Least Squares Revisited

- In slide set 4 we studied the Least Squares.
- Together with the Maximum Likelihood, it is by far the most widely used estimation method.
- Actually, under a Gaussian noise assumption the ML estimate turns out to be the LS estimate.
- Therefore, numerous modifications of the basic principle exist, including *Total Least Squares*, *Order Recursive Least Squares*, *Regularized Least Squares (Ridge Regression, the LASSO)*,...
- We will study these next.

Quick Recap of the LS

- Ordinary Least Squares (OLS) problems are either *linear* or *nonlinear*.
- Linear problems are of the form $\mathbf{x} = \mathbf{H}\theta + \mathbf{w}$ and solution is $\hat{\theta} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x}$.
- Nonlinear problems are more general minimization problems of the form

$$\hat{\theta} = \arg\min_{\theta} \sum_{n=0}^{N-1} \left(y[n] - f(x[n];\theta)\right)^2 .$$

Total Least Squares

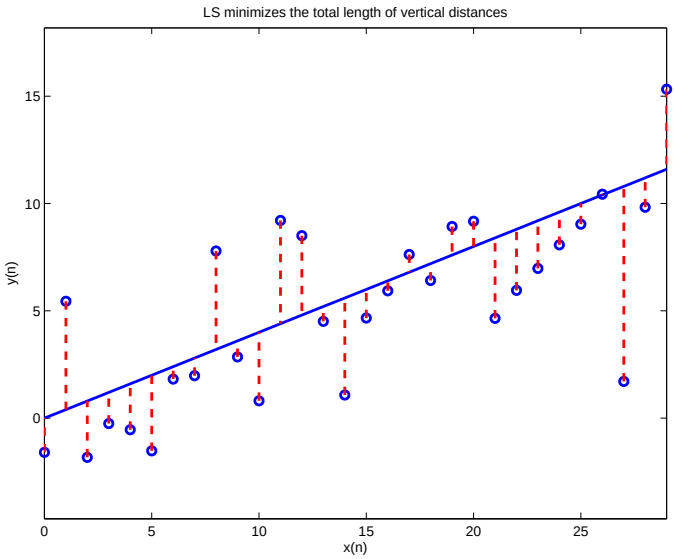
- Least squares assumes that the explaining variable(s) have no error, in other words, in the model of the ideal signal

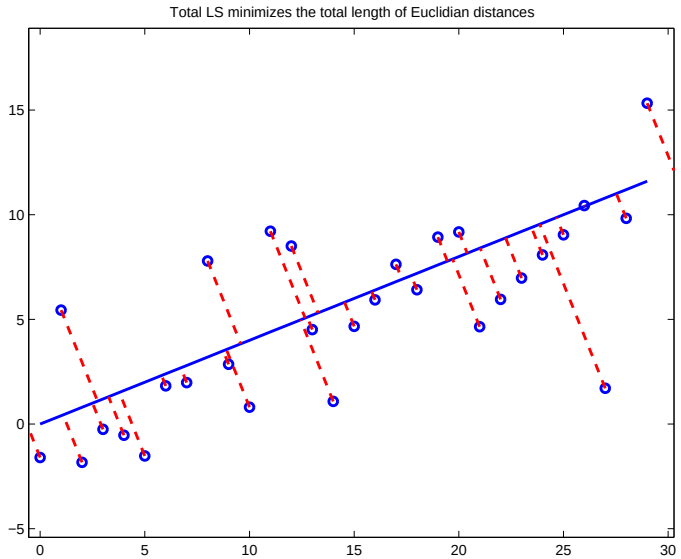
$$\mathbf{s} = \mathbf{H}\theta$$

  the matrix $\mathbf{H}$ is assumed error-free.
- If there's reason to expect that for example the time of taking the measurements is not accurate, it is possible to use *Total Least Squares* (TLS), which allows modeling of error in both $\mathbf{s}$ and $\mathbf{H}$.

Total Least Squares

- The pictures below illustrate the difference: LS minimizes the vertical distances between the observations and the model whereas TLS minimizes Euclidian distances.





Total Least Squares

- The LS problem can be reformulated as

$$\text{minimize} \quad \mathbf{e}^T\mathbf{e} \quad \text{subject to} \quad \mathbf{x} = \mathbf{H}\theta + \mathbf{e}$$

- The TLS problem is

$$\text{minimize} \quad (\mathbf{E}\mathbf{e})^T\mathbf{E}\mathbf{e} \quad \text{subject to} \quad \mathbf{x} = (\mathbf{H} + \mathbf{E})\theta + \mathbf{e}$$

Total Least Squares

- The solution uses Lagrangian multipliers, and the details are available for example, in Wikipedia or a tutorial by Markovsky et al.[1]
- The TLS parameter estimate $\hat{\theta}_{TLS}$ for the linear model $\mathbf{x} = \mathbf{H}\theta$ is obtained as follows.

1. Append the matrix $\mathbf{H}$ with the measurement data as follows: $\mathbf{A} = [\mathbf{H}, \mathbf{x}] \in \mathbb{R}^{n \times p+1}$.
2. Calculate the singular value decomposition (SVD) of the matrix $\mathbf{A}$:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T.$$

[1] I. Markovsky, S Van Huffel, "Overview of Total Least Squares Methods," *Signal Processing*, Volume 87, Issue 10, October 2007, Pages 2283-2302.

Total Least Squares

3. Now, $\mathbf{V} \in \mathbb{R}^{p+1 \times p+1}$:

$$\mathbf{V} = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{1,p+1} \\ v_{21} & v_{22} & \cdots & v_{2,p+1} \\ \vdots & \vdots & \ddots & \vdots \\ v_{p1} & v_{p2} & \cdots & v_{p,p+1} \\ v_{p+1,1} & v_{p+1,2} & \cdots & v_{p+1,p+1} \end{pmatrix}$$

Total Least Squares

4. The TLS parameter estimate is now defined by the column marked in red above:

$$\hat{\theta}_{\mathrm{TLS}} = -\frac{1}{v_{p+1,p+1}} \begin{pmatrix} v_{1,p+1} \\ v_{2,p+1} \\ \vdots \\ v_{p,p+1} \end{pmatrix}$$

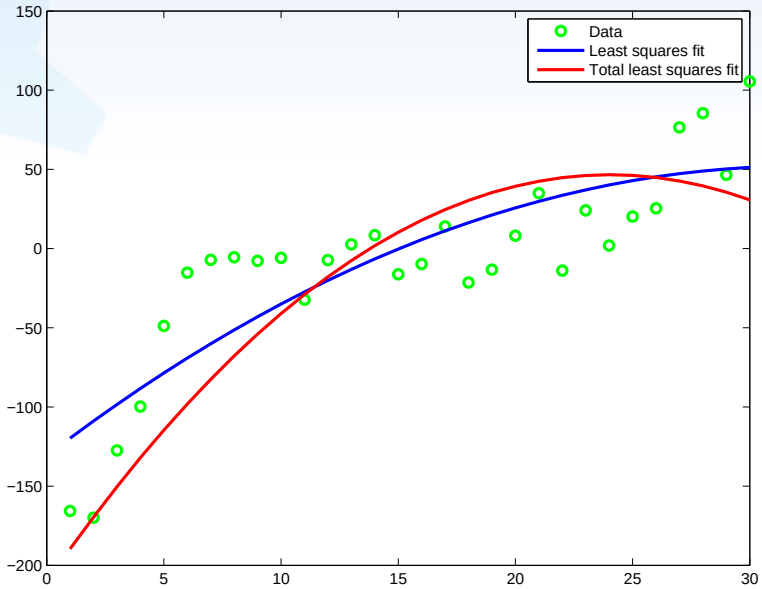
- In Matlab code, the above procedure simplifies into two lines of code:

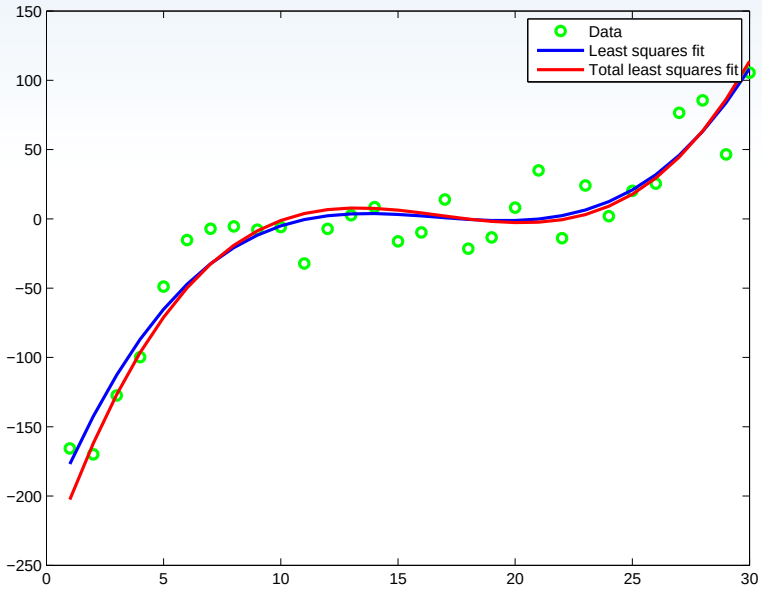
```
[U,S,V] = svd([H, x]);
theta_TLS = -V(1:end-1, end) / V(end,end);
```

Total Least Squares

- As an example, the figures below illustrate the difference between least squares and total least squares.
- The figure on the left shows the LS and TLS fits of second order polynomial, and the one on the right is the LS and TLS fits of third order polynomials to the same dataset.

Total Least Squares

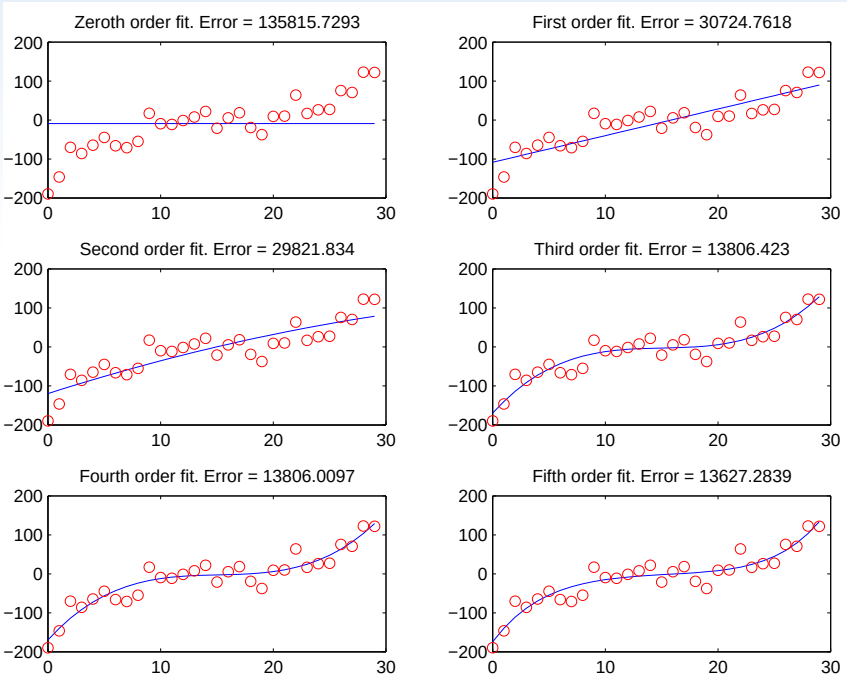




Order Recursive Least Squares

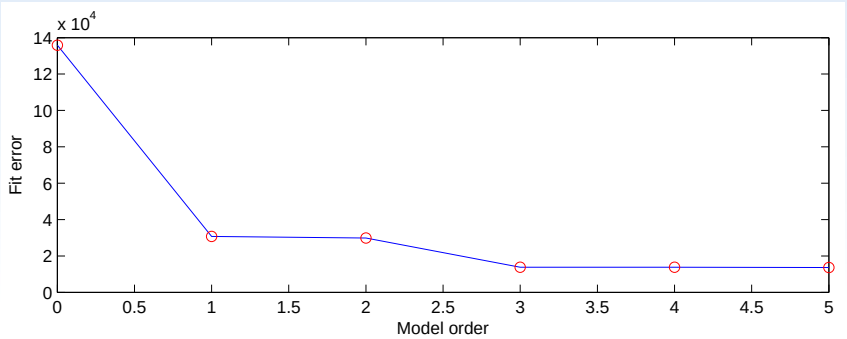
- What if the signal model is unknown? Can we easily compare models of different orders?
- The figure below shows an example of the performance of LS estimators of different order.
- The accuracy of the fit clearly increases with the order, but eventually saturates.

