Linear Least-Squares

Numerical Methods for Deep Learning

Learning Objective: Linear Least-Squares

In this module we review computational methods for linear least-squares problems, which play a role in almost any learning tasks (including deep learning).

Learning tasks:

- image classification (supervised)
- image inpainting (semi-supervised)
- ...

Numerical methods:

- singular value decomposition
- steepest descent
- conjugate gradient method

Reminder: Supervised Learning Problem

Given examples (inputs)

$$\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n] \in \mathbb{R}^{n_f \times n}$$

and labels (outputs)

$$\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_n] \in \mathbb{R}^{n_c \times n},$$

find a classification/prediction function $f(\cdot, \boldsymbol{\theta})$, i.e.,

$$f(\mathbf{y}_j, \boldsymbol{\theta}) \approx \mathbf{c}_j, \quad j = 1, \ldots, n.$$

Regression and Least-Squares

Simplest option, a linear model with $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b})$ and

$$f(\mathbf{Y}, \mathbf{W}, \mathbf{b}) = \mathbf{W}\mathbf{Y} + \mathbf{b}\mathbf{e}_n^\top \approx \mathbf{C}$$

- $\mathbf{W} \in \mathbb{R}^{n_c \times n_f}$ are *weights*
- $\mathbf{b} \in \mathbb{R}^{n_c}$ are *biases*
- $\mathbf{e}_n \in \mathbb{R}^n$ is a vector of ones

Equivalent notation:

$$f(\mathbf{Y}, \mathbf{W}, \mathbf{b}) = \begin{pmatrix} \mathbf{W} & \mathbf{b} \end{pmatrix} \begin{pmatrix} \mathbf{Y} \\ \mathbf{e}_n^\top \end{pmatrix} \approx \mathbf{C}$$

Problem may not have a solution, or may have infinite solutions (when?). Solve through optimization

$$\min_{\mathbf{W}} \frac{1}{2} \|\mathbf{W}\mathbf{Y} - \mathbf{C}\|_F^2$$

(Frobenius norm: $\|\mathbf{A}\|_F^2 = \mathrm{trace}(\mathbf{A}^\top \mathbf{A}) = \sum_{i,j} \mathbf{A}_{i,j}^2$.)

Remark: Relation to Least-Squares

Consider the regression problem

$$\min_{\mathbf{W}} \frac{1}{2}\|\mathbf{W}\mathbf{Y} - \mathbf{C}\|_F^2.$$

It is easy to see that this is equivalent to

$$\min_{\mathbf{W}} \frac{1}{2}\|\mathbf{Y}^\top\mathbf{W}^\top - \mathbf{C}^\top\|_F^2,$$

which can be solved separately for each row in $\mathbf{W}$

$$\mathbf{W}(j,:)^\top = \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{Y}^\top\mathbf{w} - \mathbf{C}(j,:)^\top\|_F^2.$$

Notation: Let $\mathbf{A} = \mathbf{Y}^\top$ and $\mathbf{X} = \mathbf{W}^\top$ (easy to add bias here), we solve

$$\min_{\mathbf{X}} \frac{1}{2}\|\mathbf{A}\mathbf{X} - \mathbf{C}^\top\|_F^2$$

Detour: Image Inpainting (Semi-Supervised)

Image inpainting: Estimate an image function $f : [0, 1]^2 \to \mathbb{R}$ from partial observations.
First: Discretize the image on a grid with $n = n_x \cdot n_y$ pixels.
We obtain $\mathbf{f} \in \mathbb{R}^n$.

The input data is given by $\mathbf{c} \in \mathbb{R}^k$ with $k < n$ where

$$\mathbf{d}_i = f(\mathbf{x}_i) + \boldsymbol{\epsilon}_i, \quad i = 1, \ldots, k,$$

where $\mathbf{x}_i$ are points in $[0, 1]^2$, and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma \mathbf{I})$.

Write the inpainting problem as a linear least-squares problem

$$\min_{\mathbf{f}} \frac{1}{2\sigma} \|\mathbf{A}\mathbf{f} - \mathbf{c}\|^2.$$

Question: What is $\mathbf{A}$? How good will this approach be?

