (bottom).

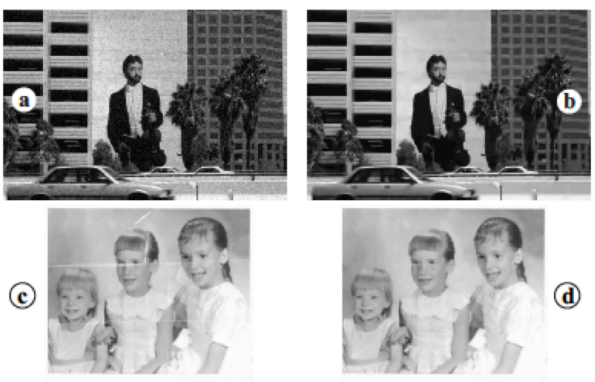


Figure 1. **Image reconstruction using a Field of Experts.** **(a)** Example image with additive Gaussian noise ($\sigma = 20$, PSNR $=$ 22.51dB). **(b)** Denoised image. (PSNR $= 28.79$dB). **(c)** Photograph with scratches. **(d)** Image inpainting using the FoE model.

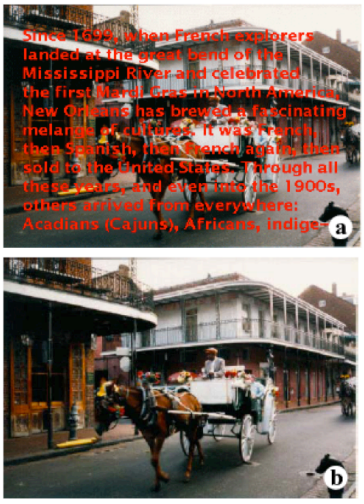


Figure 14: Image denoising or reduction of artifacts based on sparse coding approaches, the images are taken from [15].

2.6 Feature sign search

As one exemplary alternative to the popular k-SVD approach, we address the feature sign search algorithm as proposed in [14]. As before, the first problem which is addressed is the sparse representation of data given a fixed dictionary. For a given vector $\vec{x}$ and dictionary D, the mathematical objective is formalized as

$$\min_{\vec{a}} \|\vec{x} - D\vec{a}\| - 2^2 + \lambda\|\vec{a}\|_1$$

Note that the 0-norm is substituted by the 1-norm which can serve as a good approximation for the latter due to having a constant derivative, but being convex. One can even proove that in some circumstances, the two requirements actually coincide. The observation of the *feature sign search algorithm* is that if we knew the signs of $a_j \in \{-1, 0, 1\}$, the problem would be easy and could be solved analytically. Therefore, the algorithm repeatedly guesses and improves the signs and solves the resulting problem provided these signs are fixed.

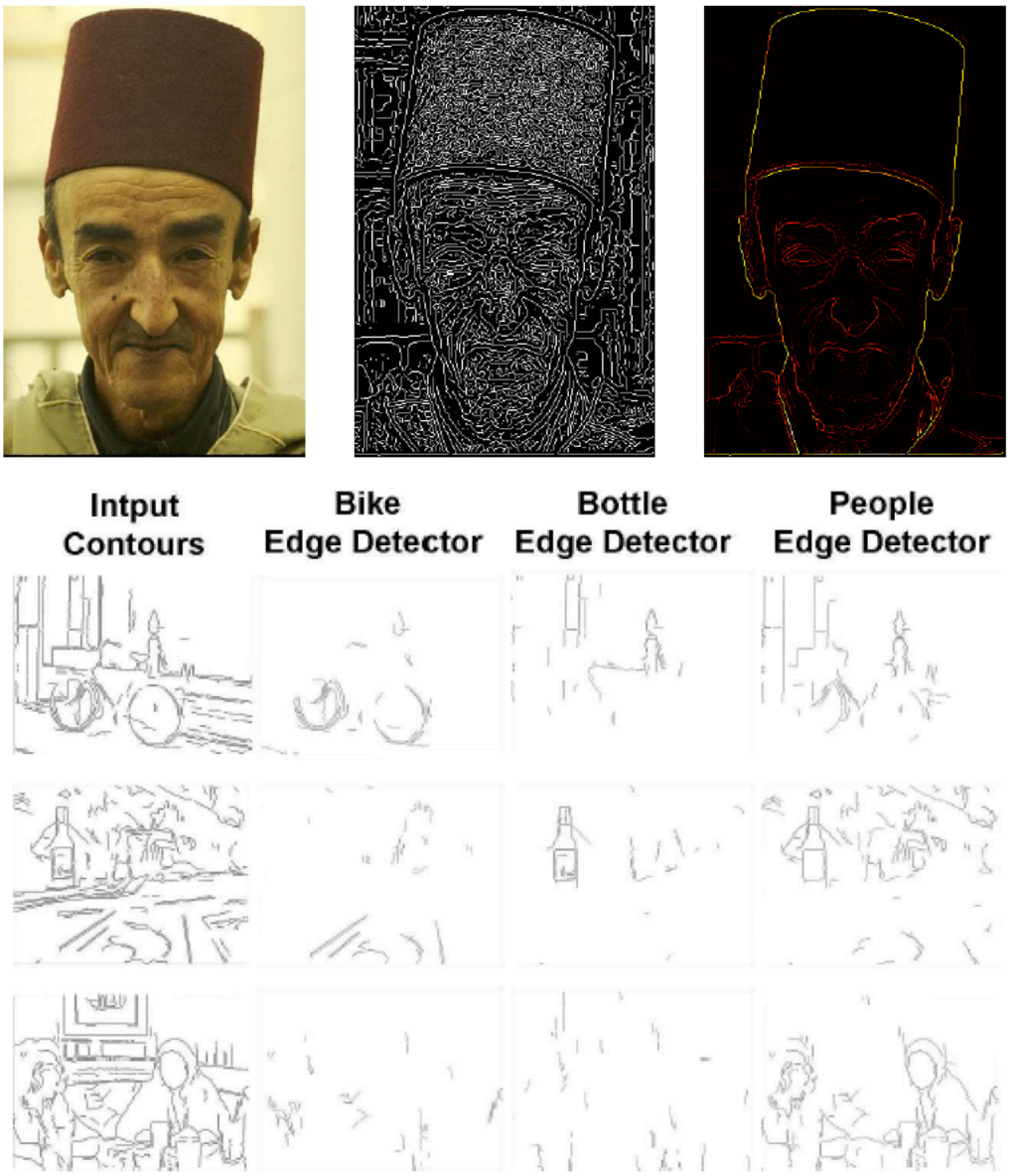


Figure 15: Detection of class specific edges based on discriminative sparse coding, to automatically extract certain entities from images, images are taken from [16].

(1) INIT:

$\vec{a} := 0$ (coefficient vector),

$\vec{\theta} := 0$ (sign vector),

$I := \emptyset$ (active set),

(2) UPDATE ACTIVE SET:

select $i \notin I$ with maximum $|(*)| := \left| \frac{\partial \|\vec{x} - D\vec{a}\|^2}{\partial \vec{d}^i} \right|$

if $(*) > \lambda$: $\theta_i := -1$, $I := I \cup \{i\}$

if $(*) < -\lambda$: $\theta_i := 1$,$I := I \cup \{i\}$

(3) UPDATE COEFFICIENTS:

D_I, $\vec{a}_I$, $\vec{\theta}_I$ denote the restrictions to I
find optimum coefficients of $\|\vec{x} - D_i\vec{a}_I\|_2^2 + \lambda\vec{\Theta}_I^t\vec{a}_I$:
these are $\hat{a}_I := (D_I^t D_I)^{-1}(D_I^t\vec{x} - \lambda\vec{\Theta}_I/2)$
pick all values on the line $[\vec{a}_I, \hat{a}_I]$ with coefficient 0
evaluate their objective value and pick minimum $\rightarrow$ new value $\vec{a}$
delete i with $a_i = 0$ from I
set $\vec{\theta}_i =$ sign of a_i

(4) CHECK FOR OPTIMALITY:

$i \in I$:
if $\frac{\partial|\vec{x}-D\vec{a}|^2}{\partial a_i} + \lambda\text{sign}(a_i) \neq 0$ GOTO (3)

$i \notin I$:
if $\left|\frac{\partial|\vec{x}-D\vec{a}|^2}{\partial a_i} > \lambda\right|$ GOTO (2)

This algorithm is guaranteed to converge to a global optimum of the costs because of the following reasons:

- the optimality check stops iff the point is optimum, since it tests
  - that the coefficients for the active set are in a local and hence global optimum of the costs
  - inclusion of coefficients not from the active set does not help
- the coefficient update (3) alone together with the first check in (4) is done only a finite number of steps since the active set is only decreased in this procedure, and unique coefficients are determined for every active set and sign vector in (3)
- (3) always decreases the cost function, since for the optimum $\hat{a}$ two situations only are possible:
  - no sign change, hence we have an optimum for the given sign vector
  - sign change, then, because of the convexity of the costs, for the first crossing point the value must be smaller, hence also for the optimum picked here
- (2) considers different active sets and sign conditions and always starts with an optimum coefficient vector respecting these sign conditions and active set. Hence it necessarily converges since there are only finitely many different conditions and the objective value is decreased by at least ϵ for every new index incorporated into I.

As a dual problem, given fixed coefficients, how to optimize a dictionary D? The problem can be formalied as

$$\min \frac{1}{1}\|X - DA\|_F^2 \text{ where } \|\vec{d^j}\|_2^2 \leq c$$

for some constant c restricting the length of the base vectors. The Lagrangian is given as

$$\frac{1}{2}(X - DA)^t(X - DA) + \sum_j \lambda_j(\|\vec{d^j}\|_2^2 - c)$$

which should be minimized w.r.t. D and maximized w.r.t. the Lagrange duals $\lambda - j \geq 0$. These problems are equivalent (referring e.g. to Slater's condition). Derivatives w.r.t. D yield $\Lambda D^t + (XA^t)^t AA^t D^t = 0$, i.e. $D^t = (AA^t + \Lambda)^{-1}(XA)^t)^t$ hence the Lagrangian becomes

$$\min_{\Lambda \geq 0} \frac{1}{2}(X^t X - XA^t(AA^t + \Lambda)^{-1}(XA^t)^t) - c\Lambda$$

for a diagonal matrix Λ for which the gradient with respect to λ_i can be easily computed as $(XA^t(AA^t + \Lambda)^{-1}e_i)^2 - c$. Hence optimization by means of a truncated gradient scheme is easily possible.

2.7 Further issues

Sparse coding constitutes an active field of research with novel approaches being proposed in the context of methods (such as e.g. hierarchical dictionaries, more robust learning algortihms, probabilistic versions), efficiency (e.g. in the context of big or streaming data), or applications (such as video noise reduction). In practice, sparse codes retrieved from example data carry the potential of more efficient and tailored encoding schemes which achieve a lower compression and, in particular, interpretable dictionaries. The latter offer novel approaches as concerns modern data analysis since they infer to some extent a meaning from data based on the universal principle of sparsity.

Literature

1. Sparse Coding with an Overcomplete Basis Set, Olshausen, Field, Vision Research 37(23), 1997

2. Emergence of single-cell receptive field properties by learning a sparse code for natural images, Olshausen, Field, Nature 381, 1996 (the classical paper proposing sparse coding in the context of machine learning and its biological inspiration)

3. Matlab code for Sparsenet is available at
http://redwood.berkeley.edu/bruno/sparsenet/

4. Lee, Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401, 1999

5. Lee, Seung, Algorithms for non-negative matrix factorization, NIPS 13, 2001

6. Hoyer, Non negative matrix factorization with sparsity constraint, JMLR 5, 2004

7. Berry, Browne, Langville, Pauca, Plemmons, Algorithms and Applications for Approximate Nonnegative Matrix Factorization, Computational Statistics and Data Analysis 2006

8. Kim, Park, Sparse NMF via alternating non-negativity-constrained least squares for microarray data analysis, Bioinformatics 23-12, 2007

9. Thurau, Kersting, Wahabzada, Bauckhage, Convex non-negative matrix factorization for massive datasets, Knowledge and Information Systems 29, 2011

10. Matlab-code for NMF from Haesun Park at
http://www.cc.gatech.edu/ hpark/nmfsoftware.php including alternating least squares solution and methods for large data sets.

11. Matlab code for NMF from Cichoci/Zdunek
http://www.bsp.brain.riken.jp/ICALAB/nmflab.html including information theoretic variants to arrive at better optima.

12. Cai/Wang, Orthogonal Matching Pursuit for Sparse Signal Decoding, IEEE Transactions on Information Theory 57, 2011

13. Aharon, Elad, Bruckstein, K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation, IEEE Transactions on Signal Processing 54, 2006

14. Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng, Efficient sparse coding algorithms , NIPS 19, 2007

15. Roth, Black, Fields of Experts, International Journal of Computer Vision, 2008

16. Mairal, Leordeanu, Bach, Hebert, Ponce, Discriminative sparse image models for class-specific edge detection and image interpretation, Computer Vision - ECCV 2008

17. Matlab code for k-SVD provided by Ron Rubinstein
http://www.cs.technion.ac.il/ ronrubin/software.html

18. Matlab code for the sparse coding using feature sign search from the NIPS paper by Lee et al.
http://ai.stanford.edu/ hllee/softwares/nips06-sparsecoding.htm

3 Compressed sensing

Compressive sampling or compressed sensing is concerned with the sampling of signals as occur in natural systems. The following central question is considered: how many measurements have to be taken to reliably reconstruct a given signal? In signal processing, it has long been a common assumption that samples have to be taken at least at twice the Nyquist rate supported by the famous Nyquist-Shannon sampling theorem which states that this is actually sufficient. Emmanuel Candes, Terence Tao and David Donoho observed that this rate is not always necessary; much fewer sensor measurements are sufficient for data provided two facts are given:

- data are sparse in some (known) base,
- measurement takes place in a way which is incoherent to the base for which a sparse signal representation holds.

This observation gave rise to a rapid development of sparse sensing algorithms which specify measurement and reconstruction protocol as well as possible base functions for representation and measurement. In addition, practical applications and devices have been developed concerning e.g. the single pixel camera from Rice, which is the base of commercial cameras in this realm, or improved protocols for magnetic resonance imaging for conventional devices shortening scanning sessions considerably.

3.1 Classical sensing

Assume a data point $\vec{x} \in \mathbb{R}^n$ is given; more generally, a continuous signal $x(t)$ could be present, but we will restrict to the discrete setting for simplicity. We think of signals such as images, audio signals, EEG, EKG, X-ray, MRI, etc. This signal should be measured, whereby possible bottlenecks occur as regards e.g. storage space, transmission efficiency of the measurement, costs or risks of the measurement. These bottlenecks require that the number of measurements taken be as small as possible. So the goal is as follows:

Measure the signal $\vec{x}$ in some way such that as few measurements as possible are taken and the signal can be reconstructed thereof.

What is a measurement? We assume that measurements are linear transforms of the data:

Definition 4 *A* measurement *of $\vec{x} \in \mathbb{R}^n$ is characterized by a vector $\vec{\varphi}_i \in \mathbb{R}^n$ giving the real value $y_i = \vec{\varphi}_i^t \vec{x}$. A measurement vector is characterized by a matrix*

$\Phi = (\vec{\varphi}_1, \ldots, \vec{\varphi}_m)$ *resulting in the vector* $\vec{y} = \Phi^t \vec{x}$*. We refer to* φ_i *as* measurement function *(albeit being a vector only for the discrete setting).*

Reconstruction *of the signal from the measurement refers to the problem of reconstructing the original signal* $\vec{x}$ *given the measurement* $\Phi^t \vec{x}$*.*

Examples for popular measurement functions include the following:

- Indicator functions, i.e. the matrix of unit vectors. For images, for example, this corresponds to one measurement being one pixel of the image, the full image being described by all pixels.
- Fourier transformations, giving rise to the vectors of Fourier coefficients of a signal. This is the background for MRI, for example.

Thus, the problem becomes as follows:

- Find measurement functions Φ and an efficient reconstruction algorithm for these functions such that
  - reasonable signals $\vec{x}$ can be reconstructed from the measurement $\Phi^t \vec{x}$ using the reconstruction algorithm,
  - the number of measurements m is as small as possible,

The Shannon-Nyquist theorem

The Shannon-Nyquist theorem offers a popular sufficient condition which sampling frequency allows the reconstruction of signals, where continuous signals in the Frequency domain are considered due to its relevance for typical signals in applications. The theorem states the following:

Theorem 5 *Assume a function* $x(t)$ *contains frequencies smaller than* B *only, i.e. the Fourier coefficients are zero:*

$$x(f) = \int x(t) \exp(-2\pi i f t) dt = 0 \textit{ for } |f| \geq B$$

Then sampling at a rate T *at most* $2B$ *(the* Nyquist rate*) is sufficient to reconstruct the signal.*

Denote the samples by $x(nT)$*, then the signal can be reconstructed as*

$$x(t) = \sum_n x(nT) \cdot \operatorname{sinc}((tT - nT)/T) \textit{ with } \operatorname{sinc}(x) = \sin(\pi x)/(\pi x)$$

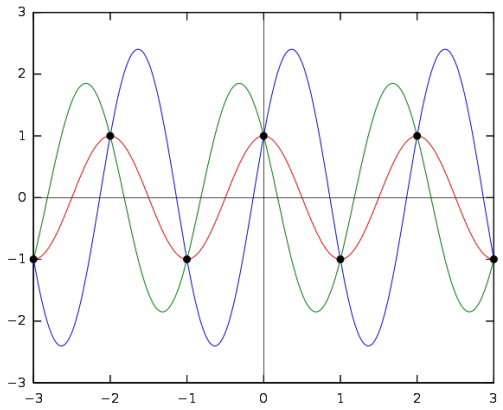


Figure 16: Different sinuses with the same frequencies and the same value at samples points where the sampling rate is lower than the Nyquist rate. Image taken from Wikipedia article on NyquistShannon sampling theorem, http://en.wikipedia.org/wiki/Nyquist-Shannon_sampling_theorem

If sampling is done at a lower frequency, then more than one underlying signal with frequencies at most B might exist which can explain the measurements, see Fig. 16. For finite vectors of dimensionality n, we can identify the entries with results at different equidistant time points. Hence we can think of arbitrary n-dimensional points as signals with frequencies smaller than $n/2$ (taking into account the phase shift of sin and cos, which account for two measurements per frequency), hence twice as many measurements, n, are sufficient (as expected).

Violations of the Nyquist sampling rate can frequently be observed in image manipulations as aliasing effects such as e.g. Moire patterns: high frequency images (due to e.g. high frequent regular structures such as textures) can display an aliasing effect when sampled at an insufficient rate. This can, for example, be the case when the camera resolution is not high enough; it can also be observed when zooming existing images, which effectively corresponds to a downsampling. High frequent components of the image might then occur in an inadequate way. Due to this fact, typically, software components first subtract high frequent components from the data before zooming, see Fig. 17.

3.2 Compressed sensing for sparse signals

The Shannon-Nyquist theorem gives a sufficient condition for the sampling rate, but it is not necessary in particular for signals which display additional characteristics. The key assumption of compressed sensing is, again, *sparsity* of the signals with respect to some fixed, known dictionary. We have already seen that natural

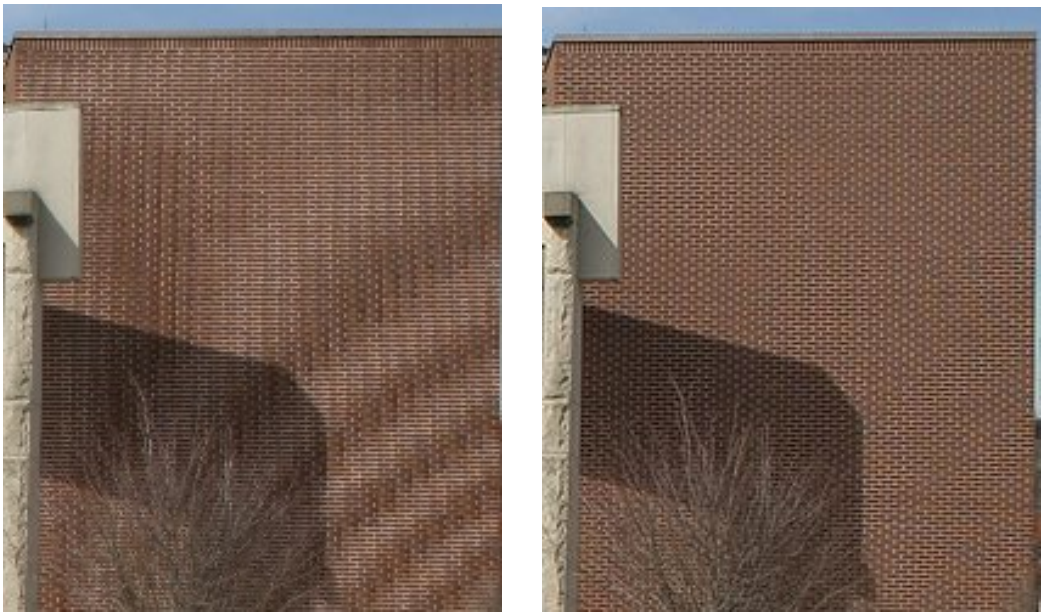


Figure 17: Moire pattern which results when zooming an image (left) without reducing the high frequent components first (right); this effect can be attributed to a violation of the Nyquist sampling theorem; http://en.wikipedia.org/wiki/Nyquist-Shannon_sampling_theorem

signals are often sparse with respect to some dictionary. We have even discussed techniques how to infer appropriate dictionaries from data. In this chapter, we assume a fixed dictionary D, which might be the output of some sparse coding scheme or which might be a classical base such as the Fourier base, wavelets, or similar.

Thus, we have the following setting:

Definition 6 *Assume a fixed dictionary D consisting of base functions $\vec{d}_1, \ldots, \vec{d}_k \in \mathbb{R}^n$ is given. Assume a signal $\vec{x} \in \mathbb{R}^n$ is measured which is* sparse in D*, i.e. $\vec{x} \approx D\vec{a}$ where $|\vec{a}|_0$ is small. Assume a measurement vector characterized by $\Phi = (\varphi_1, \ldots, \varphi_m)$ is given, leading to a measurement $\vec{y} = \Phi^t \vec{x}$.*

Sparse reconstruction *of $\vec{x}$ from the measurement $\vec{y}$ for a known dictionary D and known measurement vector Φ refers to the following algorithm:*

(I) find $\vec{a}$ with $\vec{y} \approx \Phi^t D\vec{a}$ and $|a|_0$ is small

(II) output $\vec{x} \approx D\vec{a}$

What is the ratio behind this procedure? We have the measurement $\vec{y} \approx \Phi^t \vec{x} \approx \Phi^t D\vec{a}$ because we know that the signal is sparse, i.e. $\vec{x} \approx D\vec{a}$. The idea is to

recover the coefficients $\vec{a}$ from this equality in **(I)** and then to reconstruct the signal from these sparse coefficients in **(II)**.

We already know algorithms for **(I)**: we can use OMP or feature sign search. If the base D is orthogonal, which is often the case for compressed sensing approaches, OMP even reduces to a very fast and exact algorithm. This gives us the basis algorithm for compressed sensing: assume we want to measure signals for which we know that they are sparse w.r.t. B. Then we measure using the measurement matrix Φ and reconstruct the signal $\vec{x}$ using sparse reconstruction.

The major question which remains is the following:

- Which measurement matrix Φ is suited for which dictionary D?
- How to choose the number m of measurement functions?
- Are there theoretical guarantees that this method actually works well?

In the following, we will always assume that both, the base vectors $\vec{d_i}$ are pairwise orthonormal and that the measurement functions φ_j are orthonormal, for simplicity. If not, an orthonormalization algorithm has to be used. We consider two settings: the situation of noise free signals and measurements, which allows us to elucidate the relation of suitable Φ and D , and the more realistic situation of signals with noise afterwards.