Order Recursive Least Squares



- The error of the fit is plotted below. It seems that a good order for the fit might be three.

Order Recursive Least Squares



- This gives rise to the Order-recursive LS algorithm, which computes the LSE based on an LSE of a smaller order thus saving time.
- If $\mathbf{H}_k$ is the $N \times k$ observation matrix, the estimate is obtained from

$$\hat{\theta}_k = (\mathbf{H}_k^T \mathbf{H}_k)^{-1} \mathbf{H}_k^T \mathbf{x}$$

- Is there a convenient formula for $\hat{\theta}_{k+1}$?

Order Recursive Least Squares

- The answer is *yes*:

$$\hat{\theta}_{k+1} = \begin{bmatrix} \hat{\theta}_k - \frac{(\mathbf{H}_k^T \mathbf{H}_k)^{-1} \mathbf{H}_k^T \mathbf{h}_{k+1} \mathbf{h}_{k+1}^T \mathbf{P}_k^{\perp} \mathbf{x}}{\mathbf{h}_{k+1}^T \mathbf{P}_k^{\perp} \mathbf{h}_{k+1}} \\ \frac{\mathbf{h}_{k+1}^T \mathbf{P}_k^{\perp} \mathbf{x}}{\mathbf{h}_{k+1}^T \mathbf{P}_k^{\perp} \mathbf{h}_{k+1}} \end{bmatrix}$$

where $\mathbf{P}_k^{\perp} = \mathbf{I} - \mathbf{H}_k (\mathbf{H}_k^T \mathbf{H})^{-1} \mathbf{H}_k^T$ and
$\mathbf{H}_{k+1} = [\mathbf{H}_k \ \mathbf{h}_{k+1}] \in \mathbf{R}^{N \times k+1}$

Order Recursive Least Squares

- The LS error is updated by:

$$J_{\min k+1} = J_{\min k} - \frac{(\mathbf{h}_{k+1}^T \mathbf{P}_k^{\perp} \mathbf{x})^2}{\mathbf{h}_{k+1}^T \mathbf{P}_k^{\perp} \mathbf{h}_{k+1}}$$

1. If $\mathbf{h}_{k+1}$ is orthogonal to all the previous one (columns of $\mathbf{H}_k$) then $\mathbf{H}_k^T \mathbf{h}_{k+1} = \mathbf{0}$ and

$$\hat{\theta}_{k+1} = \begin{bmatrix} \hat{\theta}_k \\ \frac{\mathbf{h}_{k+1}^T \mathbf{P}_k^{\perp} \mathbf{x}}{\mathbf{h}_{k+1}^T \mathbf{P}_k^{\perp} \mathbf{h}_{k+1}} \end{bmatrix}$$

Order Recursive Least Squares

2. The term $\mathbf{P}_k^{\perp}\mathbf{x} = (\mathbf{I} - \mathbf{H}_k(\mathbf{H}_k^T\mathbf{H}_k)^{-1}\mathbf{H}_k^T)\mathbf{x} = \mathbf{x} - \mathbf{H}_k\hat{\boldsymbol{\theta}}_k = \boldsymbol{\epsilon}_k$ is the LS error vector or the *data residual* that cannot be modeled by the columns of $\mathbf{H}_k$. It represents the component of $\mathbf{x}$ orthogonal to the space spanned by the columns of $\mathbf{H}_k$.
3. If $\mathbf{h}_{k+1}$ is nearly in the space spanned by $\mathbf{H}_k$ then, $\mathbf{P}_k^{\perp}\mathbf{h}_{k+1}$ will be small and $\mathbf{H}_{k+1}$ will be close to singular.

Sequential (Recursive) Least Squares

- In many signal processing applications the received data are obtained by sampling a continuous-time waveform - as time progresses, more data become available.
- We want to update the estimator for each new available data. *e.g. DC level*

$$\hat{A}[N] = \underbrace{\hat{A}[N-1]}_{\text{old estimate}} + \underbrace{\frac{1}{N+1}\left(x[n] - \hat{A}[N-1]\right)}_{\text{correction}}$$

$$J_{min}[N] = J_{min}[N-1] + \frac{N}{N+1}(x[N] - \hat{A}[N-1])^2$$

Sequential (Recursive) Least Squares

- Consider the minimization of the weighted LS error criterion with $\mathbf{W} = \mathbf{C}^{-1}$, where $\mathbf{C}$ is the covariance matrix of the zero mean noise

$$J = (\mathbf{x} - \mathbf{H}\theta)^T \mathbf{C}^{-1} (\mathbf{x} - \mathbf{H}\theta)$$

- From WLSE we know that

$$\hat{\theta} = (\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{C}^{-1} \mathbf{x}$$

- And from BLUE that

$$\mathbf{C}_{\hat{\theta}} = (\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H})^{-1}$$

Sequential (Recursive) Least Squares

- If $\mathbf{C}$ is diagonal or the noise is uncorrelated, then $\hat{\theta}$ can be computed sequentially in time.
- Let

$$\mathbf{C}[n] = \text{diag}(\sigma_0^2, \sigma_1^2, \ldots, \sigma_n^2)$$

$$\mathbf{H}[n] = \begin{bmatrix} \mathbf{H}[n-1] \\ \mathbf{h}^T[n] \end{bmatrix} = \begin{bmatrix} n \times p \\ 1 \times p \end{bmatrix}$$

$$\mathbf{x}[n] = [x[0]\ x[1]\ \ldots x[n]]^T$$

Sequential (Recursive) Least Squares

- Then, the batch estimator is

$$\hat{\theta}[n] = (\mathbf{H}^T[n]\mathbf{C}^{-1}[n]\mathbf{H}[n])^{-1}\mathbf{H}^T[n]\mathbf{C}^{-1}[n]\mathbf{x}[n]$$

with covariance matrix

$$\mathbf{C}_{\hat{\theta}} = \mathbf{\Sigma}[n] = (\mathbf{H}^T[n]\mathbf{C}^{-1}[n]\mathbf{H}[n])^{-1}$$

Sequential (Recursive) Least Squares

- The sequential LSE becomes:
  Estimator update:

$$\hat{\theta}[n] = \hat{\theta}[n-1] + \mathbf{K}[n]\left(x[n] - \mathbf{h}^T[n]\hat{\theta}[n-1]\right)$$

where

$$\mathbf{K}[n] = \frac{\mathbf{\Sigma}[n-1]\mathbf{h}[n]}{\sigma_n^2 + \mathbf{h}^T[n]\mathbf{\Sigma}[n-1]\mathbf{h}[n]}.$$

Covariance update:

$$\mathbf{\Sigma}[n] = (\mathbf{I} - \mathbf{K}[n]\mathbf{h}^T[n])\mathbf{\Sigma}[n-1].$$

Sequential (Recursive) Least Squares

- Commonly a forgetting factor γ is included in the RLS formulation in order to make this technique useful in estimating of time varying signals.
- If no γ is included we obtain the same solution by LS and SLS (RLS).
- A one-to-one correspondence between SLS and Kalman filter can be established.

Sequential LS - example

- *E.g.* Fourier analysis The signal model is

$$s[n] = a\cos 2\pi f_0 n + b\sin 2\pi f_0 n \qquad n \geqslant 0.$$

$\theta = [\mathbf{a}\ \mathbf{b}]$ is to be estimated by SLS. The covariance matrix of the noise should be diagonal (we assume the same variance σ^2.)

$$\mathbf{H}[1] = \begin{bmatrix} 1 & 0 \\ \cos 2\pi f_0 & \sin 2\pi f_0 \end{bmatrix}$$

$$\mathbf{x}[1] = \begin{bmatrix} x[0] \\ x[1] \end{bmatrix}$$

Sequential LS - example

- **Initialization using normal LS estimate:**
- Applying batch LSE we have

$$\begin{aligned}\hat{\theta}[1] &= \left(\mathbf{H}^T[1]\left(\frac{1}{\sigma^2}\mathbf{I}\right)\mathbf{H}[1]\right)^{-1}\mathbf{H}^T[1]\left(\frac{1}{\sigma^2}\mathbf{I}\right)\mathbf{x}[1] \\ &= \left(\mathbf{H}^T[1]\mathbf{H}[1]\right)^{-1}\mathbf{H}^T[1]\mathbf{x}[1] \qquad (1)\end{aligned}$$

- The initial covariance matrix of the estimator is

$$\mathbf{\Sigma}[1] = \left[\mathbf{H}^T[1]\left(\frac{1}{\sigma^2}\mathbf{I}\right)\mathbf{H}[1]\right]^{-1} = \sigma^2(\mathbf{H}^T[1]\mathbf{H}[1])^{-1} \qquad (2)$$

Sequential LS - example

- New "observation row" $\mathbf{h}^T[2] = [\cos 4\pi f_0 \;\; \sin 4\pi f_0] \Rightarrow$ the gain vector

$$\mathbf{K}[2] = \frac{\mathbf{\Sigma}[1]\mathbf{h}[2]}{\sigma^2 + \mathbf{h}^T[2]\mathbf{\Sigma}[1]\mathbf{h}[2]} \in \mathbf{R}^{2\times 1}$$

and

$$\hat{\theta}[2] = \hat{\theta}[1] + \mathbf{K}[2](x[2] - \mathbf{h}^T[2]\hat{\theta}[1])$$

$$\mathbf{\Sigma}[2] = (\mathbf{I} - \mathbf{K}[2]\mathbf{h}^T[2])\mathbf{\Sigma}[1]$$

Sequential LS - example

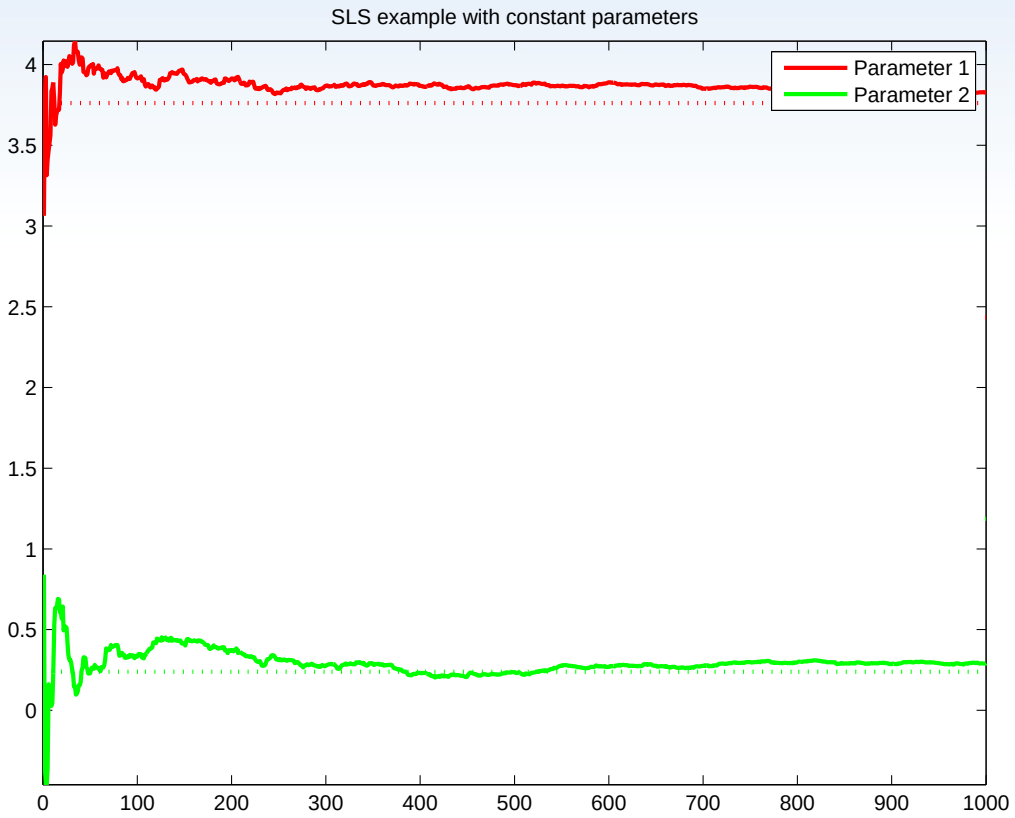
- It is also possible to incorporate a "forgetting factor" $\gamma \in [0, 1]$.
- Without γ, the estimator remembers everything and averages over a long period of time, which may be undesirable if the parameters change.
- This results in the following equations.

