Statistical Data Analysis

Dr. Jana de Wiljes

1. Dezember 2021

Universität Potsdam

Ill-posedness and Regularization

If the least squares problem is ill-posed, i.e., solution does not exist or is unstable.

Small perturbations in $\mathbf{y}$ or $\mathbf{X}$ yield large perturbations in β

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

$$\min_{\beta} \frac{1}{2}\|\mathbf{X}\beta - \mathbf{y}^\top\|^2 + \frac{\lambda}{2}\|\mathbf{G}\beta\|^2$$

Iterative Methods

Iterative Solvers for Least-Squares Regression

So far: Given $\mathbf{y} \in \mathbb{R}^n$, solve

$$\min_{\beta} \frac{1}{2} \|\mathbf{X}\beta - \mathbf{y}\|$$

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

$$\mathbf{X} \in \mathbb{R}^{n \times (p+1)} \quad \text{and} \quad \beta \in \mathbb{R}^{(p+1)}.$$

Problems:

1. Generating $\mathbf{X}^\top \mathbf{X}$ 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 $\mathbf{X}^\top \mathbf{X}$

- 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 $\beta_1, \ldots, \beta_j, \ldots$ that converges to least-squares solution β^*

$$\beta_j \longrightarrow \beta^*, \quad \text{for} \quad j \to \infty.$$

How fast does the sequence converge? Assume

$$\|\beta_{j+1} - \beta^*\| < \gamma_j \|\beta_j - \beta^*\|$$

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

$$\|\beta_{j+1} - \beta^*\| < \gamma_j \|\beta_j - \beta^*\|^2$$

Steepest Descent for Least-Squares

Consider now

$$\phi(\beta) = \frac{1}{2}\|\mathbf{X}\beta - \mathbf{y}\|^2 \quad \text{with} \quad \nabla_\beta \phi(\beta) = \mathbf{X}^\top(\mathbf{X}\beta - \mathbf{y}).$$

Steepest descent direction is $\mathbf{d}_j = \mathbf{X}^\top(\mathbf{y} - \mathbf{X}\beta_j)$ and

$$\beta_{j+1} = \beta_j + \alpha_j \mathbf{d}_j$$

How to choose α_j?

Idea: Minimize ϕ along direction $\mathbf{d}_j$

$$\alpha_j = \operatorname*{argmin}_\alpha \phi(\beta_j + \alpha \mathbf{d}_j) = \operatorname*{argmin}_\alpha \frac{1}{2}\|\alpha \mathbf{X}\mathbf{d}_j - \mathbf{r}_j\|^2$$

with residual $\mathbf{r}_j = \mathbf{y} - \mathbf{X}\beta_j$.

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

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

Algorithm: Steepest Descent for Least-Squares

for $j = 1, \ldots$

- Compute residual $\mathbf{r}_j = \mathbf{y} - \mathbf{X}\beta_j$
- Compute the SD direction $\mathbf{d}_j = \mathbf{X}^\top \mathbf{r}_j$
- Compute step size $\alpha_j = \frac{\mathbf{r}_j^\top \mathbf{X}\mathbf{d}_j}{\|\mathbf{X}\mathbf{d}_j\|^2}$
- Take the step $\beta_{j+1} = \beta_j + \alpha_j \mathbf{d}_j$

Converges linearly, i.e.,

$$\|\beta_{j+1} - \beta^*\| < \gamma \|\beta_j - \beta^*\| \quad \text{with} \quad \gamma \approx \left|\frac{\kappa - 1}{\kappa + 1}\right|$$

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

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

Can be painfully slow for ill-conditioned problems

Accelerating Steepest Descent: Post-Conditioning

Idea: Improve convergence by transforming the problem

$$\phi(\beta) = \frac{1}{2}\|\mathbf{XSS}^{-1}\beta - \mathbf{y}\|^2$$

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

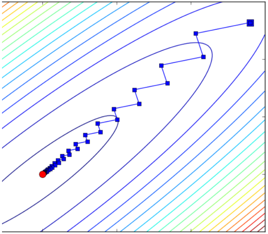
1. Set $\mathbf{z} = \mathbf{S}^{-1}\beta$ and compute

$$\mathbf{z}^* = \underset{\mathbf{z}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{XSz} - \mathbf{y}\|^2$$

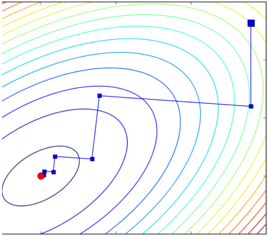
2. Then $\beta = \mathbf{Sz}$.