3.3 Base functions for compressed sensing – the noiseless case

The main quantity which allows us to judge the suitability of Φ and D is the following:

Definition 7 *The* coherence *of Φ and D is defined as the quantity*

$$\mu(\Phi, D) = \sqrt{n} \cdot \max_{i,j} |\langle \vec{\varphi}_i, \vec{dj} \rangle|$$

The coherence is always in the interval $[1, \sqrt{n}]$. The value $\sqrt{n}$ is assumed if $\Phi = D$, and the value 1 e.g. where D is the identity matrix and Φ is represented by distributed vectors with entries $\pm\sqrt{1/n}$. Intuitively, the coherence measures in how far the base functions are aligned to each other. Incoherent sets are dual in the sense that signals which have a sparse representation in D are distributed in Φ, i.e. few measurements already can capture the full information.

It can be seen that the incoherence indeed characterizes the necessary number of measurements, the necessary number of measurements scales as $\mathcal{O}(\log n)$ instead of n. More precisely, one can prove the following:

Theorem 8 *Assume $\vec{x} \in \mathbb{R}^n$ is a signal which is sparse in the orthonormal dictionary D, i.e. $|\vec{a}|_0 \leq S$ for some small S, and $\vec{x} = D\vec{a}$. Assume a measurement matrix Φ is given. Assume*

$$m \geq \text{const} \cdot \mu(\Phi, D)^2 \cdot S \cdot \log(n/\delta)$$

where m is the number of measurements, $\delta \in [0, 1]$. If measurement functions $\varphi_1, \ldots, \varphi_m$ are taken at random from the measurement matrix Φ, then with probability at least $1 - \delta$ the sparse reconstruction of $\vec{x}$ from the signal measured using these random measurement functions is correct. Further, already L_1 optimization is sufficient to recover the optimum coefficients $\vec{a}$ (i.e. we can optimize the smooth L_1 norm of the vector inbstead of the L_0 norm).

Here const is a fixed constant. The proof is contained in [3] and out of the scope of this lecture, the proof techniques relying on non-trivial statistical arguments and the proof being quite a few pages in length. We have a look at the simplest possible case to understand why this procedure is indeed reasonable: Assume D is the standard base of unit vectors and the signal $\vec{x}$ has sparsity $|\vec{a}|_0 = 1$ that means $\vec{x}$ is a multiple of a base vector $\lambda\vec{e}_k$. Assume Φ is an ON base which results from random vectors, in particular the entries are nonzero. Let's have a look at two different measurements using such random vectors: Obviously,

$$\vec{\varphi}_i^t \vec{x} = \lambda(\varphi_i)_k$$

for all i. On the other side, provided $|\vec{a}|_0 = 1$,

$$\vec{\varphi}_i^t D\vec{a} = a_{k'}(\varphi_i)_{k'}$$

for some k'. Hence, reconstruction from two measurements leads to the problem to find some k' corresponding to the nonzero entry of the coefficient vector $\vec{a}$ such that

$$\lambda \cdot (\varphi_i)_k = a_{k'} \cdot (\varphi_i)_{k'}$$
$$\lambda \cdot (\varphi_j)_k = a_{k'} \cdot (\varphi_j)_{k'}$$

Since $\varphi_i \neq \varphi_j$ with probability one (for smooth probability measures), this can only be fulfilled if we pick $k' = k$ and $a_k = \lambda$, i.e. we reconstruct the signal $\vec{x}$.

This very simple example shows the relevance of the incoherence: incoherent measurement functions capture the information present in the relevant base functions, no matter which coefficients a_k are actually nonzero. Provided the base functions and measurement functions would be corrrelated, this would not be the case: we could only observe the effects for one priorly fixed base function with one measurement.

Which orthonormal bases fulfil the requirement of incoherence? A few popular examples are the following:

- Fourier functions and the standard ON base have minimum coherence 1,
- wavelets and so-called noiselets have coherence $\approx \sqrt{2}$, and ON base D with a random ON base Φ has coherence $\sqrt{2 \log n}$ with high probability,
- in particular, it seems to be a good idea to choose D as Fourier base, wavelets, cosines, or similar, for which many natural signals are sparse, and the measurement vector Φ as random ON functions.

Note that it is crucial to choose D in such a way that the considered signals are sparse w.r.t D. Then, any measurements incoherent to D can be taken for compressed sensing. For random functions, an order of $S \cdot (\log n)^2$ measurements is then sufficient with high probability where S refers to the sparsity.

As a concrete example, let us consider the signal

$$f(t) = \sin(2.3 \cdot 2\pi \cdot t) - 0.5 \sin(5.4 \cdot 2\pi \cdot t)$$

which is sparse with respect to the Fourier base. An incoherent base to the Fourier base is the standard ON base. Thus, we take measurements at 10 random points, resulting in a vector $\vec{y} = (y_1, \ldots, y_{10})$. We could try to reconstruct the original signal in a classical (non sparse) way by minimizing the quadratic costs $\|\vec{y} - \Phi^t D\vec{a}\|_2^2$ or L_1 costs $\|\vec{y} - \Phi^t D\vec{a}\|_1$. Since both reconstructions do not aim at sparse reconstructions, the original signal is not recovered. For sparse recovery, we solve the problem of OMP: find $\vec{a}$ where $\|\vec{y} - \Phi^t D\vec{a}\|_2^2 \leq \epsilon$ and $|\vec{a}|_0$ is minimum (or $\|\vec{a}\|_1$ is minimum, which is the same in this case). This sparse recovery leads to the original signal $\vec{x}$, see Fig. 18.

3.4 Base functions for compressed sensing – the case of noise

In practice, this result is not directly applicable due to two facts: signals are usually only approximately sparse, i.e. many coefficients are close to 0, but not exactly 0. The measurement process can be subject to noise, i.e. $\vec{y}$ is only approximately equal to $\Phi^t \vec{x}$. It is possible to extend the result from the noise free case to the more realistic noisy one. Incoherence is substituted by a property of the matrix $\Phi^t D$, which brings compressed sensing close to the famous Johnson-Lindenstrauss theorem which is behind data processing techniques such as random projections.

Definition 9 *A matrix A satisfies the* restricted isometry property (RIP) *of order S if there exists a constant $\delta_S \in (0, 1)$ with*

$$(1 - \delta_S)\|\vec{a}\|_2^2 \leq \|A\vec{a}\|^2 \leq (1 + \delta_S)\|\vec{a}\|^2$$

holds for all $\vec{a}$ which are sparse with $|\vec{a}|_0 \leq S$.

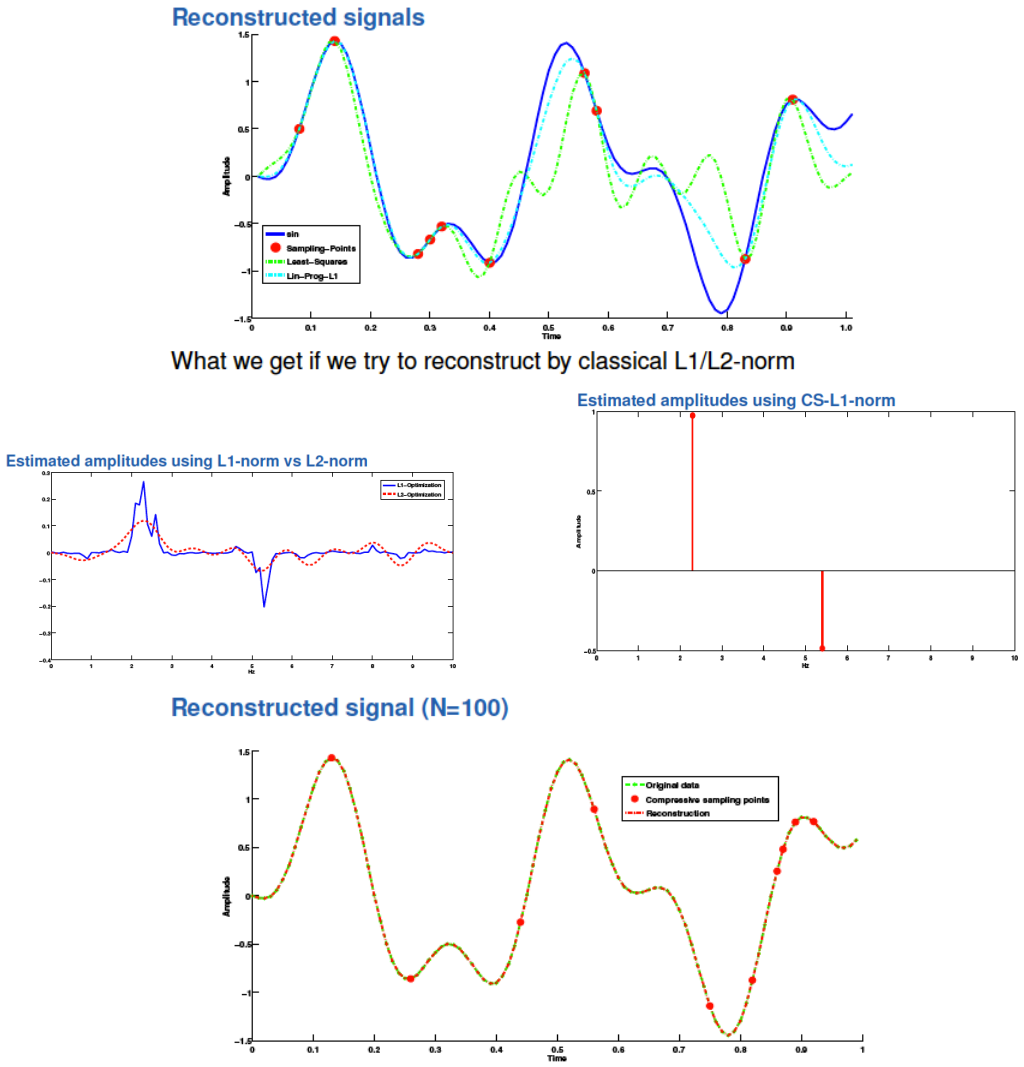


Figure 18: Reconstruction of a sparse signal from 10 measurement points: direct optimization using L_1 or L_2 norm fails in this case (top), which also manifests in the reconstructed coefficients $\vec{a}$ which are not sparse for these approaches (middle left). If using sparse reconstruction, the coefficients are retrieved (middle right), and the original signal can be reconstructed (bottom).

This means that the matrix A approximately preserves the norm of all sparse vectors $\vec{a}$. Since this holds for all sparse vectors, the information as concerns these vectors and ways to distinguish between different ones is preserved when projecting using A. Therefore, it is not surprising, that this property allows a reconstruc-

tion of the original signal. More precisely, the following holds:

Theorem 10 *Assume $\vec{y}$ is a noisy measurement of the signal $\vec{x}$, i.e. $\|\vec{y} - \Phi^t\vec{x}\|_2 \leq \epsilon$. Assume $\vec{a}^*$ is the solution of the problem*

$$\min_{\vec{a}} \|\vec{a}\|_1 \text{ such that } \|\vec{y} - \Phi^t D\vec{a}\|_2^2 \leq \epsilon^2$$

where the matrix $\Phi^t D$ satisfies RIP with $\delta_{2S} < \sqrt{2} - 1$. Then the original signal is recovered by $\vec{a}^*$ from $\vec{y}$ in the sense that

$$\|D\vec{a}^* - \vec{x}\|_2 \leq \text{const} \cdot \frac{\|\vec{x} - \vec{x}_S\|_2}{\sqrt{S}} + \text{const}' \cdot \epsilon$$

where $\vec{x}_S = D\vec{a}_S$ and $|\vec{a}_S|_0 \leq S$ constitutes a sparse coefficient vector.

This theorem limits the reconstruction error by a term which decomposes into the actual sparsity of $\vec{x}$ and the noise during the measurement, leading to worse reconstruction the more coefficients of $\vec{a}$ have a significant contribution and the noisier the measurement.

Which sensing/representation matrices fulfil RIP? We will get the same pairs as beforehand when considering incoherence. The proofs for popular choices actually closely mirror the Johnson-Lindenstrauss theorem which we mention for the sake of completeness:

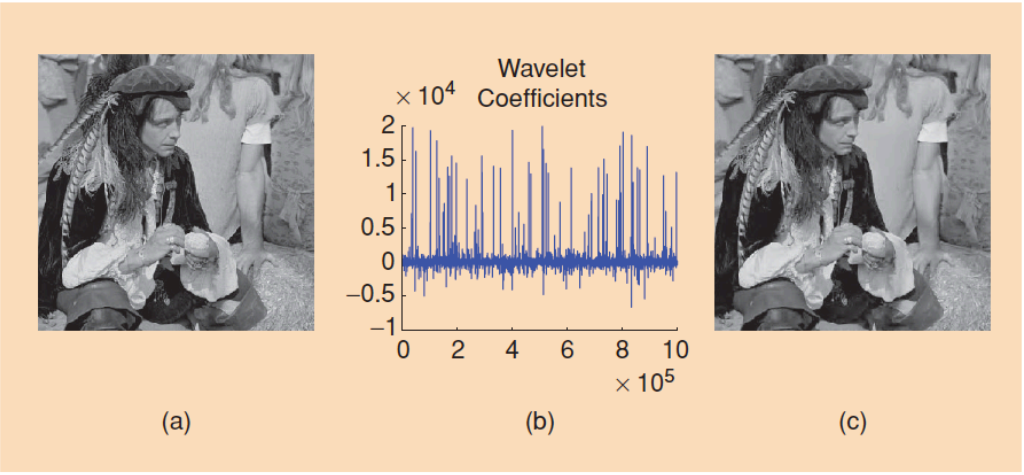
Theorem 11 *Given $\epsilon > 0$, points $Q \in \mathbb{R}^n$, $d > c \cdot (|Q|/\epsilon^2)$ (where c is a fixed constant which we do not specify here) then there exists a Lipschitz mapping $f : \mathbb{R}^n \to \mathbb{R}^d$ where*

$$(1 - \epsilon)\|\vec{u} - \vec{v}\|_2^2 \leq \|f(\vec{u}) - f(\vec{v})\|_2^2 \leq (1 + \epsilon)\|\vec{u} - \vec{v}\|_2^2 \quad \forall\, \vec{u}, \vec{v} \in Q$$

There exist constructive proofs for the theorem which give also a recipe how to choose f, random matrices A with $A_{ij} \sim \mathcal{N}(0, 1/d)$, random entries $\pm 1/\sqrt{d}$, or even sparse matrices with A_{ij} being 0 with probability $2/3$ and $\pm\sqrt{3/d}$ otherwise doing the job. This can be transferred to properties of $\Phi^t D$ which should equal e.g. one of the following to guarantee RIP with high probability:

- random column vectors on the unit sphere,
- i.i.d. entries $\sim \mathcal{N}(0, 1/m)$,
- i.i.d. entries $\pm\frac{1}{\sqrt{m}}$

where $m \geq \text{const} \cdot S \cdot \log(n/S)$. Concrete realizations for Φ and D are as follows:



[FIG1] (a) Original megapixel image with pixel values in the range [0,255] and (b) its wavelet transform coefficients (arranged in random order for enhanced visibility). Relatively few wavelet coefficients capture most of the signal energy; many such images are highly compressible. (c) The reconstruction obtained by zeroing out all the coefficients in the wavelet expansion but the 25,000 largest (pixel values are thresholded to the range [0,255]). The difference with the original picture is hardly noticeable. As we describe in "Undersampling and Sparse Signal Recovery," this image can be perfectly recovered from just 96,000 incoherent measurements.

Figure 19: Example application taken from [1]).

- Assume Φ and D are incoherent, then any m random vectors from Φ will do with high probability provided $m \geq \text{const} \cdot S \cdot (\log n)^4$.
- Choose any ON base D and random vectors Φ (with suitable probability) and $m \geq \text{const} \cdot S \cdot \log(n/S)$.

An example application for compressed sensing of an image approximated by a sparse represented by wavelets is shown in Fig. 19.

3.5 Applications

- The single pixel camera from Rice University relies on this principle to gather measurements which result from a transformation of the image by a multi-micromirror, which, mathematcially speaking, is nothing else but a random projection. Then comparably few such measurements are sufficient to capture the information in an actual image. See Fig. 20 for the hardware realization, based on which commercial versions have been developed. Fig. 21 displays an exemplary reconstruction result.

- Data compression of sparse data can be done also in the absence of the knowledge of the dictionary D. It is sufficient to encode few random projections of the data. Then, everybody who is in possession of D can decode these signals. This can be used for distributed encoding, for example.
- Sparse coding has been used to improve XRay or MRI gathering, since this reconstruction technique enables to shorten the sessions.

Literature

1. E.J. Candes, M.B. Wakin, An Introduction to Compressive Sampling, IEEE Transactions on Signal Processing 21, 2008
2. David Donoho, Compressed sensing. IEEE Trans. on Information Theory, 52(4), pp. 1289 - 1306, April 2006
3. E.Candes, J. Romberg, Sparsity and incoherence in compressive sampling, Inverse Prob. 23, 3,969-985, 2007.
4. Marco Duarte, Mark Davenport, Dharmpal Takhar, Jason Laska, Ting Sun, Kevin Kelly, and Richard Baraniuk, Single-pixel imaging via compressive sampling. (IEEE Signal Processing Magazine, 25(2), pp. 83 - 91, March 2008)
5. Emmanuel Candes, Justin Romberg, and Terence Tao, Stable signal recovery from incomplete and inaccurate measurements. (Communications on Pure and Applied Mathematics, 59(8), pp. 1207-1223, August 2006)
6. Richard Baraniuk, Mark Davenport, Ronald DeVore, Michael Wakin, A Simple Proof of the Restricted Isometry Property for Random Matrices, Constructive Approximation December 2008, Volume 28, Issue 3, pp 253-263
7. Overview about literature at http://dsp.rice.edu/cs also link to projects at Rice University

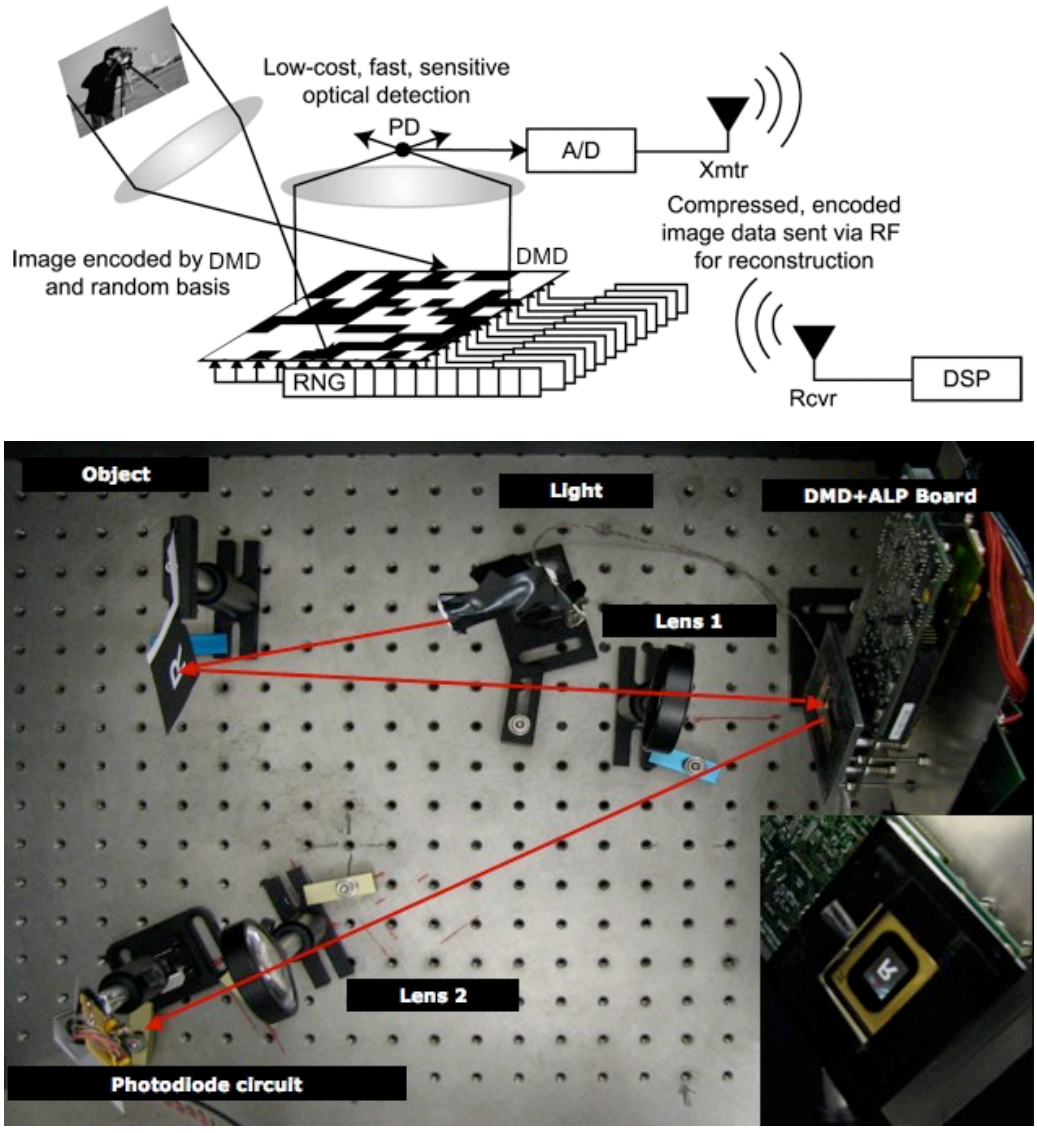


Figure 20: Hardware of the one-pixel camera which gathers pixels which correspond to a random projection of the observed scene [7]).

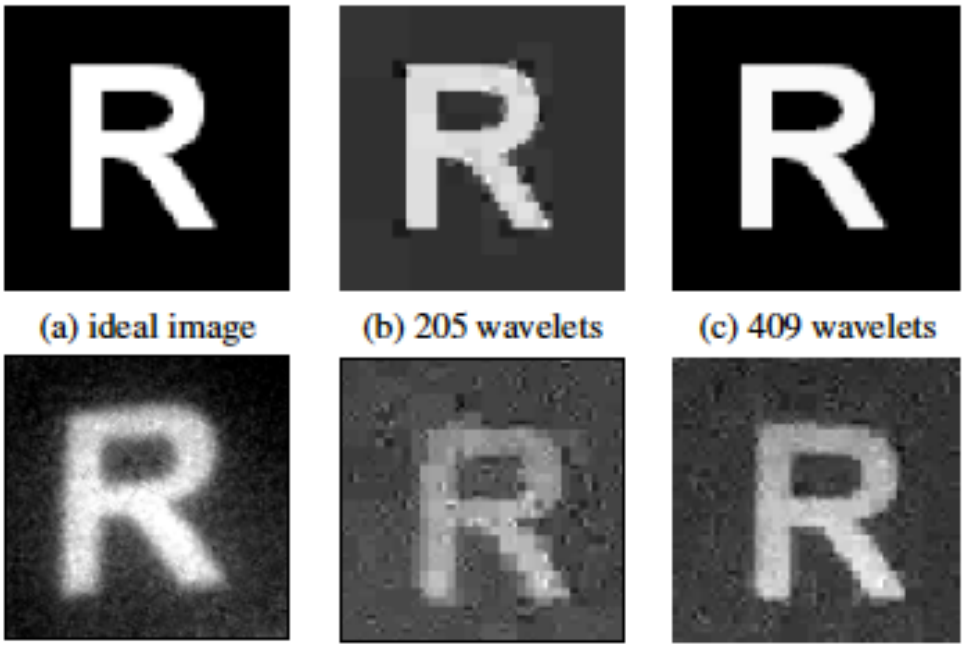


Fig. 2. *CI DMD imaging of a* 64×64 *(*$N = 4096$ *pixel) image. Ideal image (a) of full resolution and approximated by its (b) largest 400 wavelet coefficients and (c) largest 675 wavelet coefficients. (d) Conventional* 320×240 *camera image acquired at the DMD plane. CS reconstruction from (e) 1600 random measurements and (f) 2700 random measurements. In all cases, Haar wavelets were used for approximation or reconstruction.*

Figure 21: Example reconstruction of an image of the latter R [7]).