$$\mathbf{K}[2] = \frac{\mathbf{\Sigma}[1]\mathbf{h}[2]}{\sigma^2 + \mathbf{h}^T[2]\mathbf{\Sigma}[1]\mathbf{h}[2]} \in \mathbf{R}^{2\times 1}$$

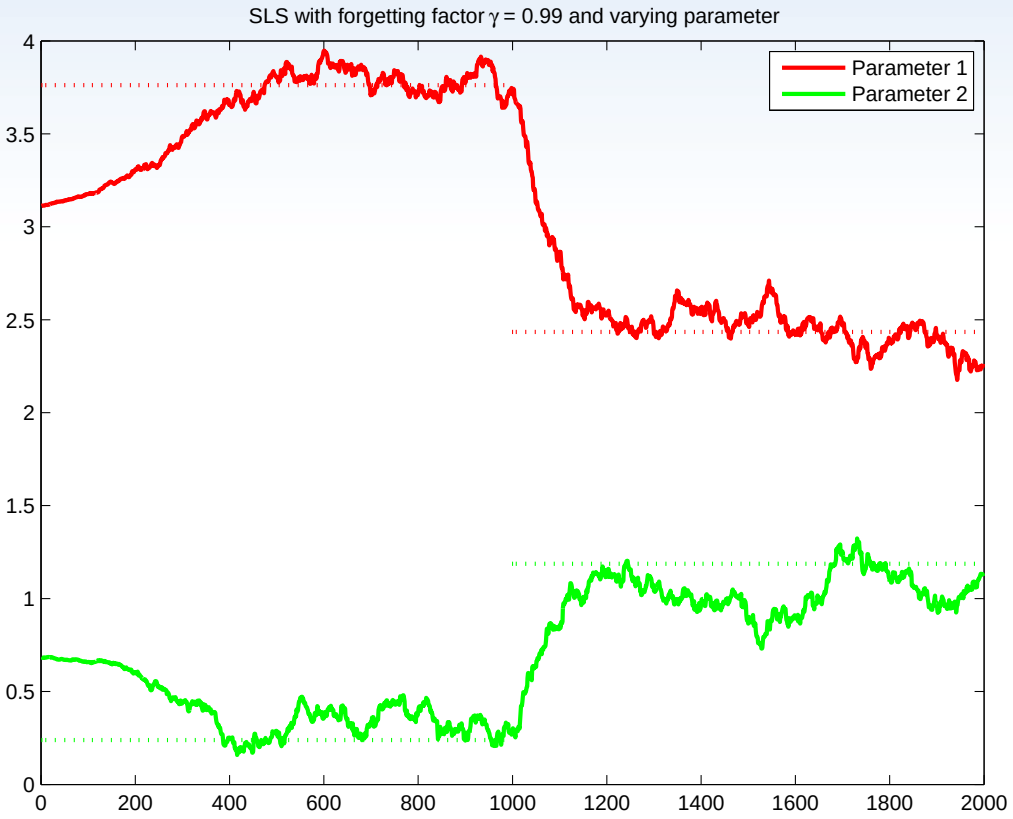
$$\hat{\theta}[2] = \hat{\theta}[1] + \mathbf{K}[2](x[2] - \mathbf{h}^T[2]\hat{\theta}[1])$$

$$\mathbf{\Sigma}[2] = \frac{1}{\gamma}(\mathbf{I} - \mathbf{K}[2]\mathbf{h}^T[2])\mathbf{\Sigma}[1]$$

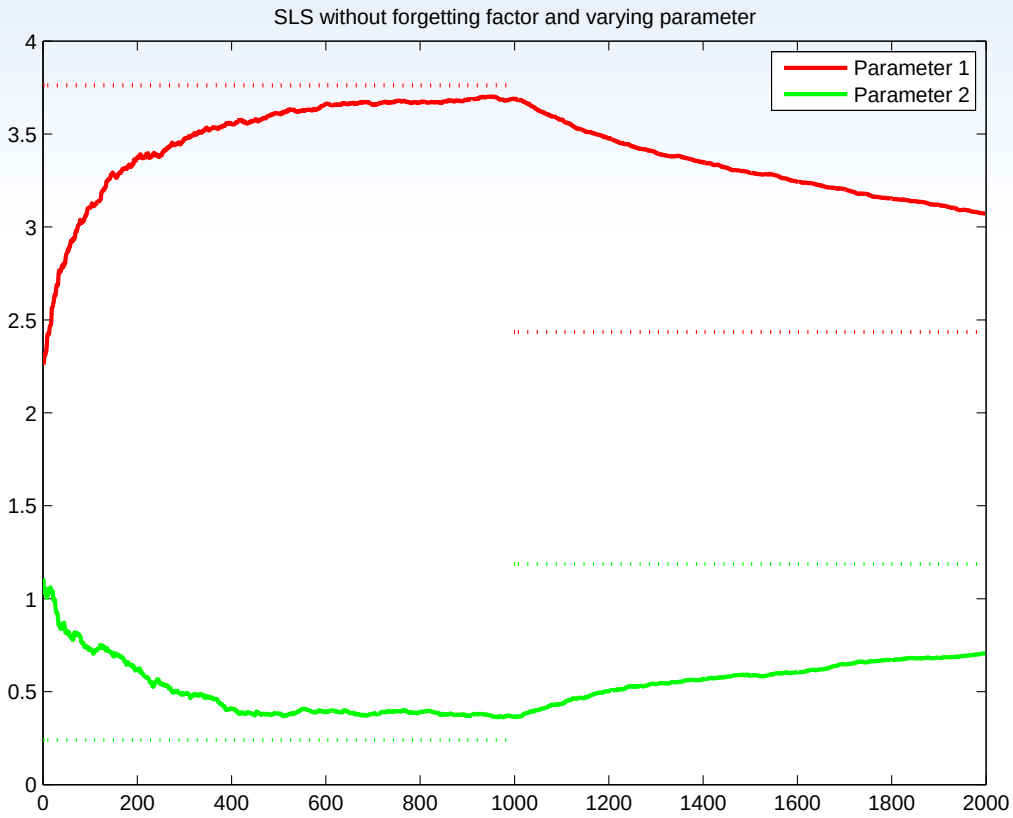
Sequential LS - example



Sequential LS - example



Sequential LS - example



Constrained Least Squares

- Some extra *a apriori* knowledge about a process might be expressed as some constraints on the paramaters.
- We assume $r < p$ linear, independent constraints

$$\mathbf{A}\theta = \mathbf{b}.$$

- Geometrically this might be expressed as something like *Find the best (in terms of LSE) parameter vector* θ *from the manifold* $\mathbf{A}\theta = \mathbf{b}$.
- Or:

$$\min_{\theta} (\mathbf{x} - \mathbf{H}\theta)^T (\mathbf{x} - \mathbf{H}\theta) \qquad \text{s.t.} \qquad \mathbf{A}\theta = \mathbf{b}.$$

Constrained Least Squares

- We determine the constrained LSE θ_c by minimizing the Lagrangian

$$J_c = (\mathbf{x} - \mathbf{H}\theta)^T(\mathbf{x} - \mathbf{H}\theta) + \lambda^T(\mathbf{A}\theta - \mathbf{b})$$

where λ is a $r \times 1$ vector of Lagrangian multipliers.

$$\frac{\partial J_c}{\partial \theta} = -2\mathbf{H}^T\mathbf{x} + 2\mathbf{H}^T\mathbf{H}\theta + \mathbf{A}^T\lambda$$

which by setting to zero produces

$$\hat{\theta}_c = \underbrace{(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x}}_{\hat{\theta}} - \frac{1}{2}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{A}^T\lambda$$

Constrained Least Squares

To find $\boldsymbol{\lambda}$ we impose the constraint

$$\mathbf{A}\boldsymbol{\theta}_c = \mathbf{A}\hat{\boldsymbol{\theta}} - \mathbf{A}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{A}^T\frac{\boldsymbol{\lambda}}{2} = \mathbf{b}$$

Therefore, we have

$$\hat{\boldsymbol{\theta}}_c = \hat{\boldsymbol{\theta}} - (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{A}^T[\mathbf{A}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{A}^T]^{-1}(\mathbf{A}\hat{\boldsymbol{\theta}} - \mathbf{b})$$

Regularized Least Squares

Regularization

- Regularization limits the size of the coefficients by adding a penalty term to the error:

$$\text{minimize } \|\mathbf{x} - \mathbf{H}\theta\|_2^2 + \lambda\|\theta\|,$$

  where $\lambda > 0$ and $\|\cdot\|$ is some norm or other penalty function.

- An alternative (but equivalent) formulation is

$$\text{minimize } \|\mathbf{x} - \mathbf{H}\theta\|_2^2 \text{ subject to } \|\theta\| < t.$$

Regularization

- Regularization is used for two reasons:
  1. The problem is ill-posed: There might be more variables than measurements ($P > N$ or even $P \gg N$), or the variables are linearly dependent of each other. In this case we need to select the best subset of features. Although there is a huge amount of subsets (2^P), efficient approximations exist. We will first study *implicit* or *embedded* feature selection and later study *explicit* or *wrapper* FS methods.
  2. The solution does not generalize well due to overlearning. Regularization prevents overlearning.

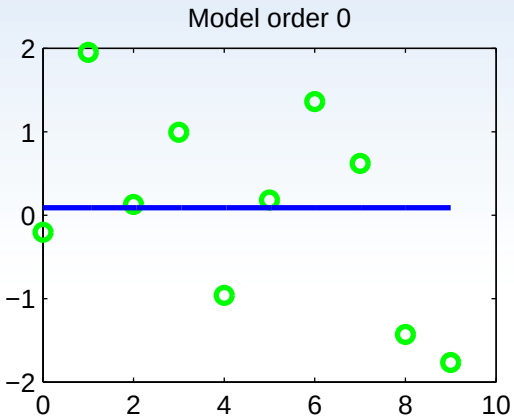
Regularization

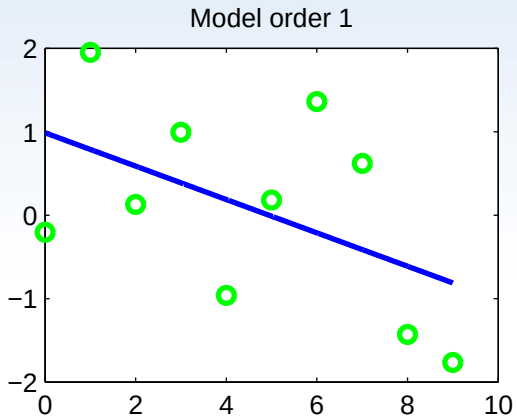
- In some cases the matrix **H** is ill-conditioned, i.e. its columns are linearly dependent and the condition number (ratio of the largest and the smallest singular value) is large. Also the singular values of **H** may gradually approach zero.
- As a result, the solution is very sensitive to perturbations in data.
- In regularization, we put additional constrains in order to obtain a more stable solution.
- The constraints may have several forms, but the easiest and most widely used penalize on the coefficient magnitude.
- This tends to make the coefficient closer to zero.