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

original problem:



post-conditioned:



Conjugate Gradient Method for Least-Squares

CG is designed to solve quadratic optimization problems

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

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

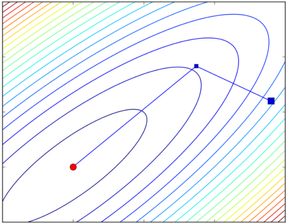
$$\underset{\beta}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{X}\beta - \mathbf{y}\|^2 = \underset{\beta}{\operatorname{argmin}} \frac{1}{2}\beta^\top \underbrace{\mathbf{X}^\top \mathbf{X}}_{=\mathbf{H}} \beta - \underbrace{\mathbf{y}^\top \mathbf{X}}_{=\mathbf{b}^\top} \beta$$

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 $\gamma_j \approx \left|\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right|^j$

Conjugate Gradient Least-Squares



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

$$\min_{\beta} \frac{1}{2} \|\mathbf{X}\mathbf{S}\mathbf{S}^{-1}\beta - \mathbf{y}\|^2$$

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

Iterative Regularization

Consider

$$\min_{\beta} \|\mathbf{X}\beta - \mathbf{b}\|^2$$

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

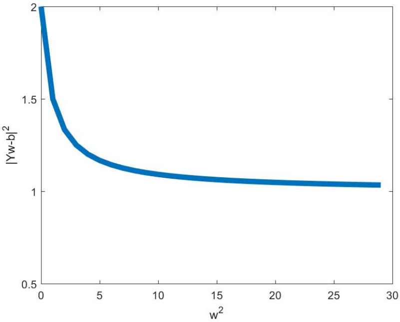
What are the properties of the iterates?

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

Iterative Regularization: L-Curve

The CGLS algorithm has the following properties

- For each iteration $\|\mathbf{X}\beta_k - \mathbf{y}\|^2 \leq \|\mathbf{X}\beta_{k-1} - \mathbf{y}\|^2$
- If starting from $\beta = 0$ then $\|\beta_k\|^2 \geq \|\beta_{k-1}\|^2$
- $\beta_1, \beta_2, \ldots$ converges to the minimum norm solution of the problem
- Plotting $\|\beta_k\|^2$ vs $\|\mathbf{X}\beta_k - \mathbf{y}\|^2$ 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"

$$\begin{aligned} \beta_1 \quad &\rightarrow \quad \|\beta_1\|^2 = \eta_1 \quad \|\mathbf{X}\beta_1 - \mathbf{y}\|^2 = \rho_1. \\ \beta_2 \quad &\rightarrow \quad \|\beta_2\|^2 = \eta_2 \quad \|\mathbf{X}\beta_2 - \mathbf{y}\|^2 = \rho_2. \end{aligned}$$

How to decide which one is better?

Cross Validation - 2

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

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

Idea: If $\|\mathbf{X}_{\mathrm{CV}}\beta_1 - \mathbf{y}_{\mathrm{CV}}\|^2 \leq \|\mathbf{X}_{\mathrm{CV}}\beta_2 - \mathbf{y}_{\mathrm{CV}}\|^2$, then β_1 is a better solution that β_2.

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

$$\lambda^* = \operatorname*{argmin}_{\lambda} \|\mathbf{X}_{\mathrm{CV}}\beta(\lambda) - \mathbf{y}_{\mathrm{CV}}\|^2,$$

where $\beta(\lambda) = \operatorname{argmin}_\beta \frac{1}{2}\|\mathbf{X}\beta - \beta\|^2 + \frac{\lambda}{2}\|\beta\|^2$.

Cross Validation - 3

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

Need a final testing set.

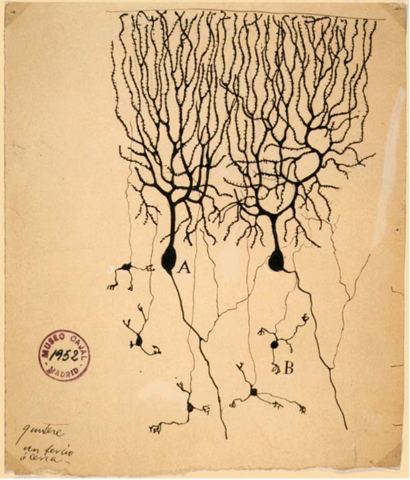
Procedure

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

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

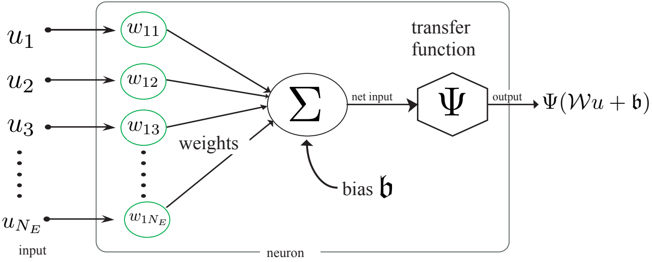
Neural Networks

Motivation from biology



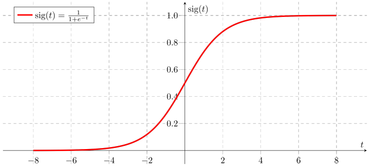
By Santiago Ramn y Cajal in 1899 see
https://de.wikipedia.org/wiki/Santiago_Ramn_y_Cajalfordetails