4 Support feature machine

In machine learning and data analysis, there exist many different classification and regression techniques enabling data analysts to infer a function or classification prescription from given examples only. One of the most popular formalisms is currently offered by the support vector machine (SVM), with many different variants being available for classification, regression, outlier detection, etc.

The key idea of SVM is to accompany the task to find a discriminative function by a powerful regularization scheme such that the generalization ability is usually excellent: the large margin trick, which prefers solutions where data have a large distance to the decision boundary. The idea of the support feature machine is to substitute this regularization scheme by an alternative one, obtaining the same excellent generalization but more interpretable classifiers: the goal is to find the sparsest discriminative prescription for the data where sparsity refers to the number of input features which are actually used.

This alternative regularization scheme turns out to be particularly useful in classification scenarios as often observed e.g. in modern biomedical application scenarios: here, the number of features often by far outnumbers the size of the given data set. Examples are e.g. the classification of microarrays, spectral data such as mass spectra or hyperspectral images, MRI images, EEG, etc. Hence the problem of finding an adequate classification is highly ill-posed and regularization is central to obtain valid results. The SVM imposes a large margin such that the resulting classifier has good generalization ability, but the obtained model is usually not interpretable at all, and applicants do not have any insight whatsoever on the ratio why a certain decision is taken by SVM. The SFM substitutes the margin requirement by a sparsity as concerns the number of used input dimensions. This way, a sparse and usually also interpretable model results where applicants can directly judge which input dimensions are relevant for the learned model.

4.1 The linear SVM

Assume data $\vec{x}_i \in \mathbb{R}^n$ together with class labels $y_i \in \{-1, 1\}$ are given. The task is to find a linear function

$$\begin{aligned} f\colon \mathbb{R}^n &\to \mathbb{R} \\ \vec{x} &\mapsto \vec{w}^t \vec{x} \end{aligned}$$

such that

$$f(\vec{x}_i) \cdot y_i > 0 \quad \forall\, i$$

i.e. the output class should coincide with the fact on which side of the hyperplane defined by $\vec{w}$ the point $\vec{x}_i$ is located. Here, we assume that the separating hyper-

plane goes through the origin, extending inputs by a constant 1 if this is not the case. The standard SVM formulation is then given as

$$\min_{\vec{w}\in\mathbb{R}^n} \|\vec{w}\|_2^2 \text{ such that } \vec{w}^t\vec{x}_i y_i \geq 1 \quad \forall i$$

where, after scaling, we can always assume a margin of at least 1. It is well known that this constitutes a convex problem for which efficient quadratic solvers can be used e.g. by means of its dual formulation. The output, however, will usually consist of a vector $\vec{w}$ with distributed entries, so interpretability of the resulting model is not clear.

4.2 Sparsity constraint for the linear SVM

The support feature machine aims at the solution which uses the smallest number of features instead, formalized as

$$\min_{\vec{w}\in\mathbb{R}^n} |\vec{w}|_0 \text{ such that } \vec{w}^t\vec{x}_i y_i \geq 1 \quad \forall\, i$$

Hence the large margin of SVM is traded for a sparse vector for SFM. We can expect a good generalization ability from this regularization, but unlike SVM, the result is interpretable.

Approximation

However, there is a price to pay: Unlike the former, this problem is NP hard, and even approximations within large constants are unlikely to exist. Therefore approximations or heuristics are necessary. The 0-norm is numerically particularly tricky due to the fact that it is discontinuous and partially constant. We consider the following approximation by the 1-norm, which is very common (remember the results for sparse coding, for example, where the 1-norm can provably be substituted for the 0-norm). Then we obtain the problem:

$$\min_{\vec{w}\in\mathbb{R}^n} \left(\sum_{j=1}^{n} \ln(\epsilon + |w_j|) \right) \text{ such that } \vec{w}^t\vec{x}_i y_i \geq 1 \quad \forall\, i$$

where $0 < \epsilon < 1$.

How good is this approximation? Assume $\vec{w}_0$ is a solution of the original

problem, $\vec{w}_l = \vec{w}_l(\epsilon)$ a solution of the approximation. Then we find

$$\begin{aligned}
& \sum_j \ln(\epsilon + |(w_l)_j|) \;\leq\; \sum_j \ln(\epsilon + |(w_0)_j|) \\
\iff & \sum_{(w_l)_j=0} \ln(\epsilon) + \sum_{(w_l)_j\neq 0} \ln(\epsilon + |(w_l)_j|) \;\leq\; \sum_{(w_0)j=0} \ln(\epsilon) + \sum_{(w_0)_j\neq 0} \ln(\epsilon + |(w_0)_j|) \\
\iff & (n - |\vec{w}_l|_0) \ln \epsilon + \sum_{(w_l)_j\neq 0} \ln(\epsilon + |(w_l)_j|) \;\leq\; (n - |\vec{w}_0|_0) \ln \epsilon + \sum_{(w_0)_j\neq 0} \ln(\epsilon + |(w_0)_j|)
\end{aligned}$$

which is, after some further calculations, equivalent to

$$|\vec{w}_l|_0 \;\leq\; |\vec{w}_0|_0 + \left|\frac{1}{\ln \epsilon}\right| \cdot \left(- \sum_{(w_0)_j\neq 0} \ln(\epsilon + |(w_0)_j|) + \sum_{(w_l)_j\neq 0} \ln(\epsilon + |(w_l)_j|) \right)$$

Since we can always limit the size of possible solutions in the 1-norm depending on the data only, the right hand sight is of order $\mathcal{O}(1/|\ln \epsilon|)$, i.e. we recover the original problem for small ϵ.

The approximation is still not easy to solve, giving rise to possible local optima. But unlike the 0-norm, we have now a continuous cost function.

Frank-Wolfe Algorithm

For optimization of the costs, a general method, the so-called Frank-Wolfe technique is used.

Definition 12 *Assume a problem of the form*

$$\min_{\vec{x}} f(\vec{x}) \textit{ such that } \vec{x} \in P$$

is given with P specifying a polygonal region. Then the Frank-Wolfe method consists of the following algorithm:

init $\vec{x}_0 \in P$
repeat
  $\min f(\vec{x}_k) + \nabla f(\vec{x}_k)^t(\vec{x} - \vec{x}_k)$ *such that* $\vec{x} \in P$
  this yields optimum $\vec{x}_{\text{opt}}$
  adapt $\vec{x}_{k+1} = \vec{x}_k + \alpha_k(\vec{x}_{\text{opt}} - \vec{x}_k)$ *where* α_k *is optimized using line search*

We denote $z_k(\vec{x}) := \nabla f(\vec{x}_k)^t(\vec{x} - \vec{x}_k)$ for short. Note that $f(\vec{x}_k)$ is a constant, so we could also delete it. What is the rationale behind this technique? For convex f, it converges to a global optimum, otherwise only a local one is found, whereby

convergence is of order $\mathcal{O}(1/k)$ in general. The idea of the method is to first approximate the problem by a Taylor approximation of first order at the current point. Then, taking into account the linear constraints, the minimization problem is a classical linear optimization problem where a simplex solver or any other LP solver can provide a solution. Since the found boundary point is not necessarily optimum for the original (nonlinear) problem, a line search determines the optimum in the direction of this candidate.

One can actually show that the search direction $\vec{x}_{\text{opt}} - \vec{x}_k$ is always a descent direction which, per construction, stays in the feasible region. Consider

$$\nabla f(\vec{x}_k)^t(\vec{x}_{\text{opt}} - \vec{x}_k) = z_k(\vec{x}_{\text{opt}}) \leq z_k(\vec{x}_k) = 0$$

This holds since an optimum is obtained in $\vec{x}_{\text{opt}}$. This implies that either

- $z_k(\vec{x}_{\text{opt}}) = 0 \leq z_k(\vec{x})$ for all $\vec{x} \in P$ since it is optimum. In this case, we have $\nabla f(\vec{x}_{\text{opt}})^t(\vec{x} - \vec{x}_k) \geq 0$ for all $\vec{x}$, i.e. $\vec{x}_{\text{opt}}$ is a local optimum (at the boundary).
- Alternatively, $0 > z_k(\vec{x}_{\text{opt}}) = \nabla f(\vec{x}_{\text{opt}})^t(\vec{x} - \vec{x}_k)$ and hence the search directions correlates to the negative gradient, i.e. it improves the costs.

Support feature machine

For the SFM, the Frank-Wolfe algorithm is used. For this purpose, we rephrase the problem as follows

$$\min_{\vec{w}} \left(\sum_{j=1}^{n} \ln(\epsilon + |w_j|) \right) \text{ such that } y_i \vec{w}^t \vec{x}_i \geq 1$$

This is equivalent to

$$\min_{\vec{v},\vec{w}} \left(\sum_{j=1}^{n} \ln(\epsilon + v_j) =: g(\vec{v}) \right) \text{ such that } y_i \vec{w}^t \vec{x}_i \geq 1 \text{ and } v_j \geq w_j, v_j \geq -w_j$$

The Frank-Wolfe method yields

init $\vec{v}_0 = (1, \ldots, 1) = \vec{w}_0$
repeat
  find solution for:
  $\min_{\vec{v},\vec{w}}$ $g(\vec{v}_k) + \nabla g(\vec{v}_k)^t(\vec{v} - \vec{v}_k)$
  such that $y_i \vec{w}^t \vec{x}_i \geq 1$
  $v_j \geq w_j, v_j \geq -w_j$
  this yields $\vec{v}_{\text{opt}}$, $\vec{w}_{\text{opt}}$
  compute optimum $g(\vec{v}_k + \alpha(\vec{v}_{\text{opt}} - \vec{v}_k))$
  set $\vec{w}_{k+1} = \vec{w}_k + \alpha(\vec{w}_{\text{opt}} - \vec{w}_k)$

Considering the gradient of g yields

$$\nabla g(\vec{v}_k) \cdot \vec{v} = \sum_{j=1}^{n} \frac{v_j}{(v_k)_j}$$

assuming $\epsilon \approx 0$, hence we have the problem

$$\begin{aligned} \min_{\vec{v},\vec{w}} \quad & \sum_{j=1}^{n} \frac{v_j - (v_k)_j}{(v_k)_j} \\ \text{such that} \quad & y_i \vec{w}^t \vec{x}_i \geq 1 \\ & v_j \geq w_j, v_j \geq -w_j \end{aligned}$$

Setting $v'_j := v_j/(v_k)_j$ we obtain, since $\sum (v_k)_j$ is constant for the optimization,

$$\begin{aligned} \min_{\vec{v}',\vec{w}} \quad & \sum_j v'_j \\ \text{such that} \quad & y_i \vec{w}^t \vec{x}_i \geq 1 \\ & v'_j \geq w_j/(v_k)_j, v'_j \geq -w_j/(v_k)_j \end{aligned}$$

Setting $w'_j = w_j/(w_k)_j$, we obtain

$$\begin{aligned} \min_{\vec{v}',\vec{w}} \quad & \sum_j v'_j \\ \text{such that} \quad & y_i (\vec{w}')^t (\vec{x}_i * \vec{w}_k) \geq 1 \\ & v'_j \geq w'_j, v'_j \geq -w'_j \end{aligned}$$

where $\vec{x}_i * \vec{w}_k$ refers to a component-wise multiplication. v'_j necessarily yields the absolute value of w'_j, hence naming the latter w_j, we get:

$$\begin{aligned} & \min_{\vec{w}} \sum_j |w_j| \text{ such that } y_i \vec{w}^t (\vec{x}_i * \vec{w}_k) \geq 1 \\ & \text{set } \vec{w}_{k+1} = \vec{w}_k * \vec{w}_{\text{opt}} \end{aligned}$$

The latter is valid since $\sum \ln |w_j|$ is a concave function, hence optima of the line from $\vec{w}_k$ to $\vec{w}_{\text{opt}}$ are at an end point. Thus, we have the optimum paramter $\alpha = 1$ which, taking into account the rescaling of $\vec{w}$ in the optimization problem, yields to the latter update. In summary, the following algorithm results

init $\vec{w}_0 = (1, \ldots, 1)$
repeat
    find solution for:
    $\min_{\vec{w}} \sum_j |w_j|$ such that $y_i \vec{w}^t (\vec{x}_i * \vec{w}_k) \geq 1$
    set $\vec{w}_{k+1} := \vec{w}_k * \vec{w}_{\text{opt}}$

It is possible to include an explicit bias b in the formalization, which we simply ignored and simulate as additional feature 1 of the data if necessary. Note, however, that this simulation can give slightly different results.

An alternative formalization

This formalization has recently been slightly rephrased in [3], based on the observation that the demand of a margin at least 1 for all points is possibly too strong. Thus, this demand is weakened, and an additional normalization is imposed instead to prevent trivial solutions $\vec{w} = 0$, as follows:

$$\begin{aligned} \min_{\vec{w}} \quad & \sum_j \ln(\epsilon + |w_j|) \\ \text{such that} \quad & y_i \vec{w}^t \vec{x}_i \geq 0 \\ \vec{w}^t(\mu^+ - \mu^-) = 1 & \end{aligned}$$

where

$$\mu^+ = \sum_{y_i=1} \vec{x}_i / \sum_{y_i=1} 1 \text{ and } \mu^- = \sum_{y_i=-1} \vec{x}_i / \sum_{y_i=-1} 1$$

refers to the centres of the two classes. This latter demand is a normalization fixing a certain correlation to the line in between the class centres. Similarly as before, this leads to a Frank-Wolfe algorithm of the form

init $\vec{w}_0 = (1, \ldots, 1)$
repeat
    find solution for:
    $\min_{\vec{w}} \sum_j |w_j|$
    such that $y_i \vec{w}^t (\vec{x}_i * \vec{w}_k) \geq 0$ and $\vec{w}^t(\mu^+ - \mu^-) = 1$
    set $\vec{w}_{k+1} := \vec{w}_k * \vec{w}_{\text{opt}}$

In this realm, extended formalizations for two settings have also been proposed: the case of misclassifications, realized in the standard way by introducing slack variables; and the case of imbalanced data, where different costs can be introduced for errors of the two classes.

Fig. 22 shows a comparison of the ability of the techniques to find the relevant features.

5 Exemplary application

Applications of SFM are in classification tasks where the number of input features exceeds the number of given examples, and an interpretability of the result in

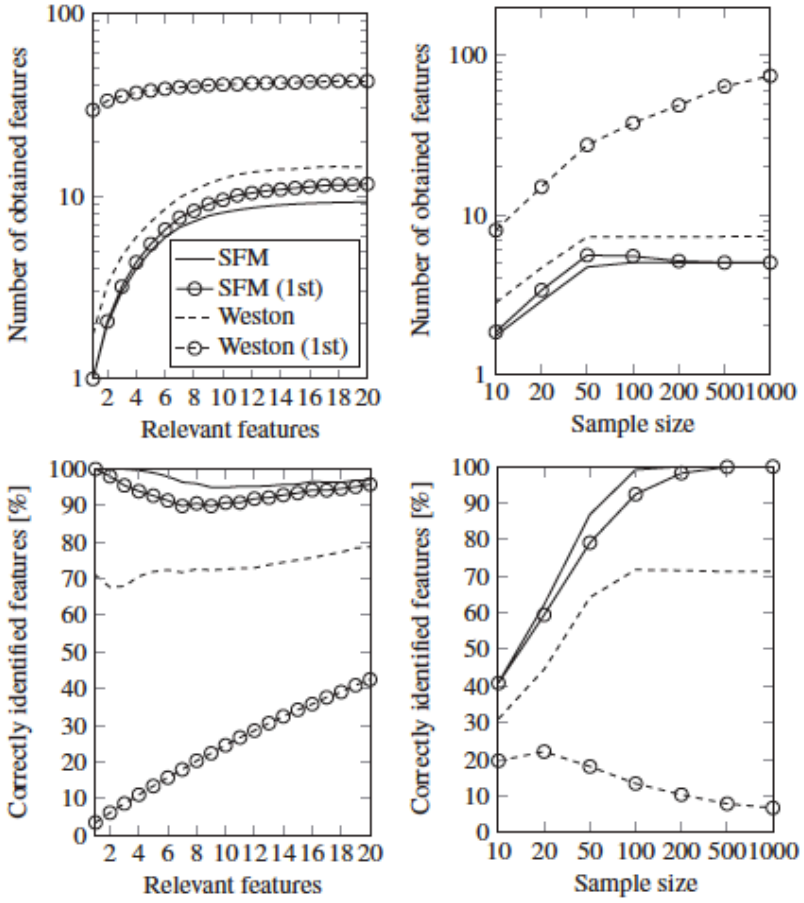


Figure 22: Results of different SFM formalizations for artificial data where ground truth is available, image taken from [3].

terms of relevant features is beneficial. These settings occur typically in areas such as

- Microarray data: classes investigated in an experiment such as diseases or conditions are characterized by potential genes as measured in microarrays in a highly parallel fashion. It is vital for further biomedical analysis to identify potentially relevant genes.
- Spectral data characterize (mixtures of) substances by means of an intensity profile for a range of wavelengths. Identifying the the most relevant wavelength can allow to infer the most relevant discriminative ingredients which correspond to the found features.

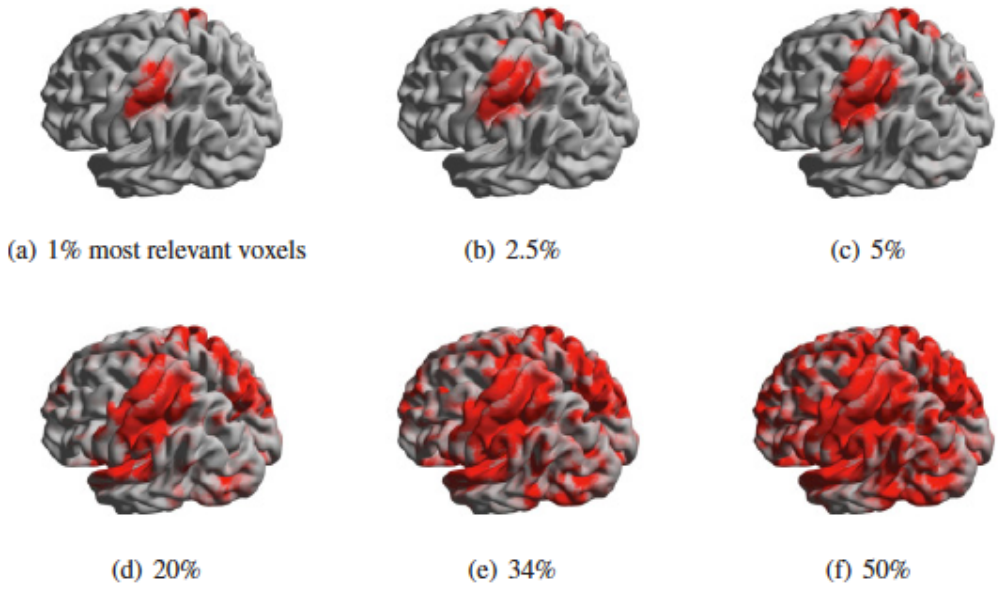


Figure 23: Voxels identified as most relevant in an fMRI study, image taken from [3].

- FMRI studies constitute a popular non invasive technique to gather a snapshot of brain activity while performing certain tasks. The identification of the most relevant voxels carries potential of giving hints about how a certain task is actually solved by humans. An example for such brain imaging is found in Fig. 23. Subjects were asked to press a button with the left/right after seeing a happy/sad tag, and, with some delay, a go-signal.

5.1 A general remark on shrinkage

Finding sparse models is not only a topic in classification, but also standard regression models. Assume data $\vec{x}_1, \ldots, \vec{x}_m \in \mathbb{R}^n$ are accompanied by real valued outputs $y_i \in \mathbb{R}$. One of the most popular regression models for linear regression is the classical ridge regression which optimizes the costs

$$E(\vec{w}) = \frac{1}{2} \sum_{i=1}^{m} |y_i - \vec{w}^t \vec{x}_i|^2 + \beta \cdot \sum_{j=1}^{n} w_j^2$$

with explicit analytical solution by the pseudoinverse $\vec{w} = (X^t X + \beta I)^{-1} X^t Y$ where X and Y refer to the matrix/vector of data and outputs, respectively. The

penalty term ensures that the actual estimate of all data is shrunken towards the average, instead of a direct minimization of the squared error only.

The great benefit of ridge regression is its explicit analytical model (and the fact of a unique solution), but results are not necessarily interpretable because the model is usually not sparse. The penalty $\sum_j w_j^2$ accounts for shrinkage of the entries, but without actually enforcing the entries to exactly become 0. Therefore, alternatives have been proposed, such as the Lasso regression

$$E(\vec{w}) = \frac{1}{2}\sum_{i=1}^{m} |y_i - \vec{w}^t\vec{x}_i|^2 + \beta \cdot \sum_{j=1}^{n} |w_j|$$

or smooth combinations of both via the so-called elastic net. Lasso has the benefit of enforcing exact 0 for the irrelevant components, but convex optimization techniques are necessary for its solution.

We do not have a look at sparse regression techniques in this context, see e.g. [5] for an overview, but we want to highlight a nice statistical motivation why shrinkage might actually give more accurate results than averaging. This phenomenon is known in the statistical literature as the Stein paradox [4]. Assume you observe a discrete event such as the batting ability of baseball players. How to estimate the average success rate? For one player, the best you can do is to estimate the average batting ability (probability to bat) by the number of successes normalized by the number of trials. We refer to this quantity as y. What is the best you can do if you simultaneously estimate the betting ability of two (independent) players? Again, the best (in a precise statistical sense) you can do is to estimate both abilities with its respective average.

And now comes the Stein paradox: starting from three independent (and possibly even entirely unrelated) events, you can do better than taking simply the single averages! The James-Stein estimator predicts the value

$$\left(1 - \frac{(m-2)\sigma^2}{|y-\bar{y}|^2}\right) \cdot (y - \bar{y}) + \bar{y}$$

instead of y, where $\bar{y}$ is the mean value of all y and σ its standard deviation. This formula implies that the average y is shrunken towards the global average $\bar{y}$ for every player whereby the degree of shrinkage depends on the standard deviation. One can show that this estimator is better than the individual mean when considering the overall error (albeit for each individual player, the average might be better, the overall squared error is smaller for the James-Stein estimator in an exact statistical sense). An example of this fact is shown in Fig. 24 when estimating the batting ability of baseball players from the first 45 trials. Thereby, the real batting ability is taken as average number of hits for the remaining season. Thus, this is

an exact statistical result which proofs that, on average, shrinkage of the values helps.

This comes somewhat as a paradox, since the individual events are not necessarily related to each other, i.e. we could combine at least three entirely unrelated events and get, on average, a better estimator than the individual means by shrinkage towards the overall mean value!

It is known that also the James-Stein estimator is not optimum (taking an absolute value for the shrinkage size yielding a better, though still not optimum estimator. An optimum estimator for the global least squares error is currently not known.

