M2 MIAGE
/MIAGE ID App
2025-2026

OPTIMIZATION FOR DATA SCIENCE

January 19, 2026

Roadmap

- Today (session 6/8): → Recap on stochastic gradient
  → Regularization
- Wednesday (session 7/8) → Exercises regularization
  → last year's exam
- Next week (session 8/8)
  Lab / course project (due early February)

RECAP ON STOCHASTIC GRADIENT
(through Exercise 1 of Tutorial 3)

Data: $\{(x_i, y_i)\}_{i=1..n}$ $\quad x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$

Model: $x \mapsto x^T w$, parameterized by $w \in \mathbb{R}^d$

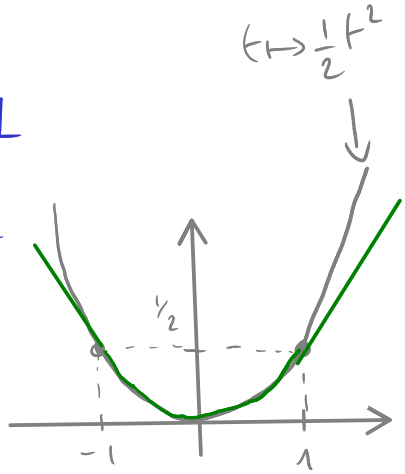
Goal: Find w such that $x_i^T w \approx y_i \quad \forall i = 1..n$

The exercise: Pick w that solves the optimization problem

$$(P) \quad \underset{w \in \mathbb{R}^d}{\text{minimize}} \quad f(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(x_i^T w - y_i)$$

where $\ell : t \mapsto \begin{cases} \frac{1}{2}t^2 & \text{if } |t| < 1 \\ |t| - \frac{1}{2} & \text{otherwise} \end{cases}$

Huber loss
(popular choice in statistics to deal with large errors/values of $x_i^T w - y_i$)



↳ $\forall i = 1..n$, $w \mapsto \ell(x_i^T w - y_i)$ is C^1

a) Justify that 0 is a lower bound on (P). Is it its minimum value?

$\forall i = 1..n$, $\ell(x_i^T w - y_i) \geq 0$ by definition ($\frac{1}{2}t^2 \geq 0$, $|t| - \frac{1}{2} \geq 0$ when $|t| \geq 1$)

hence $f(w) = \frac{1}{n}\sum_{i=1}^{n} \ell(x_i^T w - y_i) \geq 0$

$\Rightarrow 0$ is a lower bound on f

0 is not necessarily the minimum value of f

Indeed, $0 = \min_{w \in \mathbb{R}^d} f(w) \iff \exists w^* \in \mathbb{R}^d,\ f(w^*) = 0$

$\iff \exists w^* \in \mathbb{R}^d,\ \ell(x_i^T w^* - y_i) = 0 \quad \forall i = 1..n$

$\iff \exists w^* \in \mathbb{R}^d,\ x_i^T w^* - y_i = 0 \quad \forall i = 1..n$

NB: Interpolation regime

b) $f \in C^1$, $\nabla f(w) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\ell'(x_i^T w - y_i)}_{\in \mathbb{R}} \underbrace{x_i}_{\in \mathbb{R}^d}$

$$\ell'(t) = \begin{cases} 1 & \text{if } t > 1 \\ -1 & \text{if } t < -1 \\ t & \text{if } |t| \leq 1 \end{cases}$$

Write down the gradient descent iteration for (P) using a constant stepsize

Iteration k of GD: $w_{k+1} = w_k - \alpha \nabla f(w_k)$

$\alpha > 0$ (constant stepsize)

$= w_k - \frac{\alpha}{n}\sum_{i=1}^{n} \ell'(x_i^T w_k - y_i) x_i$

↳ If $w_k \in \operatorname{argmin}_{w \in \mathbb{R}^d} f(w)$, $\nabla f(w_k) = 0$ and thus $w_{k+1} = w_k$

↳ $\|\nabla f(v) - \nabla f(w)\| \leq L \|v - w\|$

c) f is $C_L^{1,1}$ ($C^1 + \nabla f$ is L-Lipschitz continuous) with $L = \frac{1}{n}\sum_{i=1}^{n} \|x_i\|^2$. How can we use L to choose the constant stepsize? Give two other strategies for choosing a stepsize in GD.

