Solving the Least-Squares Problem in Practice

Numerical Methods for Deep Learning

Iterative Solvers for Least-Squares Regression

Last time: 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: Generating $\mathbf{A}^\top \mathbf{A}$ and solving normal equations is too costly for large-scale problems.

Iterative Solvers for Least-Squares Regression

Last time: 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: Generating $\mathbf{A}^\top \mathbf{A}$ and solving normal equations is too costly for large-scale problems.

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

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

Excellent references: Numerical Optimization [?], iterative linear algebra [?], general introduction [?]

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

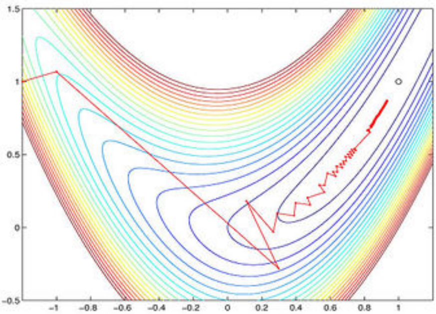
- If γ_j is bounded away from 0 and 1 the convergence is linear
- If $\gamma_j \to 0$ the convergence is superlinear
- If $\gamma_j \to 1$ the convergence is sublinear

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

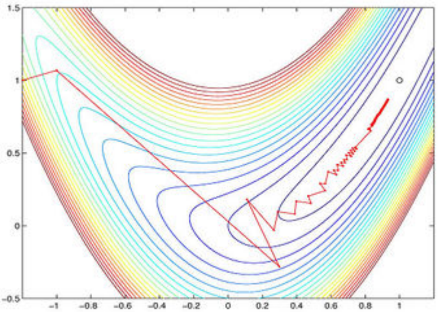
Most basic iterative technique for solving $\min_{\mathbf{x}} \phi(\mathbf{x})$



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

Steepest Descent

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



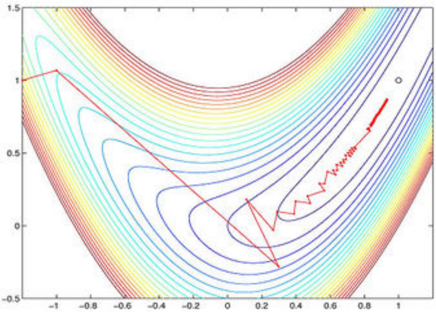
$$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{d}_j \quad \text{with} \quad \mathbf{d}_j = -\nabla\phi(\mathbf{x}_j).$$

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

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

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?

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

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

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.,

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

Can be painfully slow for ill-conditioned problems

Accelerating Steepest Descent: Post-Conditioning

Idea: Improve convergence by transforming the problem

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

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

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

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

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

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

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:

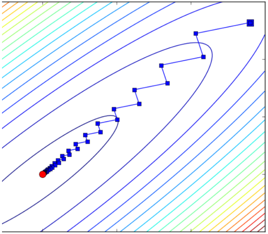
1. Set $\mathbf{z} = \mathbf{S}^{-1}\mathbf{x}$ and compute

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

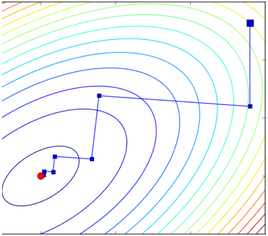
2. Then $\mathbf{x} = \mathbf{S}\mathbf{z}$.

Pick $\mathbf{S}$ such that $\mathbf{A}\mathbf{S}$ 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

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

Plot errors $\|\mathbf{x}_j - \mathbf{x}^*\|$ 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

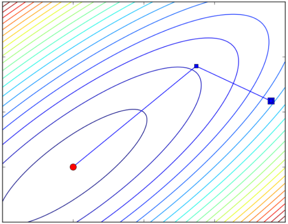
```
function x = cgls(A,c,k)
n = size(A,2);
x = zeros(n,1);
d = A'*c;   r = c;
normr2 = d'*d;
for j=1:k
    Ad = A*d; alpha = normr2/(Ad'*Ad);
    x  = x + alpha*d;
    r  = r - alpha*Ad;
    d  = A'*r;
    normr2New = d'*d;
    beta = normr2New/normr2;
    normr2 = normr2New;
    d = d + beta*d;
end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

Conjugate Gradient Least-Squares

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

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

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



Iterative Regularization

- Assume that $\mathbf{A}$ has a null space
- The matrix $\mathbf{A}^\top\mathbf{A}$ is not invertible
- Can we still use CGLS to solve(?) the least squares problem

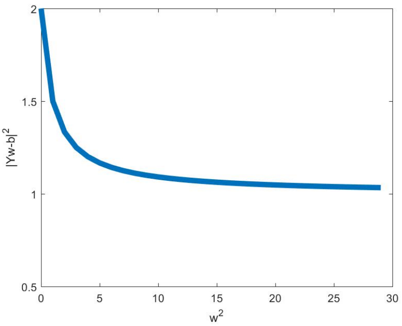
What are the properties of CGLS iterations?

Excellent introduction to computational inverse problems [**? ? ?**]

Iterative Regularization: L-Curve

The CGLS algorithm has the following properties

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



Cross Validation

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"

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

How to decide which one is better?

Cross Validation

Measure how well can each of the solutions predict new data.

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

Then 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 $\mathbf{x}_2$.

Cross Validation

Measure how well can each of the solutions predict new data.

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

Then 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 $\mathbf{x}_2$.

In general, if the solution depends on some hyper-parameter λ then the best one is

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

Cross Validation

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

Cross Validation

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!

Coding: Iterative Methods for Regression

Outline:

- Dataset: MNIST / CIFAR 10
- write a steepest descent specific to the problem

  `function x = sdLeastSquares(A,c,x0,maxIter)`
- write a conjugate gradient code

  `function x = cgLeastSquares(A,c,maxIter)`

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.