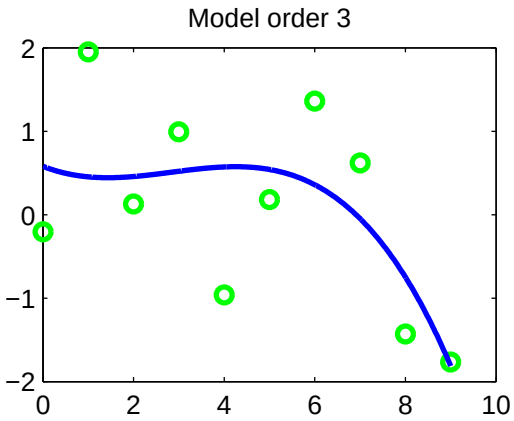
Overlearning

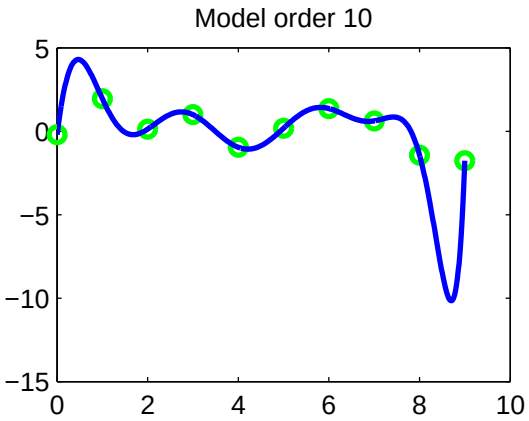
- An example of overlearning is illustrated below. The data consists of a noisy sinusoid, and we attempt to fit polynomial models of various orders.
- As the model order increases, the training error decreases.
- However, it seems that the most complex models starts to learn the noise, as well.
- This phenomenon is called *overlearning*.

Overlearning









Overlearning

- In the above plot, orders 0 and 1 are obviously too small.
- On the other hand, order 10 has the best fit to the training data, but is hardly the correct model (sinusoid) because the complexity of 10th order polynomial allows modeling the noise, as well.
- The solution is to use *regularization*, which attempts to push the coefficients towards zero thus simplifying the model.
- Regression models of this class include *ridge regression* and *the LASSO*.

Ridge Regression

- In Ridge regression (in non-regression context also called *Tikhonov regularization*) we attempt to minimize

$$\|\mathbf{x} - \mathbf{H}\theta\|_2^2 + \lambda\|\theta\|_2^2, \tag{3}$$

  where $\lambda > 0$ controls the trade-off between the regularizing constraints and the minimization of the sum of residuals. ($\|\cdot\|_2$ is the Euclidean norm.)
- A generalized version of the above multiplies θ with a matrix Γ:

$$\|\mathbf{x} - \mathbf{H}\theta\|_2^2 + \lambda\|\Gamma\theta\|_2^2, \tag{4}$$

Ridge Regression

- Using $\mathbf{\Gamma}$ one can for example penalize for the magnitudes of the derivatives in which case we favor smooth solutions.
- Regularization, here, means the trade-off between smoothness and the fidelity of the fit.
- Small λ puts more emphasis in the fidelity of the fit whereas large values of λ emphasize the smoothness constraint. The value of λ is also related to the sensivity of $\mathbf{H}$ and $\mathbf{x}$ to perturbation.
- The solution minimizing (4) can be shown to be

$$\hat{\theta}_{\text{reg}} = [\mathbf{H}^T\mathbf{H} + \lambda\mathbf{\Gamma}^T\mathbf{\Gamma}]^{-1}\mathbf{H}^T\mathbf{x}$$

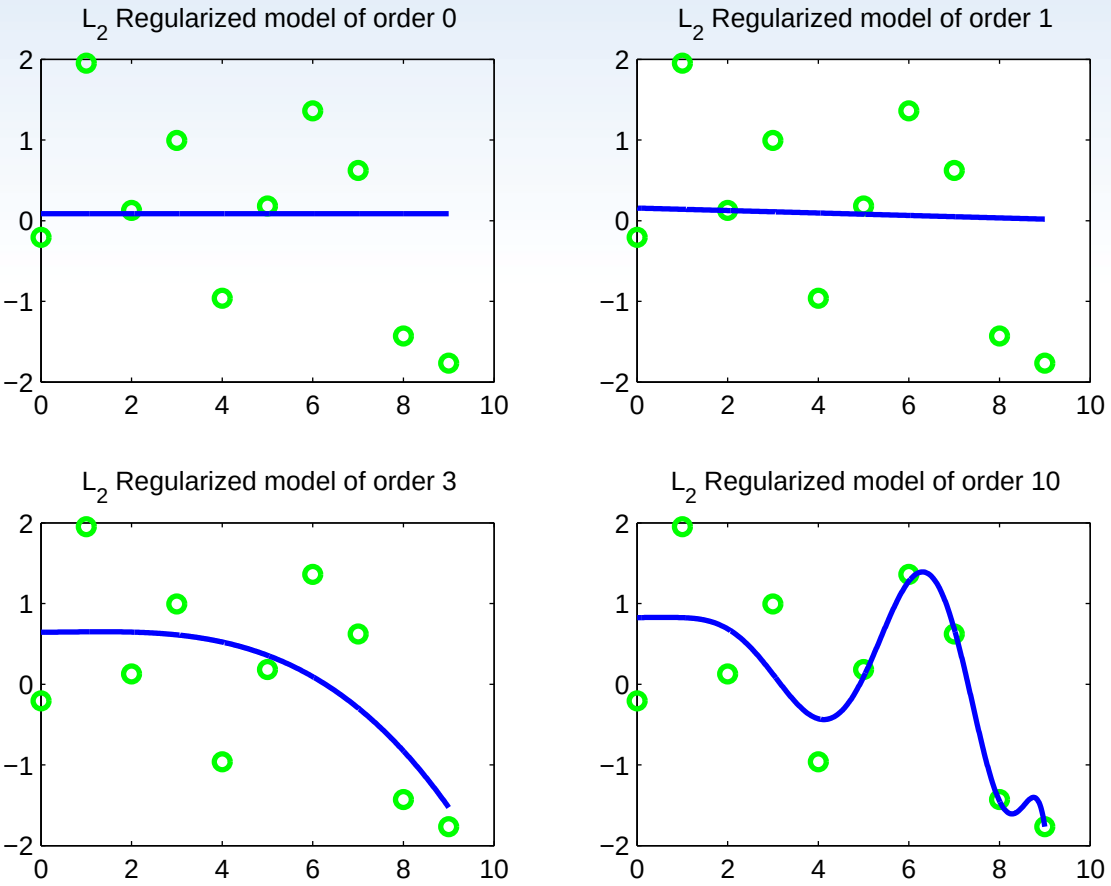
Ridge Regression

- In the restricted (and more widely used case) of (3) the solution is given by

$$\hat{\theta}_{reg} = [\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}]^{-1}\mathbf{H}^T\mathbf{x}$$

- In our earlier example case, regularization simplifies the more complex models, see below.
- Note1: In the example, we have modified the above formula by setting the upper left element of the matrix $\lambda\mathbf{I}$ to zero. This has the effect that the constant term is not penalized (i.e., a_0 in the model $y = a_0 + a_1x + a_2x^2 + a_Nx^N$). This is usually desirable.
- Note2: In this case we set (more or less arbitrarily) $\lambda = 1000$. The larger the value of λ, the smoother the result.

Ridge Regression



The LASSO

- The definition of the LASSO (least absolute shrinkage and selection operator) algorithm is very similar to the ridge regression.
- The only difference is that now we penalize the ℓ_1-norm of the parameter vector:

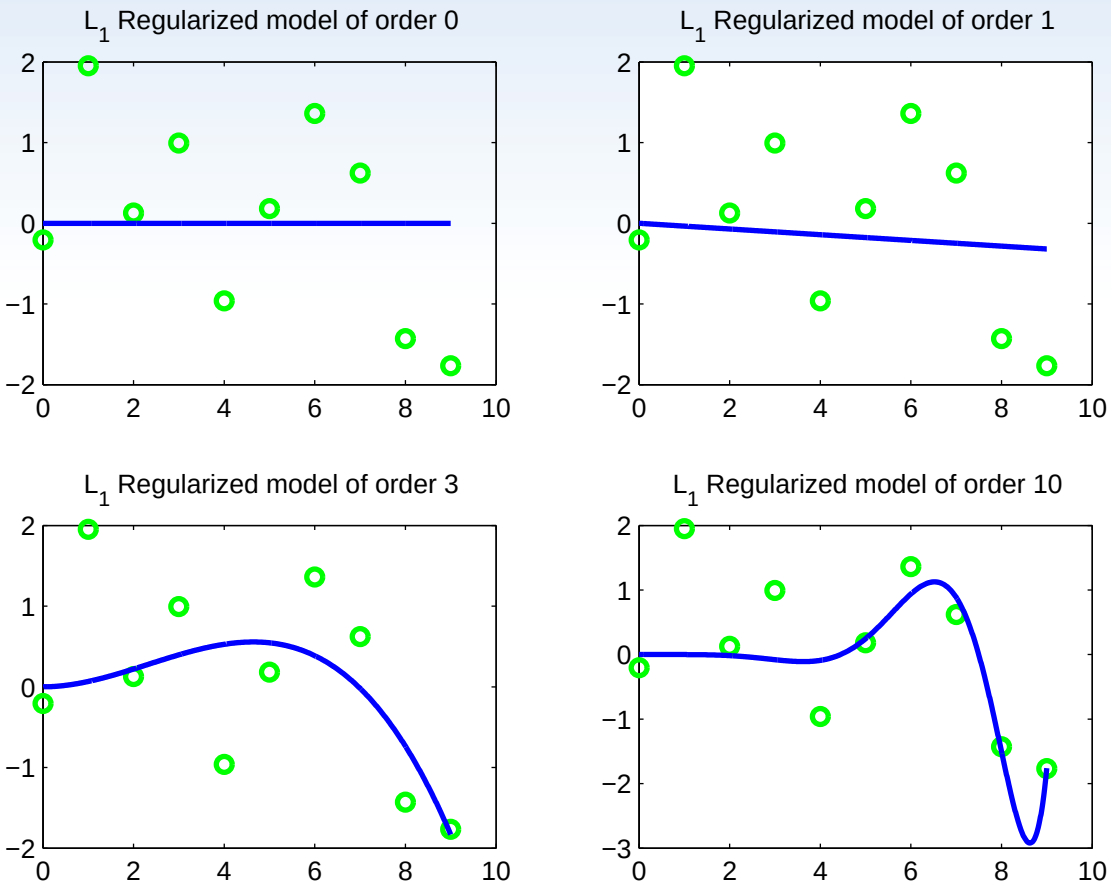
$$\theta_{\text{LASSO}} = \|\mathbf{x} - \mathbf{H}\theta\|_2^2 + \lambda\|\theta\|_1.$$

- This problem may be solved using quadratic programming or the *Least Angle Regression* algorithm.

The LASSO

- Free implementations available:
  - The LASSO, Elastic Net and LARS for **regression** problems: Sjöstrand, K.: *Matlab implementation of LASSO, LARS, the elastic net and SPCA* (2005), http://www2.imm.dtu.dk/pubdb/p.php?3897.
  - The LASSO, Elastic Net and LARS with Logistic Regression for **classification** problems: http://www-stat.stanford.edu/~tibs/glmnet-matlab/.
- Also Matlab has LASSO implementation since R2012a. Easy to use but not very fast.
- The result of using ℓ_1 regularization in our test case is shown below(with $\lambda = 1000$).

