Optimization Models

EECS 127 / EECS 227AT

Laurent El Ghaoui

EECS department
UC Berkeley

Fall 2018

LECTURE 22

Applications in Machine Learning

The greater the uncertainty, the bigger the gap between what you can measure and what matters, the more you should watch out for overfitting - that is, the more you should prefer simplicity.

Tom Griffiths

Outline

1. Supervised learning
   - Generic supervised learning problem
   - Least-squares and variants
   - Support vector machines
   - Logistic regression
   - Neural networks
   - Kernel trick

2. Unsupervised learning
   - Generalized low-rank models
   - Matrix completion
   - Sparse covariance selection

What is supervised learning?

In supervised learning, we are given

- A matrix of data points $X = [x_1, \ldots, x_m]$, with $x_i \in \mathbf{R}^n$;
- A vector of "responses" $y \in \mathbf{R}^m$.

The goal is to use the data and the information in y (the "supervised" part) to form a model. In turn the model can be used to *predict* a value $\hat{y}$ for a (yet unseen) new data point $x \in \mathbf{R}^n$.

- If the response vector contains real entries, the problem is called "regression";
- If the response vector contains binary (more generally, categorical) entries, the problem is called "classification".

Generic form of supervised learning problem

Many classification and regression problems can be written as

$$\min_{w} \; L(X^T w, y) + \lambda p(w)$$

where

- $X = [x_1, \ldots, x_n]$ is a $m \times n$ matrix of data points.
- $y \in \mathbf{R}^m$ contains a response vector (or labels).
- w contains classifier/regression coefficients.
- L is a "loss" function that depends on the problem considered.
- p is a *penalty function* that is usually data-independent (more on this later).
- $\lambda \geq 0$ is a regularization parameter.

Prediction/classification rule: depends only on $w^T x$, where $x \in \mathbf{R}^n$ is a new data point.

Popular loss functions

- Squared loss: (for linear least-squares regression)

$$L(z, y) = \|z - y\|_2^2.$$

- l_1-norm loss: (useful to deal with outliers)

$$L(z, y) = \|z - y\|_1.$$

- Hinge loss: (for SVMs)

$$L(z, y) = \sum_{i=1}^{m} \max(0, 1 - y_i z_i)$$

- Logistic loss: (for logistic regression)

$$L(z, y) = -\sum_{i=1}^{m} \log(1 + e^{-y_i z_i}).$$

Linear least-squares

Basic model:

$$\min_w \ \|X^T w - y\|_2^2$$

where

- $X = [x_1, \ldots, x_n]$ is the $m \times n$ matrix of data points.
- $y \in \mathbf{R}^m$ is the "response" vector,
- w contains regression coefficients.
- $\lambda \geq 0$ is a regularization parameter.

Prediction rule: $y = w^T x$, where $x \in \mathbf{R}^n$ is a new data point.

Example

Auto-regressive models

Linear "auto-regressive" model for time-series: y_t linear function of y_{t-1}, y_{t-2}

$$y_t = w_1 + w_2 y_{t-1} + w_3 y_{t-2}, \quad t = 1, \ldots, T.$$

This writes $y_t = w^T x_t$, with x_t the "feature vectors"

$$x_t := (1, y_{t-1}, y_{t-2}), \quad t = 1, \ldots, T.$$

Model fitting via least-squares: we minimize the sum-of-squares of errors

$$\min_w \|X^T w - y\|_2^2$$

Prediction rule: once we've "learnt" w, we can make a prediction for time $T+1$: $\hat{y}_{T+1} = w_1 + w_2 y_T + w_3 y_{T-1} = w^T x_{T+1}$.

Variants of least-squares

- LASSO and elastic net:

$$\min_w \ \|X^T w - y\|_2^2 + \lambda \|w\|_2^2 + \mu \|w\|_1,$$

  where λ, μ are usually chosen via cross-validation, with λ controlling robustness to noise, and μ controlling sparsity of w.

- Least absolute deviation model:

$$\min_w \ \|X^T w - y\|_1 + \lambda \|w\|_2^2 + \mu \|w\|_1,$$

  which is less sensitive to outliers in data than the previous model.

- Group LASSO:

$$\min_w \ \|X^T w - y\|_2^2 + \lambda \sum_{k=1}^{p} \|w_k\|_2,$$

  where $w = (w_1, \ldots, w_p)$, and each $w_k \in \mathbf{R}^{n_k}$, with $n_1 + \ldots + n_p = n$. Allows to select groups of features.

Binary classification via SVM

Data

We are given a *training* data set:

- Feature vectors: data points $x_i \in \mathbf{R}^p$, $i = 1, \ldots, n$.
- Labels: $y_i \in \{-1, 1\}$, $i = 1, \ldots, n$.

Examples:

Feature vectors	Labels
Companies' corporate info	default/no default
Stock price data	price up/down
News data	price up/down
News data	sentiment (positive/negative)
Emails	presence of a keyword
Genetic measures	presence of disease

Linear classification

Using the training data set $\{x_i, y_i\}_{i=1}^n$, our goal is to find a classification rule $\hat{y} = f(x)$ allowing to predict the label $\hat{y}$ of a new data point x.

Linear classification rule: assumes f is a combination of the sign function and a linear (in fact, affine) function:

$$\hat{y} = \mathbf{sign}(w^T x + b),$$

where $w \in \mathbf{R}^p$, $b \in \mathbf{R}$ are given.

The goal of a linear classification algorithm is to find w, b, using the training data.