Optimality Conditions for Least-Squares

To minimize a function need to differentiate and equate to 0

$$\frac{\partial\left(\frac{1}{2}\|\mathbf{AX}-\mathbf{C}^\top\|_F^2\right)}{\partial\mathbf{X}} = 0$$

Compute the derivatives in three steps

1. $$\frac{\partial\left(\frac{1}{2}\|\mathbf{R}\|_F^2\right)}{\partial\mathbf{R}} = ???$$

2. $$\frac{\partial\left(\mathbf{AX}\right)}{\partial\mathbf{X}} = ???$$

3. Use chain rule

Least-Squares: Normal Equations

The necessary and sufficient optimality conditions for the least-squares problem are

$$\frac{\partial \left(\frac{1}{2}\|\mathbf{AX} - \mathbf{C}^\top\|_F^2\right)}{\partial \mathbf{X}} = \mathbf{A}^\top(\mathbf{AX} - \mathbf{C}^\top) = 0$$

Reorganize to obtain the **normal equations**

$$\mathbf{X} = (\mathbf{A}^\top\mathbf{A})^{-1}(\mathbf{A}^\top\mathbf{C}^\top).$$

Here, $\mathbf{A}^\top\mathbf{A} \in \mathbb{R}^{n_f \times n_f}$ must be invertible, i.e.,

- sufficient amount of data $(n > n_f)$
- data is linearly independent

Coding: Least-Squares for Classification

1. Write a code for solving

$$\min_{\mathbf{W},\mathbf{b}} \frac{1}{2}\|\mathbf{W}\mathbf{Y} + \mathbf{b}\mathbf{e}_n^\top - \mathbf{C}\|^2$$

   and apply it to the MNIST test data from

   `http://yann.lecun.com/exdb/mnist/`

2. Solve the problem using the normal equations derived above.
3. Use optimal weights to predict labels for test data. How well does your solution generalize?
4. Visualize the rows of $\mathbf{W}$ as images.

Ill-posedness and the SVD

If the data is linearly dependent or close to be linearly dependent, least-squares problem gives no good solution [2, 6, 3].
Understanding can be gained by the *Singular Value Decomposition* (SVD) (e.g., [1, Ch. 8])

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$$

where $\mathbf{U} \in \mathbb{R}^{n_f \times n_f}, \mathbf{V} \in \mathbb{R}^{n_f \times n}$ satisfy

$$\mathbf{U}^\top\mathbf{U} = \mathbf{I}, \quad \text{and} \quad \mathbf{V}^\top\mathbf{V} = \mathbf{I}$$

Diagonal of $\boldsymbol{\Sigma}$ contains the singular values $\sigma_1 \geq ... \sigma_{n_f} \geq 0$

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_{n_f} \end{pmatrix}$$

Ill-posedness and Regularization

Important is the *effective rank*: If $\sigma_j \ll \sigma_k$ for all $j \geq k$, then the effective rank of the problem is k.

If $k < n_f$, the least squares problem is ill-posed, i.e., solution does not exist or is unstable.
Small perturbations in $\mathbf{C}$ or $\mathbf{A}$ yield large perturbations in $\mathbf{X}$

Solve regularized problem: For some $\lambda > 0$ and matrix $\mathbf{G}$

$$\min_{\mathbf{X}} \frac{1}{2}\|\mathbf{A}\mathbf{X} - \mathbf{C}^\top\|_F^2 + \frac{\lambda}{2}\|\mathbf{G}\mathbf{X}\|_F^2$$

Exercise: solve the regularized least-squares problem

$$\mathbf{X} = (\mathbf{A}^\top\mathbf{A} + \lambda\mathbf{G}^\top\mathbf{G})^{-1}\mathbf{A}^\top\mathbf{C}^\top$$