$\alpha \in \left(0, \frac{2}{L}\right)$ is a good stepsize

$\rightarrow$ $f \in C_L^{1,1} \Rightarrow \alpha = \frac{1}{L}$ is a good stepsize for GD

(leads to convergence and convergence rates)

$\rightarrow$ Other possibilities: $w_{k+1} = w_k - \alpha_k \nabla f(w_k)$ $\alpha_k > 0 \ \forall k$

- α_k decreasing sequence (ex: $\alpha_k = \frac{1}{\sqrt{k+1}}$, $\alpha_k = \frac{2}{k+2}, \dots$)

  ⊕ Guaranteed cv, can choose it independently of L/f

  ⊖ Stepsize can get small very quickly, causing numerical stagnation

- α_k adaptive / computed using current information at iteration k

  (Ex) Line search

  Pick α_k as the largest value in $\{1, \frac{1}{2}, \frac{1}{4}, \dots\}$ such that

  $$f(w_k - \alpha_k \nabla f(w_k)) < f(w_k) - c\,\alpha_k \|\nabla f(w_k)\|^2$$

  for some $c > 0$

  ⊕ Uses function information

  ⊖ Costs more than constant/decreasing stepsizes, need to evaluate f

d) $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$, where $f_i(w) = \ell(x_i^T w - y_i)$ depends only on the data point (x_i, y_i)

Write the stochastic gradient iteration with a generic stepsize.

SG (Stochastic Gradient) iteration

$$w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k)$$

$\alpha_k > 0$

i_k random index drawn from $\{1, \dots, n\}$

e) Suppose that a unit of cost corresponds to one access to an x_i

In terms of this unit of cost, what is:

- the cost of a GD iteration?
- the cost of a SG iteration?

GD: Compute $\nabla f(w_k) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w_k) = \frac{1}{n}\sum_{i=1}^{n} \ell'(x_i^T w - y_i)\, x_i$

$\Rightarrow$ Cost: n accesses to x_i's $\quad (n)$

SG: Compute $\nabla f_{i_k}(w_k) = \ell'(x_{i_k}^T w - y_{i_k})\, x_{i_k}$ $\Rightarrow$ 1 access to an x_i $\quad (x_{i_k})$

Cost 1

NB: An epoch: n accesses to a data point in a dataset of size n

1 iteration of GD costs 1 epoch, 1 iteration of SG costs $\frac{1}{n}$ epoch

⇒ with the same budget of epochs, can do n times more SG iterations than GD iterations

h) Batch stochastic gradient : Draw n_b elements of the dataset at every iteration

i) Write the batch SG iteration

$$w_{k+1} = w_k - \alpha_k \times \frac{1}{n_b} \sum_{i \in S_k} \nabla f_i(w_k)$$

where $\alpha_k > 0$

S_k is a set of n_b indices drawn randomly in $\{1, \dots, n\}$ with or without replacement

$n_b = 1 \Rightarrow$ SG

$n_b = n$ + drawing without replacement $\Rightarrow$ GD

ii) What is the statistical advantage of batch methods?

When $n_b > 1$, the method has reduced variance compared to vanilla SG ($n_b = 1$), in the sense that different runs will vary less when $n_b > 1$ than when $n_b = 1$.

iv) Suppose that increasing n_b from 1 to $\frac{n}{10}$ consistently improves the performance while increasing n_b from $\frac{n}{10}$ to n ——— deteriorates the performance. How can you explain this behavior?

→ Mini-batch regime (here $n_b > 1$, $n_b \leq \frac{n}{10}$): cheaper iterations than GD, better performance than SG because of smaller variance and more information at every iteration

(⚠ Predicting the best batch size in advance is not easy !)