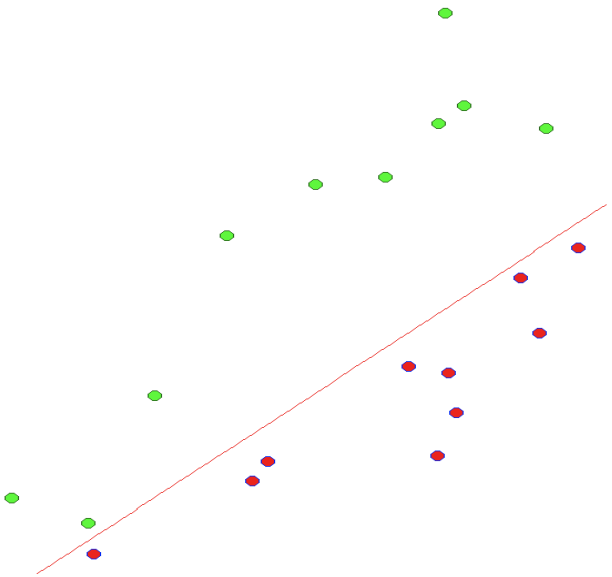
Separable data

The data is linearly separable if there exist a linear classification rule that makes no error on the training set.

This is a set of linear inequalities constraints on (w, b):

$$y_i(w^T x_i + b) \geq 0, \quad i = 1, \ldots, n.$$

Strict separability corresponds the the same conditions, but with strict inequalities.

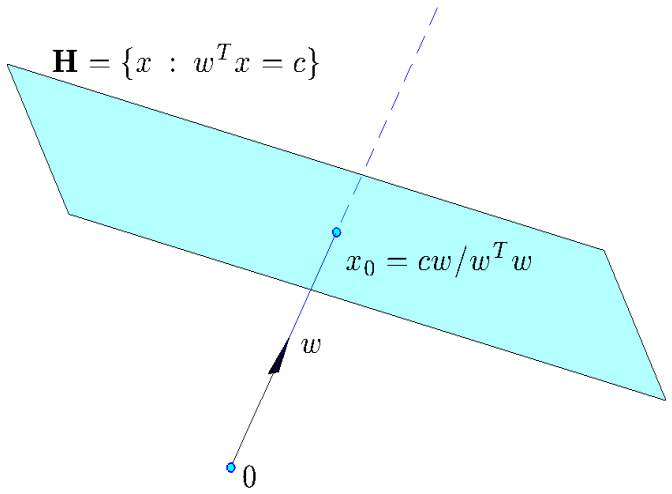


Geometrically: the hyperplane

$$\{x \ : \ w^T x + b = 0\}$$

perfectly separates the positive and negative data points.

Linear algebra flashback: hyperplanes



Geometrically, a hyperplane

$$\mathcal{H} = \{w : w^T x = c\}$$

is a translation of the set of vectors orthogonal to w. The direction of the translation is determined by w, and the amount by $c/\|w\|_2$.

The projection of 0 onto $\mathcal{H}$ is $x_0 = cw/(w^T w)$, hence the distance from 0 to $\mathcal{H}$ is

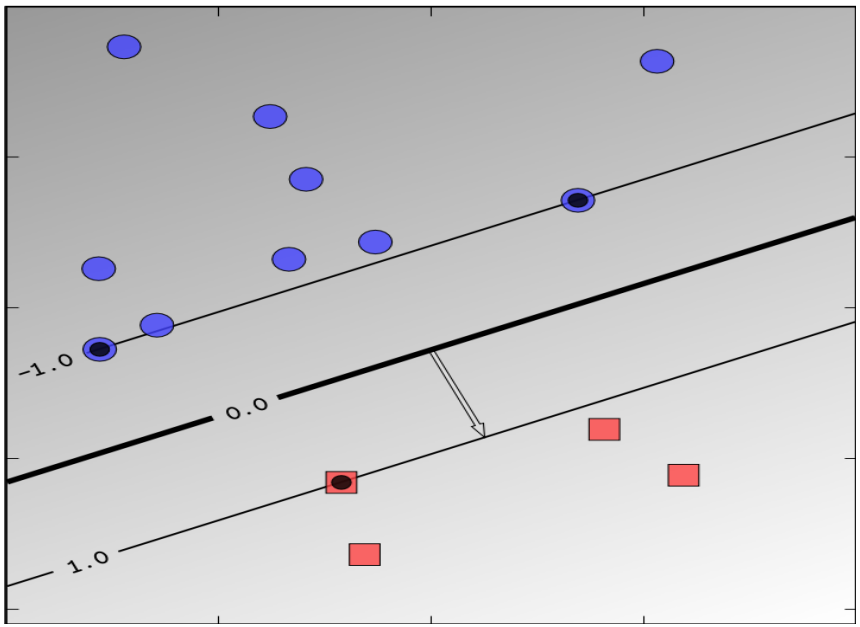
$$\text{dist}(0, \mathcal{H}) = \|x_0\|_2 = |c|/\|w\|_2.$$

Geometry (cont'd)

Assuming strict separability, we can always rescale (w, b) and work with

$$y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, n.$$

Amounts to make sure that negative (resp. positive) class contained in half-space $w^T x + b \leq -1$ (resp. $w^T x + b \geq 1$).



The distance between the two boundaries $(w^T x + b = \pm 1)$ is equal to $2/\|w\|_2$.

Thus the "margin" $\|w\|_2$ is a measure of how well the hyperplane separates the data apart.

Non-separable data

Separability constraints are homogeneous, so WLOG we can work with

$$y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, n.$$

The above is the condition for the data to be strictly separable. If it is infeasible, we try to minimize "slacks":

$$\min_{w,b,s} \sum_{i=1}^{n} s_i \;\; : \;\; s \geq 0, \quad y_i(w^T x_i + b) \geq 1 - s_i, \quad i = 1, \ldots, n.$$

The above is an LP. We can also minimize a combination of slacks and (squared) margin, leading to the QP

$$\min_{w,b,s} \sum_{i=1}^{n} s_i + \lambda \|w\|_2^2 \;\; : \;\; s \geq 0, \quad y_i(w^T x_i + b) \geq 1 - s_i, \quad i = 1, \ldots, n.$$

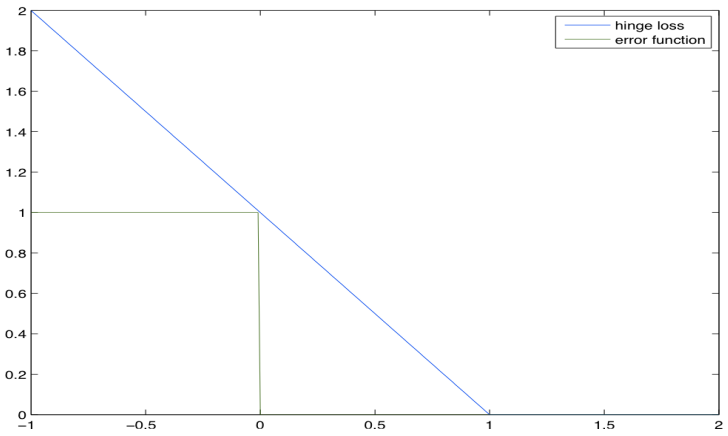
The above is the standard SVM model found in libraries.