Literature

- Nello Cristianini, John Shawe-Taylor: Kernel Methods for Pattern Analysis, Cambridge University Press, Cambridge, 2004
- J. Weston, A. Elisseeff, B. Schölkopf, M. Tipping, Use of the zero-norm with linear models and kernel methods, JMLR 3, 1439-1461, 2003
- S. Klement, S. Anders, T. Martinetz, The support feature machine: classification with the least number of features and application to neuroimaging data, Neural Computation 25, 1548-1584, 2013
- B. Efron, C. Morris, Stein's Paradox in Statistics, Scientific American, 236(5): 119-127, 1977.
- Fast sparse regression and classification, Jerome H. Friedman, International Journal of Forecasting, 2012, vol. 28, issue 3, pages 722-738
- Matlab code for SFM is available at http://www.inb.uni-luebeck.de/tools-demos/support-feature-machine

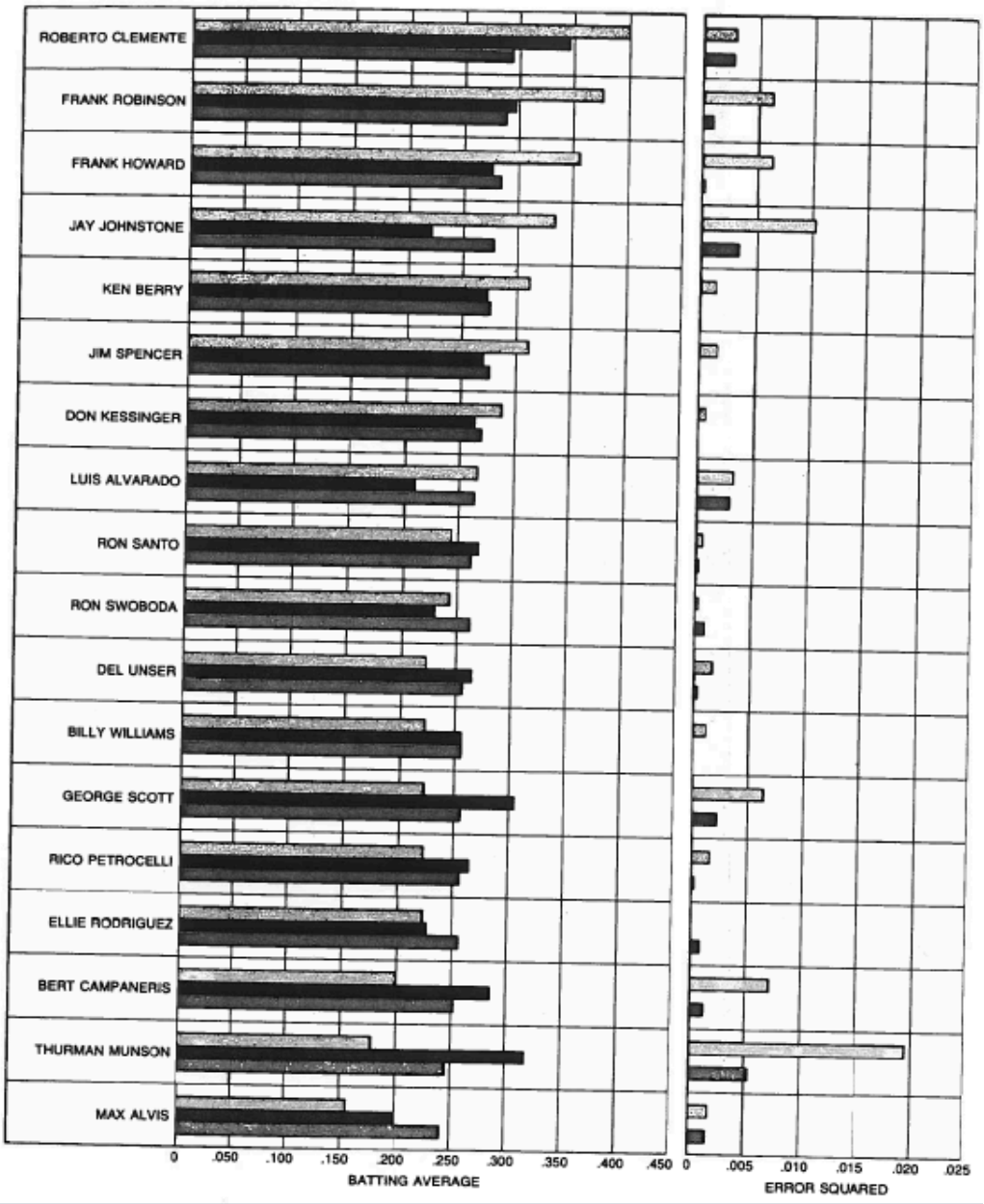


Figure 24: James-Stein estimator in comparison to the individual average for the batting ability of baseball players; the bars correspond to the average over 45 trials (top per row), average over the season (middle per row) and the James-Stein estimator from the first 45 trials (bottom per row); image taken from [4].

6 Core vector machine

In this chapter, we are interested in data classification, regression, or outlier detection in general, thereby focussing on one of the most popular techniques in the last decades: a technology typically referred to as support vector machine, as pioneered by Vapnik as early as 1995. We will investigate this technology in a very relevant context of modern data analysis: what to do if data sets are big? Note that the challenge of 'big data' has been proclaimed as one of the major problems in this decade e.g. by the White House; research programs in the same topic are funded by German research funding agencies such as DFG or BMBF.

Our motivation is as follows: We have already seen the basic formulation of a linear classification by means of a support vector machine in the last chapter. Given data $\vec{x}_1, \ldots, \vec{x}_m \in \mathbb{R}^n$ together with class labels $y_1, \ldots, y_m \in \mathbb{R}$ (ignoring a potential bias, i.e. offset from the origin) the task is to find a separating hyperplane characterized by $\vec{w}$ with maximum margin, i.e.

$$\begin{array}{ll} \min_{\vec{w}} & \|\vec{w}\|_2^2 \\ \text{such that} & y_i \vec{w}^t \vec{x}_i \geq 1 \quad \forall\, i \end{array}$$

The benefits of such a model are manifold. On the one hand, excellent generalization ability is achieved due to the large margin approach, i.e. the objective to have small weight length $\vec{w}$. On the other hand, the problem constitutes a convex problem with a unique optimum and guaranteed convergence to it. In addition, as we will see later, this formalization is just the most basic one, extensions towards more complex problems such as regression, incorporation of possible errors etc. being simple. Further, the restriction to linear systems can be extended to general nonlinear functions by means of so-called kernelization, a we will also see later.

However, one very fundamental restriction remains: the optimization problem as shown above has worst case complexity $\mathcal{O}(m^3)$, m being the number of data points. Thus, it is infeasible to use SVM directly for big data sets. In practice, a variety of extensions has been proposed to speed up the scheme to much faster methods, most of the techniques being heuristics, or still having a large worst case complexity. There is one notable exception: the so-called core vector machine as proposed by Tsang and Kwok constitutes a variant of SVM which provably solves the problem approximately, and which provably converges in linear time only. It is one example of the usage of so-called core sets in machine learning. This modification will be the subject of this chapter.

Approximation algorithms

First, we would like to stress the fact that it is often worthwhile to give up exact solutions of a given problem but to restrict to approximations only. Let's consider

a classical example:

- Consider the metric TSP: given a number of points and pairwise distances d_{ij} which fulfill the conditions of a distance, in particular the triangle inequality, what is a shortest tour through all these points?

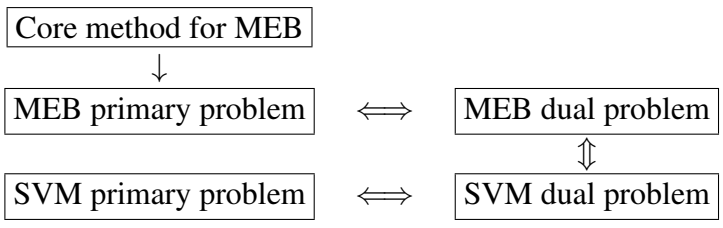
  It is well known that this problem is NP hard, thus we do not expect efficient algorithms for its solution. Interestingly, there are efficient algorithms which find an approximate solution, though. As an example, the spanning tree heuristics computes a minimum spanning tree (using e.g. Prim's algorithm in $\mathcal{O}(m^3)$) and outputs the visited points in preorder. One can show that the resulting tour is at most twice as long as an optimum one, i.e. a polynomial 2-approximation of the NP hard metric TSP is possible.

- If an euclidean TSP is considered, i.e. points are elements of a finite dimensional euclidean vector space, a polynomial time approximation scheme (PTAS) is possible, i.e. for every constant $c \geq 0$ exists a polynomial time algorithm which computes tours of length no longer than $(1 + 1/c)$ times the optimum length, as shown by Arora [1].

- Similar approximation schemes are well known for a number of NP hard problems such as shortest superstring, multiple alignment, vertex cover, etc.

In such cases, approximation is the key to making the problem feasible. Luckily, SVM is polynomial per se, having cubic complexity. However, a speed up of algorithms in the polynomial domain can be crucial if big data are dealt with: in such settings anything more complex than linear is simply out of the question. For optimization problems as occur in machine learning, restricting to approximate optimization makes particular sense: the given optimization problem is obtained as an instance of training examples which represent the underlying aim of learning a function or classification. It is expected that this sampling yields an approximate representation anyway. Hence good solutions are likely sufficient since exact optima would result in approximations of the underlying setting only due to the sampling of data from the true distribution.

The principle idea of core techniques and the way in which an approximation is obtained is particularly elegant in this case. The basic idea is shown in the following scheme:



An SVM can be formulated as a primal or equivalent dual problem. The same holds for the MEB. Because the duals of SVM and MEB are identical, solutions to the dual MEB also yield a solution for the primary SVM. Thus an approximate solution for a particular MEB can be transferred to an approximate solution for SVM. In the following, we start with the MEB, and we will present the single steps one after the other, finally putting all pieces together.

6.1 Minimum enclosing ball

The following geometric problem will be central to core algorithms.

Definition 13 *The* minimum enclosing ball *problem (MEB) is the following: Given data $\vec{x}_1, \ldots, \vec{x}_m \in \mathbb{R}^n$, find a minimum enclosing ball, i.e.*

$$\begin{array}{ll} \min_{R^2,\vec{c}} & R^2 \\ \textit{such that} & \|\vec{c} - \vec{x}_i\|_2^2 \leq R^2 \quad \forall\, i \end{array}$$

This formulation is referred to as the primary problem *for MEB. If the data form a set S, we refer to the unique MEB as* $\mathrm{MEB}(S)$.

Note that we use R^2 to indicate that this value must be non negative. The problem is a linear optimization problem with (convex) quadratic constraints. For what follows, we will need to remember a classic duality theorem of convex optimization:

Theorem 14 *Assume the following primary problem is given*

$$\min_{\vec{x}} f(\vec{x}) \ \textit{such that}\ g_i(\vec{x}) \leq 0 \quad \forall\, i$$

where $f, g_i : \mathbb{R}^n \to \mathbb{R}$ are convex functions. Then the (Lagrangian) dual problem *is given as*

$$\max_{\alpha_i \geq 0} \inf_{\vec{x}} L(\vec{x}, \vec{\alpha}) := \max_{\alpha_i \geq 0} \inf_{\vec{x}} \left(f(\vec{x}) + \sum_i \alpha_i g_i(\vec{x}) \right)$$

A saddle point *of L with respect to the parameters $\vec{x}$ and $\vec{\alpha}$ of L is defined as a point $(\bar{x}, \bar{\alpha})$ with*

$$L(\vec{x}, \bar{\alpha}) \geq L(\bar{x}, \bar{\alpha}) \geq L(\bar{x}, \vec{\alpha})$$

for all $\vec{x}$ with $g_i(\vec{x}) \leq 0$ and $\vec{\alpha} \geq 0$.

The Slater condition *holds if there is a point $\bar{x}$ with $g_i(\bar{x}) < 0$ for all i.*

The Karush, Kuhn, Tucker conditions *(KKT) specify in how far the primary problem and its Lagrangian dual are equivalent. It holds:*

- *Assume $(\bar{x}, \bar{\alpha})$ is a saddle point of the Lagrangian $L(\vec{x}, \vec{\alpha})$ with respect to the coefficients $\vec{x}$ and $\vec{\alpha}$ with $\bar{\alpha}_i \geq 0$ then $\bar{x}$ is an optimum for the primary problem and it holds $\bar{\alpha}_i g_i(\bar{x}) = 0$ for all i.*
- *Conversely, if the Slater condition is satisfied, we find the following:*
  - *If $\bar{x}$ is an optimum for the primary problem then there is $\bar{\alpha}$ with $\bar{\alpha}_i \geq 0$ such that $(\bar{x}, \bar{\alpha})$ is a saddle point of the Lagrangian with respect to the parameters $\vec{x}$ and $\vec{\alpha}$ of L.*
  - *If the optimum value f_{opt} of the primal problem is finite, then there are coefficients $\bar{\alpha}$ with $\alpha_i \geq 0$ such that*

$$f_{opt} = \inf_{\vec{x}} L(\vec{x}, \bar{\alpha}) = \max_{\alpha_i \geq 0} \inf_{\vec{x}} L(\vec{x}, \vec{\alpha})$$

Hence the primal and dual problem are equivalent for convex functions provided the Slater condition is satisfied and the optimum is finite. Further, the KKT conditions allow us to simplify the problem by a number of constraints which must hold at an optimum.

Obviously, MEB fulfills the Slater condition and has a finite optimum, thus its dual is equivalent:

$$\max_{\alpha_i \geq 0} \min_{R^2, \vec{c}} \left(R^2 + \sum_{i=1}^{m} \alpha_i \left(\|\vec{c} - \vec{x}_i\|_2^2 - R^2 \right) \right)$$

The KKT conditions lead to the following conditions for an optimum. Since we are looking for a saddle point, derivatives with respect to the parameters $\vec{c}$, R^2 are zero. Hence:

- ∇_{R^2}:

$$\begin{aligned} 1 - \sum_{i=1}^{m} \alpha_i &= 0 \\ \Rightarrow \sum_{i=1}^{m} \alpha_i &= 1 \end{aligned}$$

- ∇_c:

$$\begin{aligned} 2 \cdot \sum_{i=1}^{m} \alpha_i \left(\vec{c} - \vec{x}_i \right) &= 0 \\ \Rightarrow \sum_{i=1}^{m} \alpha_i \vec{c} &= \sum_{i=1}^{m} \alpha_i \vec{x}_i \\ \overset{\sum \alpha_i = 1}{\Rightarrow} \vec{c} &= \sum_{i=1}^{m} \alpha_i \vec{x}_i \end{aligned}$$

- In addition, the condition $\bar{\alpha}_i g_i(\bar{x}) = 0$ leads to

$$\begin{aligned} 0 &= \sum_{i=1}^{m} \alpha_i \left(\|\vec{c} - \vec{x}_i\|_2^2 - R^2 \right) \\ \Rightarrow R^2 &= \frac{\sum_{i=1}^{m} \alpha_i \|\vec{c} - \vec{x}_i\|_2^2}{\sum_{i=1}^{m} \alpha_i} \\ &= \sum_{i=1}^{m} \alpha_i \| \sum_{j=1}^{m} \alpha_j \vec{x}_j - \vec{x}_i \|_2^2 \end{aligned}$$

taking into account our earlier results that $\sum_{i=1}^{m} \alpha_i = 1$ and $c = \sum_{i=1}^{m} \alpha_i \vec{x}_i$.

Because of these equalities, the Lagrangian becomes:

$$\begin{aligned} L(X, \vec{\alpha}) &= \sum_{i=1}^{m} \alpha_i \| \sum_{j=1}^{m} \alpha_j \vec{x}_j - \vec{x}_i \|_2^2 \\ &\quad + \sum_{i=1}^{m} \alpha_i \left(\| \sum_{j=1}^{m} \alpha_j \vec{x}_j - \vec{x}_i \|_2^2 - \sum_{k=1}^{m} \alpha_k \| \sum_{j=1}^{m} \alpha_j \vec{x}_j - \vec{x}_k \|_2^2 \right) \\ &= \sum_{i=1}^{m} \alpha_i \| \sum_{j=1}^{m} \alpha_j \vec{x}_j - \vec{x}_i \|_2^2 \\ &= \sum_{i=1}^{m} \alpha_i \left(\vec{x}_i^t \vec{x}_i - 2 \cdot \sum_{j=1}^{m} \alpha_j \left(\vec{x}_j^t \vec{x}_i \right) + \sum_{k=1}^{m} \sum_{l=1}^{m} \left(\alpha_k \alpha_l \cdot \vec{x}_k^t \vec{x}_l \right) \right) \\ &= \sum_{i=1}^{m} \left(\alpha_i \vec{x}_i^t \vec{x}_i \right) - 2 \cdot \sum_{i=1}^{m} \sum_{j=1}^{m} \left(\alpha_i \alpha_j \vec{x}_i^t \vec{x}_j \right) + \sum_{i=1}^{m} \alpha_i \left(\sum_{k=1}^{m} \sum_{l=1}^{m} \left(\alpha_k \alpha_l \cdot \vec{x}_k^t \vec{x}_l \right) \right) \\ &= \sum_{i=1}^{m} \left(\alpha_i \vec{x}_i^t \vec{x}_i \right) - 2 \cdot \sum_{i=1}^{m} \sum_{j=1}^{m} \left(\alpha_i \alpha_j \vec{x}_i^t \vec{x}_j \right) + \sum_{k=1}^{m} \sum_{l=1}^{m} \left(\alpha_k \alpha_l \cdot \vec{x}_k^t \vec{x}_l \right) \\ &= \sum_{i=1}^{m} \alpha_i \vec{x}_i^t \vec{x}_i - \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j \vec{x}_i^t \vec{x}_j \\ &= - \left(\vec{\alpha}^t K(X, X) \vec{\alpha} - \sum_{i=1}^{m} \alpha_i \vec{x}_i^t \vec{x}_i \right) \end{aligned}$$

where $K(X, X) = \left(\vec{x}_i^t \vec{x}_j \right)_{ij}$ is the matrix of dot products of the vectors. This yields the dual problem for MEB.

Theorem 15 *The* dual problem *for MEB has the form*

$$\min_{\alpha_i \geq 0, \sum_i \alpha_i = 1} \vec{\alpha}^t K(X,X)\vec{\alpha} - \sum_{i=1}^{m} \alpha_i \vec{x}_i^t \vec{x}_i$$

This problem can be solved using standard constrained convex problem solvers such as CPlex or Matlab, the required time effort is $\mathcal{O}(m^3)$. This, however, can be infeasible if m is large.

Surprisingly, for MEB, there exist approximation algorithms which run in linear time only independently of the dimensionality of the data. To achieve this, the optimization problem is weakened to an approximation problem:

Definition 16 *The* approximate MEB *is the following problem, given an approximation parameter $\epsilon > 0$: given data $\vec{x}_1, \ldots, \vec{x}_m$, find a ball centred in $\vec{c}$ which encloses all data points $\vec{x}_i$ and the radius of which is at most $(1+\epsilon)$ times the minimum possible radius.*

The key observation is that a small set of points actually already determines an approximate MEB.

Definition 17 *Given data $\vec{x}_1, \ldots, \vec{x}_m$, a* core set *$S$ is a subset of these data such that the following holds: assume R and $\vec{c}$ are radius and center of $\mathrm{MEB}(S)$. Then*

$$\|\vec{x}_i - \vec{c}\|_2^2 \leq R^2(1+\epsilon)^2 \quad \forall\, i = 1, \ldots, m$$

There exists a simple algorithm to find such a core set from a given set of points X:

choose $S := \{\vec{x}_i\}$ for some random point,
choose $\vec{x}_j \in X \backslash S$ where $\|\vec{x}_i - \vec{x}_j\|_2^2$ is maximal,
set $S := S \cup \{\vec{x}_j\}$,
repeat
    find $\mathrm{MEB}(S)$ with radius R and centre $\vec{c}$,
    if there is $\vec{x}_k$ with $\|\vec{x}_k - \vec{c}\|_2 > R(1+\epsilon)$:
        $S := S \cup \{\vec{x}_k\}$
until no such $\vec{x}_k$ can be found

Per construction, the algorithms provides a core set S. This core set determines an enclosing ball which approximates the minimum enclosing ball of X with parameter ϵ per construction. What is the run time of the algorithm? For every loop, we have $\mathcal{O}(|S|^3)$ to solve the MEB problem for S (e.g. by solving the respective dual), and $\mathcal{O}(m)$ to sift through all data points. The key observation is, that the number of loops is actually limited by a constant, it is of order $\mathcal{O}(1/\epsilon^2)$ regardless of data dimensionality and number of points! Since S is restricted in size by the number of loops, there results a linear time algorithm.

Theorem 18 *The approximation algorithm for MEB requires at most $\mathcal{O}(1/\epsilon^2)$ loops.*

The proof is geometric in nature. The following observation is valid: Assume $\vec{c}$, R are centre and radius of $\mathrm{MEB}(P)$ for some set P. Then, a closed half-space which contains $\vec{c}$ contains at least one point with distance R from $\vec{c}$ in P. (This is intuitive if considered geometrically. A solid proof shows that if this is not the case, a smaller MEB could be found by slightly moving the centre and reducing the radius.)

Now we set

$$\Delta := \max\{\|\vec{x}_i - \vec{x}_j\|_2 \mid \vec{x}_i, \vec{x}_j \in P\} \geq R$$

Denote by R_t and $\vec{c}_t$ radius and centre of the ball found in iteration t of the algorithm and by S_t the core set after the t-th iteration. It holds

$$R_{t+1} \geq \left(1 + \epsilon^2/16\right) R_t$$

because of the following:

- Case $\|\vec{c}_t - \vec{c}_{t+1}\|_2 < (\epsilon/2) \cdot R_t$: assume point $\vec{p}$ is added to S in step t. Then

$$\begin{aligned} R_{t+1} \geq \|\vec{p} - \vec{c}_{t+1}\|_2 &\geq \|\vec{p} - \vec{c}_t\|_2 - \|\vec{c}_t - \vec{c}_{t+1}\| \\ &\geq (1+\epsilon)R_t - \epsilon/2 \cdot R_t = (1+\epsilon/2)R_t \\ &\geq \left(1 + \epsilon^2/16\right) R_t \end{aligned}$$

- Case $\|\vec{c}_t - \vec{c}_{t+1}\|_2 \geq (\epsilon/2) \cdot R_t$: Consider the hyperplane through $\vec{c}_t$ which is perpendicular to the line from $\vec{c}_t$ to $\vec{c}_{t+1}$. Consider the half-space defined by this hyperplane which does not contain $\vec{c}_{t+1}$. Because of the observation above there must exist a point $\vec{x}$ in S_t which has distance at least R_t from $\vec{c}_t$ and which is contained in this half-space. Thus it holds since we have a MEB of S_t in the next step

$$\begin{aligned} R_{t+1} &\geq \|\vec{x} - \vec{c}_{t+1}\|_2 \\ &= \sqrt{\|\vec{x} - \vec{c}_t\|_2^2 + \|\vec{c}_t - \vec{c}_{t+1}\|_2^2} \\ &\geq \sqrt{R_t^2 + \epsilon^2/4 \cdot R_t^2} \\ &\geq R_t(1 + \epsilon^2/16) \end{aligned}$$

  using the Pythagorean theorem, because $\sqrt{1+\epsilon^2/4} \geq (1+\epsilon^2/16)$ is valid iff $1 + \epsilon^2/4 \geq (1+\epsilon^2/16)^2 = 1 + \epsilon^2/8 + \epsilon^4/256$ which holds for $\epsilon \leq \sqrt{32}$. Reasonable ranges for ϵ are close to 0, hence this is valid.

We find that $R_0 \geq \Delta/4$ so we have after $64/\epsilon^2$ steps:

$$\text{radius} \geq \left(1+\epsilon^2/16\right)^{64/\epsilon^2} R_0 \geq 64/\epsilon^2 \cdot \epsilon^2/16 \cdot R_0 \geq \Delta$$

hence an MEB has to be present at this step at the latest. Thereby, $R_0 \geq \Delta/4$ holds because the first two points are contained in the core set per definition, and the second point has maximum distance from the first one. □

Core algorithms are of direct relevance for diverse tasks in computer graphics where shape matching or detection of intersection can be related to finding MEBs. The algorithm above is particularly simple. Using more advanced techniques, one can reduce the size of core sets to $\mathcal{O}(1/\epsilon)$.

It will turn out that a variation of MEB will be more useful since its dual problem has a slightly more general form: the center-constrained MEB problem which fixes the last coefficient of the center to 0.