The LASSO

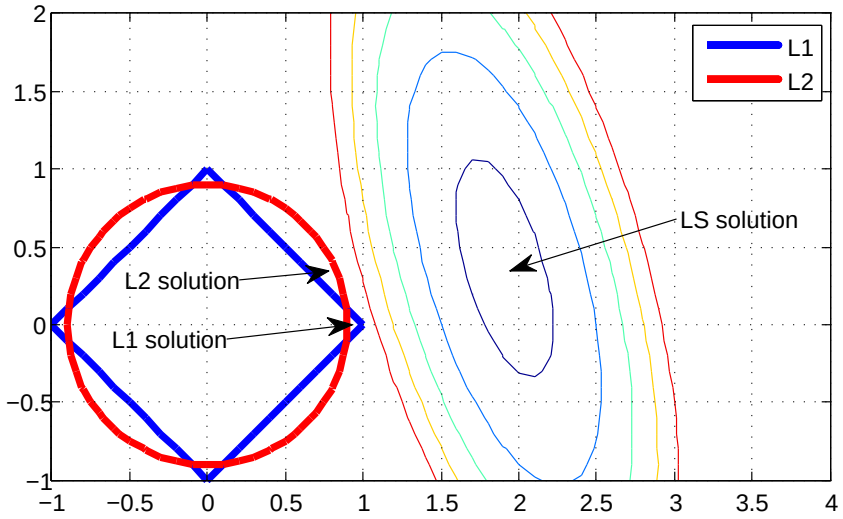

L_1 Regularized model of order 0
L_1 Regularized model of order 1
L_1 Regularized model of order 3
L_1 Regularized model of order 10

The LASSO

- It can be seen that the results are similar to ridge regression.
- However, there is a significant difference: the LASSO solution is *sparse*, i.e., many of the parameters are zero.
- For example, in the 10th order LASSO model, only 4 coefficients of 11 are nonzero (for powers 6, 7, 8 and 9). For ridge regression, all 11 coefficients are nonzero.
- The LASSO uses only the most important features thus simplifying the model automatically.
- This is the key property of LASSO: it incorporates feature and model selection into the optimization.

Why LASSO Result is Sparse?

- Graphically, the isosurfaces of the ℓ_1 penalty are diamonds, whose corners tend to have the minimum prediction cost.



Bayesian Interpretation

- The ℓ_1 regularization can also be formulated in a Bayesian manner: Assuming a Laplacian prior for the coefficients is equivalent to the ℓ_1 penalty [2].
- This way, it can be solved by an EM / MCMC algorithm, and allows alternative isosurfaces instead of the diamond and circle.
- See, e.g., Figure 5 of the original paper: http://icml2008.cs.helsinki.fi/papers/574.pdf.
- This version of LASSO is implemented in the pmtk3 toolbox: http://code.google.com/p/pmtk3/.

[2] F. Caron and A. Doucet, "Sparse Bayesian nonparametric regression," *25th International Conference on Machine Learning (ICML)*, Helsinki, 2008.

Choice of the regularization coefficient λ

- A crucial question to the performance is the selection of the regularization parameter λ in

$$\text{minimize } \|\mathbf{x} - \mathbf{H}\theta\|_2^2 + \lambda\|\theta\|,$$

  for either the ℓ_1 or the ℓ_2 norm.
- A helpful tool in this is the *regularization path* that plots the coefficient values for all $\lambda > 0$ (the largest λ is the case where all coefficients go to zero).

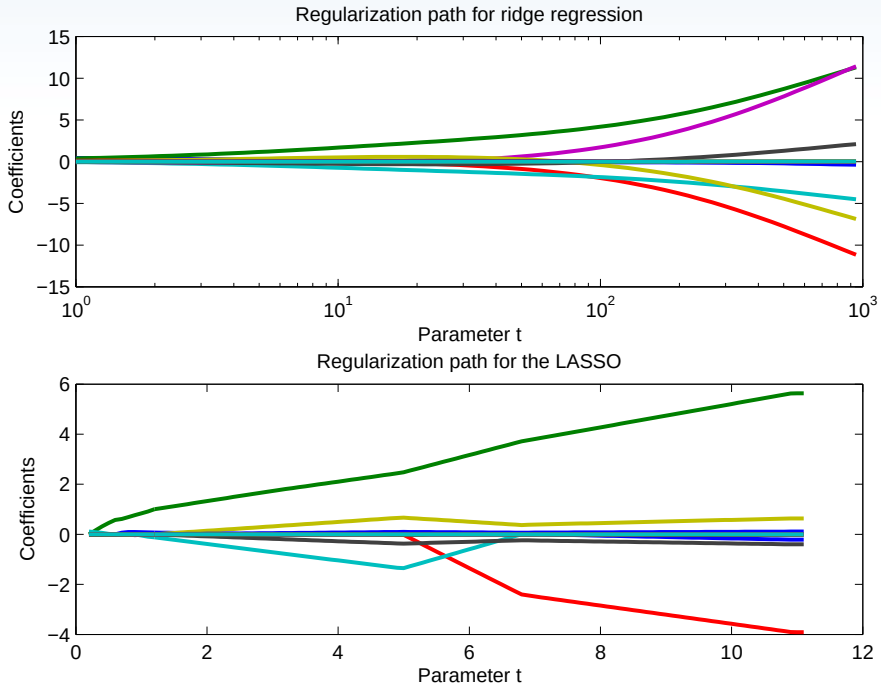
The Regularization Path

- The picture below shows the regularization paths of our problem for ridge regression (top) and the LASSO (bottom).
- The graph shows the coefficients for each regularization parameter value. Each coefficient has different color.
- The number of nonzero coefficients is always full 11 for ridge regression, and varies between 1 and 9 for the LASSO.
- Note that the paths are plotted for the equivalent problems

$$\text{minimize } \|\mathbf{x} - \mathbf{H}\theta\|_2^2 \text{ subject to } \|\theta\| < t$$

The Regularization Path

- An equivalent (left-right-flipped) plot could be done for λ, but this is the usual way due to technical reasons.



Selection of λ

- So, which point to select along the regularization path (i.e., what is a good value for λ)?
- The selection is usually done by *crossvalidation* (CV).
- k-fold crossvalidation does the following:
  1. Split the data randomly into k disjoint sets (called folds) of equal size.
  2. Train the model using all except one of the folds.
  3. Test the model performance on the remaining fold.
  4. The estimate of error is the mean of all k left-out-fold cases.
- Typically people use 10-fold CV or 5-fold CV.
- A special case is *leave-one-out* CV, where $k = N$, i.e., there are as many folds as samples.

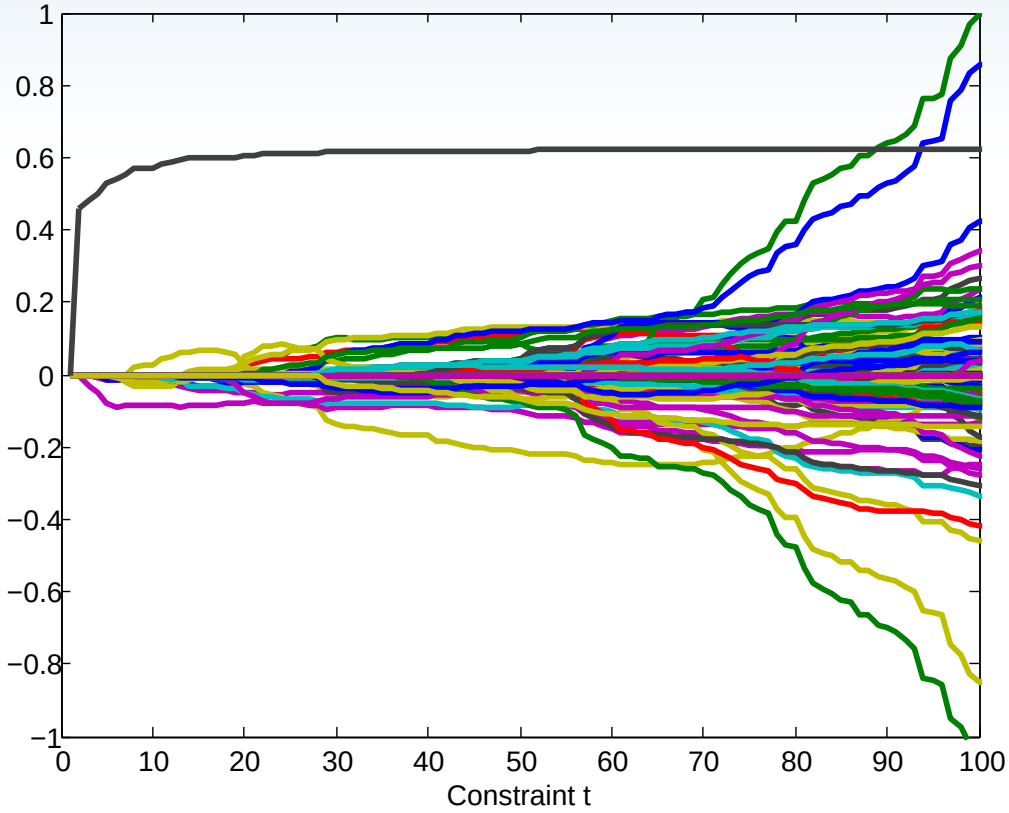
Example: Selection of λ

- In our recent paper[3] in ICANN2011 conference (http://www.cis.hut.fi/icann11/), we study LASSO regression for predicting face prevalence from brain fMRI data.
- There are $N = 244$ measurements from $P = 602$ sensors and one response variable.
- Thus, this is a case where $P \gg N$.
- We apply the LASSO (among other methods) for constrained regression.

[3] J-P. Kauppi, H. Huttunen, H. Korkala, I. Jääskeläinen, M. Sams and J. Tohka, "Face Prediction from fMRI Data During Movie Stimulus: Strategies for Feature Selection," *ICANN2011*.

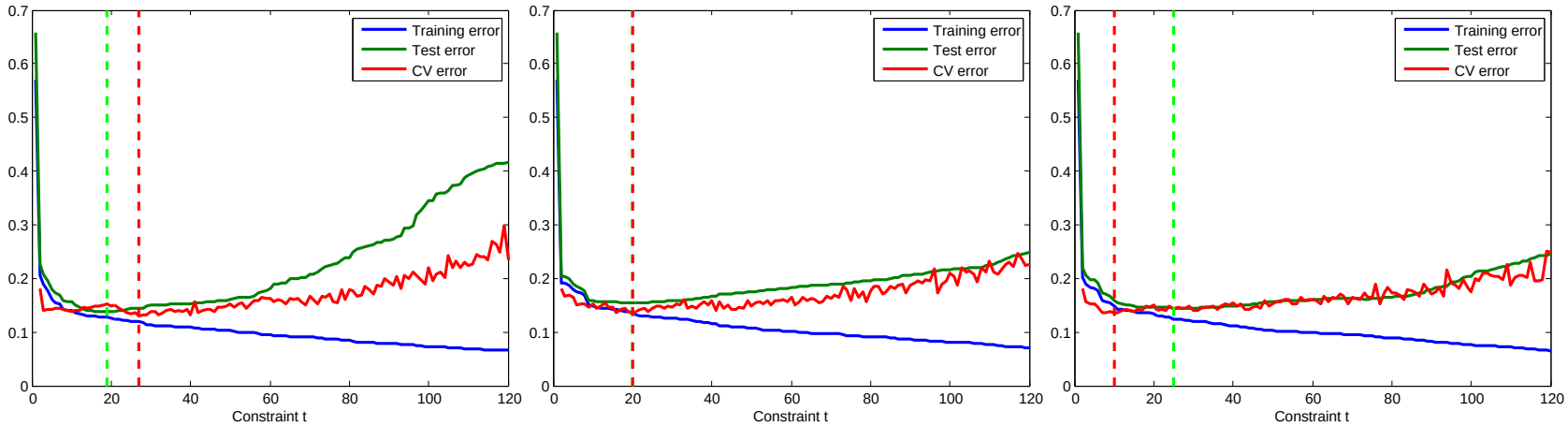
Example: Selection of λ

- The beginning of the regularization path is shown below.