Hinge loss function

The previous LP can be interpreted as minimizing the "hinge" loss function

$$L(w, b) := \sum_{i=1}^{m} \max(1 - y_i(w^T x_i + b), 0).$$

This serves as an approximation to the number of errors made on the training set:



Minimizing the total number of errors amounts to solve

$$\sum_{i=1}^{m} E(y_i(w^T x_i + b))$$

where E is the "0-1 loss", shown in green. The blue curve corresponds to the hinge loss.

Dual problem

Denote by $\mathcal{C}^+$ (resp. $\mathcal{C}^-$) the set of points x_i with $y_i = +1$ (resp. -1). Consider maximazing margin, assuming perfect separability:

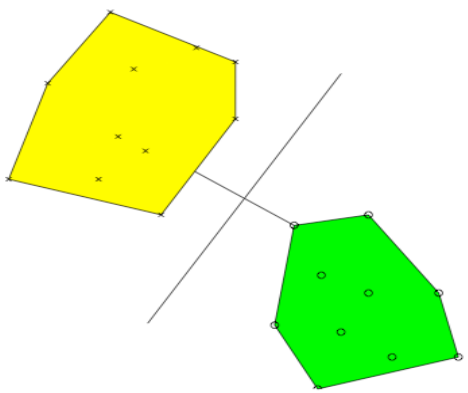
$$\min_w \|w\|_2 \ : \ y_i(w^T \hat{x}_i + b) \geq 1, \ \ i = 1, \ldots, n.$$

Dual:

$$\min_{x_+, x_-} \|x_+ - x_-\|_2 \ : \ x_+ \in \mathbf{Co}\mathcal{C}^+, \ \ x_- \in \mathbf{Co}\mathcal{C}^-,$$

where $\mathbf{Co}\mathcal{C}$ denotes convex hull of set $\mathcal{C}$, that is:

$$\mathbf{Co}\mathcal{C} = \left\{ \sum_{i=1}^q \lambda_i x_i \ : \ x_i \in \mathcal{C}, \ \ \lambda \geq 0, \ \ \sum_{i=1}^q \lambda_i = 1 \right\}.$$



Dual problem amounts to find the smallest distance between the two classes, each represented by the convex hull of its points. The optimal hyperplane sits at the middle of the line segment joining the two closest points.

Logistic model

We model the probability of a label Y to be equal $y \in \{-1, 1\}$, given a data point $x \in \mathbf{R}^n$, as:

$$P(Y = y \mid x) = \frac{1}{1 + \exp(-y(w^T x + b))}.$$

This amounts to modeling the *log-odds ratio* as a linear function of X:

$$\log \frac{P(Y = 1 \mid x)}{P(Y = -1 \mid x)} = w^T x + b.$$

- The decision boundary $P(Y = 1 \mid x) = P(Y = -1 \mid x)$ is the hyperplane with equation $w^T x + b = 0$.
- The region $P(Y = 1 \mid x) \geq P(Y = -1 \mid x)$ (*i.e.*, $w^T x + b \geq 0$) corresponds to points with predicted label $\hat{y} = +1$.

Maximum-likelihood

The likelihood function is

$$l(w, b) = \prod_{i=1}^{m} \frac{1}{1 + e^{-y_i(w^T x_i + b)}}.$$

Now maximize the log-likelihood:

$$\max_{w,b} \; L(w, b) := -\sum_{i=1}^{m} \log(1 + e^{-y_i(w^T x_i + b)})$$

In practice, we may consider adding a regularization term

$$\max_{w,b} \; L(w, b) + \lambda r(w),$$

with $r(w) = \|w\|_2^2$ or $r(x) = \|w\|_1$.

Neural networks

Single-layer neural network:

$$\min_{w,W} \; L(Z^\top w, y) \;\; : \;\; Z = \max(0, WX).$$

- Here W, w and Z are variables, with Z representing "hidden" features;
- L is loss function;
- Reduces to ordinary least-squares if we impose $W = I$, $Z = X$;
- Can be extended to multiple "layers".

Model is not convex, and a very active research area ("deep" learning).

Kernel trick

Motivation: nonlinear regression

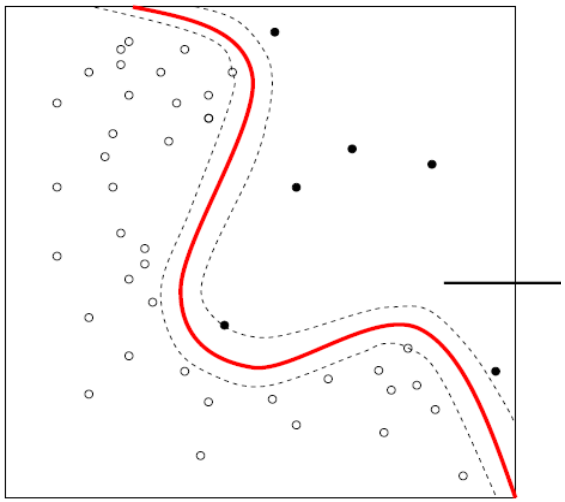
Nonlinear auto-regressive model for time-series: y_t quadratic function of y_{t-1}, y_{t-2}

$$y_t = w_1 + w_2 y_{t-1} + w_3 y_{t-2} + w_4 y_{t-1}^2 + w_5 y_{t-1} y_{t-2} + w_6 y_{t-2}^2.$$

This writes $y_t = w^T \phi(x_t)$, with $\phi(x_t)$ the augmented feature vectors

$$\phi(x_t) := \left(1, y_{t-1}, y_{t-2}, y_{t-1}^2, y_{t-1} y_{t-2}, y_{t-2}^2\right).$$

Everything the same as before, with x replaced by $\phi(x)$.