Definition 19 *The* center constraint minimum enclosing ball problem *(CMEB) is the problem, given data* $(\vec{x}_i, z_i) \in \mathbb{R}^{n+1}$*,* $i = 1, \ldots, m$*, find a radius* R *and centre* $(\vec{c}, 0) \in \mathbb{R}^{n+1}$ *such that* $\|\vec{x}_i - \vec{c}\|_2^2 + |z_i|^2 \leq R^2$ *and* R *is smallest with this property.*

The core algorithm can easily be adapted for the CMEB: in every loop, we solve the CMEB instead of the MEB. The proof that a finite approximate core set with size limited by the approximation quality is found remains valid. Similar as before, the dual problem can be derived as follows:

The dual of CMEB has the form

$$\max_{\alpha_i \geq 0} \min_{R^2, \vec{c}} \left(R^2 + \sum_{i=1}^{m} \alpha_i \left(\|\vec{c} - \vec{x}_i\|_2^2 + z_i^2 - R^2 \right) \right)$$

so we find

- ∇_{R^2}: $\sum_{i=1}^{m} \alpha_i = 1$
- ∇_c: $2 \cdot \sum_{i=1}^{m} \alpha_i \left(\vec{c} - \vec{x}_i\right) = 0 \overset{\sum \alpha_i = 1}{\Rightarrow} \vec{c} = \sum_{i=1}^{m} \alpha_i \vec{x}_i$

In addition, the condition $\bar{\alpha}_i g_i(\bar{x}) = 0$ leads to

- $R^2 = \sum_{i=1}^{m} \alpha_i \left(\| \sum_{j=1}^{m} \alpha_j \vec{x}_j - \vec{x}_i \|_2^2 + z_i^2 \right)$

Because of these equalities, the Lagrangian becomes:

$$
\begin{aligned}
L(X,\vec{\alpha}) &= \sum_{i=1}^{m}\alpha_i\|\sum_{j=1}^{m}\alpha_j\vec{x}_j-\vec{x}_i\|_2^2 \\
&+\sum_{i=1}^{m}\alpha_i\left(\|\sum_{j=1}^{m}\alpha_j\vec{x}_j-\vec{x}_i\|_2^2+z_i^2-\sum_{k=1}^{m}\alpha_k\|\sum_{j=1}^{m}\alpha_j\vec{x}_j-\vec{x}_k\|_2^2\right) \\
&=\sum_{i=1}^{m}\alpha_i\left(\|\sum_{j=1}^{m}\alpha_j\vec{x}_j-\vec{x}_i\|_2^2+z_i^2\right) \\
&=\sum_{i=1}^{m}\alpha_i\left(-2\sum_{j=1}^{m}\alpha_j\vec{x}_j^t\vec{x}_i+\sum_{kl}\alpha_k\alpha_l\vec{x}_l^t\vec{x}_k+\vec{x}_i^t\vec{x}_i+z_i^2\right) \\
&=-\sum_{ij}\alpha_i\alpha_j\vec{x}_i^t\vec{x}_j+\sum_{i=1}^{m}\left(\alpha_i\vec{x}_i^t\vec{x}_i+z_i^2\right) \\
&=-\left(\vec{\alpha}^tK(X,X)\vec{\alpha}-\sum_{i=1}^{m}\left(\alpha_i\vec{x}_i^t\vec{x}_i+z_i^2\right)\right)
\end{aligned}
$$

Theorem 20 *The* dual problem *for CMEB has the form*

$$\min_{\alpha_i\geq 0,\sum_i\alpha_i=1}\vec{\alpha}^tK(X,X)\vec{\alpha}-\sum_{i=1}^{m}\alpha_i(\vec{x}_i^t\vec{x}_i+z_i^2)$$

Because of $\sum_{i=1}^{m}\alpha=1$ *this is equivalent to*

$$\min_{\alpha_i\geq 0,\sum_i\alpha_i=1}\vec{\alpha}^tK(X,X)\vec{\alpha}-\sum_{i=1}^{m}\alpha_i(\vec{x}_i^t\vec{x}_i+z_i^2-\eta)$$

for any $\eta\in\mathbb{R}$

Note that the linear term has the form $\bar{\alpha}^t(\mathrm{diag}(K(X,X)+Z^2-\eta\mathbf{1})$ where Z^2 is a diagonal matrix with arbitrary non-negative entries and η accounts for a constant possibly negative shift. Hence arbitrary linear terms $\vec{\alpha}^t\vec{v}$ for a vector $\vec{v}$ can be described this way, so that the dual CMEB problem is a general quadratic problem of the form

$$\min_{\alpha_i\geq 0,\sum_i\alpha_i=1}\vec{\alpha}^tM\vec{\alpha}+\vec{\alpha}^t\vec{v}+k$$

with positive semidefinite matrix M, vector $\vec{v}$ and constant k.

6.2 Core vector outlier detection

Assume data $\vec{x}_1, \ldots, \vec{x}_m \in \mathbb{R}^n$ are given. Outlier detection aims at a classification which classifies all observed data as well as similar data as 'normal' while data which are dissimilar should be classified as 'outlier'. One popular approach for outlier detection which uses strategies behind SVM tries to find a linear separation of all given points from the origin (the latter considered as a default outlier) such that the margin of this classification is large. Again, we drop the bias for simplicity. We arrive at the primary problem

$$\begin{array}{ll} \min_{\vec{w}} & \frac{1}{2}\|\vec{w}\|_2^2 \\ \text{such that} & \vec{w}^t\vec{x}_i \geq 1 \quad \forall\, i \end{array}$$

We will change this formalization in two respects. The first observation is, that this problem cannot necessarily be solved, which is why the constraints are turned to soft constraints by means of slack variables

$$\begin{array}{ll} \min_{\vec{w},\xi_i} & \frac{1}{2}\|\vec{w}\|_2^2 + \frac{C}{2}\sum \xi_i^2 \\ \text{such that} & \vec{w}^t\vec{x}_i \geq 1 - \xi_i \quad \forall\, i \end{array}$$

where the size of ξ_i indicates in how far the constraints are violated, and the constant C balances the two objectives, to arrive at a large margin, or at correct classification of all points. In this formulation, there is always a solution in the interior of the feasible region, i.e. Slater's condition is satisfied.

The second observation is more subtle. We are finally interested in a formalization of a problem such that its dual looks like a dual CMEB problem. If we consider the dual problem of the outlier detection, we observe that no constraint of the form $\sum_i \alpha_i = 1$ is present. This constraint can be introduced by means of a small, reasonable variation. We do actually not know which margin is best, and the minimum distance 1 of the constraint is a bit arbitrary. Instead, we can explicitly optimize this quantity as follows:

Definition 21 *The primary problem of* outlier detection *by means of a linear separation has the form*

$$\begin{array}{ll} \min_{\vec{w},\rho,\xi_i} & \frac{1}{2}\|\vec{w}\|_2^2 - \rho + \frac{C}{2}\sum \xi_i^2 \\ \textit{such that} & \vec{w}^t\vec{x}_i \geq \rho - \xi_i \quad \forall\, i \end{array}$$

The dual problem of outlier detection is given as

$$\max_{\alpha_i \geq 0} \min_{\vec{w},\rho,\xi_i} \frac{1}{2}\|\vec{w}\|_2^2 - \rho + \frac{C}{2}\sum_{i=1}^{N} \xi_i^2 + \sum_{i=1}^{N} \alpha_i \left(\rho - \xi_i - \vec{w}^t\vec{x}_i\right)$$

The KKT conditions yield the following:

- $\nabla_{\vec{w}}$: $\vec{w} = \sum \alpha_i \vec{x}_i$
- ∇_{ξ_i}: $\xi_i = \frac{\alpha_i}{C}$
- $\nabla\rho$: $\sum \alpha_i = 1$
- $\sum_i \alpha_i(\rho - \xi_i - \vec{w}^t\vec{x}_i) = 0 \Rightarrow \rho = \sum_{ij} \alpha_i\alpha_j\vec{x}_i^t\vec{x}_j + \sum_i \alpha_i^2/C$

Hence the Lagrangian becomes

$$\begin{aligned}
&\frac{1}{2}\|\sum_{i=1}^{m}\alpha_i\vec{x}_i\|_2^2 - \sum_{ij}\alpha_i\alpha_j\vec{x}_i^t\vec{x}_j - \sum_{i=1}^{m}\frac{\alpha_i^2}{C} + \frac{C}{2}\sum_{i=1}^{m}\frac{\alpha_i^2}{C^2} \\
&+ \sum_{i=1}^{m}\alpha_i\left(\sum_{jk}\alpha_j\alpha_k\vec{x}_j^t\vec{x}_k + \sum_{j=1}^{m}\frac{\alpha_j^2}{C} - \frac{\alpha_i}{C} - \sum_{j=1}^{m}\alpha_j\vec{x}_j^t\vec{x}_i\right) \\
=& -\frac{1}{2}\sum_{ij}\alpha_i\alpha_j\vec{x}_i^t\vec{x}_j - \frac{1}{2}\sum_{i=1}^{m}\frac{\alpha_i^2}{C}
\end{aligned}$$

Thus the dual problem is

$$\begin{aligned}
&\max_{\alpha_i\geq 0, \sum\alpha_i = 1} -\frac{1}{2}\left(\vec{\alpha}^t K(X,X)\vec{\alpha} + \frac{1}{C}\sum_{i=1}^{m}\alpha_i^2\right) \\
=&\max_{\alpha_i\geq 0, \sum\alpha_i = 1} -\frac{1}{2}\left(\vec{\alpha}^t K'(X,X)\vec{\alpha}\right)
\end{aligned}$$

where $K'(X,X)$ is the matrix $\left(\vec{x}_i^t\vec{x}_j + \frac{\delta_{ij}}{C}\right)_{ij}$.

Theorem 22 *The* dual problem of outlier detection *is*

$$\max_{\alpha_i\geq 0, \sum\alpha_i = 1} -\frac{1}{2}\left(\vec{\alpha}^t K'(X,X)\vec{\alpha}\right)$$

where $K'(X,X)$ is the matrix with entries $\vec{x}_i^t\vec{x}_j + \frac{\delta_{ij}}{C}$ at position (i,j).

We can compare the dual problem of outlier detection with the dual problem of MEB. We obtain the two dual formulations:

$$\begin{array}{ll}
\text{CMEB:} & \\
\max & -\frac{1}{2}\left(\vec{\alpha}^t K(X,X)\vec{\alpha} - \sum_i \alpha_i(\vec{x}_i^t\vec{x}_i + z_i^2 - \eta)\right) \\
\text{with} & \alpha_i \geq 0 \\
& \sum\alpha_i = 1
\end{array}$$

and

Outlier detection:	
max	$-\frac{1}{2}\vec{\alpha}^t K'(X,X)\vec{\alpha}$
with	$\alpha_i \geq 0$
	$\sum \alpha_i = 1$

One can see that these two problems have exactly the same form provided $z_i^2 = \max_j \|\vec{x}_j\|_2^2 - \|\vec{x}_i\|_2^2$, $\eta = \max_i \|\vec{x}_i\|_2^2$ and $K'(X,X)$ equals $K(X,X)$. What is the consequence? We only have to map the inputs $\vec{x}_i$ of the outlier detection problem to inputs of the CMEB of slightly different form to get exact equivalence of the duals. Then, the core algorithm for CMEB offers an approximate solution of the outlier detection problem by means of a linear mapping with linear costs as follows:

Core algorithm for outlier detection

input: data $\vec{x}_i$ for outlier detection
set data for CMEB as:
$\vec{x}'_i = (\vec{x}_i, 1/\sqrt{C} \cdot \vec{e}_i)$ where $\vec{e}_i$ is unit vector i
$z_i = \max_j \|\vec{x}_j\|_2 - \|\vec{x}'_i\|_2$
solve CMEB for $(\vec{x}'_i, z_i)$
this gives coefficients $\vec{\alpha}$
the outlier detection function is given as:
$\vec{x} \mapsto \text{sign}\left(\sum_i \alpha_i \vec{x}_i^t \vec{x}_i - \sum_{ij} \alpha_i \alpha_j \vec{x}_i^t \vec{x}_j - \sum_i \alpha_i^2 / C\right)$

The latter follows from the fact that we have modeled outlier detection as linear functions with margin ρ: $\vec{w}^t \vec{x}_i \leq \rho$, hence using the KKT for $\vec{w}$ and ρ we end up with this functional form. Note that the sum has to be taken over coefficients corresponding to the found core set only, i.e. a representation of the solution in terms of a fixed number of summands is found. Obviously, it is easily possible to change the offset ρ if this yields better results.

6.3 Kernelization

Before proceeding to similar algorithms which allow us to derive efficient core optimization algorithms for linear classification and linear regression, we have a look at a central step to make these linear models much more powerful, so-called kernelization.

In practice, it is very unlikely that outliers can be characterized by a half space of the data space for which the origin is not included. Rather we expect a nonlinear (and bounded) form of the outlier region. So-called kernelization allows us to extend a linear model towards a more general nonlinear form by combining it with a fixed nonlinear preprocessing of the data. Since a nonlinear representation

of data in a high dimensional kernel or feature space is much more flexible it can be expected that the original problem is much simpler, ideally linear in the transformed view.

This technique bears one drawback: to achieve sufficient flexibility, a high dimensionality is necessary such that computations can become infeasible. Due to this fact, the preprocessing is usually only implicit by means of a so-called kernel.

Definition 23 *A* kernel *is a function*

$$k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$$

such that a feature map $\Phi : \mathbb{R}^n \to H$ *exists for some Hilbert space* H*, the* feature space*, such that*

$$k(\vec{x}, \vec{y}) = \Phi(\vec{x})^t \Phi(\vec{y})$$

where the latter refers to the inner product of the Hilbert space H*. Note that this implies that* k *is symmetric.*

A Hilbert space is a generalization of a euclidean vector space (it is a complex vector space with inner product which is complete), and we can simply think of it as a standard usually very high (possibly infinite) dimensional euclidean vector space for our purposes. Essentially, a kernel is just a shortcut for a nonlinear feature mapped followed by an inner product, thus enabling efficient computation of this combination. Interestingly, one can show that a function is a valid kernel without actually constructing the mapping Φ in some cases.

Popular kernels include the following:

- *Gaussian kernel*: $k(\vec{x}, \vec{y}) = \exp(-\|\vec{x} - \vec{y}\|_2^2/\sigma^2)$ where σ is a parameter indicating the smoothness of the kernel (larger σ corresponding to smoother mappings). Interestingly, the corresponding Φ would map to an infinite dimensional space, thus the Gaussian kernel has a high degree of flexibility. Another beneficial aspect of the Gaussian kernel is its limited support: only vectors $\vec{x}$ and $\vec{y}$ which are close result in an output value which is significantly different from zero. The Gaussian kernel is probably the most common kernel which is often used per default.

- *Linear kernel*: $k(\vec{x}, \vec{y}) = \vec{x}^t \vec{y}$ is a kernel per definition since it is directly expressed as inner product. Here the feature map is simply the identity $\Phi(\vec{x}) = \vec{x}$, such that the linear kernel can only be used if a linear mapping is already sufficient. This can be the case for very high dimensional data, for example.

- *Polynomial kernel:* $k(\vec{x}, \vec{y}) = (\vec{x}^t\vec{y} + 1)^d$ for a fixed degree $d > 0$ of polynomials. Its flexibility is limited according to the degree d.

Interestingly, kernelization even offers an interface to extend vectorial techniques to the domain of non-vectorial data such as trees or graph structures: as long as the output of k is a real number which can be interpreted as inner product in some Hilbert space, the technique can be used. We will say a few words about this issue in the next chapter where we focus on time series data.

Another interesting aspect of kernels is that one can check whether a function k is a valid kernel without actually explicitly constructing Φ. There are principled ways to test the kernel property such as e.g. the so-called Mercer condition. Further closure properties of kernels as regards e.g. summation allow constructing new kernels from known ones. In addition, given a finite set of data $\vec{x}_i$ only and its pairwise kernel values $K(X, X) = (k(\vec{x}_i, \vec{x}_j))_{i,j}$, one can easily check whether these values stem from an underlying kernel: this is the case if and only if the matrix $K(X, X)$, the *Gram matrix*, has the properties as induced by an inner product, i.e. it is a symmetric positive semidefinite matrix thus relating to a positive definite quadratic form. Kernelization extends outlier detection to a nonlinear mapping which, in case of the popular Gaussian kernel, has limited support. Thus, we arrive at a very suitable functional form for outlier detection. The dual problem has the kernelized form

$$\max_{\alpha_i \geq 0, \sum \alpha_i = 1} -\frac{1}{2}\left(\vec{\alpha}^t K'(X, X)\vec{\alpha}\right)$$

where

$$K'(X, X) = \left(k\left(\vec{x}_i, \vec{x}_j\right) + \frac{\delta_{ij}}{C}\right)_{ij}$$

thus we can also express the dual problem in terms of the kernel only. Provided we can approximately solve this problem with parameters α_i, we arrive at the outlier function

$$\vec{x} \mapsto \operatorname{sign}\left(\sum_{i=1}^{m} \alpha_i k(\vec{x}_i, \vec{x}_i) - \sum_{ij} \alpha_i \alpha_j k(\vec{x}_i, \vec{x}_j) - \sum_{i=1}^{m} \frac{\alpha_i^2}{C}\right)$$

which is expressed in terms of the kernel only.

How to solve the kernelized dual? Matching this dual problem to the underlying CMEB, we arrive at the task to solve the CMEB for vectors of the form

$$\left(\Phi\left(\vec{x}_i\right), \frac{1}{\sqrt{C}} \cdot \vec{e}_i\right)$$

where we do not know the mapping Φ but its corresponding kernel only. So we would like to solve CMEB using core methods for a kernel

$$k'(\vec{x}_i, \vec{x}_j) = k(\vec{x}_i, \vec{x}_j) + \delta_{ij}/C$$

which implicitly characterizes the data. Luckily, CMEB can be dealt with using the kernel only, as follows:

Core kernel CMEB algorithms:

given points $(\Phi(\vec{x}_i), z_i)$ via a kernel $K' = (k'(\vec{x}_i, \vec{x}_j))_{ij} = (k'_{ij})_{ij}$,
choose $S :=$ some random index $\{i\}$,
choose $j \neq i$ with maximum $k'_{ii} + k'_{jj} - 2k'_{ij}$,
set $S := S \cup \{j\}$,
repeat
  find $\mathrm{CMEB}(S)$ with radius r and centre $\vec{c}$ via its dual:
    $\min_{\alpha_i \geq 0, \sum \alpha_i = 1} \vec{\alpha}^t K' \vec{\alpha} - \sum \alpha_i (k'_{ii} + z_i^2)$
    this implicitly gives the centre $(\sum \alpha_i \Phi(\vec{x}^i), 0)$
    and radius $r^2 = \sum \alpha_i (k'_{ii} + |z_i|^2) - \sum \alpha_i \alpha_j k'_{ij}$
  if there is l with $k'_{ll} - 2\sum \alpha_i k'_{li} + \sum \alpha_i \alpha_j k'_{ij} + |z_k|^2 > r^2(1+\epsilon)^2$:
    $S := S \cup \{l\}$
until no such l can be found
output: dual variables α_i yield approximate solution of dual problem

In consequence, we arrive at the following approximation algorithm for kernel outlier detection:

Core kernel outlier detection:

input: data $\vec{x}_i$ for outlier detection
set data for CMEB as:
  kernel $K' = (k(\vec{x}_i, \vec{x}_j) + \delta_{ij}/C)_{ij}$, $z_i^2 = \max_j k'_{jj} - k'_{ii}$
solve kernel CMEB for these data
this gives coefficients $\vec{\alpha}$
the outlier detection function is given as:
  $\vec{x} \mapsto \mathrm{sign}\left(\sum_i \alpha_i k(\vec{x}_i, \vec{x}) - \sum_{ij} \alpha_i \alpha_j k(\vec{x}_i, \vec{x}_j) - \sum_i \frac{\alpha_i^2}{C}\right)$

Having found this general scheme, it is possible to transfer other SVM problems to core algorithms. We will exemplarily look at classification and regression only.

6.4 Core vector classification

Assume labeled points $(\vec{x}_i, y_i) \in \mathbb{R}^n \times \{-1, 1\}$ with $i \in \{1, \ldots, m\}$ are given. Assume a fixed kernel k is chosen with corresponding feature map Φ. We are

interested in finding a linear classifier of the data

$$\vec{x} \mapsto \text{sign}(\vec{w}^t\vec{x} + b)$$

such that the classification coincides with the given training labels on the data and the margin of the classifier is as large as possible. This can be formalized in the following primary problem:

Definition 24 *The* primary two class SVM problem *is the following: given data* $\vec{x}_i$*, labels* y_i *and a kernel* k *with feature map* Φ*.*

$$\begin{array}{ll} \min_{\vec{w},b,\rho,\xi_i} & \frac{1}{2}\left(\|\vec{w}\|_2^2 + b^2\right) - \rho + \frac{C}{2}\sum \xi_i^2 \\ \textit{such that} & y_i(\vec{w}^t\Phi(\vec{x}_i) + b) \geq \rho - \xi_i \end{array}$$

$C \geq 0$ *is a constant quantifying the error as measured by the slack variables* ξ_i*.* ρ *refers to the margin.*

The Lagrangian becomes

$$\max_{\alpha_i \geq 0} \min_{\vec{w},b,\rho,\xi_i} \frac{1}{2}\left(\|\vec{w}\|_2^2 + b^2\right) - \rho + \frac{C}{2}\sum_{i=1}^{m} \xi_i^2 + \sum_{i=1}^{m} \alpha_i \left(\rho - \xi_i - y_i\left(\vec{w}^t\Phi\left(\vec{x}_i\right) + b\right)\right)$$

The KKT conditions yield the following:

- $\nabla\vec{w}$: $\vec{w} = \sum \alpha_i y_i \Phi(\vec{x}_i)$
- ∇b: $b = \sum \alpha_i y_i$
- $\nabla\xi_i$: $\xi_i = \frac{\alpha_i}{C}$
- $\nabla\rho$: $\sum \alpha_i = 1$
- $\sum \alpha_i(\rho - \xi_i - y_i(\vec{w}t\Phi(\vec{x}_i) + b)) = 0$ hence:

$$\rho = \sum_{i=1}^{m}\left(\frac{\alpha_i^2}{C} + \sum_{j=1}^{m}\left(\alpha_i\alpha_j y_i y_j k\left(\vec{x}_i, \vec{x}_j\right) + \alpha_i\alpha_j y_i y_j\right)\right)$$

Using these terms, we can simplify the dual problem to obtain the following:

Theorem 25 *The* dual two class SVM problem *is given as*

$$\max_{\alpha_i \geq 0, \sum \alpha_i = 1} -\frac{1}{2}\left(\sum_{ij} \alpha_i\alpha_j y_i y_j k(\vec{x}_i, \vec{x}_j) + \sum_{ij} \alpha_i\alpha_j y_i y_j + \sum_{i=1}^{m} \frac{\alpha_i^2}{C}\right)$$

This corresponds to a kernel CMEB problem provided we set

$$k'(\vec{x}_i, \vec{x}_j) = y_i y_j k(\vec{x}_i, \vec{x}_j) + y_i y_j + \delta_{ij}/C$$

Note that this is a valid kernel with feature map

$$\Phi'(\vec{x}_i) = \left(y_i \Phi(\vec{x}_i), y_i, \vec{e}_i/\sqrt{C} \right)$$

Further, we set $z_i^2 = \max_j k(\vec{x}_j, \vec{x}_j) - k(\vec{x}_i, \vec{x}_i)$. This yields a core algorithm for SVM classification by means of the kernel CMEB problem:

Core vector classification:

input: data $\vec{x}_i$ and label y_i
set data for CMEB as:
kernel $K' = (y_i y_j k(\vec{x}_i, \vec{x}_j) + y_i y_j + \delta_{ij}/C)_{ij}$, $z_i^2 = \max_j k'_{jj} - k'_{ii}$,
$\eta = \max_j k'_{jj}$
solve kernel CMEB for these data
this gives coefficients $\vec{\alpha}$
the SVM classifier function is given as:
$\vec{x} \mapsto \operatorname{sign}\left(\sum_i \alpha_i y_i k(\vec{x}_i, \vec{x}) + \sum_i \alpha_i y_i\right)$

6.5 Core vector regression

Assume points $\vec{x}_1, \ldots, \vec{x}_N \in \mathbb{R}^n$ and real-valued outputs $y_i \in \mathbb{R}$ are given. Provided a kernel k with feature map Φ is given, a kernel regression trains a function of the form

$$\vec{x} \mapsto \vec{w}^t \Phi(\vec{x}) + b$$

The training objective is to get as many data points as possible approximately right while preserving a large margin. Thereby, different possible formalizations are popular, one being the so-called ϵ-tube: a data point is correctly predicted iff its image is within ϵ of the desired value. As before, we can optimize this value together with the data and arrive at the formalization

Definition 26 *The* primal SVM regression problem *is given as*

$$\begin{array}{ll} \min_{\vec{w},b,\epsilon,\xi_i,\xi_i^*} & \frac{1}{2}(\|\vec{w}\|_2^2 + b^2) + \frac{C}{2}\sum(\xi_i^2 + (\xi_i^*)^2) + \epsilon \\ \textit{where} & y_i - (\vec{w}^t\Phi(\vec{x}_i) + b) \leq \epsilon + \xi_i \\ & (\vec{w}^t\Phi(\vec{x}_i) + b) - y_i \leq \epsilon + \xi_i^* \end{array}$$

where C quantifies the error relevance.