Example: Selection of λ

- The errors for training set (blue), test set (green) are shown below for three test cases together with CV error (red).
- The hope is that the minimum of the CV error (red dashed line) would be the same as that of the test error (green dashed line).



Example: Selection of λ

- The middle is a case with correct CV error minimum, the others are slightly off their targets. However, none of the λ values selected by CV is totally wrong.
- Usually it is advisable to select a slightly simpler model than the CV suggests.
- A typical choice is to select the smallest t (largest λ) whose CV error is at most one standard error (calculated from the CV folds) away from the optimum.
- In the above figures, this would move the dashed red line a few units to the left.

The Elastic Net

- The Elastic Net is a compromise between ridge regression and the LASSO.
- The cost function to be minimized includes both constraints:

$$\text{minimize } \|\mathbf{x} - \mathbf{H}\boldsymbol{\theta}\|_2^2 + \lambda \left(\alpha\|\boldsymbol{\theta}\|_1 + (1-\alpha)\|\boldsymbol{\theta}\|_2^2\right)$$

  with $\lambda \geqslant 0$ and $0 \leqslant \alpha \leqslant 1$.
- If $\alpha = 1$ this becomes the LASSO and if $\alpha = 0$ this is ridge regression.

The Elastic Net

- Pros:
  - More flexibility
  - In the $P > N$ case the solution is unique even for more than N variables (if $\alpha \neq 1$). For LASSO (the case $\alpha = 1$), you can only construct unique models of up to N variables.
- Cons:
  - Yet another parameter to cross-validate. It is necessary to test all combinations of α and λ, which can be slow.

The LARS

- The LARS (least angle regression) algorithm was discovered by a colleague of the authors of the famous book: Hastie, Tibshirani, Friedman: "Elements of Statistical Learning"[4] while commenting on their manuscript.
- The LARS algorithm was later added to the second edition of the book.
- The LARS is a fast iterative algorithm, whose regularization path is close to that of the LASSO.
- The LARS can be modified to give the LASSO solution.

[4] http://www-stat.stanford.edu/~tibs/ElemStatLearn/

The LARS Algorithm

- Denote our model as usual $\mathbf{x} = \mathbf{H}\theta + \mathbf{w}$ with $\mathbf{H} = [\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_P]$.
  1. Initialize: Set all coefficients to $\theta = 0$ and standardize all $\mathbf{h}_j$ to have zero mean and unit norm. Set the active set (with nonzero coefficients) to $\mathcal{S} = \emptyset$.
  2. Calculate the model residual error: $\mathbf{r} = \mathbf{x} - \mathbf{H}\theta$.
  3. Find the column $\mathbf{h}_j$ of $\mathbf{H}$ most correlated with $\mathbf{r}$ (i.e., with largest $|\langle \mathbf{h}_j, \mathbf{r} \rangle|$) and add the column index to the active set $\mathcal{S}$.
  4. Move the columns in $\mathcal{S}$ a towards their joint LS coefficients until some column $\mathbf{h}_k$ has equally large absolute correlation with the residual $|\langle \mathbf{h}_k, \mathbf{r} \rangle|$. The original LARS paper[5] gives the formula for the optimal step size.
  5. Add k to $\mathcal{S}$ and return to step 4.

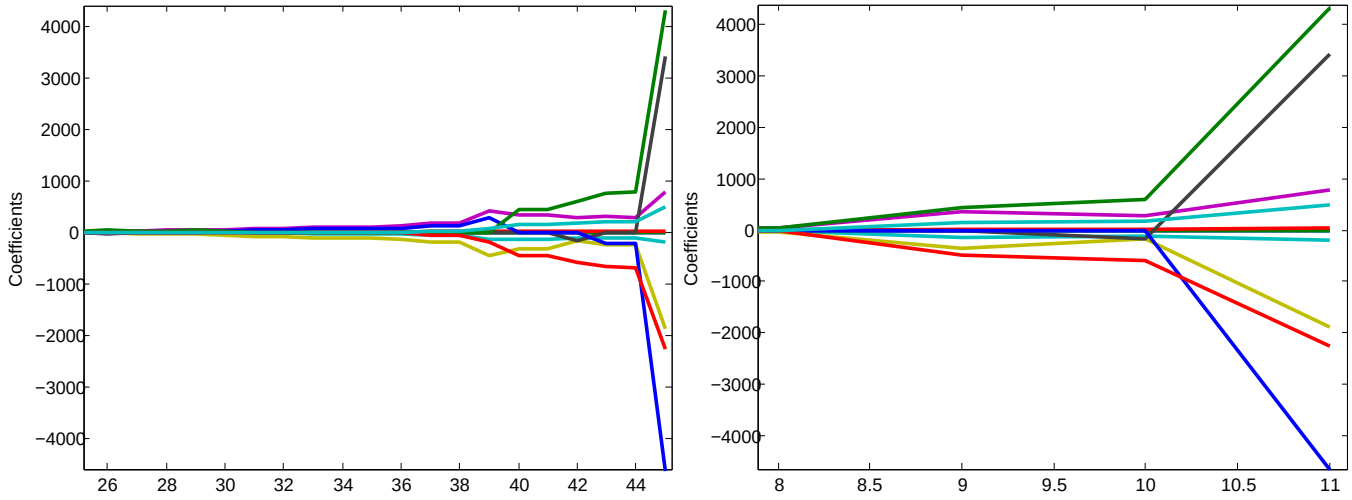
[5] Efron, Hastie, Johnstone, and Tibshirani, "Least Angle Regression". Annals of Statistics 32 (2): pp. 407-499, 2004.

LARS for LASSO

- It can be shown that the LARS algorithm can be used for finding the LASSO path.
- Their paths differ only at points where one coefficient changes sign (i.e., crosses zero).
- The fastest way to calculate LASSO path is thus using LARS with the modification at step 4:
  4. Move the columns in $\mathcal{S}$ a small step towards the LS coefficient: $\langle \mathbf{h}_j, \mathbf{r} \rangle$ for all $j \in \mathcal{S}$ until some column $\mathbf{h}_k$ has equally large correlation with the residual $\langle \mathbf{h}_k, \mathbf{r} \rangle$. *If a nonzero coefficient crosses zero, drop the variable from $\mathcal{S}$.*

LARS for LASSO

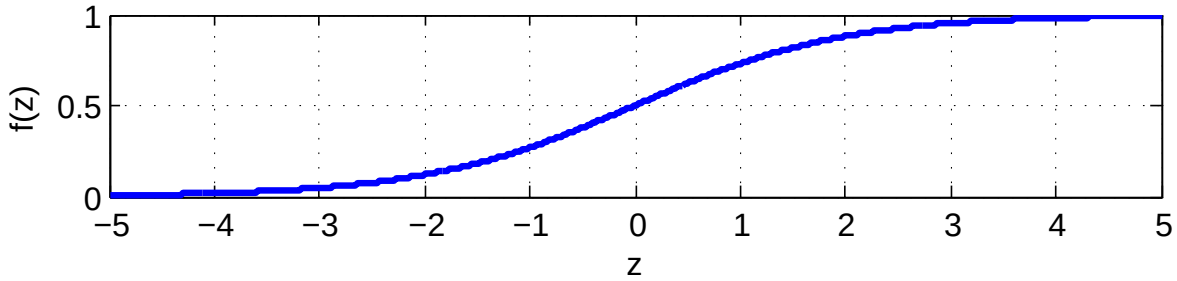
- A comparison of regularization paths is shown below (left: LASSO, right: LARS). The differences occur at zero crossings. For example the diabetes data set does not have any zero crossings, so the paths are identical.



Logistic Regression

- The LASSO, LARS, etc. can be extended to classification problems as well.
- This is done using *logistic regression*.
- Logistic regression models the posterior PDF using the logistic function:

$$f(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}.$$



Logistic Regression

- The logistic regression models the PDF for the class $k = 1, 2, \ldots, K$ as

$$p_k(\mathbf{x}) = \frac{\exp(\theta_k^T \mathbf{x})}{1 + \sum_{j=1}^{K} \exp(\theta_j^T \mathbf{x})}, \text{ for } k \neq K, \text{ and} \quad (5)$$

$$p_K(\mathbf{x}) = \frac{1}{1 + \sum_{j=1}^{K} \exp(\theta_j^T \mathbf{x})}, \quad (6)$$

where $\mathbf{x} = (1, x_1, x_2, \ldots, x_p)^T$ denotes the data and $\theta_k = (\theta_{k0}, \theta_{k1}, \theta_{k2}, \ldots, \theta_{kp})^T$ are the coefficients of the model.

Logistic Regression

- For the binary case, the probability of a class is given by:

$$p(\mathbf{x}) = \frac{1}{1 + \exp(\theta^T \mathbf{x})}$$

- Fast algorithms include the following:
  - J. Friedman, T. Hastie, R. Tibshirani, "Regularized Paths for Generalized Linear Models via Coordinate Descent," *Journal of Statistical Software* (2010).
  - B. Krishnapuram, L. Carin, M. Figueiredo and E. Hartemink, "Sparse multinomial logistic regression: Fast algorithms and generalization bounds," *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2005).

Logistic Regression

- Matlab implementation: http://www-stat.stanford.edu/~tibs/glmnet-matlab/.

ICANN MEG Mind Reading Challenge

- We participated in the Mind reading challenge from MEG data organized in the ICANN 2011 conference. [6] [7]

Rank	Team	Affiliation	Accuracy (%)
1.	Huttunen et al.	Tampere University of Technology	68.0
2.	Santana et al.	Universidad Politecnica de Madrid	63.2
3.	Jylänki et al.	Aalto University	62.8
4.	Tu & Sun (1)	East China Normal University	62.2
5.	Lievonen & Hyötyniemi	Helsinki Institute for Information Technology	56.5
6.	Tu & Sun (2)	East China Normal University	54.2
7.	Olivetti & Melchiori	University of Trento	53.9
8.	Van Gerven & Farquhar	Radboud University Nijmegen	47.2
9.	Grozea	Fraunhofer Institute FIRST	44.3
10.	Nicolaou	University of Cyprus	24.2

[6] H.Huttunen, J-P.Kauppi, J.Tohka, "Regularized logistic regression for mind reading with parallel validation," Proc. ICANN/PASCAL2 Challenge: MEG Mind-Reading. Aalto University publication series (2011).

[7] H.Huttunen, T.Manninen, J-P.Kauppi, J.Tohka, "Mind Reading with Regularized Multinomial Logistic Regression," *Machine Vision and Applications*, November 2012.

ICANN MEG Mind Reading Challenge Data

- The competition data consists of MEG recordings while watching five different movie categories.
- The data contains measurements of 204 planar gradiometer channels at 200Hz rate, segmented into samples of one second length.
- The task was to design and implement a classifier that takes as an input the MEG signals and produces the predicted class label.
- The training data with annotations had 727 one-second samples, and the secret test data 653 unlabeled samples.

ICANN MEG Mind Reading Challenge Data

- 50 samples of the training data were special, because they were recorded on the same day as the test data.