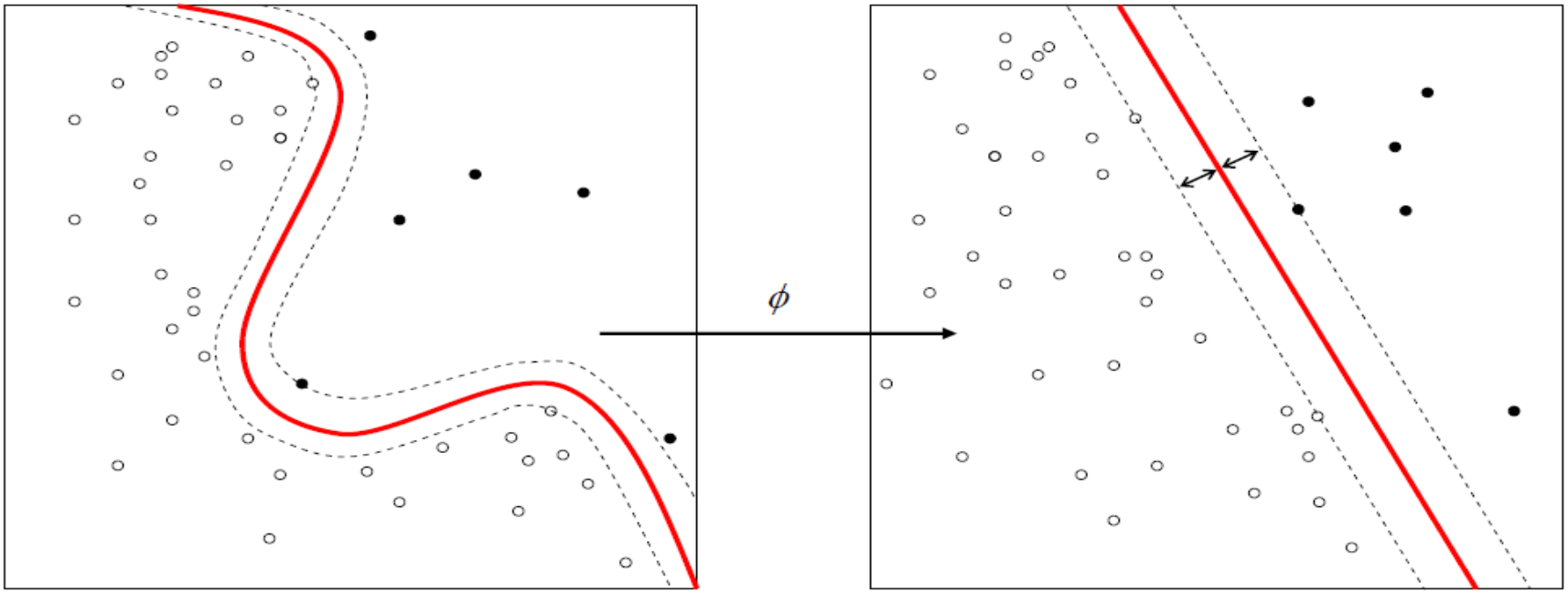
Non-linear (*e.g.*, quadratic) decision boundary

$$w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2 + b = 0.$$

Writes $w^T \phi(x) + b = 0$, with $\phi(x) := (x_1, x_2, x_1^2, x_1 x_2, x_2^2)$.

Augmenting data dimension

In principle, it seems may augment the dimension of the feature space to make the data linearly separable[1]



How do we do it in a computationally efficient manner?

[1] See the video at `http://www.youtube.com/watch?v=3liCbRZPrZA`.

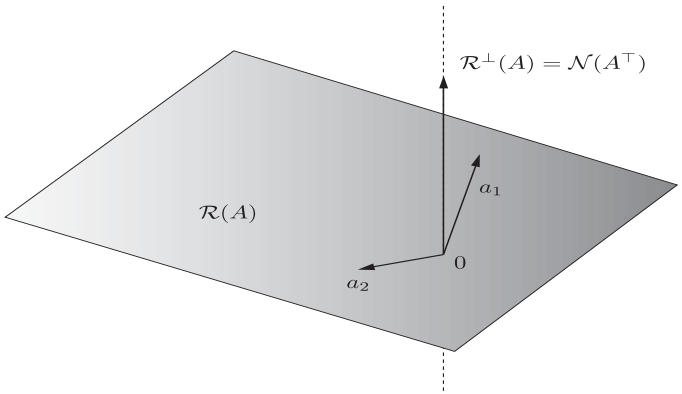
Key result

For the generic problem:

$$p^* \doteq \min_w \; L(X^T w) + \lambda \|w\|_2^2$$

the optimal w lies in the span of the data points $(x_1, \ldots, x_m)$: $w = Xv$ for some vector $v \in \mathbf{R}^m$.

Proof: From the fundamental theorem of linear algebra,



any $w \in \mathbf{R}^n$ can be written as the sum of two *orthogonal* vectors:

$$w = Xv + r$$

where $X^T r = 0$; that is, r is in the nullspace $\mathcal{N}(X^T)$. Figure shows the case $X = A = (a_1, a_2)$.

Since we are using the Euclidean norm as penalty, the optimal r is zero:

$$p^* = \min_{v,r} \; L(X^T X v) + \lambda \left(\|r\|_2^2 + \|Xv\|_2^2 \right) = \min_v \; L(X^T X v) + \lambda \|Xv\|_2^2.$$

Consequence of key result

For the generic problem:

$$\min_w \; L(X^T w) + \lambda \|w\|_2^2$$

the optimal w can be written as $w = Xv$ for some vector $v \in \mathbf{R}^m$.