The dual problem is given as

$$\begin{array}{ll} \max_{\alpha_i\alpha_i^*} \min_{\vec{w},b,\epsilon,\xi_i,\xi_i^*} & (\|\vec{w}\|_2^2 + b^2)/2 + \frac{C}{2}\sum(\xi_i^2 + (\xi_i^*)^2) + \epsilon \\ & - \sum \alpha_i(\epsilon + \xi_i - y_i + (\vec{w}^t\Phi(\vec{x}_i) + b)) \\ & - \sum \alpha_i^*(\epsilon + \xi_i^* + y_i - (\vec{w}^t\Phi(\vec{x}_i) + b)) \end{array}$$

The KKT conditions yield

- $\nabla\vec{w}$: $\vec{w} = \sum(\alpha_i - \alpha_i^*)\Phi(\vec{x}_i)$
- ∇b: $b = \sum(\alpha_i - \alpha_i^*)$
- $\nabla\xi_i$: $\xi_i = \alpha_i/C$
- $\nabla\xi_i^*$: $\xi_i^* = \alpha_i^*/C$
- $\nabla\epsilon$: $\sum(\alpha_i + \alpha_i^*) = 1$
- and a condition which determines ϵ (we do not need this here).

hence we arrive at the dual problem:

Theorem 27 Dual core vector regression *is described by the problem*

$$\begin{aligned} \max_{\alpha_i,\alpha_i^*\geq 0,\sum\alpha_i+\alpha_i^*=1} & -\frac{1}{2}\cdot\sum_{ij}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(k_{ij} + 1) \\ & -\frac{1}{2}\cdot\sum_{i=1}^{m}\alpha_i^2/C - \frac{1}{2}\cdot\sum_{i=1}^{m}(\alpha_i^*)^2/C \\ & +\sum_{i=1}^{m}(\alpha_i - \alpha_i^*)y_i \end{aligned}$$

This problem is of the form

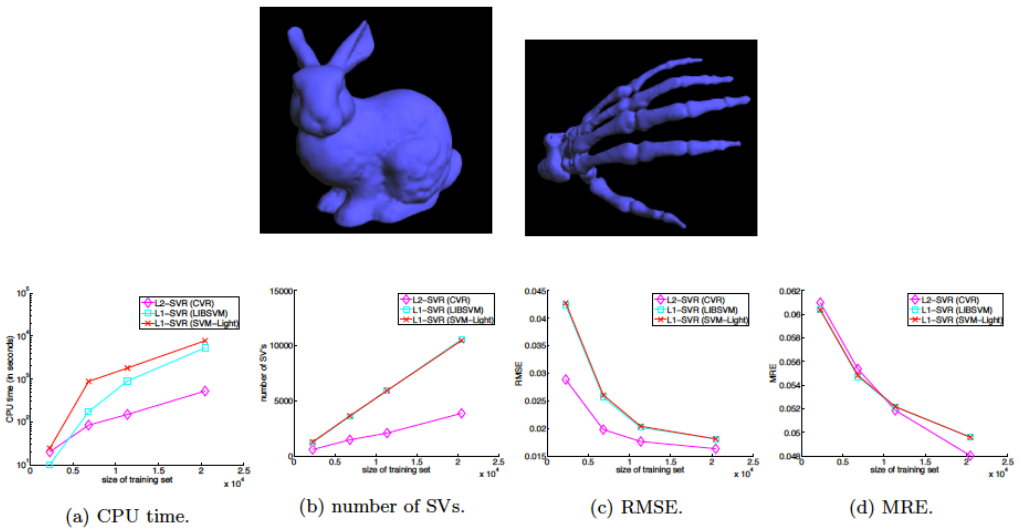
$$\max_{\alpha_i,\alpha_i^*\geq 0,\sum\alpha_i+\alpha_i^*=1} \quad -\tfrac{1}{2}\cdot\begin{pmatrix}\vec{\alpha}\\ \vec{\alpha^*}\end{pmatrix}^T\tilde{K}\begin{pmatrix}\vec{\alpha}\\ \vec{\alpha^*}\end{pmatrix} + \begin{pmatrix}\vec{\alpha}\\ \vec{\alpha^*}\end{pmatrix}^t\begin{pmatrix}\vec{y}\\ -\vec{y}\end{pmatrix}$$

where

$$\tilde{K} = \begin{pmatrix} K + \mathbf{1}\mathbf{1}^t + 1/C\cdot\mathbf{I} & -(K + \mathbf{1}\mathbf{1}^t) \\ -(K + \mathbf{1}\mathbf{1}^t) & K + \mathbf{1}\mathbf{1}^t + 1/C\cdot\mathbf{I}\end{pmatrix}$$

$\tilde{K}$ is a valid kernel. Therefore we can use the core algorithm for kernel CMEB with kernel $\tilde{K}$ and $|z_i|^2 = \max_j(\tilde{K}_{jj} \pm y_j) - \tilde{K}_{ii} \pm y_i$.

Using this technique, further linear time algorithms can be derived for similar settings, including, for example:



(a) CPU time. (b) number of SVs. (c) RMSE. (d) MRE.

Figure 25: Example applications of core vector methods: (top) implicit surface modelling by mapping data to a constant, this way defining an implicit surface as zero set. The CVM is recursively used in three steps with decreased kernel width, where steps two and three are used to predict (correct) the errors. (bottom) example behavior of CVM in comparison to two popular SVM implementations on a benshmark example. Images taken from [6].

- classification in the presence of imbalanced classes where errors are weighted differently, as primary problem:

$$\begin{array}{ll} \min_{\vec{w},b,\rho,\xi_i} & \frac{1}{2}(\|\vec{w}\|_2^2 + b^2) - \rho + \sum C_i \xi_i^2 \\ \text{where} & y_i(\vec{w}^t \Phi(\vec{x}_i) + b) \geq \rho - \xi_i \end{array}$$

- ranking SVM
- implicit surface modelling

A few applications of core vector machines are depicted in the following image.

Literature

- PTAS for Euclidean Traveling Salesman and Other Geometric Problems, Sanjeev Arora, Journal of the ACM, 45(5):753782, 1998
- Mihai Badoiu, Kenneth L. Clarkson. Smaller core-sets for balls. SODA '03: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2003

- Piyush Kumar and E. Alper Yildirim. Minimum volume enclosing ellipsoids and core sets, Journal of Optimization Theory and Applications, 126 (1) pp. 1-21 (2005).
- Ivor W. Tsang, James T. Kwok, Pak-Ming Cheung. Core vector machines: Fast SVM training on very large data sets. Journal of Machine Learning Research, 6:363-392, 2005.
- Ivor W. Tsang, Andras Kocsor, James T. Kwok. Simpler core vector machines with enclosing balls. Proceedings of the Twenty-Fourth International Conference on Machine Learning (ICML), Corvallis, Oregon, USA, June 2007.
- Ivor W. Tsang, James T. Kwok, Kimo T. Lai, Core Vector Regression for Very Large Regression Problems, ICML 2005
- C++ code for CVM can be found at http://www.c2i.ntu.edu.sg/ivor/cvm.html

7 Kernels for structures

We have just considered fast realizations of classification/regression/outlier detection by means of core techniques. Besides this ingenious idea to achieve a speed up, there is another key point behind the formalism of such problems via its duals which we would like to stress: in the dual formalization of the problem, data are addressed by means of kernels only. That means, we have a complete formalization of the machine learning classifier as regards training and classification at our disposal even if we do not know an explicit vectorial representation of the data provided we are capable of computing the kernel on the given data set only.

in machine learning, this observation has led to a direction of research which can be summarized under the umbrella of structure kernels: if is requires a kernel only as the interface to the data, we are no longer bound to vectorial data, rather, arbitrary possibly discrete data types can be dealt with provided there are efficient ways to define and compute a kernel for such data. This opens the way towards machine learning methods for general data structures such as sequential data, tree structures, or graphs, as occur in bioinformatics, natural language processing, time series processing, logic, etc.

In this chapter, we will focus on a kernel which has been proposed for time series data, the DTW kernel, which extends one of the most popular dissimilarity measure for time series to the field of kernels. Afterwards, we will have a short glimpse at principled techniques how to device kernels for structured data as well as alternatives how to treat time series in machine learning tasks and their challenges.

7.1 Dynamic Time Warping

We denote time series as $X = (\vec{x}(1), \ldots, \vec{x}(N))$ or $Y = (\vec{y}(1), \ldots, \vec{y}(M)) \in (\mathbb{R}^n)^*$. Assume a local dissimilarity measure $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is available, of which we assume non negativity. d allows us to compare single entries of given times series, but how to extend this measure to a comparison of the full signal? One popular approach, given time series of a fixed length $N = M$ only, is to simply sum over the distances of all entries of the time series. This has two drawbacks: it is applicable for time series of the same length only. Further, it relies on the assumption that the timing of both series is the same, i.e. events which should be compared are located at the same relative time points.

Due to these restrictions, there exist alternative approaches which do take different time scales and lengths into account. The basic idea of dynamic time warping is to align the entries of the two time series consecutively such that their mutual distance is minimum. Thereby, it is assumed that the actual duration of the signal is not relevant, rather its principled form only. Hence it is possible to observe one

entry in X for a longer duration in Y and vice versa. Formally, we obtain the following definition:

Definition 28 Dynamic Time Warping (DTW) *is defined as follows. A* warping path *of length L for time series X and Y is a sequence of indices $P = (p_1, \ldots, p_L) \in (\{1, \ldots, N\} \times \{1, \ldots, M\})^*$ such that*

- $p_1 = (1, 1)$, $P_L = (N, M)$,
- $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ *for all* $l < L$

The costs along the warping path P are defined as

$$d_P(X, Y) = \sum_{l=1}^{L} d(x((p_l)_1), y((p_l)_2))$$

The DTW costs are defined as the costs of an optimum warping path:

$$D(X, Y) = \min_P d_P(X, Y)$$

Hence a warping path aligns the entries of two time series consecutively, taking possible different time frequencies into account. DTW computes the costs along such an optimum alignment.

Attention: DTW does not provide a metric in general. It is symmetric, but it can be 0 without entries being identical. As an example, warping the two time series (x_1, x_2) and (x_1, x_1, x_2, x_2) where the second one is just a longer version of the first one leads to a distance 0. Due to this fact, also the triangle inequality is not necessarily fulfilled. Take as an example the three time series $X = (x_1, x_2, x_3)$, $Y = (x_1, x_2, x_2, x_3)$, $Z = (x_1, x_3, x_3)$ with $D(X, Y) = 0$, $D(X, Z) = 1$, $D(Y, Z) = 2$, assuming costs 1 for mismatches. Hence $D(X, Y) + D(X, Z) < D(Y, Z)$. This price is paid for by the fact that the DTW respects invariance to time delays. We will nevertheless sloppily address DTW as a metric in the following.

Two challenges occur:

- How to compute DTW efficiently?
- How to turn DTW into a valid kernel e.g. for time series classification using support vector machines?

Denote the restriction to the first entries by $X[1 : i] = (x(1), \ldots, x(i))$. The key for an efficient computation of DTW lies in the Bellman equation

$$\begin{array}{rcl} D(X[1:i], Y[1:j]) & = & d(x(i), y(j)) \\ & & + \min\{ \quad D(X[1:i-1], Y[1:j-1]), \\ & & \quad\quad D(X[1:i-1], Y[1:j]), \\ & & \quad\quad D(X[1:i], Y[1:j-1])\} \end{array}$$

which corresponds to the observation, that we have to arrive at an alignment path ending at i and j from one of the preceding three corners, whereby the minimum one gives best results due to the non-negativity of the values. This equation allows us to efficiently compute DTW in quadratic time using dynamic programming as follows:

init $D(0,0) = 0$
init $D(i,0) = D(0,j) = \infty$ for all $i, j > 0$
for $i = 1$ to N do
    for $j = 1$ to M do
        $D(i,j) := d(x(i), y(j)) +$
            $\min\{D(i-1,j-1), D(i-1,j), D(i,j-1)\}$

Having computed the values, we can also restore the optimum warping path by backwards tracking, starting from $D(N,M)$ and tracking the indices corresponding to the respective minimum in the recursion.

There exist variations of DTW which make it even more efficient, and which prevent highly degenerate warping paths, by forcing the path to stay inside a region along the diagonal (i,i).

- The Sakoe-Shiba band restricts the indices (i,j) towards the band $|i-j| \leq T$ for a fixed T. We can enforce this constraint by initializing $D(i,j) = \infty$ for all $|i-j| > T$ and by restricting the loops for its computation to indices inside the band only. Besides avoiding degenerate paths, this accelerates the computational complexity to linear costs.

- The Itakura parallelogram restricts to a parallelogram with slope S along the diagonal. A similar speed-up of the computation is possible, whereby the time is still quadratic (according to the size of the region of the parallelogram).

- Data adaptive restriction by means of multi-scale DTW: the idea is to deal with time series data in a hierarchical fashion, iteratively subsampling time points at time scales of increasing frequency. While the first DTW path is determined in the standard way, the regions are restricted to refinements of the index pairs of the previous warping path in iterative refinements.

There exist principled alternatives to DTW which are similar in nature such as local or global alignment of sequences, which is very common in bioinformatics. Instead of allowing index repetitions, differences in length and timing are accounted for by gaps of the alignment. Unlike DTW, global alignment yields a metric provided the local scoring d has metric properties.

7.2 DTW kernel

DTW itself does not yield a valid distance measure or kernel, thus the question occurs how to efficiently turn it to such a measurement. As already mentioned in the last section, it is possible to use principled preprocessing steps to turn a given matrix of dissimilarities into a valid Gram matrix, thereby changing the values as little as possible. First, a standard way to turn dissimilarities to similarities is offered by double centering (entries g_{ij} of the matrix are computed from distances d_{ij} by the operation $g_{ij} = -0.5(d_{ij} - \sum_k d_{kj}/n - \sum_k d_{ik}/n + \sum_{kk'} d_{kk'}/n^2)$). For euclidean data, this corresponds to a centering of points in 0 and computation of the standard euclidean dot product. In general, the result is a valid kernel iff G is positive semidefinite. If not, corrections of the negative eigenvalues are possible such as

- *clip*: set negative eigenvalues to 0,
- *flip*: flip the sign of negative eigenvalues,
- *shift*: add the absolute size of the minimum negative value to all eigenvalues,
- *square*: consider GG^t instead of G, this corresponds to a feature representation of a point x by the similarity values $(g(x, x_i))_i$ corresponding to a row or column of G.

One very interesting aspect of such preprocessing is the question how we can devise techniques to extend this preprocessing to future data points which have not yet been taken into account in the corrections. One can see the following:

- *clip*: assume the diagonal decomposition $G = UDU^t$ of the Gram matrix. Then, $G_{\text{clip}} = UDCU^t$ with C being a diagonal matrix with entries 1 for non-negative entries of D and 0 otherwise. This yields $G \cdot UCU^t = UDU^tUCU^t = UDCU^t = G_{\text{clip}}$, hence we obtain the clipped matrix from the Gram matrix by means of a simple linear transformation. For an extension, it is easy to apply the same linear transformation UCU^t to the novel similarities which describe the new data points.
- *flip*: similarly, we find $G_{\text{flip}} = UDCU^t$ with C being the diagonal matrix with entries 1 where D is non-negative, and entries -1 for the other diagonal values, hence the transformation corresponds to a linear transformation of the given similarities and a corresponding out-of-sample extension.
- *shift*: unlike the previous preprocessing techniques, shift cannot be realized by a linear mapping. It has been discussed e.g. in the contribution [Chen et al.] that one convenient technology is to simple let similarities

for out-of-sample data unaltered. Due to this fact shift is less common as preprocessing for kernels.

- *square*: For new data x, the feature representation $(g(x, x_i))_i$ and the corresponding similarities as measured in the standard dot product are used.

As an alternative to double centering, the simpler transformation $-D$ or point wise application of the exponential $\exp -d$ are often used to turn dissimilarities to similarities, with according additional preprocessing to guarantee positive semidefiniteness (psd).

For DTW, a similar approach has been directly integrated into the computation technique by Cuturi, the DTW kernel:

Definition 29 *The* global alignment *kernel is defined as*

$$k_{GA}(X, Y) = \sum_{P \text{ warping path}} \exp(-d_P(X, Y))$$

The global alignment kernel corresponds to a soft minimum over all possible warping paths. Thus, it usually provides a more stable similarity measure than DTW itself, since it does not restrict to the minimum alignment path only, but e.g. smoothes outliers by taking all possibilities into account to some degree.

We find

$$\begin{aligned} k_{GA}(X, Y) &= \textstyle\sum_P \exp(-d_P(X, Y)) \\ &= \textstyle\sum_P \prod_l \exp(-d(x((p_l)_1), y((p_l)_2))) \\ &= \textstyle\sum_P \prod_l k(x((p_l)_1), y((p_l)_2)) \end{aligned}$$

where $k(x, y) = \exp(-d(x, y))$. One can show that this is a kernel provided $k/(1 + k)$ is positive semidefinite on the given data entries. (A rather technical proof can be found in [Cuturi,Vert et al.].) Cuturi et al. propose to use, for example, a measure of the form $s/(1 - s)$ for a given kernel s since $(s/(1 - s))/(1 + (s/(1 - s))) = s$ yields the original kernel s back. As an example, this gives $k(x, y) = 1/2 \exp(-|x - y|^2/\sigma^2)/(1 - 1/2 \exp(-|x - y|^2/\sigma^2))$ for the Gaussian kernel. In practice, as pointed out in [Cuturi,Vert et al.], most practically relevant kernels s have the property that also $s/(1 + s)$ is positive semidefinite for a given finite data et.

How to compute k_{GA} efficiently? It can be shown that a similar Bellmann equation as for DTW holds:

Theorem 30 *Define* $M(ij) := k_{GA}(X[1 \dots i], Y[1 \dots j])$. *Then*

$$M(ij) = (M(i, j - 1) + M(i - 1, j - 1) + M(i - 1, j)) \cdot k(x_i, y_j)$$

holds.

This claim follows because of the following identity:

$$\begin{aligned}
M(i,j) &= k_{GA}(X[1\ldots i], Y[1\ldots j]) \\
&= \textstyle\sum_{P \text{ path to } (i,j)} \prod_{l \le L_P} k(x((p_l)_1), y((p_l)_2)) \\
&= k(x_i, y_j) \cdot \textstyle\sum_{P \text{ path to } (i,j)} \prod_{l < L_P} k(x((p_l)_1), y((p_l)_2)) \\
&= k(x_i, y_j) \cdot \Big(\textstyle\sum_{P \text{ path to } (i-1,j)} \prod_{l \le L_P} k(x((p_l)_1), y((p_l)_2)) \\
&\qquad + \textstyle\sum_{P \text{ path to } (i,j-1)} \prod_{l \le L_P} k(x((p_l)_1), y((p_l)_2)) \\
&\qquad + \textstyle\sum_{P \text{ path to } (i-1,j-1)} \prod_{l \le L_P} k(x((p_l)_1), y((p_l)_2)) \Big) \\
&= k(x_i, y_j) \cdot (M(i-1,j) + M(i,j-1) + M(i-1,j-1))
\end{aligned}$$

This allows an efficient computation of the kernel by means of dynamic programming leading to a quadratic time complexity.

Similar to the Sakoe-Shiba bound for DTW, the question occurs whether a restriction of the kernel to paths for which the indices fulfill the relation $|i-j| \le T$ is possible. The problem is to find a formalization which yields a valid kernel. It has been proposed by Cuturi to pick the following band weighting kernel on $\mathbb{N}$: $w(i,j) = (1 - |i-j|/T)_+$. Provided a kernel s on real vectors, we can use the tensor kernel $s \otimes w$ on the space $\mathbb{R}^n \otimes \mathbb{N}$. Taking $s \otimes w / (1 - s \otimes w)$ we end up with a band limited kernel for time series with positioning $((x(1),1), \ldots, (x(N),N))$ induced by the Gaussian kernel of the form

$$\sum_P \prod_l \frac{w((p_l)_1, (p_l)_2) \exp(-|x((p_l)_1) - y((p_l)_2)|^2/\sigma^2)}{2 - w((p_l)_1, (p_l)_2) \exp(-|x((p_l)_1) - y((p_l)_2)|^2/\sigma^2))}$$

which is restricted to nonzero entries for indices $|i-j| \le T$ only. Since the same optimality equation as before holds, this allows us to compute the kernel in linear time only by restricting to indices where the kernel is non vanishing.

7.3 Structure kernel

There exist principled alternatives which allow us to construct kernels for time series or more general data structures based on general principles. It has been pointed out by Haussler that kernels are closed with respect to a number of operations such as

- multiplication by positive constant,
- exponentiation,
- pointwise product
- pointwise limit

- projection
- ...

These operations allow us to construct complex kernels from simple basic constituents. Popular examples for such kernels which have been proposed for sequences are the following:

- *String kernel*: there exist different kernels which essentially count and weight the number of common subsequences of two given (symbolic) time series. The string subsequence kernel computes the term $\varphi(X) \cdot \varphi(Y)$ where $\varphi(X)$ extracts all not necessarily contiguous substrings from a given series, mapping to the feature weights $\sum_{\text{index sequence i yields s}} \lambda^{\text{length(i)}}$ for some $\lambda \in (0, 1)$. Hence two sequences are similar if they contain the same short substrings. It is obvious per definition that this yields a valid kernel, but its computational efficiency is not clear. [Lodhi et al.] develop a dynamic programming scheme to compute the kernel efficiently taking the length of the sequences as iteration quantity.

  As an alternative, the spectrum kernel introduced by [Leslie et al.] counts only common contiguous substrings of length k. Then, computation of the kernel can be done efficiently in linear time relying on suffix trees for the occurring substrings.

- *Fisher kernel*: The Fisher kernel is based on the assumption that a probabilistic model is available to describe the given data $p(X, \theta)$. For time series, one method of choice can be a hidden Markov model (HMM), for example. The parameters θ have been trained such that the model is representative for the given data, e.g. by optimizing the average log likelihood. Then, instead of comparing the data X and Y (which is not directly possible due to the different structure), we compare characteristics of the probabilistic model as concerns the observed data. More precisely, we compare the gradient of the log likelihood of each data point, the so-called Fisher score:

  $$\frac{\partial \mathcal{L}(X)}{\partial \theta} \cdot U^{-1} \cdot \frac{\partial \mathcal{L}(Y)}{\partial \theta}$$

  where $\mathcal{L}$ is the data log likelihood induced by $p.$ and U denotes the Fisher information matrix

  $$E_{p(X,\theta)} \left(\frac{\partial \mathcal{L}(X)}{\partial \theta}^t \frac{\partial \mathcal{L}(X)}{\partial \theta} \right)$$

  which locally scales the manifold of model parameters according to its influence on the data. Often, the Fisher matrix is even dropped and the simple euclidean norm is taken to compare the tangent vectors.