Error Estimation

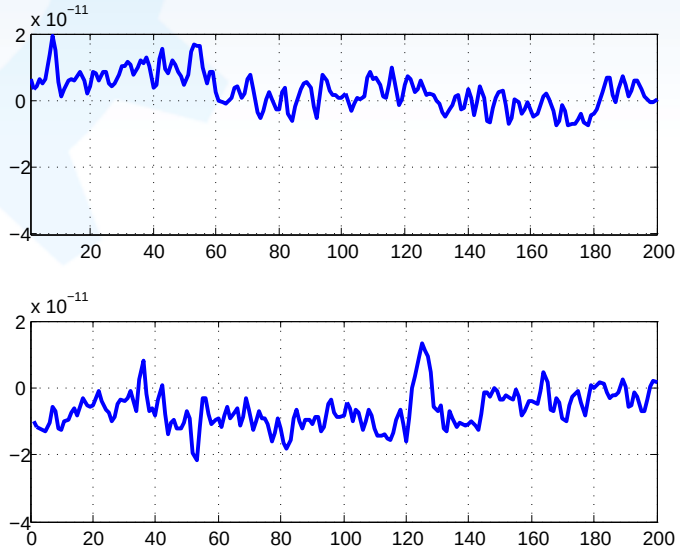
- For the performance assessment, we used a leave-one-out -kind of error estimate with special emphasis on the 50 samples from the second day.
- The computation was parallel in the Techila grid at TUT consisting of approx. 1000 cores. This is critical for the development, because in parallel you get feedback in minutes instead of days.
- The usage is very simple: The following code in Matlab spreads the computing to the entire campus.

```
gridfor n = 1 : N
   model = trainLogRegModel(trainingData(setdiff(1:N, n)));
   error(n) = testLogRegModel(trainingData(n), model);
end
```

The Curse of Dimensionality

- Since each measurement is a time series, it cannot be fed to classifier directly.
- Instead, we calculated a number of common quantities.
- In all, our pool of features consists of $11 \times 204 = 2244$ features. This means $2244 \times 5 = 11220$ parameters.

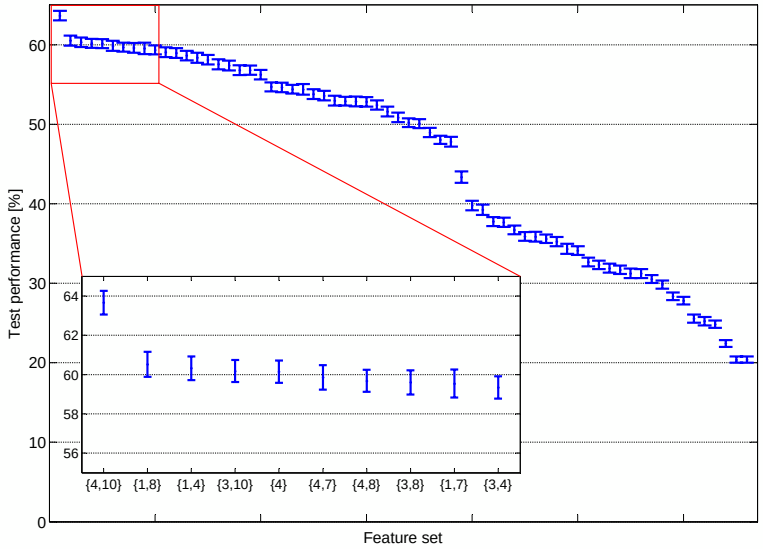
The Curse of Dimensionality



Intercept	$x_i^{(1)} = \hat{b}_i$
Slope	$x_i^{(2)} = \hat{a}_i$
Variance (d)	$x_i^{(3)} = \frac{1}{N}\sum_{n=1}^{N} \tilde{s}_i^2(n)$
Std. dev. (d)	$x_i^{(4)} = \sqrt{x_i^{(3)}}$
Skewness (d)	$x_i^{(5)} = \frac{1}{N}(x_i^{(4)})^{-3}\sum_{n=1}^{N} \tilde{s}_i^3(n)$
Kurtosis (d)	$x_i^{(6)} = \frac{1}{N}(x_i^{(4)})^{-4}\sum_{n=1}^{N} \tilde{s}_i^4(n)$
Variance	$x_i^{(7)} = \frac{1}{N}\sum_{n=1}^{N}(s_i(n) - \hat{b}_i)^2$
Std. dev.	$x_i^{(8)} = \sqrt{x_i^{(7)}}$
Skewness	$x_i^{(9)} = \frac{1}{N}(x_i^{(8)})^{-3}\sum_{n=1}^{N}(s_i(n) - \hat{b}_i)^3$
Kurtosis	$x_i^{(10)} = \frac{1}{N}(x_i^{(8)})^{-4}\sum_{n=1}^{N}(s_i(n) - \hat{b}_i)^4$
Fluctuation	$x_i^{(11)} = \frac{1}{N-1}\left\lvert\sum_{n=2}^{N} \mathrm{sgn}(s_i(n) - s_i(n-1))\right\rvert$

Feature Selection

- The set of 11220 features is prohibitively large for any feature selector.
- Thus, we experimented with all pairs of quantities among the 11. Their cross-validated performance is shown below.

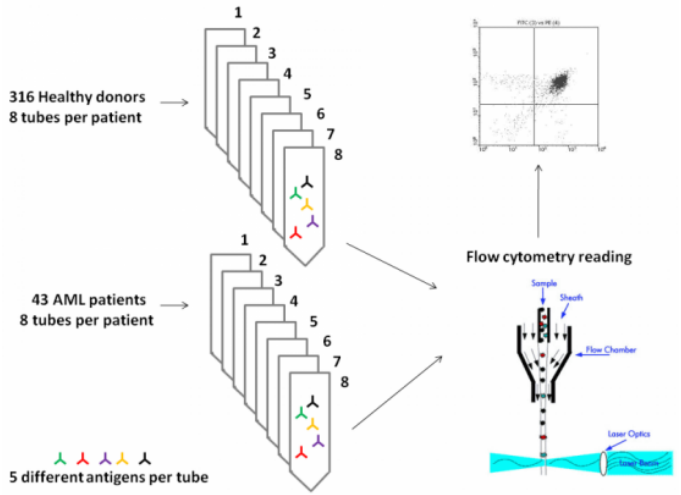


Feature Selection

- Luckily we did not in fact have the time to test all combinations before submission:
  - The best combination $[4, 10] = \{\text{stddev + kurtosis}\} \Rightarrow$ **57%** accuracy on the test set
  - The third best combination (our selection) $[1, 4] = \{\text{intercept + stddev}\} \Rightarrow$ **68%** on the test set
- Why did this happen? Overfitting on the training set?
- Lesson learned: If in doubt, use the simplest features possible (in our case, lower order moments).

DREAM6 AML Challenge

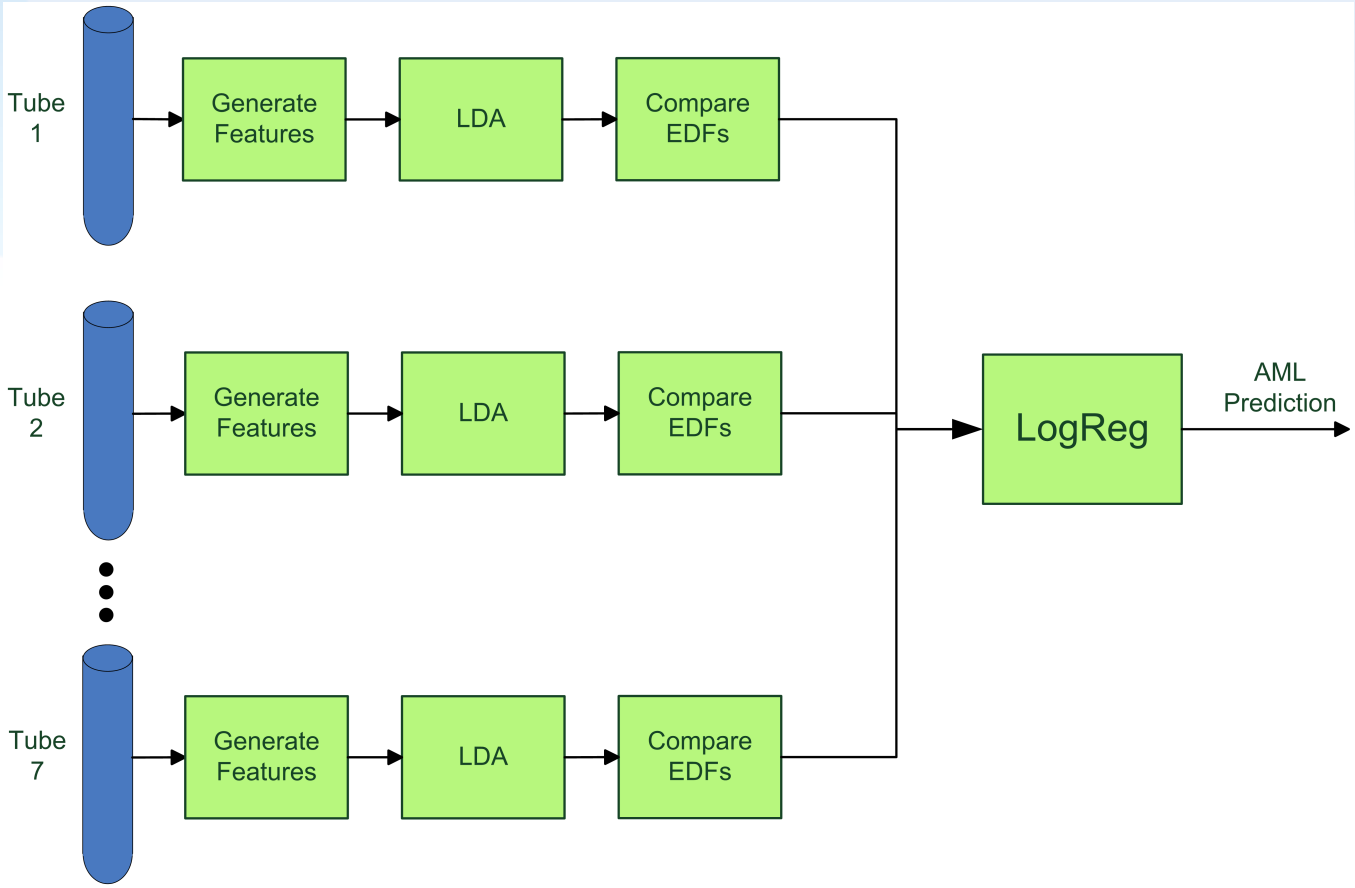
- The task was to design a binary classifier for discriminating AML cancer positive patients from healthy donors.
- The data consists of 8×5 measurements, i.e., there were 8 tubes each having 5 measurements. Thus the dimensions can not be simply concatenated.



Our Solution

- Our solution projects the measurements from each tube into a scalar using the LDA.
- In addition to plain cytometry measurements, we also generated extra features including inverses of the variables as well as multiplications and divisions with each two variable combination (in order to circumvent the restrictions of the linear model).
- For each tube, we trained a kernel density estimate of the distributions of healthy and AML-affected patients.
- The **Cramér-von Mises distance** between the model and test distribution is used as the input to the sparse logistic regression.