→ Large-batch regime ($n_b \geq \frac{n}{10}$): iterations are more expensive than SG and the convergence slows down

ii) If n_b processors are available, what is the interest of batch methods with batch size n_b?

With n_b processors, can evaluate $\{\nabla f_i(w_k)\}_{i \in S_k}$ in parallel, and so a batch method with batch size n_b will run at a cost comparable to a vanilla SG method

NB: Explains why most ML implementations use batch sizes 64, 128, 256, ...

Motivation for the last part of the course

→ So far we have looked at problems of the form $\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(w)$

assuming access to (training) data

→ Goal was to minimize training loss / find a model that fits the data

→ ML / Data science has other concerns:

Not part of the optimization problem above

- Want model to generalize to unseen data / Want a model that does not overfit
- Want model that is as simple as possible (for interpretability)

$\Rightarrow$ these concerns can be incorporated in optimization using regularization

Regularized optimization problem

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \quad f(w) + \lambda \Omega(w)$$

- $f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$ "data-fitting term" (training loss)
- $\Omega : \mathbb{R}^d \to \mathbb{R}$ "regularization term", does not depend on data
- $\lambda \geq 0$: "regularization parameter" (weight of regularization in the optimization problem)

$\lambda = 0 \Rightarrow$ $\underset{w \in \mathbb{R}^d}{\text{minimize}} \; f(w)$ (what we have done so far)

this may happen numerically for finite λ

$\lambda \to \infty \Rightarrow$ the problem becomes equivalent to

$\underset{w \in \mathbb{R}^d}{\text{minimize}} \; \Omega(w)$: (here the term in f no longer matters)

Regularizer Ω: Used to penalize vectors that do not satisfy a certain property or structure

$\rightarrow$ Replaces constraints in many ML formulations

Two main regularization choices: ℓ_2 and ℓ_1 regularization

(1) ℓ_2 regularization (aka ridge regularization, Tychonov regularization, ...)

$$\Omega(w) = \frac{1}{2}\|w\|^2 = \frac{1}{2}\sum_{j=1}^{d} w_j^2$$

(quadratic function, $C^{1,1}$, strongly convex)

→ Used to prevent over-fitting

→ Reduces the sensitivity to problem data

→ If f is convex, then $f + \lambda\Omega$ is λ-strongly convex for $\lambda > 0$

Ex) Ridge regression (linear regression + ℓ_2 regularization)

$$\min_{w \in \mathbb{R}^2} \underbrace{\frac{1}{4}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2}_{f(w)} \qquad n = d = 2$$

$$X = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$\lambda = 0$ $\quad \operatorname{argmin}_{w \in \mathbb{R}^d} f(w) = \left\{ \begin{bmatrix} 1 \\ w_2 \end{bmatrix} \mid w_2 \in \mathbb{R}^d \right\}$

(convex problem with infinitely many solutions)

$\lambda > 0$ $\quad \operatorname{argmin}_{w \in \mathbb{R}^d} f(w) = \left\{ \begin{bmatrix} \frac{1}{1+2\lambda} \\ 0 \end{bmatrix} \right\}$

As $\lambda \to \infty$, $\begin{bmatrix} \frac{1}{1+2\lambda} \\ 0 \end{bmatrix} \to \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\lambda = 0$ f convex hence $w \in \operatorname{argmin} f \iff \nabla f(w) = 0$

$$\iff \frac{1}{2} X^T(Xw - y) = 0$$

$$X = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad y = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\iff X^T X w - X^T y = 0$$

$$X^T X = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$X^T y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\iff \begin{cases} w_1 - 1 = 0 \\ 0 = 0 \end{cases}$$