Hence, training problem depends only on the "kernel" matrix $K := X^T X$:

$$\min_v \; L(Kv) + \lambda v^T K v.$$

$K = (x_i^\top x_j)_{1 \le i,j \le m}$ contains the scalar products between all data point pairs.

The prediction/classification rule depends on the scalar products between new point x and the data points $x_1, \ldots, x_m$:

$$w^T x = v^T X^T x = v^T k, \;\; k := X^T x = (x^T x_1, \ldots, x^T x_m).$$

Once K is formed (this takes $O(nm^2)$), then the training problem has only m variables. When $n >> m$, this leads to a dramatic reduction in problem size.

How about the nonlinear case?

In the nonlinear case, we simply replace the feature vectors x_i by some "augmented" feature vectors $\phi(x_i)$, with ϕ a non-linear mapping.

Example: in classification with quadratic decision boundary, we use

$$\phi(x) := (x_1, x_2, x_1^2, x_1 x_2, x_2^2).$$

This leads to the modified kernel matrix

$$K_{ij} = k(x_i, x_j) = \phi(x_i)^T \phi(x_j), \quad 1 \leq i, j \leq m.$$

The "kernel function" k provides information about the metric in the feature space, *e.g.*:

$$\|\phi(x) - \phi(z)\|_2^2 = k(x, x) - 2k(x, z) + k(z, z).$$

The computational effort involved in solving the training problem and making a prediction, depends only on our ability to quickly evaluate the kernel function.

Polynomial and Gaussian kernels

There is a large variety (a zoo?) of kernel functions, some adapted to structure of data (text, images, etc). Two generic examples:

- Polynomial kernel: here $\phi(x)$ is the vector formed with all the products between the components of $x \in \mathbf{R}^n$, up to degree d, then for any two vectors $x, z \in \mathbf{R}^n$,

$$\phi(x)^T \phi(z) = (1 + x^T z)^d.$$

  Computational effort grows linearly in n (instead of n^d with a naïve approach).

- Gaussian kernel:

$$k(x, z) = \exp\left(-\frac{\|x - z\|_2^2}{2\sigma^2}\right),$$

  where $\sigma > 0$ is a scale parameter. Allows to ignore points that are too far apart. Corresponds to a non-linear mapping ϕ to infinite-dimensional feature space.

Note: kernel methods do not work with l_1-norm regularization!

What is unsupervised learning?

In unsupervised learning, we are given a matrix of data points $X = [x_1, \ldots, x_m]$, with $x_i \in \mathbf{R}^n$; we wish to learn some condensed information from it.

Examples:

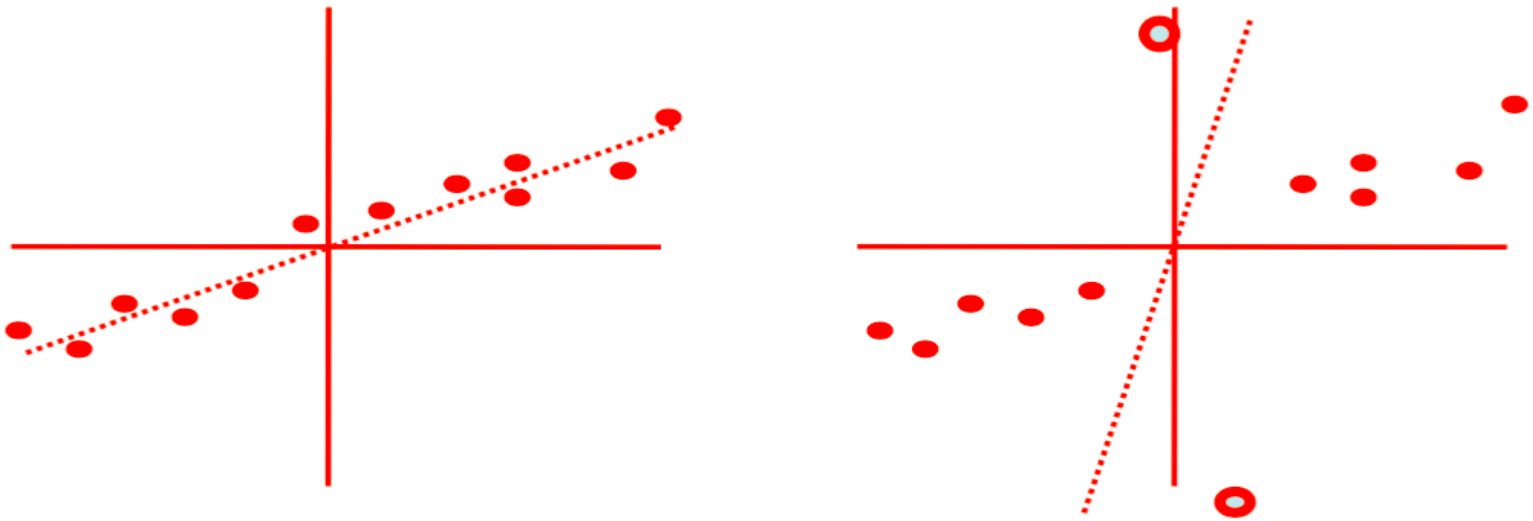
- Find one or several direction of maximal variance.
- Find a low-rank approximation or other structured approximation.
- Find correlations or some other statistical information (*e.g.*, graphical model).
- Find clusters of data points.

Motivation

PCA has many restrictions, including

- it works only on fully known matrix (no missing entries);
- it cannot handle different data types, such as Boolean, categorical, non-negative, etc;
- it is very sensitive to outliers, an artefact due to the squared Frobenius (variance) being the underlying metric used.

Gross errors of even one/few points can completely throw off PCA



Reason: Classical PCA minimizes ℓ_2 error, which is susceptible to gross outliers

Low-rank models and alternate minimization

For $X \in \mathbf{R}^{p \times m}$, ordinary rank-$k$ model solves

$$\min_{L,R} \|X - LR^T\|_F \ : \ L \in \mathbf{R}^{p \times k}, \ \ R \in \mathbf{R}^{m \times k},$$

by minimization over L, R alternatively. This is essentially PCA, if we work with a column-centered data matrix.