Our Solution



The Model

- The final prediction model becomes surprisingly simple:

$$P(\mathrm{AML} \mid \boldsymbol{\omega}) = \frac{1}{1 + \exp\left(0.31 - 29.9\omega_4^{(0)} - 9.5\omega_5^{(0)} + 35.2\omega_4^{(1)} + 51.5\omega_5^{(1)}\right)},$$

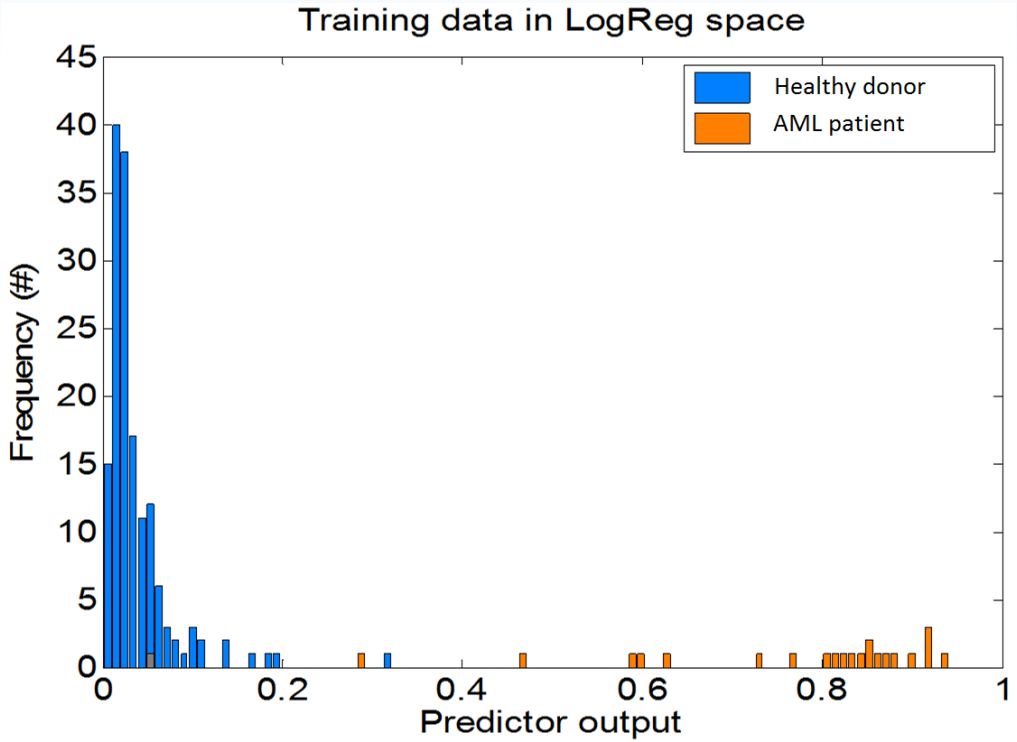
- In the above model, $\omega_k^{(0)}$ and $\omega_k^{(1)}$ are the Cramér-von Mises distances of the test sample for the k'th tube from the corresponding healthy and AML-affected distributions, respectively.
- The sparse model is **economical**, because only 2 out of 7 flow cytometry measurements are needed. This is what separates our solution from the other submissions.

The Model

- N. Aghaeepour, et al., "Critical assessment of automated flow cytometry data analysis techniques," *Nature Methods*, Feb. 2013.

The Result

- The LOO prediction result for the training data is shown below.



Alternative Feature Selection Methods

- The above methods are called *embedded feature selection methods*, because the feature selection is embedded in the cost function.
- More traditional approaches do feature selection explicitly:
    - **Forward stepwise selection** starts with empty set of features and iteratively adds the one that decreases the CV error the most. Matlab Statistics toolbox have implementations: `stepwisefit` and `sequentialfs`. The former is for regression and the latter for any cost function (e.g., classification error).

Alternative Feature Selection Methods

- **Backward elimination** starts with all features and iteratively removes the one that increases the CV error the most.
- **Floating selection** alternates forward and backward selection steps.
- General search heuristics such as **genetic algorithms** or **simulated annealing** have been used, as well. Their problem is randomness, which increases the variance of the result.

- Although there are 2^P possible solutions, the above heuristics produce reasonably good approximations.

Example run of `sequentialfs`

```
-----------------------------------------
Start forward sequential feature selection:

Initial columns included:  none
Columns that can not be included:  none

Step 1, added column 46, criterion value 998.029
Step 2, added column 40, criterion value 841.208
Step 3, added column 19, criterion value 572.885
Step 4, added column 48, criterion value 514.084
Step 5, added column 1, criterion value 473.465
Step 6, added column 30, criterion value 382.142
Step 7, added column 9, criterion value 329.322
Step 8, added column 2, criterion value 274.638
Step 9, added column 25, criterion value 257.528
Step 10, added column 7, criterion value 238.946
Step 11, added column 10, criterion value 219.969
Step 12, added column 54, criterion value 205.456
Step 13, added column 13, criterion value 191.416
Step 14, added column 52, criterion value 184.532
Step 15, added column 27, criterion value 177.366
Step 16, added column 43, criterion value 172.81
Step 17, added column 57, criterion value 170.307
Step 18, added column 56, criterion value 168.306
Step 19, added column 49, criterion value 166.119
Step 20, added column 20, criterion value 163.762
Step 21, added column 28, criterion value 161.622
Step 22, added column 61, criterion value 158.633
Step 23, added column 35, criterion value 156.619
Step 24, added column 18, criterion value 154.186
Step 25, added column 41, criterion value 152.792
Step 26, added column 22, criterion value 151.181
Step 27, added column 51, criterion value 149.77
Step 28, added column 23, criterion value 148.283
Step 29, added column 15, criterion value 146.752
Step 30, added column 45, criterion value 144.22
Step 31, added column 42, criterion value 142.466
Step 32, added column 37, criterion value 141.273
Step 33, added column 59, criterion value 140.76
Step 34, added column 55, criterion value 140.478
Step 35, added column 5, criterion value 140.082
Step 36, added column 12, criterion value 139.816
Step 37, added column 36, criterion value 138.885
Step 38, added column 6, criterion value 138.597
Step 39, added column 8, criterion value 138.584

Final columns included:  1 2 5 6 7 8 9 10 12 13 15
18 19 20 22 23 25 27 28 30 35 36 37 40 41 42 43 45
46 48 49 51 52 54 55 56 57 59 61
-----------------------------------------
```

Dimensionality Reduction Techniques

- Another alternative applies dimension reduction techniques before LS.
  - **Principal component regression** first drop the dimension using a linear transform such as the PCA followed by the LS.
  - A similar approach but in some sense more optimal one is **partial least squares**.
- The problem with transform approach is that the transform matrix may be unstable. That is, adding or deleting a sample from the training set may have a large effect in the transformation.

Numerical Implementation

Numerical Implementation of the linear OLS

- $\mathbf{H}^T\mathbf{H}$ must be non-singular for its inverse to exist. This may cause problems when implementing the estimation numerically.
- Recall, that originally we started with the signal model

$$\mathbf{s} = \mathbf{H}\theta$$

- If $\mathbf{H}$ were square, the solution would be as simple as

$$\theta = \mathbf{H}^{-1}\mathbf{s}$$

Numerical Implementation of the linear OLS

- Or, as we do not have $\mathbf{s}$, but a noisy version $\mathbf{x}$, we can write

$$\hat{\theta} = \mathbf{H}^{-1}\mathbf{x}$$

  which gives a perfectly viable estimator in this very restricted special case.
- This gives another interpretation of the linear LS approach: it is a problem of solving a linear system.
- Since the observation matrix seldom is a square matrix, we'll have to resort to a *pseudoinverse*.
- Depending on the rank of the matrix $\mathbf{H}$ the calculation of the pseudoinverse $\mathbf{H}^{+}$ is done in different ways.

Numerical Implementation of the linear OLS

1. **Case** $N \geqslant P$**.** If the observation matrix has full column rank[8], the pseudoinverse is the familiar one

$$\mathbf{H}^{+} = (\mathbf{H}^{T}\mathbf{H})^{-1}\mathbf{H}^{T}$$

In practice we seldom compute the inverse of the matrix $\mathbf{H}^{T}\mathbf{H}$. Typically QR factorization is used instead ($\mathbf{Q}$ is a orthogonal matrix[9] and $\mathbf{R}$ upper triangular and invertible). The QR factorization makes the calculation of the inverse more numerically stable.

[8] A full column rank matrix has columns that are linearly independent. In other words there are more data points than functions and the functions are linearly independent.

[9] An orthogonal matrix is a square matrix $\mathbf{Q}$ whose transpose is its inverse: $\mathbf{Q}^{T}\mathbf{Q} = \mathbf{Q}\mathbf{Q}^{T} = \mathbf{I}$.

Numerical Implementation of the linear OLS

- The orthogonalization simplifies the matrix inversion in the linear LS problem:

$$\mathbf{H}^T\mathbf{H} = \mathbf{R}^T\mathbf{Q}^T\mathbf{Q}\mathbf{R}$$

and the normal equation $\mathbf{H}^T\mathbf{H}\theta = \mathbf{H}^T\mathbf{x}$ simplifies to

$$\mathbf{R}^T\mathbf{R}\theta = \mathbf{R}^T\mathbf{Q}^T\mathbf{x}$$

$$\mathbf{R}\theta = \mathbf{Q}^T\mathbf{x}$$

which can be solved very quickly because $\mathbf{R}$ is triangular.

Numerical Implementation of the linear OLS

2. **Case** $N < P$**.** In some cases the number of functions can exceed the number of data points (for example when estimating the covariance matrix of several measured variables with only a few data points) or the columns may be linearly dependent. In such cases the pseudoinverse is usually solved using *singular value decomposition* (SVD). Any $m \times n$ matrix $\mathbf{A}$ can ve factored as

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}},$$

where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices and $\mathbf{\Sigma}$ is a diagonal matrix with so called *singular values* on the diagonal.

Numerical Implementation of the linear OLS

For example, if the observation matrix is

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 3 & 4 \\ 1 & 4 & 5 \end{pmatrix}$$

corresponding to the linear model
$x[n] = A + Bn + C(n+1) + w[n]$, the rank of $\mathbf{H}$ is only 2 and we can't calculate the usual pseudoinverse $(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$ (the matrix $\mathbf{H}^T\mathbf{H}$ is singular). However, the

Numerical Implementation of the linear OLS

SVD helps to see what causes the singularity. Now, the decomposition $\mathbf{H} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$ becomes

$$\mathbf{U} = \begin{pmatrix} -0.1064 & 0.7673 & 0.5655 & 0.0750 & 0.2732 \\ -0.2518 & 0.4864 & -0.8099 & 0.0081 & 0.2099 \\ -0.3972 & 0.2056 & 0.0591 & -0.5489 & -0.7037 \\ -0.5425 & -0.0752 & 0.0495 & 0.7735 & -0.3150 \\ -0.6879 & -0.3561 & 0.1358 & -0.3077 & 0.5356 \end{pmatrix}$$

and

$$\boldsymbol{\Sigma} = \begin{pmatrix} 9.3969 & 0 & 0 \\ 0 & 1.3034 & 0 \\ 0 & 0 & 0.0000 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{V} = \begin{pmatrix} -0.2113 & 0.7887 & -0.5774 \\ -0.5774 & -0.5774 & -0.5774 \\ -0.7887 & 0.2113 & 0.5774 \end{pmatrix}$$

Numerical Implementation of the linear OLS

For full rank matrices, the diagonal of $\mathbf{\Sigma}$ is all nonzero, and the pseudoinverse can be calculated simply by inverting the diagonal elements and transposing. However, in this case, one element is zero, and it can't be inverted. The natural way of proceeding is to leave the zero element on the diagonal as is, i.e., use the matrix

$$\mathbf{\Sigma}^{+} = \begin{pmatrix} \frac{1}{9.3969} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{1.3034} & 0 & 0 & 0 \\ 0 & 0 & 0.0000 & 0 & 0 \end{pmatrix}$$

Numerical Implementation of the linear OLS

Thus, the pseudoinverse of $\mathbf{H}$ becomes

$$\mathbf{H}^{+} = \mathbf{V}\boldsymbol{\Sigma}^{+}\mathbf{U}^{\mathsf{T}} = \begin{pmatrix} 0.4667 & 0.3000 & 0.1333 & -0.0333 & -0.2000 \\ -0.3333 & -0.2000 & -0.0667 & 0.0667 & 0.2000 \\ 0.1333 & 0.1000 & 0.0667 & 0.0333 & 0.0000 \end{pmatrix}$$

Finally, the estimator becomes

$$\hat{\theta} = \mathbf{H}^{+}\mathbf{x} = \mathbf{V}\boldsymbol{\Sigma}^{+}\mathbf{U}^{\mathsf{T}}\mathbf{x}$$

which is *the minimum length least squares solution.*