$\lambda > 0$ $w \in \operatorname{argmin} f + \lambda\Omega \iff \frac{1}{2} X^T(Xw - y) + \lambda w = 0$

$$\iff \begin{cases} w_1 - 1 + 2\lambda w_1 = 0 \\ \lambda w_2 = 0 \end{cases}$$

$$\iff \begin{cases} w_1 = \frac{1}{1+2\lambda} \\ w_2 = 0 \end{cases}$$

Consider the perturbed problem

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \quad \frac{1}{4}\|X_\varepsilon w - y\|^2 + \frac{\lambda}{2}\|w\|^2$$

$$X_\varepsilon = \begin{bmatrix} 1 & 0 \\ 0 & \varepsilon \end{bmatrix} \quad \text{with } 0 < \varepsilon \ll 1 \text{ small perturbation}$$

$$X_\varepsilon \begin{bmatrix} 1 \\ \varepsilon^{-1} \end{bmatrix}$$

$\lambda = 0$ $\operatorname{argmin}_{w \in \mathbb{R}^d} f(w) = \left\{ \begin{bmatrix} 1 \\ \varepsilon^{-1} \end{bmatrix} \right\}$ $\varepsilon^{-1} = \frac{1}{\varepsilon}$

With Perturbation, so from ∞ number of solution to 1 solution with small perturbation, ε solution has norm $O(\varepsilon^{-1})$

$\lambda > 0$

$$\text{argmin}_{w \in \mathbb{R}^d} f(w) = \left\{ \begin{bmatrix} \frac{1}{1+2\lambda} \\ \frac{1}{\varepsilon+2\lambda} \end{bmatrix} \right\}$$

For small ε, $\frac{1}{\varepsilon+2\lambda} \simeq O\left(\frac{1}{\lambda}\right)$

↳ Solving a problem with ℓ_2 regularization

$$\text{minimize}_{w \in \mathbb{R}^d} \underbrace{f(w)}_{C^1} + \underbrace{\frac{\lambda}{2}\|w\|^2}_{C^1} \quad \text{with } f \in C^1$$

$$\underbrace{\qquad\qquad}_{C^1}$$

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

GD:

$$w_{k+1} = w_k - \alpha_k \nabla\left(f + \frac{\lambda}{2}\|\cdot\|^2\right)(w_k)$$

Formulas for derivatives

$$= w_k - \alpha_k \left(\nabla f(w_k) + \lambda w_k\right)$$

$$= \underbrace{w_k - \alpha_k \nabla f(w_k)}_{\text{GD update for minimize } f(w)} \underbrace{- \lambda \alpha_k w_k}_{\text{Effect of regularization, called "weight decay" in ML packages}}$$

$$w_{k+1} = \underbrace{(1-\lambda\alpha_k)}_{\in (0,1) \text{ if } \lambda\alpha_k < 1} w_k - \alpha_k \nabla f(w_k)$$

NB: Weight decay also applies to Stochastic gradient methods

$$w_{k+1} = (1 - \lambda \alpha_k) w_k - \alpha_k \nabla f_{i_k}(w_k)$$

(2) ℓ_1 regularization (aka LASSO regularization)

$$\Omega(w) = \|w\|_1 = \sum_{j=1}^{d} |w_j|$$

convex function (not C^1!)

→ Used to promote vectors with (many) zero coordinates, aka sparse vectors

→ For linear models ($x \mapsto x^T w$), used for automatic feature selection

→ Used to compute "simple" models

Sparsity (having many zero coefficients) is a form of simplicity

Illustration: LASSO (Least Absolute Self-Shrinkage Operator) regression

$$\min_{w \in \mathbb{R}^d} \underbrace{\frac{1}{4}\|Xw - y\|^2 + \lambda \|w\|_1}_{f(w)}$$

$$X = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$\lambda = 0$ $\quad \operatorname{argmin} f(w) = \left\{ \begin{bmatrix} 1/2 \\ 1 \end{bmatrix} \right\}$

$\frac{1}{2} < \lambda < 1$ $\quad \operatorname{argmin} f(w) = \left\{ \begin{bmatrix} 0 \\ 1-\lambda \end{bmatrix} \right\}$

$1 \leq \lambda$ $\quad \operatorname{argmin} f(w) = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\}$

$0 < \lambda \leq 1/2$ $\quad \operatorname{argmin} f(w) = \left\{ \begin{bmatrix} 1/2 - \lambda \\ 1-\lambda \end{bmatrix} \right\}$

→ How do you solve ℓ_1 regularized problems?