Note that $(LR^T)_{ij} = l_i^T r_j$, where

$$L = \begin{pmatrix} l_1^T \\ \vdots \\ l_p^T \end{pmatrix}, \ \ R = \begin{pmatrix} r_1^T \\ \vdots \\ r_m^T \end{pmatrix},$$

Thus we can write the above problem as

$$\min_{L,R} \sum_{i,j} \mathcal{L}(X_{ij}, l_i^T r_j) \ : \ l_i \in \mathbf{R}^k, \ \ i = 1, \ldots, p, \ \ r_j \in \mathbf{R}^k, \ \ j = 1, \ldots, m,$$

with $\mathcal{L}(a, b) = (a - b)^2$.

Generalization

Generalized low-rank model solves

$$\min_{L,R} \sum_{i,j} \mathcal{L}(X_{ij}, l_i^T r_j) + \sum_i p_i(l_i) + \sum_j q_j(r_j),$$

where $\mathcal{L}$ is convex, and functions p_i, q_j are convex penalties.

- The problem is not convex—but it is with respect to X, R (resp. X, L) when L (resp. R) is fixed.
- We can solve the problem by alternative minimization over L, R.
- In most cases, there is no guarantee of convergence to a global minimum.
- Playing with different losses and penalties we can model a lot of useful situations.

Robust PCA

In robust PCA we seek to decompose the input data matrix X into a sum of a sparse and a low-rank component:

$$X = LR^T + S, \quad L \in \mathbf{R}^{n \times k}, \quad R \in \mathbf{R}^{m \times k}, \quad S \text{ sparse.}$$

We can model this with

$$\mathcal{L}(a, b) = |a - b|,$$

leading to

$$\min_{L,R} \sum_{i,j} |X_{ij} - l_i^T r_j| = \|X - LR^T\|_1,$$

where $\|Z\|_1$ is the sum of the absolute values of the entries of matrix Z.

The l_1-norm is chosen as a heuristic to make the matrix in the norm sparse.

Example: video analytics

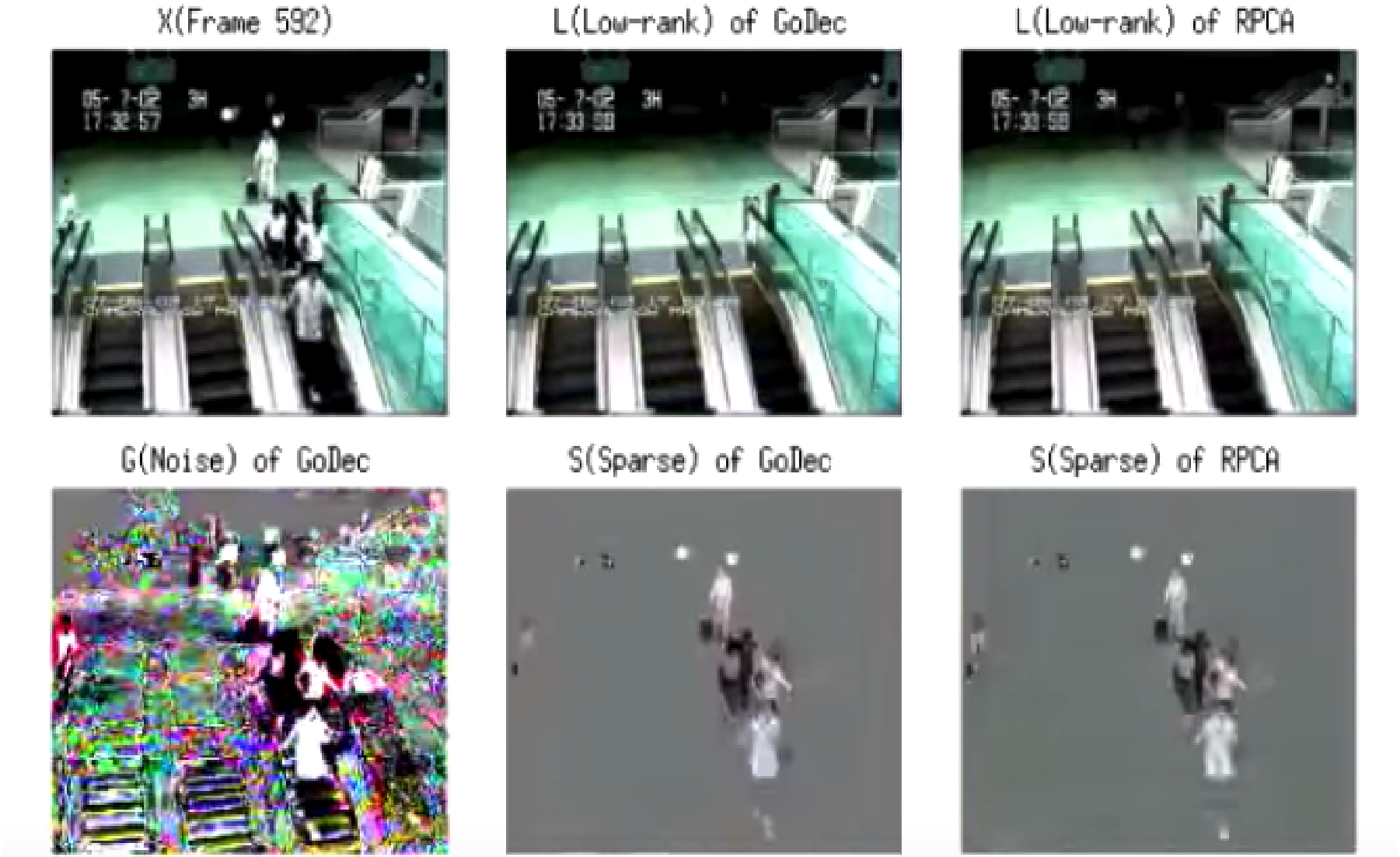


Figure: See https://www.youtube.com/watch?v=BTrbow8u4Cw

Non-negative matrix factorization

Non-negative matrix factorization (NNMF) is a variant on PCA where the factors are required to be non-negative:

$$X = LR^T, \text{ with } L \geq 0, \quad R \geq 0,$$

with inequalities understood component-wise. This problem arises when the data matrix is itself non-negative.