The Bias-Variance Decomposition

Assume $\mathbf{C}^\top = \mathbf{A}\mathbf{X}_{\text{true}} + \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma\mathbf{I})$, $\lambda > 0$ fixed, $\mathbf{G} = \mathbf{I}$.
Then setting $\mathbf{A}_\lambda^\dagger = (\mathbf{A}^\top\mathbf{A} + \lambda\mathbf{I})^{-1}$

$$\begin{aligned}
\mathbf{X} - \mathbf{X}_{\text{true}} &= \mathbf{A}_\lambda^\dagger \mathbf{A}^\top \mathbf{C}^\top - \mathbf{X}_{\text{true}} \\
&= \left(\mathbf{A}_\lambda^\dagger \mathbf{A}^\top \mathbf{A} - \mathbf{I}\right) \mathbf{X}_{\text{true}} + \mathbf{A}_\lambda^\dagger \mathbf{A}^\top \boldsymbol{\epsilon} \\
&= -\lambda \mathbf{A}_\lambda^\dagger \mathbf{X}_{\text{true}} + \mathbf{A}_\lambda^\dagger \mathbf{A}^\top \boldsymbol{\epsilon}
\end{aligned}$$

Error depends on $\boldsymbol{\epsilon} \rightsquigarrow$ take expectation

$$\begin{aligned}
\mathbb{E}\|\mathbf{X} - \mathbf{X}_{\text{true}}\|_F^2 &= \mathbb{E}\|\mathbf{A}_\lambda^\dagger \mathbf{A}^\top \boldsymbol{\epsilon} - \lambda \mathbf{A}_\lambda^\dagger \mathbf{X}_{\text{true}}\|_F^2 \\
&= \overbrace{\lambda^2 \|\mathbf{A}_\lambda^\dagger \mathbf{X}_{\text{true}}\|_F^2}^{\|\text{bias}\|_{\text{F}}^2} + \overbrace{\sigma^2 \text{trace}\left(\mathbf{A}{\mathbf{A}_\lambda^\dagger}^T \mathbf{A}_\lambda^\dagger \mathbf{A}^\top\right)}^{\text{variance}}
\end{aligned}$$

Take home: No such thing as exact recovery!

Iterative Solvers for Least-Squares Regression

So far: Given $\mathbf{Y} \in \mathbb{R}^{n_f \times n}$ and $\mathbf{C} \in \mathbb{R}^{n_c \times n}$, solve

$$\min_{\mathbf{X}} \frac{1}{2} \left\| \mathbf{A}\mathbf{X} - \mathbf{C}^\top \right\|_F^2$$

directly using $\mathbf{X}^* = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{C}^\top$. Here

$$\mathbf{A} = \begin{pmatrix} \mathbf{Y}^\top & \mathbf{e}_n \end{pmatrix}, \quad \text{and} \quad \mathbf{X} = \mathbf{W}^\top \in \mathbb{R}^{(n_f+1) \times n_c}.$$

Problems:

1. Generating $\mathbf{A}^\top \mathbf{A}$ and solving normal equations is too costly for large-scale problems.
2. Exact solution not useful when problem is ill-posed $\rightsquigarrow$ add explicit regularization or do so implicitly by early stopping.

Iterative methods that avoid working with $\mathbf{A}^\top \mathbf{A}$

- Steepest descent
- Conjugate gradient for least-squares (CGLS)

Excellent references: Numerical Optimization [4], iterative linear algebra [5], general introduction [1]

Iterative Methods

General idea - obtain a sequence $\mathbf{X}_1, \ldots, \mathbf{X}_j, \ldots$ that converges to least-squares solution $\mathbf{X}^*$

$$\mathbf{X}_j \longrightarrow \mathbf{X}^*, \quad \text{for} \quad j \to \infty.$$

How fast does the sequence converge? Assume

$$\|\mathbf{X}_{j+1} - \mathbf{X}^*\| < \gamma_j \|\mathbf{X}_j - \mathbf{X}^*\|$$

where all $\gamma_j < 1$. Then

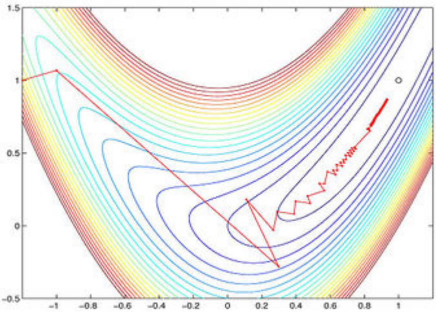
- If γ_j is bounded away from 0 and 1 the convergence is linear
- If $\gamma_j \to 0$ the convergence is superlinear
- If $\gamma_j \to 1$ the convergence is sublinear