Neuron



u_1
u_2
u_3
u_{N_E}
input
w_{11}
w_{12}
w_{13}
w_{1N_E}
weights
$\sum$
net input
transfer function
Ψ
output
$\Psi(\mathcal{W}u + \mathfrak{b})$
bias $\mathfrak{b}$
neuron

Activation function example: sigmoid



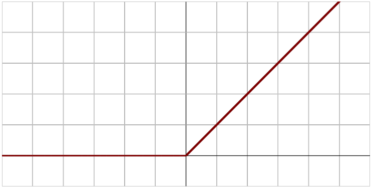
Sigmoid function:

$\mathrm{sig}(t) = \frac{1}{1+e^{-t}}$

Properties:

- Derivative: $\dfrac{1 + e^{-x} + xe^{-x}}{\left(1 + e^{-x}\right)^2}$
- $\mathrm{sig}'(t) = \mathrm{sig}(t)\,(1 - \mathrm{sig}(t))$

Activation function example: ReLu



Rectified linear unit:

$$\text{ReLu}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$
$$= \max\{0, x\}$$

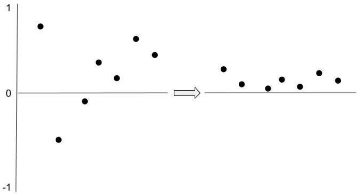
Properties:

- Derivative:

$$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases} \tag{1}$$

- very popular for Deep RL
- Dying ReLU problem - vanishing gradient problem.

Activation function example: Softmax



Softmax:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \qquad \text{for } i = 1, \ldots, K$$

and $\mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$.

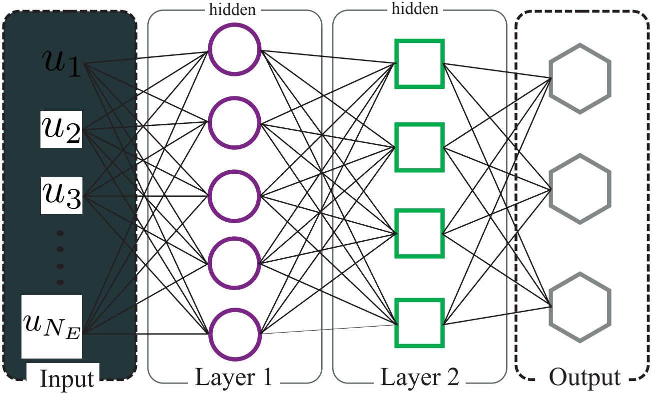
Properties:

- Derivative:

$$\frac{\partial}{\partial q_k} \sigma(\mathbf{q}, i) = \sigma(\mathbf{q}, i)(\delta_{ik} - \sigma(\mathbf{q}, k)). \tag{2}$$

- used in to normalize the output (map to a probability distribution)
- also used in RL to convert action values into action probabilities

Multilayer perceptron



Training Neural Network

1. Choose network architecture:
    - activation functions
    - hidden layers (shallow or deep)
    - number of neurons
    - etc.
2. Choose appropriate loss function E, e.g., least squares
3. Find minima via:
    - stochastic gradient descent
    - Backpropagation

Stochastic Gradient Descent

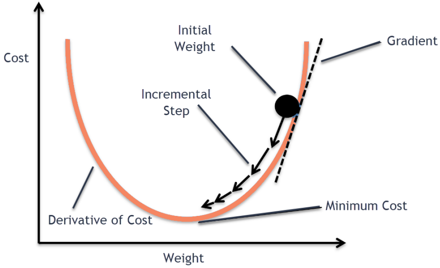


Image ref: https://morioh.com/p/bc6bc20e9739 and https://medium.com/38th-street-studios/exploring-stochastic-gradient-descent-with-restarts-sgdr-fa206c38a74e

Iterative weight improvement:

$$w := w - \eta \nabla E_i(w). \tag{3}$$

Backpropagation

Backpropagation

Backpropagation

Backpropagation

References i

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

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.

References ii

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

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

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

C. R. Vogel.
Computational Methods for Inverse Problems.
SIAM, Philadelphia, 2002.