We can model this with

$$\min_{L,R} \sum_{i,j} (X_{ij} - l_i^T r_j)^2 \ : \ L \geq 0, \quad R \geq 0,$$

corresponding to penalties p_i, q_j all chosen to be equal to

$$p(z) = \begin{cases} 0 & \text{if } z \geq 0 \\ +\infty & \text{otherwise.} \end{cases}$$

Sparse PCA

In sparse PCA we seek to approximate a matrix by a low-rank one, each factor being sparse:

$$X = LR^T, \text{ with } L, R \text{ sparse.}$$

We can model this with

$$\mathcal{L}(a, b) = |a - b|,$$

leading to

$$\min_{L,R} \sum_{i,j} (X_{ij} - l_i^T r_j)^2 + \|L\|_1 + \|R\|_1.$$

Again the l_1-norm is chosen as a heuristic to make the matrix in the norm sparse.

Example

Sparse PCA of New York Times headlines

Table: Words associated with the top 5 sparse principal components in NYTimes

1st PC (6 words)	2nd PC (5 words)	3rd PC (5 words)	4th PC (4 words)	5th PC (4 words)
million	point	official	president	school
percent	play	government	campaign	program
business	team	united_states	bush	children
company	season	u_s	administration	student
market	game	attack		
companies				

Note: the algorithm found those terms without any information on the subject headings of the corresponding articles (unsupervised problem).

Matrix completion problem

Matrix completion is the problem of filling unknown entries of a partially known matrix. A classical assumption is that the completion should be made so that the completed matrix has the lowest rank prossible.

Approach based on regularized PCA:

$$\min_{L,R,X\in\mathcal{X}} \|X - LR^T\|_F^2 + \gamma\left(\|L\|_F^2 + \|R\|_F^2\right) \;:\; L \in \mathbf{R}^{n\times k},\;\; R \in \mathbf{R}^{m\times k},$$

with X a variable, and $\mathcal{X}$ the set of $n \times m$ matrices that have the required given entries.

- Alternating minimization works the same! Just add missing entries in X as variables.
- Some theoretical results show that if the locations of missing entries are randomly distributed, convergence to the global minimum is guaranteed.
- In practice, for this to work, missing entries should not follow a clear pattern (*e.g.*, they should not all be located at the bottom in a time-series matrix).

Generalized low-rank models

Summary

- Generalized low-rank models offer a very flexible way to model data.
- It is always based on the key low-rank assumption, and generalizes standard PCA in many directions.
- In general, GLRMs are not convex, and convergence is not guaranteed.
- It is often a good idea to add a squared penalty to the loss function.

Sparse covariance selection

We'd like to draw a graph that describes the links between the features (*e.g.*, words).

- Edges in the graph should exist when some strong, natural metric of similarity exist between features.
- For better interpretability, a *sparse* graph is desirable.
- Various motivations: portfolio optimization (with sparse risk term), clustering, etc.

Here we focus on exploring *conditional independence* within features.

Gaussian assumption

Let us assume that the data points are zero-mean, and follow a multi-variate Gaussian distribution: $x \simeq \mathcal{N}(0, \Sigma)$, with Σ a $p \times p$ covariance matrix. Assume Σ is positive definite.

Gaussian probability density function:

$$p(x) = \frac{1}{(2\pi \det \Sigma)^{p/2}} \exp((1/2)x^T \Sigma^{-1} x).$$

where $X := \Sigma^{-1}$ is the *precision* matrix.

Conditional independence

The pair of random variables x_i, x_j are *conditionally independent* if, for x_k fixed ($k \neq i, j$), the density can be factored:

$$p(x) = p_i(x_i)p_j(x_j)$$

where p_i, p_j depend also on the other variables.

- *Interpretation:* if all the other variables are fixed then x_i, x_j are independent.
- *Example:* Gray hair and shoe size are independent, conditioned on age.

Conditional independence

C.I. and the precision matrix

Theorem 1 (C.I. for Gaussian RVs)

The variables x_i, x_j are conditionally independent if and only if the i, j element of the precision matrix is zero:

$$(\Sigma^{-1})_{ij} = 0.$$

Proof.

The coefficient of $x_i x_j$ in $\log p(x)$ is $(\Sigma^{-1})_{ij}$. □

Sparse precision matrix estimation

Let us encourage sparsity of the precision matrix in the maximum-likelihood problem:

$$\max_X \log\det X - \mathbf{Tr}SX - \lambda\|X\|_1,$$

with $\|X\|_1 := \sum_{i,j} |X_{ij}|$, and $\lambda > 0$ a parameter.

- The above provides an invertible result, even if S is not positive-definite.
- The problem is convex, and can be solved in a large-scale setting by optimizing over column/rows alternatively.

Dual

Sparse precision matrix estimation:

$$\max_X \ \log\det X - \mathbf{Tr} SX - \lambda \|X\|_1.$$

Dual:

$$\min_U \ -\log\det(S+U) \ : \ \|U\|_\infty \leq \lambda.$$

Block-coordinate descent: Minimize over one column/row of U cyclically. Each step is a QP.