The sequence converges quadratically if γ_j is bounded away from 0 and 1 and

$$\|\mathbf{X}_{j+1} - \mathbf{X}^*\| < \gamma_j \|\mathbf{X}_j - \mathbf{X}^*\|^2$$

Steepest Descent

Most basic iterative technique for solving $\min_{\mathbf{x}} \phi(\mathbf{x})$



$$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{d}_j \quad \text{with} \quad \mathbf{d}_j = -\nabla\phi(\mathbf{x}_j).$$

Interpretation 1: $\mathbf{d}_{j+1}$ maximizes local descent, i.e., solves

$$\min_s \phi(\mathbf{x}_j) + \mathbf{d}^\top \nabla\phi(\mathbf{x}_j) \quad \text{subject to} \quad \|\mathbf{d}\|_2 = 1.$$

Interpretation 2: $\mathbf{d}_j$ is orthogonal to level sets of ϕ at $\mathbf{x}_j$.

Steepest Descent for Least-Squares

Consider now

$$\phi(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{c}\|^2 \quad \text{with} \quad \nabla_{\mathbf{x}}\phi(\mathbf{x}) = \mathbf{A}^\top(\mathbf{A}\mathbf{x} - \mathbf{c}).$$

Steepest descent direction is $\mathbf{d}_j = \mathbf{A}^\top(\mathbf{c} - \mathbf{A}\mathbf{x}_j)$ and

$$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{d}_j$$

How to choose α_j? Idea: Minimize ϕ along direction $\mathbf{d}_j$

$$\alpha_j = \arg\min_{\alpha} \phi(\mathbf{x}_j + \alpha \mathbf{d}_j) = \arg\min_{\alpha} \frac{1}{2}\|\alpha \mathbf{A}\mathbf{d}_j - \mathbf{r}_j\|^2$$

with residual $\mathbf{r}_j = \mathbf{c} - \mathbf{A}\mathbf{x}_j$.

This leads to simple quadratic equation in 1D whose solution is

$$\alpha_j = \frac{\mathbf{r}_j^\top \mathbf{A}\mathbf{d}_j}{\|\mathbf{A}\mathbf{d}_j\|^2}$$

Algorithm: Steepest Descent for Least-Squares

for $j = 1, \ldots$

- Compute residual $\mathbf{r}_j = \mathbf{c} - \mathbf{A}\mathbf{x}_j$
- Compute the SD direction $\mathbf{d}_j = \mathbf{A}^\top \mathbf{r}_j$
- Compute step size $\alpha_j = \frac{\mathbf{r}_j^\top \mathbf{A}\mathbf{d}_j}{\|\mathbf{A}\mathbf{d}_j\|^2}$
- Take the step $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{d}_j$

Converges linearly, i.e.,

$$\|\mathbf{X}_{j+1} - \mathbf{X}^*\| < \gamma \|\mathbf{X}_j - \mathbf{X}^*\| \quad \text{with} \quad \gamma \approx \left| \frac{\kappa - 1}{\kappa + 1} \right|$$

Here, κ depends on condition number of $\mathbf{A}$, i.e.,

$$\kappa \approx \frac{\sigma^2_{\mathbf{max}}}{\sigma^2_{\mathbf{min}}}$$

Can be painfully slow for ill-conditioned problems

Accelerating Steepest Descent: Post-Conditioning

Idea: Improve convergence by transforming the problem

$$\phi(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{S}\mathbf{S}^{-1}\mathbf{x} - \mathbf{c}\|^2$$

Here: $\mathbf{S}$ is invertible
Solve in two steps:

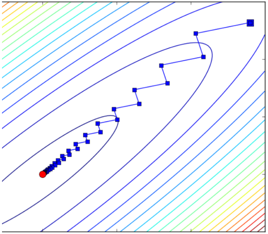
1. Set $\mathbf{z} = \mathbf{S}^{-1}\mathbf{x}$ and compute

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \frac{1}{2}\|\mathbf{A}\mathbf{S}\mathbf{z} - \mathbf{c}\|^2$$

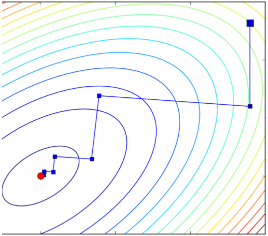
2. Then $\mathbf{x} = \mathbf{S}\mathbf{z}$.