ISTA (Iterative Soft-Thresholding Algorithm)

which a version of a more generic algorithm called proximal gradient

ISTA iteration for minimize $f(w) + \lambda \|w\|_1$ with $f(x)$

- Given w_k and $\alpha_k > 0$, compute w_{k+1} coordinate wise

$$\forall j = 1..d, \quad [w_{k+1}]_j = \begin{cases} [w_k - \alpha_k \nabla f(w_k)]_j - \lambda \alpha_k & \text{if } [w_k - \alpha_k \nabla f(w_k)]_j > \lambda \alpha_k \\ [w_k - \alpha_k \nabla f(w_k)]_j + \lambda \alpha_k & \text{if } [w_k - \alpha_k \nabla f(w_k)]_j < -\lambda \alpha_k \\ 0 & \text{otherwise} \end{cases}$$

ISTA looks at the gradient descent update for the un-regularized problem $w_k - \alpha_k \nabla f(w_k)$ and compares its coordinates with $\pm \lambda \alpha_k$

⇒ the iterates of ISTA have at least as many 0s as that

of GD

NB: For convex f, ISTA can be accelerated using momentum

$\Rightarrow$ FISTA

(3) Proximal gradient methods

$$(P) \quad \underset{w \in \mathbb{R}^d}{\text{minimize}} \; f(w) + \lambda \Omega(w)$$

f C^1, data-fitting term

$\lambda \geq 0$

Ω : regularizer, convex (but not necessarily C^1)

Proximal gradient: Solve a sequence of auxiliary optimization problems, called subproblems, based on the gradient of f

Iteration k

$$w_{k+1} \in \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ f(w_k) + \nabla f(w_k)^T (w - w_k) + \frac{1}{2\alpha_k} \|w - w_k\|^2 + \lambda \Omega(w) \right\}$$

where $\alpha_k > 0$

- $f(w_k) + \nabla f(w_k)^T (w - w_k)$: Approximation of $f(w)$ around w_k
- $\frac{1}{2\alpha_k} \|w - w_k\|^2$: Proximal term: forces w_{k+1} to stay close to w_k
- $\lambda \Omega(w)$: Keep regularization as in (P)

$\rightarrow$ If the subproblems can be solved easily, then proximal gradient is a good method for (P) (and it has convergence rate guarantees like GD)

- When $\Omega(w) = \|w\|_1$, proximal gradient is ISTA
- When $\Omega(w) = 0$ (no regularization), proximal gradient is gradient descent

$$\operatorname*{argmin}_w \left\{ f(w_k) + \nabla f(w_k)^T (w - w_k) + \frac{1}{2\alpha_k} \|w - w_k\|^2 \right\} = \left\{ w_k - \alpha_k \nabla f(w_k) \right\}$$

- When $\Omega(w) = \frac{1}{2}\|w\|^2$ (l_2 regularization), the proximal gradient iteration has a closed-form expression:

$$w_{k+1} = \frac{1}{1+\lambda\alpha_k} w_k - \frac{\alpha_k}{1+\lambda\alpha_k} \nabla f(w_k)$$

$\frac{1}{1+\lambda\alpha_k}$: weight decay

$\frac{\alpha_k}{1+\lambda\alpha_k}$: "gradient decay" ($\neq$ from GD applied to the problem)

Remark: Many regularization functions aim at producing sparse solutions in a structured way by combining l_1 and l_2 regularization

Ex) Group LASSO

$$w = \begin{bmatrix} w_{g_1} \\ \vdots \\ w_{g_m} \end{bmatrix} \begin{matrix} \updownarrow d_1 \\ \\ \updownarrow d_m \end{matrix}$$

$d_1 + d_2 + \dots + d_m = d$

Groups of variables that should have the same importance in the model

$$\Omega(w) = \sum_{j=1}^{m} \|w_{g_j}\| = \left\| \begin{bmatrix} \|w_{g_1}\| \\ \vdots \\ \|w_{g_m}\| \end{bmatrix} \right\|_1$$