These are three examples of time series kernels, which can deal with time series of different sizes. The differences of the three methods are as follows:

- DTW kernel: can take into account time different time scales. It is suitable for continuous vectors. Quadratic complexity, but speed-up to linear time by windowing is possible.
- String kernel: relies on (contiguous or non-contiguous) substrings shared by the sequences. In its original form it has been proposed for symbolic sequences. It has quadratic complexity if dynamic programming is used, suffix trees scale this down to linear complexity.
- Fisher-kernel: general principle to turn a probabilistic model into kernels. With HMM, it provides a kernel which takes into account the dependency of the model parameterization with respect to the considered data. Can be used for continuous or discrete time series. The complexity depends on the computation of the model likelihood, for trained models the model complexity determines the effort. For HMM using dynamic programming, the likelihood can be computed in linear time.

7.4 Time windows

Despite these approaches, a simple time window technique is still by far the most prominent technique to deal with time series data: instead of a full time series, a time window of fixed size is considered, for which the simple euclidean distance is often chosen. There are quite a few research approaches which discuss the feasibility of this simple approach, such as:

- proposals of more suitable metrics which take the time structure into account, e.g. as proposed by Lee/Verleysen,
- discussions of the meaninglessness of clustering algorithms based on sliding windows. Essentially, this technique relies on the observation that the statistics smoothens if every position of a long time series appears at every position of the fixed time windows due to the sliding window approach. If one prototype is used, the result will be determined by the average over all entries (up to small differences at the borders). This has been made more formal, accompanied by experiments, by Keogh et al.
- Discussions on how preprocessing can help using what is called unfolding, i.e. choosing not subsequent points but a suitable time lag τ for time windows of the form $[x_t, x_{t+\tau}, x_{t+2\tau}, \ldots]$. This way, trivial matches of time windows due to statistical effects are avoided, by choosing τ in a way to

minimize the correlations of subsequent data in the window. A formalization and experiments have been reported by Simon/Lee/Verleysen. See Fig. 26

- One basic observation, which underlies the popularity of time series window strategies, is due to the classical Taken's embedding theorem: this claims that every attractor of a dynamic system can be recovered by delayed observations $[x(t), x(t+\tau), \ldots, x(t+2n\tau)]$ where n is the dimensionality of the manifold, and τ is appropriately chosen as regards orbits of the system. The problem is how to estimate n and how to choose τ appropriately, with some recipes existing in classical time series analysis.

Literature

- Elzbieta Pekalska and Robert P.W. Duin, The Dissimilarity Representation for Pattern Recognition. Foundations and Applications. World Scientific, Singapore, December 2005.
- Fast Global Alignment Kernels, In Proceedings of the 28th International Conference on Machine Learning (ICML-11) (June 2011), pp. 929-936 by Marco Cuturi
- Cuturi, Vert, Birkendes, Matsui, A kernel for time series based on alignments, arXiv:cs/0610033v1
- Yihua Chen, Eric K. Garcia, Maya R. Gupta, Ali Rahimi, and Luca Cazzanti. 2009. Similarity-based Classification: Concepts and Algorithms. J. Mach. Learn. Res. 10 (June 2009), 747-776
- T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In Advances in Neural Information Processing Systems 11, 1998.
- Convolution Kernels on Discrete Structures Haussler, 1999, Technical Report
- Text classification using string kernels, Lodhi, Saunders, Shawe-Taylor, Christianini, Watkins, JMLR 2, 2002
- The spectrum kernel: a string kernel for SVM protein classification, Leslie, Eskin, Noble, 2002.

- Soeren Sonnenburg, Gunnar Raetsch, Sebastian Henschel, Christian Widmer, Jonas Behr, Alexander Zien, Fabio de Bona, Alexander Binder, Christian Gehl, and Vojtech Franc. The SHOGUN Machine Learning Toolbox. Journal of Machine Learning Research, 11:1799-1802, June 2010. http://www.shogun-toolbox.org/page/home/

- Generalization of the Lp norm for time series and its application to Self-Organizing Maps J.A. Lee, M. Verleysen, WSOM'05

- Clustering of Time Series Subsequences is Meaningless: Implications for Past and Future Research (2003) Keogh, Lin, IEEE ICDM

- Simon, Geoffroy ; Lee, John Aldo ; Verleysen, Michel. Unfolding preprocessing for meaningful time series clustering.. In: Neural networks : the official journal of the International Neural Network Society, Vol. 19, no. 6-7, p. 877-88 (2006)

- Takens, F. (1981) Detecting strange attractors in turbulence. Lecture Notes in Mathematics 898, Berlin:Springer-Verlag

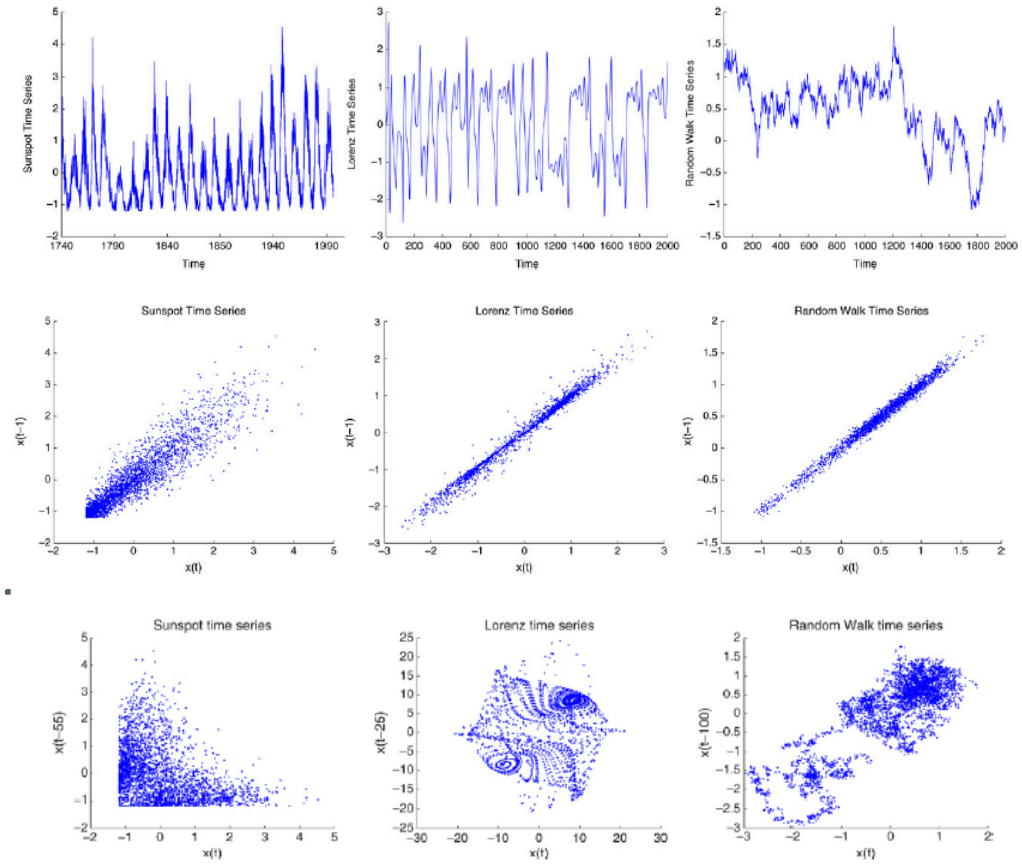


Figure 26: Time series data (top) and their embedding using time lag 1 (middle) or optimum time lag according to uncorrelatedness of consecutive entries (bottom). Images are taken from Simon/Lee/Verleysen, Unfolding preprocessing for meaningful time series clustering

8 Gaussian Processes

Gaussian Processes (GPs) have been proposed around 1996 by MacKay, Rasmussen, Williams in the area of machine learning, but they have been around e.g. in the area of geostatistics earlier under different names. With machine learning being fundamentally linked to probabilistic modelling, GPs are contained in standard textbooks on machine learning as long as the latter take a probabilistic view such as the popular textbooks from Bishop or Murphy. There exists an excellent textbook about Gaussian processes from Rasmussen/Williams. Nevertheless, GPs definitely constitute an active area of research (and they are highly non trivial) such that their place in this lecture is justified ;-) We will focus on the basics (i.e. regression) in this summary to point out the underlying fundamental motivation.

The principled aim of GPs is to get rid of any parameterization of the models at all, but to substitute this by a prior over all possible functions or models rather than an explicit parametric form. Coming from a classical Bayesian modeling of regression, one can see that this is actually feasible provided a few assumptions hold.

Classical Bayesian linear regression

Assume there are given points $\vec{x}_i \in \mathbb{R}^n$ and outputs $y_i \in \mathbb{R}$; denote X and Y for the matrix of inputs and vector of outputs. We would like to learn a linear regression

$$f(\vec{x}) = \vec{w}^t \vec{x}$$

based on these data (neglecting offsets as usual). We assume that the outputs are observations

$$y_i = f(\vec{x}_i) + \epsilon_i$$

where ϵ_i is noise which is i.i.d. according to $\mathcal{N}(0, \sigma_n^2)$. $\mathcal{N}$ refers to the Gaussian distribution with center and variance as paramaters. How to determine the weights? *Bayes rule* yields

$$p(\vec{w}|Y, X) = \frac{p(Y|X, \vec{w})p(\vec{w})}{p(Y|X)}$$

We find for the *marginal likelihood*

$$p(Y|X) = \int p(Y|X, \vec{w})p(\vec{w})d\vec{w}$$

which is independent of the weights, hence a normalization only. For the *weight prior* we assume a Gaussian distribution

$$p(\vec{w}) \sim \mathcal{N}(0, \Sigma_p)$$

with zero mean and a fixed covariance matrix Σ_p, often a simple diagonal matrix $\delta \cdot \mathbf{I}$ (this corresponds to an independence of the weight entries). Then the *likelihood* equals a multivariate Gaussian

$$\begin{aligned} p(Y|X,\vec{w}) &= \prod_i p(y_i|\vec{x}_i,\vec{w}) \\ &= \prod_i \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{(y_i - \vec{w}^t\vec{x}_i)^2}{2\sigma_n^2}\right) \\ &= \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp\left(-\frac{1}{2\sigma_n^2}\|Y - X^t\vec{w}\|^2\right) \\ &= \mathcal{N}(X^t\vec{w}, \sigma_n^2\mathbf{I}) \end{aligned}$$

due to the independence assumption. Therefore, we find

$$\begin{array}{l} p(\vec{w}|Y,X) = \\ \text{norm} \cdot \exp\left(-\frac{1}{2}\vec{w}^t\Sigma_p^{-1}\vec{w}\right) \cdot \exp\left(-\frac{1}{2\sigma_n^2}\|Y - X^t\vec{w}\|^2\right) = \\ \text{norm} \cdot \exp\left(-\frac{1}{2}(\vec{w} - \vec{w}_m)^t(XX^t/\sigma_n^2 + \Sigma_p^{-1})(\vec{w} - \vec{w}_m)\right) = \Sigma_n^{-2}A^{-1}XY \\ \mathcal{N}(\vec{w}_m, A^{-1}) \end{array}$$

where

$$\vec{w}_m = \sigma_n^{-2}(\sigma_n^{-2}XX^t + \Sigma_p^{-1})^{-1}XY =,$$
$$A = XX^t/\sigma_n^2 + \Sigma_p^{-1}$$

and norm is a normalization factor. To see this equality, one can compute for the exponents of the Gaussians (with 'const' referring to terms without $\vec{w}$ which just contribute to normalization factors for the Gaussians):

$$\begin{aligned} & -\tfrac{1}{2}(\vec{w} - \vec{w}_m)^t A(\vec{w} - \vec{w}_m) \\ = \; & -\tfrac{1}{2}\vec{w}^t A\vec{w} + \vec{w}^t A\vec{w}_m - \tfrac{1}{2}\vec{w}_m^t A\vec{w}_m \\ = \; & -\tfrac{1}{2}\vec{w}^t\Sigma_p^{-1}\vec{w} - \tfrac{1}{2\sigma_n^2}\vec{w}^t XX^t\vec{w} + \tfrac{1}{\sigma_n^2}\vec{w}^t AA^{-1}XY + \text{const} \\ = \; & -\tfrac{1}{2}\vec{w}^t\Sigma_p^{-1}\vec{w} - \tfrac{1}{2\sigma_n^2}\vec{w}^t XX^t\vec{w} + \tfrac{1}{\sigma_n^2}\vec{w}^t XY - \tfrac{1}{2\sigma_n^2}Y^tY + \text{const} \\ = \; & -\tfrac{1}{2}\vec{w}^t\Sigma_p^{-1}\vec{w} - \tfrac{1}{2\sigma_n^2}\|Y - X^t\vec{w}\|^2 + \text{const} \end{aligned}$$

Thus, under these assumptions, we have a closed form expression for the likelihood of the parameters of a linear function given the observed data. There exist different classical ways to infer a function prescription from this form:

- **MAP:** Maximum a posteriori estimate of the weights is often underlying classical parametric prediction. The weights $\vec{w}$ with maximum likelihood are taken, i.e.

$$\vec{w}_{\text{opt}} = \vec{w}_m = \sigma_n^{-2}(\sigma_n^{-2}XX^t + \Sigma_p^{-1})^{-1}XY$$

which yields the prediction

$$\vec{x} \mapsto \vec{w}_m^t \vec{x} = ((XX^t + \sigma_n^2 \Sigma_p^{-1})^{-1} XY)^t \vec{x}$$

which corresponds to standard ridge regression.

- **Bayesian prediction:** Bayesian prediction takes into account all possible weights weighted according to their likelihood. We obtain an explicit expression of the probability of a value $f(\vec{x})$ for a given data point $\vec{x}$ and the observed data as follows:

$$\begin{aligned} & p(f(\vec{x})|\vec{x}, X, Y) \\ = & \int p(f(\vec{x})|\vec{x}, \vec{w}, X, Y) \cdot p(\vec{w}|X, Y) d\vec{w} \\ = & \int p(f(\vec{x})|\vec{x}, \vec{w}) \cdot p(\vec{w}|X, Y) d\vec{w} \end{aligned}$$

because the result of f for $\vec{x}$ is fixed once we know $\vec{w}$. Note that $p(f(\vec{x})|\vec{x}, \vec{w})$ is peaked at $\vec{w}^t \vec{x}$, further we can express the weight distribution as Gaussian $\mathcal{N}(\vec{w}_m, A^{-1})$. Since the linear transformation of a Gaussian is again a Gaussian, we arrive at the closed form solution

$$p(f(\vec{x})|\vec{x}, X, Y) = \mathcal{N}\left(\frac{1}{\sigma_n^2} \vec{x}^t A^{-1} XY, \vec{x}^t A^{-1} \vec{x}\right)$$

This expression has the advantage that we obtain an explicit distribution for all outputs: e.g. we can can not only determine the most likely value of the function but also the confidence of any interval.

Kernelization

As usual, linear functions are often not sufficient for modeling real systems. Therefore, we use our standard trick, we consider a kernel mapping $k(\vec{x}, \vec{y}) = \Phi(\vec{x})^z \Phi(\vec{y})$ for a given kernel k with underlying feature map Φ. We can plug a kernel into the linear mapping and obtain the following distribution in a Bayesian model

$$\mathcal{N}\left(\frac{1}{\sigma_n^2} \Phi(\vec{x})^t A^{-1} \Phi Y, \Phi(\vec{x})^t A^{-1} \Phi(\vec{x})\right)$$

where Φ refers to the matrix of training inputs X preprocessed using Φ, and

$$A = \sigma_n^{-1} \Phi \Phi^t + \Sigma_p^{-1}$$

One problem of this expression is that it is not yet in kernelized form, in the sense that we cannot compute the distribution based on k only, but we still need Φ. For a kernelized form, we have to restrict to terms $\Phi^t \Phi$ only. Further, inversion

of the matrix A is rather costly / infeasible because its dimensionality equals the (possibly infinite) dimensionality of the feature space.

Thus, we rearrange the terms. Define

$$K = \Phi^t \Sigma_p \Phi$$

The dimensionality of K is only the number of data, hence it is much easier to invert than A. Further, K is a kernel matrix per definition. We will later use this K instead of the original kernel k, which corresponds to the assumption that the noise prior Σ_p for the weights is already a part of the kernel and it need not be added separately.

We find:

$$\sigma_n^{-2}\Phi(K + \sigma_n^2\mathbf{I}) = \sigma_n^{-2}\Phi(\Phi^t\Sigma_p\Phi + \sigma_n^2\mathbf{I}) = A\Sigma_p\Phi$$

hence multiplying by $(K+\sigma_n^2\mathbf{I})^{-1}$ from the right and A^{-1} from the left, we obtain

$$\sigma_n^{-2}A^{-1}\Phi = \Sigma_p\Phi(K + \sigma_n^2\mathbf{I})^{-1}$$

Hence the mean value of the Gaussian can be expressed as

$$\sigma_n^2\Phi(\vec{x})^t A^{-1}\Phi Y = \Phi(\vec{x})^t\Sigma_p\Phi(K + \sigma_n^2\mathbf{I})^{-1}Y = K(\vec{x})(K + \sigma_n^2\mathbf{I})^{-1}Y$$

which refers to evaluations of the 'new' kernel K at $\vec{x}$ and the training data, and a matrix inversion of the smaller kernel matrix K only, denoting $K(\vec{x}) = \Phi(\vec{x})^t\Sigma_p\Phi$. Hence this allows an 'efficient' computation of the mean value (efficient meaning cubic time, theoretical computer science would call this efficient, machine learning probably not if training sets become too big ...).

Similarly, we can derive an efficient computation of the covariance matrix by relying on the so-called matrix inversion lemma:

Lemma 31 (Matrix inversion lemma) *For $Z \in \mathbb{R}^{n\times n}$, $W \in \mathbb{R}^{m\times m}$, and $U, V \in \mathbb{R}^{n\times m}$ we find*

$$(Z + UWV^t)^{-1} = Z^{-1} - Z^{-1}U(W^{-1} + V^tZ^{-1}U)^{-1}V^tZ^{-1}$$

The matrix inversion lemma has been designed to allow an efficient inversion of a matrix which is subject to a low rank perturbation only. Here, we set $Z = \Sigma_p^{-1}$, $W^{-1} = \sigma_n^2\mathbf{I}$, $V = U = \Phi$. Hence

$$(\Sigma_p^{-1} + \Phi\sigma_n^{-2}\Phi^t)^{-1} = \Sigma_p - \Sigma_p\Phi(\sigma_n^2\mathbf{I} + \Phi^t\Sigma_p\Phi)^{-1}\Phi^t\Sigma_p$$

Hence the following equation expresses the covariance matrix

$$\begin{aligned} & \Phi(\vec{x})^t A^{-1} \Phi(\vec{x}) \\ = \; & \Phi(\vec{x})^t (\sigma_n^{-2} \Phi \Phi^t + \Sigma_p^{-1})^{-1} \Phi(\vec{x}) \\ = \; & \Phi(\vec{x})^t (\Sigma_p - \Sigma_p \Phi (\sigma_n^2 \mathbf{I} + \Phi^t \Sigma_p \Phi)^{-1} \Phi^t \Sigma_p) \Phi(\vec{x}) \\ = \; & \Phi(\vec{x})^t \Sigma_p \Phi(\vec{x}) - \Phi(\vec{x})^t \Sigma_p \Phi (K + \sigma_n^2 \mathbf{I})^{-1} \Phi^t \Sigma_p \Phi(\vec{x}) \\ = \; & K(\vec{x}) - K(\vec{x})^t (K + \sigma_n^2 \mathbf{I})^{-1} K(\vec{x}) \end{aligned}$$

This expression uses the kernel only, hence it can be computed efficiently (meaning in cubic time). Thus, we have a closed form Bayesian solution of kernelized linear regression assuming a reasonable statistical model as detailed above.

Gaussian processes

Bayesian modeling relies on a prior on the weights and an explicit parametric modeling of the function. The idea behind Gaussian process regression is to get rid of the need of an explicit parametric form, but to do inference in the space of functions and to directly impose a *prior over functions* instead. For this purpose, the basic Bayesian expression is evaluated as

$$p(\vec{y}|\vec{x}, X, Y) = \int p(\vec{y}|f, \vec{x}) p(f|X, Y) df$$

where integration is done over the space of functions f instead of model parameters. This requires a suitably equipped space of functions where such integrals can be evaluated. Typically, one restricts to squared integrable functions and the operator $\langle f, g \rangle = \int \overline{f(x)} g(x) dx$ which induces a corresponding σ-algebra and a measurable space. We will not elaborate about details as regards these issues but we will assume that all terms and definitions which we will use are possible.

Regarding the above formula, we need to evaluate $p(\vec{y}|f, \vec{x})$. This is directly given by $f(\vec{x})$ and a possibly additional noise model. In addition, we need to evaluate $p(f|X, Y)$ where we will rely on Bayes rule, as before. Thus, we need to specify a suitable prior $p(f)$ over functions.

Definition 32 *A* **Gaussian process** *is a prior distribution over functions $f : \mathbb{R}^N \to \mathbb{R}$ which is defined as follows:*

- *we have a collection of random variables which are indexed by the variables $\vec{x}_i$, such that for any finite collection of variables $\vec{x}_1, \ldots, \vec{x}_n$ the distribution $(f(\vec{x}_1), \ldots, f(\vec{x}_n))$ is a joint Gaussian distribution,*
- *the mean value of variable $f(\vec{x}_i)$ is defined by $m(\vec{x}_i) = E(f(\vec{x}_i))$,*

- *the covariance matrix entry for $\vec{x}_i$ and $\vec{x}_j$ is defined as $k(\vec{x}_i, \vec{x}_j) = E((f(\vec{x}_i) - m(\vec{x}_i)) \cdot (f(\vec{x}_j) - m(\vec{x}_j)))$, where k is a suitable kernel to ensure symmetry and positive definiteness of the matrix.*

We write

$$f \sim GP(m(\vec{x}), k(\vec{x}, \vec{x}'))$$

for short.

The rationality behind this definition is that it is actually not necessary to define the functions as a whole, but it is sufficient to define the functions evaluated for every possible finite set of inputs only. Then, a distribution of the functions evaluated at a finite set of points corresponds to a distribution over the finite set of points only. The assumption of Gaussianity specifies the probability of the underlying functions: the function values are centered around some value $m(\vec{x})$ with Gaussian noise. Correlations of neighbored entries $\vec{x}$ and $\vec{x}'$ are specified by the measure k which must be chosen to give a valid covariance matrix for every set of entries.

Actually, we already know which functions fulfill these conditions: we have to ensure symmetry and positive definiteness of any matrix which arises from a finite set of entries, this is precisely what we refer to as *kernel function* k!

Why is such a definition a valid definition of a prior over functions?

We have to guarantee the *consistency/marginalization property*: for any subset of points $p(f(\vec{x}_{i_1}), \ldots, f(\vec{x}_{i_l}))$ must be consistent with $p(f(\vec{x}_1), \ldots, f(\vec{x}_n))$. This means that we arrive at the former probability by marginalizing over all entries not contained in the set. For two variable sets, the marginal is

$$p(\vec{x}) = \int p(\vec{x}, \vec{y}) d\vec{y}$$

If we plug in Gaussians, this reads as

$$(\vec{x}, \vec{y}) \sim \mathcal{N}\left(\begin{pmatrix} \vec{m}(\vec{x}) \\ \vec{m}(\vec{y}) \end{pmatrix}, \begin{pmatrix} A & C \\ C^t & B \end{pmatrix}\right) \Rightarrow \vec{x} \sim \mathcal{N}(\vec{m}(\vec{x}), A)$$

hence Gaussians defined via its mean and covariance matrix automatically fulfill this property! Usually, we set $m(f(\vec{x})) = 0$ (one can use a different prior if e.g. a trend of f is expected). Further, $k(f(\vec{x}), f(\vec{x}'))$ is chosen as a kernel such as the Gaussian kernel

$$k(f(\vec{x}), f(\vec{x}')) = \sigma_f^2 \exp(-\frac{1}{2l^2}|\vec{x} - \vec{x}'|^2) + \sigma_n^2 \delta_{\vec{x},\vec{x}'}$$

Why is a Gaussian prior reasonable?