Pick $\mathbf{S}$ such that $\mathbf{AS}$ is better conditioned.

original problem:



post-conditioned:



Exercise: Steepest Descent for Least-Squares

Goal: Program steepest descent and solve a simple problem.

To verify your code generate data using

$$\mathbf{c} = \mathbf{A}\mathbf{x}_{\mathrm{true}} + \boldsymbol{\epsilon}.$$

where $\boldsymbol{\epsilon}$ is random with zero mean and standard deviation 0.1 and

$$\mathbf{Y} = \begin{pmatrix} 1 & 1+a \\ 1 & 1+2a \\ 1 & 1+3a \end{pmatrix} \quad \text{and} \quad \mathbf{x}_{\mathrm{true}} = \begin{pmatrix} 1 \\ 1.2 \end{pmatrix}.$$

Plot errors $\|\mathbf{x}_j - \mathbf{x}_{\mathrm{true}}\|$ for $j = 1, \ldots$ and $a \in \{1, 10^{-2}, 10^{-5}\}$.

Conjugate Gradient Method for Least-Squares

CG is designed to solve quadratic optimization problems

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^\top \mathbf{H}\mathbf{x} - \mathbf{b}^\top \mathbf{x}$$

with $\mathbf{H}$ symmetric positive definite. In our case

$$\arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{c}\|^2 = \arg\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^\top \underbrace{\mathbf{A}^\top \mathbf{A}}_{=\mathbf{H}} \mathbf{x} - \underbrace{\mathbf{c}^\top \mathbf{A}}_{=\mathbf{b}^\top} \mathbf{x}$$

CG improves over SD by using previous step (not a memory-less method) and constructing a basis for the solution.

Facts:

- terminates after at most n steps (in exact arithmetic)
- good solutions for $j \ll n$
- convergence $\gamma_j \approx \left|\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right|^j$

CGLS: Conjugate Gradient Least-Squares

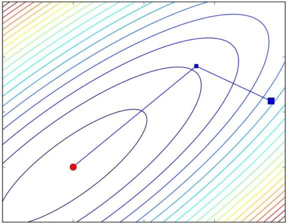
```
function x = mycgls(A,c,k,tol)
n = size(A,2);
x = zeros(n,1);
d = A'*c;   r = c;
normr2 = d'*d;
for j=1:k
    Ad = A*d; alpha = normr2/(Ad'*Ad);
    x =x+alpha*d;
    r = r - alpha*Ad;
    s = A'*r;
    normr2New = d'*d;
    if normr2New<tol;return; end
    beta = normr2New/normr2;
    normr2 = normr2New;
    d = s + beta*d;
end
```

Conjugate Gradient Least-Squares

- Uses the structure of the problem to obtain stable implementation
- Typically converges much faster than SD
- Accelerate using post conditioning

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{ASS}^{-1}\mathbf{x} - \mathbf{c}\|^2$$

- Faster convergence when eigenvalues of $\mathbf{S}^\top\mathbf{A}^\top\mathbf{AS}$ are clustered.



Iterative Regularization

Consider

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$$

- Assume that $\mathbf{A}$ has non-trivial null space
- The matrix $\mathbf{A}^\top \mathbf{A}$ is not invertible
- Can we still use iterative methods (CG, CGLS, ...)?

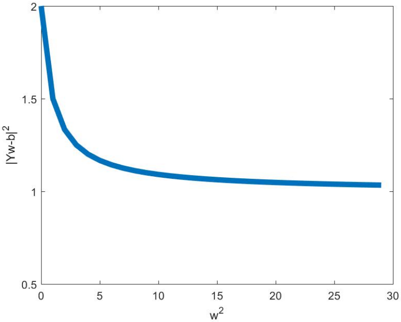
What are the properties of the iterates?