Example

Data: Interest rates

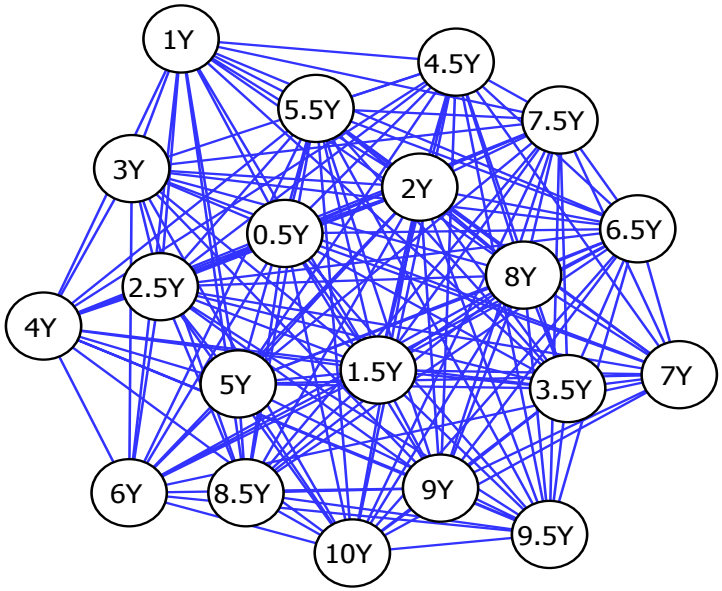


Figure: Using covariance matrix ($\lambda = 0$).

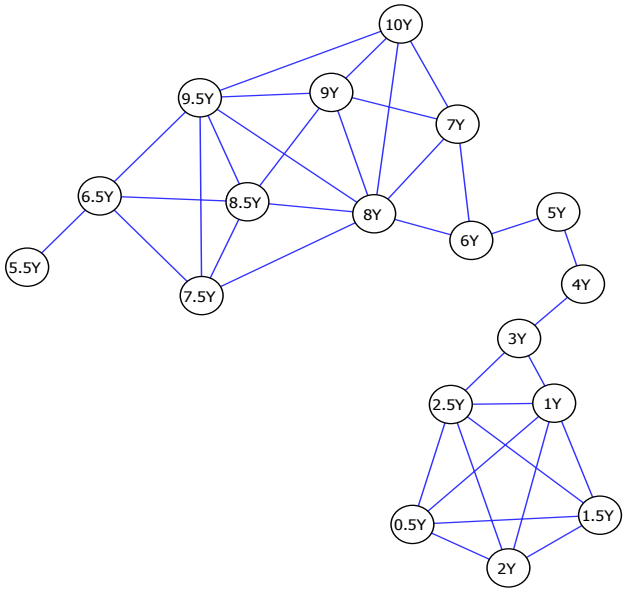
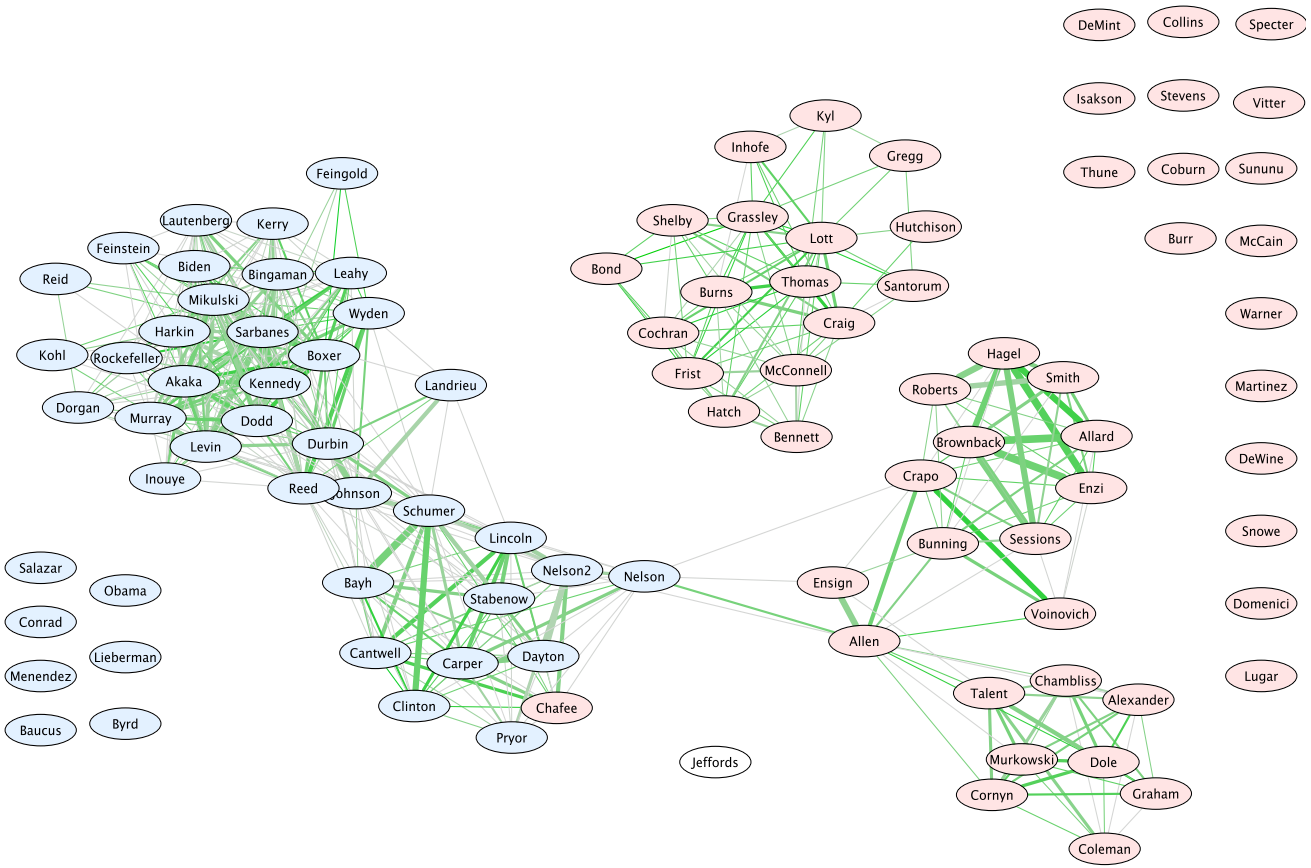


Figure: Using $\lambda = 0.1$.

The original precision matrix is dense, but the sparse version reveals the maturity structure.

Example

Data: US Senate voting, 2002-2004



Again the sparse version reveals information, here political blocks within each party.