Consider $f(\vec{x}) = \Phi(\vec{x})^t \vec{w}$ with prior $\vec{w} \sim \mathcal{N}(0, \Sigma_p)$. Note that linear transformations of Gaussians are Gaussian, hence we observe a joined Gaussian distribution of the function values. We find for the expectation

$$E(f(\vec{x})) = \Phi(\vec{x})^t E(\vec{w}) = 0$$

and for the correlation

$$E(f(\vec{x})f(\vec{x}')) = \Phi(\vec{x})^t E(\vec{w}\vec{w}^t)\Phi(\vec{x}') = \Phi(\vec{x})^t \Sigma_p \Phi(\vec{x}')$$

Hence we arrive at a joint Gaussian distribution with zero mean and covariance matrix with entries

$$\Phi(\vec{x})^t \Sigma_p \Phi(\vec{x}')$$

That means, Gaussian process priors correspond to generalized linear mappings with feature map Φ!

Gaussian process regression

Noise free case

Assume $\vec{f}$ is the vector of observations for X, $\vec{f_*}$ is the vector of inference points X_*. Assume a kernel k is chosen for the Gaussian process prior, we refer to matrices given by k with K. Due to the model, we assume a Gaussian distribution

$$p(\vec{f}, \vec{f_*}|X, X_*) \sim \mathcal{N}\left(0, \begin{pmatrix} K(X,X) & K(X_*,X) \\ K(X_*,X)^t & K(X_*,X_*) \end{pmatrix}\right)$$

We are interested in the conditional distribution $p(\vec{f_*}|X_*, X, \vec{f})$.

Lemma 33 Conditioning of Gaussians: *Assume*

$$p(\vec{x}, \vec{y}) \sim \mathcal{N}\left(\begin{pmatrix} \vec{m}(\vec{x}) \\ \vec{m}(\vec{y}) \end{pmatrix}, \begin{pmatrix} A & C \\ C^t & B \end{pmatrix}\right)$$

Then

$$p(\vec{x}|\vec{y}) \sim \mathcal{N}(\vec{m}(\vec{x}) + CB^{-1}(\vec{y} - \vec{m}(\vec{y})), A - CB^{-1}C^t)$$

This can be computed as follows:

$$p(\vec{x}, \vec{y}) \sim$$
$$\exp\left(-\tfrac{1}{2}\begin{pmatrix} \vec{x} - \vec{m}(\vec{x}) \\ \vec{y} - \vec{m}(\vec{y}) \end{pmatrix}^t \begin{pmatrix} A & C \\ C^t & B \end{pmatrix} \begin{pmatrix} \vec{x} - \vec{m}(\vec{x}) \\ \vec{y} - \vec{m}(\vec{y}) \end{pmatrix}\right) =$$
$$\exp\left(-\tfrac{1}{2}\begin{pmatrix} \vec{x} - \vec{m}(\vec{x}) \\ \vec{y} - \vec{m}(\vec{y}) \end{pmatrix}^t \begin{pmatrix} I & 0 \\ -B^{-1}C^t & I \end{pmatrix} \begin{pmatrix} M/B & 0 \\ 0 & B \end{pmatrix} \begin{pmatrix} I & -CB^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} \vec{x} - \vec{m}(\vec{x}) \\ \vec{y} - \vec{m}(\vec{y}) \end{pmatrix}\right)$$
$$= (*)$$

with $(M/B) := A - CB^{-1}C^t$ the so-called Schur-complement of $M = \begin{pmatrix} A & C \\ C^t & B \end{pmatrix}$
This is valid because of

$$\begin{pmatrix} I & -CB^{-1} \\ 0 & I \end{pmatrix} \cdot M = \begin{pmatrix} A - CB^{-1}C^t & 0 \\ C^t & B \end{pmatrix}$$

and

$$\begin{pmatrix} A - CB^{-1}C^t & 0 \\ C^t & B \end{pmatrix} \cdot \begin{pmatrix} I & 0 \\ -B^{-1}C^t & I \end{pmatrix} = \begin{pmatrix} A - CB^{-1}C^t & 0 \\ 0 & B \end{pmatrix}$$

hence, using the matrix equality $XMY = Z \Rightarrow M^{-1} = YZ^{-1}X$

$$M^{-1} = \begin{pmatrix} I & 0 \\ -B^{-1}C^t & I \end{pmatrix} \begin{pmatrix} A - CB^{-1}C^t & 0 \\ 0 & B \end{pmatrix}^{-1} \begin{pmatrix} I & -CB^{-1} \\ 0 & I \end{pmatrix}$$

Thus, we can separate the term into two factors

$(*) =$
$\exp\left(-\frac{1}{2}(\vec{x} - \vec{m}(\vec{x}) - CB^{-1}(\vec{y} - \vec{m}(\vec{y}))^t (M/B)^{-1}(\vec{x} - \vec{m}(\vec{x}) - CB^{-1}(\vec{y} - \vec{m}(\vec{y}))\right)$
$\cdot \exp\left(-\frac{1}{2}(\vec{y} - \vec{m}(\vec{y}))^t B^{-1}(\vec{y} - \vec{m}(\vec{y}))\right)$
$\sim p(\vec{x}|\vec{y}) \cdot p(\vec{y})$

where $p(\vec{y}) \sim \mathcal{N}(\vec{m}(\vec{y}), B)$ and

$$p(\vec{x}|\vec{y}) \sim \mathcal{N}(\vec{m}(\vec{x}) + CB^{-1}(\vec{y} - \vec{m}(\vec{y})), A - CB^{-1}C^t)$$

Hence we find from

$$p(\vec{f}, \vec{f_*}|X, X_*) \sim \mathcal{N}\left(0, \begin{pmatrix} K(X,X) & K(X_*,X) \\ K(X_*,X)^t & K(X_*,X_*) \end{pmatrix}\right)$$

by using this formula that $p(\vec{f_*}|X_*, X, \vec{f})$ is distributed according to the Gaussian

$$\mathcal{N}(K(X,X_*)^t K(X,X)^{-1}\bar{f}, K(X_*,X_* - K(X,X_*^t)K(X,X)^{-1}K(X,X_*))$$

This is the model inference: given an observation $X, \bar{f}$, we infer the distribution of the function values for X_* using this prior over all functions as Gaussian with mean

$$K(X,X_*)^t K(X,X)^{-1}\bar{f}$$

and covariance matrix as above. Note that this corresponds to noise free generalized linear regression with kernel K!

The case of noise

We assume a vector of observations

$$Y = \bar{f} + \vec{\epsilon}$$

given data X where we assume i.i.d. noise with variance σ_n^2. The covariance matrix of the Gaussian process becomes for entries $\vec{x}$ and $\vec{x}'$:

$$k(\vec{x}, \vec{x}') + \sigma_n^2 \delta_{\vec{x},\vec{x}'}$$

Thus, the prior distribution is given as

$$p(Y, \vec{f}_*|X_*, X) \sim \mathcal{N}\left(0, \begin{pmatrix} K(X,X) + \sigma_n^2 \mathbf{I} & K(X, X_*) \\ K(X, X_*)^t & K(X_*, X_*) \end{pmatrix}\right)$$

Conditioning as above yields the inference: $p(\bar{f}_*|X_*, X, Y)$ is distributed according to

$$\mathcal{N}(K(X,X_*)^t(K(X,X)+\sigma_n^2\mathbf{I})^{-1}\bar{f}, K(X_*,X_*-K(X,X_*^t)(K(X,X)+\sigma_n^2\mathbf{I})^{-1}K(X,X_*))$$

We can compare this result to Bayesian inference for a generalized linear model with kernel $K = \Phi^t \Sigma_p \Phi$: we obtain exactly the same result! Hence Bayesian inference for generalized linear models and GP regression are identical under these assumptions!

Note that for a single point $\vec{x}$ we obtain the mean $k^t(K + \sigma_n^2\mathbf{I})^{-1}Y$ which is a linear function in the kernel values $k = (k(\vec{x}, \vec{x}_i))_i$ and the variance $k(\vec{x}, \vec{x}) - k^t(K + \sigma_n^2\mathbf{I})^{-1}k$ which is a quadratic form in k.

Now GP regression proceeds algorithmically as follows:

Gaussian Process Regression:
choose: covariance function given by kernel k, noise of measurement σ_n^2
input: X, Y data, test point $\vec{x}$
compute $A^{-1} := (K + \sigma_n^2 I)^{-1}$ via Cholesky decomposition
compute mean output $(k(\vec{x}, \vec{x}_i))_i^t A^{-1} Y$
compute variance $k(\vec{x}, \vec{x}) - (k(\vec{x}, \vec{x}_i))_i^t A^{-1} (k(\vec{x}, \vec{x}_i))_i$

See Fig. 27 for an example result for a simple GP regression for three data points.

As already said, a typical kernel is the Gaussian kernel with the form

$$k(\vec{x}, \vec{x}') = \sigma_f^2 \exp(-1/(2l^2)(\vec{x} - \vec{x}')^2) + \sigma_n^2 \delta_{\vec{x},\vec{x}'}$$

where the kernel parameters define the prior of the resulting functions, and, thus, have a severe influence on the outcome in particular for small samples sizes. The influence is as follows:

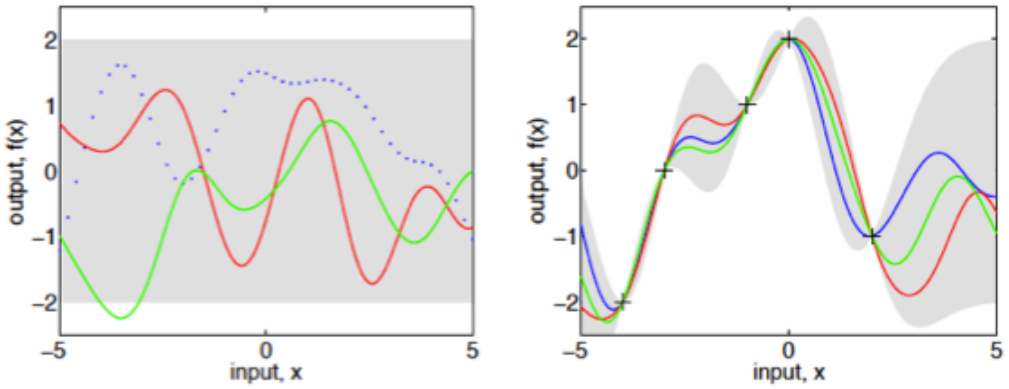


Figure 27: GP regression. On the left, three functions are chosen randomly according to the prior; the dots depict the generated points for one of the examples. Grey indicates the 95% confidence interval which is flat due to a zero mean. On the right, the GP distribution is shown when conditioning for five observed values in the noise free case. The functions must go through the observed values. The confidence interval is reduced according to the closeness from observed data due to a smooth underlying kernel. Images are taken from [1].

- kernel width l: for large l, functions vary smoothly since far away points still have a high correlation. For small l, a more rapid change of the function is encouraged.
- σ_n: for large noise variance, the observation are expected to be noisy, i.e. a large deviation from the mean value is tolerated especially for the observed values. In the limit of vanishing variance, the observed values must be interpolated.
- σ_f: this scales the deviation of the values from the expectation as an overall range, i.e. it scales the width of the confidence interval.

The influence of different parameter choices is depicted in Fig. 28.

What are benefits of GPs as compared to a simple parametric Bayesian treatment of the data using an according kernel. We can immediately observe the following:

- GPs are more costly in its standard form, being cubic w.r.t number of data.
- The results of GPs depends on parameter choices similar to Bayesian regression: the choice of the kernel determines the result, hence for (usually) unknown parameters we have the same problems as for classical regression.

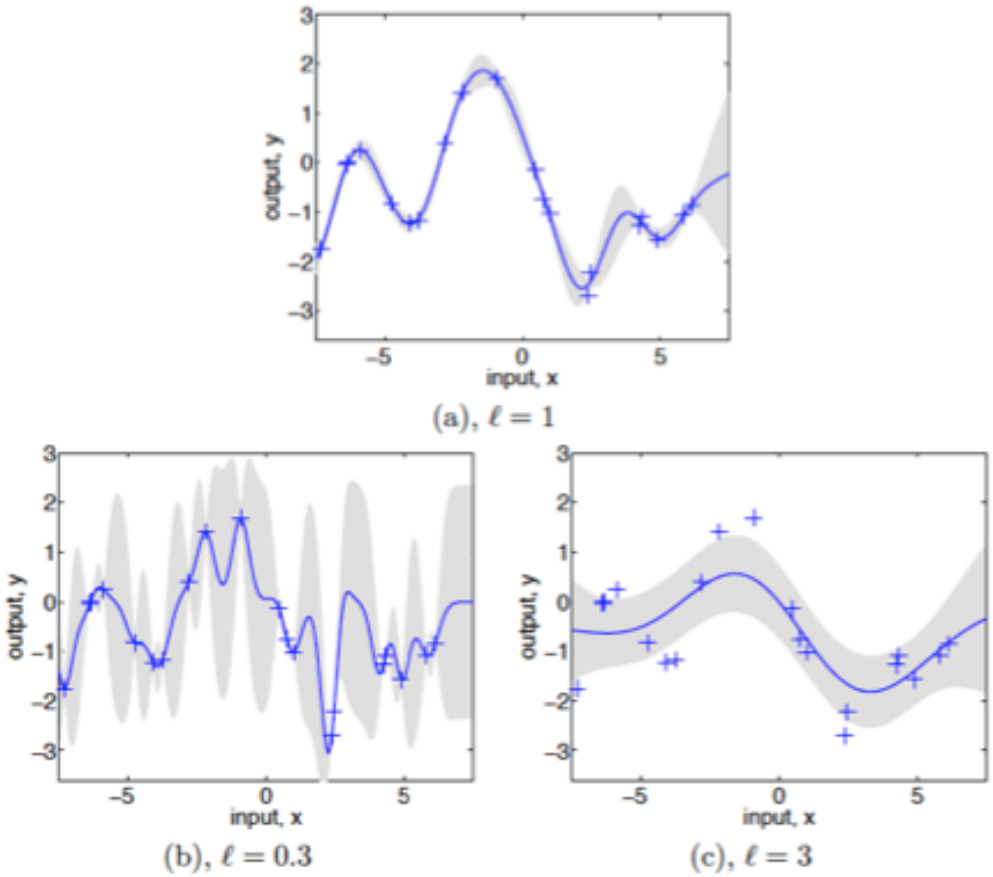


Figure 28: Example results for different choices of the kernel parameters for relatively few data. hyperparameters are $(l, \sigma_f, \sigma_n) = (1, 1, 0.1)$ for (a), $(0.3, 10.08, 0.00005)$ for (b) and $(3.0, 1.16, 0.89)$ for (c). Images are taken from [1].

- GPs are more principled in the sense that they make the statistical assumptions very clear; by relying on priors over functions, they allow a few more general things in a very easy way (if you know how to do probabilistic inference, of course . . .)

Because of these issues, GPs are not the method of choice in all cases. But they seem very valuable if auxiliary information is present, or highly noisy data have to be dealt with.

We give one example for a direct possibility where GP modeling helps.

Metaparameter optimization

One principled way to avoid often tedious trial and error for metaparameter optimization can be based on a direct *optimization of the likelihood*. The logarithm of the marginal likelihood for given data is

$$\log p(Y|X) = \log \int p(Y|X,f)p(f|X)df$$

with $p(Y|X,f) \sim \mathcal{N}(f, \sigma_n^2 I)$ and $p(f|X) \sim \mathcal{N}(0, K)$ with kernel matrix K, hence the integrand is distributed according to $\mathcal{N}(0, K + \sigma_n^2 \mathbf{I})$ This allows us to evaluate the integral as sum

$$-\frac{1}{2}Y^t(K + \sigma_n^2\mathbf{I})^{-1}Y - \frac{1}{2}\log|K + \sigma_n^2\mathbf{I}| - \frac{n}{2}\log 2\pi$$

which decomposes into a term characterizing the data fit, a penalty term for model complexity, and a constant. One can try to optimize these costs directly with respect to a kernel parameter by means of a gradient, taking derivatives of the first two terms. A typical landscape is depicted in Fig. 29. However, it is not guaranteed to find a global optimum this way, and local optima of the likelihood can exist, see Fig. 29.

A similar question is which kernel functions to use. Every kernel yields a valid GP, but the kernels are differently suited for different applications areas. One can characterize important properties of the kernel which correspond to important characteristics of the resulting functions, which might guide the choice, such as

- stationary: depends on $\vec{x} - \vec{x}'$, invariance to translations
- isotropic: depends on $|\vec{x} - \vec{x}'|$, invariance to rigid motion
- dot product: depends on $\vec{x}^t\vec{x}'$, invariance to rotation around origin

Another important characteristic of a kernel is that it is *non degenerate*, which refers to the fact that the resulting kernel matrices do not have a limited rank, but the rank increases with the number of data. This property is very important to guarantee in the limit the convergence to the true underlying distribution. Indeed, one can show that under suitable conditions on the kernel, GPs converge towards the underlying distribution for sufficiently many data, i.e. the limit result is then independent of the prior.

Fig. 30 summarizes a few popular kernel choices.

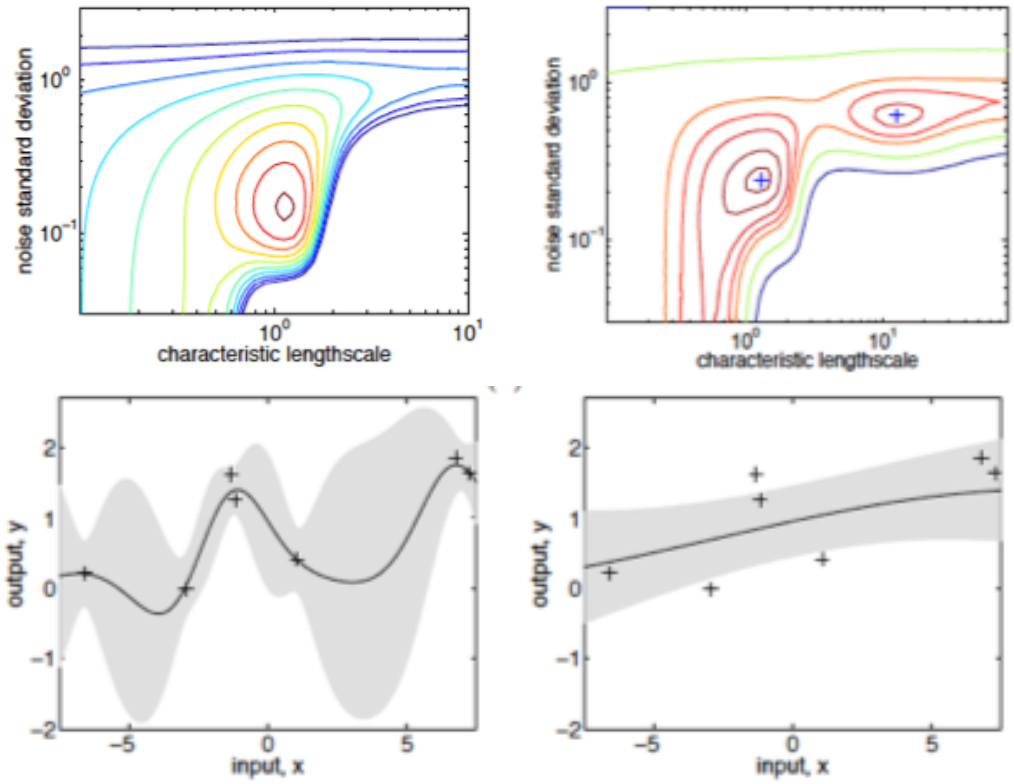


Figure 29: Upper row: example likelihood for two models, the right one has two optima, corresponding to the GP regression as shown in the lower row. Images are taken from [1].

Further issues

GPS being an active field of research, a variety of issues are tackled. We just mention a few lines of research:

- Efficient approximations: For large data cubic complexity is obviously infeasible. Approximations have been proposed around different lines such as subsampling of the data, low rank approximations of K i.e. using PCA or the Nyström approximation, decomposition of the data into subsets and their treatment by committee machines, etc.

- Gaussian process classification: surprisingly, classification is much more complex than GP regression, because the resulting posterior is in general no Gaussian. Popular solutions for this problem are for example the standard Laplacian approximation for the resulting distribution.

covariance function	expression	S	ND
constant	σ_0^2	√	
linear	$\sum_{d=1}^{D} \sigma_d^2 x_d x'_d$		
polynomial	$(\mathbf{x} \cdot \mathbf{x}' + \sigma_0^2)^p$		
squared exponential	$\exp(-\frac{r^2}{2\ell^2})$	√	√
Matérn	$\frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{\ell} r\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{\ell} r\right)$	√	√
exponential	$\exp(-\frac{r}{\ell})$	√	√
γ-exponential	$\exp\left(-(\frac{r}{\ell})^\gamma\right)$	√	√
rational quadratic	$(1 + \frac{r^2}{2\alpha\ell^2})^{-\alpha}$	√	√
neural network	$\sin^{-1}\left(\frac{2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}}'}{\sqrt{(1+2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}})(1+2\tilde{\mathbf{x}}'^\top \Sigma \tilde{\mathbf{x}}')}}\right)$		√

Figure 30: A few popular kernels and their characteristics. $r = |\vec{x} - \vec{x}'|$. S means stationary, ND means non degenerate. The table is taken from [1].

Literature

1. Gaussian Processes for Machine Learning, Carl Rasmussen, Christopher Williams, MIT Press, 2006

2. Pattern Recognition and Machine Learning, Christopher Bishop, Springer, 2006

3. Machine Learning, A probabilistic perspective, Kevin Murphy, MIT, 2012

9 Some form of summary

The topics which were tackled in this lecture span a representative range as concerns data processing. Albeit they have been presented independent of each other, they can be combined in various ways, or they are partially competing approaches for the same tasks. Tasks addressed by the techniques concern mostly one of two aspects

- data preprocessing and data preparation for further processing or data inspection; here the focus lies on data which go beyond typical real-vectors, by addressing times series / structures (SFA and DTW) or addressing underlying structure given by sparsity (Sparse coding and compressed sensing)
  - Slow feature analysis (SFA): Linearly transform high dimensional time series such that the new representation represents important time invariant principles.
  - Time series and structure kernel (DTW): How to define kernels for structured objects such as time series?
  - Sparse coding: Represent vectors linearly using sparse coefficients and a universal dictionary
  - Compressed sensing: How to measure data efficiently which displays a certain structure (sparsity)?
- data classification / clustering / regression; here classical data formats obtained using e.g. the just mentioned preprocessing are tackled. However, the techniques go beyond classical data analysis in addressing one specific point as particular aim and exemplarily showing its feasibility
  - Support feature machine (SFM): Linear classification based on few input coefficients, aim: interpretability for high dimensional data
  - Core vector machine (CVM): generalized (i.e. including kernel) linear classification/regression suited for big data, aim: efficiency for big data by devising linear time training, constant time classification (due to limited core set)
  - Gaussian processes (GP): regression without assuming an explicit parametric form of the classifier but imposing a prior directly over functions, aim: probabilistic modeling brought to an extremal form, opens principled possibilities to deal with meta-parameter optimization, missing data, different data format, etc. Excellent e.g. in the situation of few noisy data points, provides output distribution and related quantities such as confidence.