Excellent introduction to computational inverse problems [2, 6, 3]

Iterative Regularization: L-Curve

The CGLS algorithm has the following properties

- For each iteration $\|\mathbf{A}\mathbf{x}_k - \mathbf{c}\|^2 \leq \|\mathbf{A}\mathbf{x}_{k-1} - \mathbf{c}\|^2$
- If starting from $\mathbf{x} = 0$ then $\|\mathbf{x}_k\|^2 \geq \|\mathbf{x}_{k-1}\|^2$
- $\mathbf{x}_1, \mathbf{x}_2, \ldots$ converges to the minimum norm solution of the problem
- Plotting $\|\mathbf{x}_k\|^2$ vs $\|\mathbf{A}\mathbf{x}_k - \mathbf{c}\|^2$ typically has the shape of an L-curve



Cross Validation - 1

Finding good least-squares solution requires good parameter selection.

- λ when using Tikhonov regularization (weight decay)
- number of iteration (for SD and CGLS)

Suppose that we have two different "solutions"

$$\begin{array}{lcll} \mathbf{x}_1 & \rightarrow & \|\mathbf{x}_1\|^2 = \eta_1 & \|\mathbf{A}\mathbf{x}_1 - \mathbf{c}\|^2 = \rho_1. \\ \mathbf{x}_2 & \rightarrow & \|\mathbf{x}_2\|^2 = \eta_2 & \|\mathbf{A}\mathbf{x}_2 - \mathbf{c}\|^2 = \rho_2. \end{array}$$

How to decide which one is better?

Cross Validation - 2

Goal: Gauge how well the model can predict new examples.

Let $\{\mathbf{A}_{\mathrm{CV}}, \mathbf{c}_{\mathrm{CV}}\}$ be data that is **not used** for the training

Idea: If $\|\mathbf{A}_{\mathrm{CV}}\mathbf{x}_1 - \mathbf{c}_{\mathrm{CV}}\|^2 \leq \|\mathbf{A}_{\mathrm{CV}}\mathbf{x}_2 - \mathbf{c}_{\mathrm{CV}}\|^2$, then $\mathbf{x}_1$ is a better solution that $\mathbf{x}_2$.

When the solution depends on some hyper-parameter(s) λ, we can phrase this as bi-level optimization problem

$$\lambda^* = \arg\min_{\lambda} \|\mathbf{A}_{\mathrm{CV}}\mathbf{x}(\lambda) - \mathbf{c}_{\mathrm{CV}}\|^2,$$

where $\mathbf{x}(\lambda) = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{x}\|^2 + \frac{\lambda}{2}\|\mathbf{x}\|^2$.

Cross Validation - 3

To assess the final quality of the solution cross validation is not sufficient (why?).

Need a final testing set.

Procedure

- Divide the data into 3 groups $\{\mathbf{A}_{\text{train}}, \mathbf{A}_{\text{CV}}, \mathbf{A}_{\text{test}}\}$.
- Use $\mathbf{A}_{\text{train}}$ to estimate $\mathbf{x}(\lambda)$
- Use $\mathbf{A}_{\text{CV}}$ to estimate λ
- Use $\mathbf{A}_{\text{test}}$ to assess the quality of the solution

Important - we are not allowed to use $\mathbf{A}_{\text{test}}$ to tune parameters!

Σ : Linear Least-Squares

$$\min_{\mathbf{W}} \frac{1}{2}\|\mathbf{WY} - \mathbf{C}\|_F^2 \quad \rightsquigarrow \quad \min_{\mathbf{x}} \frac{1}{2}\|\mathbf{Ax} - \mathbf{b}\|_F^2$$

- optimality conditions: normal equations
- typically requires regularization
- direct: Tikhonov weight decay $\rightsquigarrow$ need to choose λ
- iterative: steepest descent, CG $\rightsquigarrow$ need to choose maximum number of iterations.
- classification: how to interpret output **WY** (not a probability!)
- image inpainting: effective choice of regularizer and parameter.

References

[1] U. M. Ascher and C. Greif. A First Course on Numerical Methods. SIAM, Philadelphia, 2011.

[2] P. C. Hansen. *Rank-deficient and discrete ill-posed problems*. SIAM Monographs on Mathematical Modeling and Computation. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998.

[3] P. C. Hansen. *Discrete inverse problems*, volume 7 of *Fundamentals of Algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2010.

[4] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer Science & Business Media, New York, Dec. 2006.

[5] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Second Edition. SIAM, Philadelphia, Apr. 2003.

[6] C. R. Vogel. *Computational Methods for Inverse Problems*. SIAM, Philadelphia, 2002.