Unconstrained optimization

Pierre Hellier

Formulation and optimality

General formulation

Solve for $x^* = arg \min_{\mathbf{x} \in C} L(x)$

- $L : \mathbb{R}^n \rightarrow \mathbb{R}$ is the loss function
- $\mathbf{x} \in \mathbb{R}^n$ is a vector of n variables
- C is the set of admissible solutions
- Objective: find vector of minimal value $\mathbf{x}^*$, i.e. $\forall \mathbf{x} \in C, L(\mathbf{x}^*) < L(\mathbf{x})$

In the following, the loss, or function, will be denoted L or F
For convenience, without loss of generality, $C = \mathbb{R}^n$

Recall: necessary condition on gradient

Gradient is a necessary condition for optimality

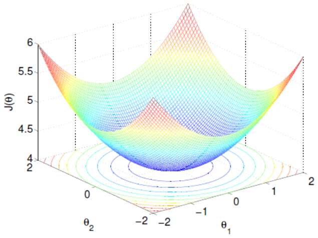
if x^* is a local minimum of function f (on some ball around x^*), then $\nabla_x F(x^*) = 0$

where

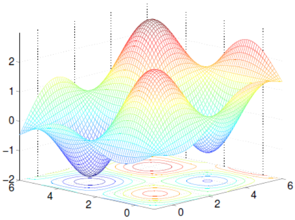
$$\forall h \in \mathbb{R}^n, t \in \mathbb{R}, \nabla_x F(x)^T h = \lim_{t \to 0} \frac{F(x + th) - F(x)}{t}$$

Examples local/global minima

$$F(x) = \frac{1}{2}x^T P x + q^T x + r$$
with P positive definite matrix



$$F(x) = \cos(x_1 - x_2) + \sin(x_1 + x_2) + \frac{x_1}{4}$$



Exercice of suboptimality

Problem

- Let us define

$$F(x) = x_1^4 + x_2^4 - 4x_1x_2$$

- Compute gradient of F
- Compute stationary points
- How many local and global minima?
- $3D$ plot of the function using matplotlib (optional)

Solution

- Gradient
$$\nabla F(x) = \begin{pmatrix} 4x_1^3 - 4x_2 \\ -4x_1 + 4x_2^3 \end{pmatrix}$$
- Stationary points $\nabla F(x) = 0$
- Three solutions $\theta_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\theta_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $\theta_3 = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$
- Only two minima (compute function values)

Assess solutions

Problem

How to express another optimality condition?
With the second derivatives, or hessian matrix

Second condition of optimality

Problem

if F is twice differentiable, there exists a unique symmetric matrix $H(x) \in \mathbb{R}^{n\times n}$ such that

$$F(x+h) = F(x) + \nabla F(x)^T h + h^T H(x) h + ||h||^2 o(h)$$

H is the second derivative matrix

$$H(x) = \begin{pmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \frac{\partial^2 F}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 F}{\partial^2 x_n} \end{pmatrix}$$

Second condition of optimality

Problem

Let F be a twice-differentiable function. If x^* is a minimum, then $\nabla F(x^*) = 0$ and $H(x^*)$ is a positive definite matrix

Remarks:

- H is positive definite if and only if all eigenvalues are positive
- H is negative definite if and only if all eigenvalues are negative
- For $n = 1$, that means that the derivative is zero and the second derivative is positive
- If for a stationary point, H is negative definite, that means that the point is a local maximum

Exercice

Problem

Let F be a quadratic loss function, $F(x) = \frac{1}{2}x^T Px + q^T x + r$.
Compute gradient and hessian

Solution

- $\nabla F(x) = Px + q$
- $H(x) = P$

Exercice

Problem

Let

$$F(x) = x_1^4 + x_2^4 - 4x_1x_2$$

Compute hessian matrix and assess the optimality of stationary points

Solution

- Three solutions $\theta_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\theta_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $\theta_3 = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$
- $H(x) = \begin{pmatrix} 12x_1^2 & -4 \\ -4 & 12x_2^2 \end{pmatrix}$

	θ_1	θ_2	θ_3
Hessian	$\begin{pmatrix} 0 & -4 \\ -4 & 0 \end{pmatrix}$	$\begin{pmatrix} 12 & -4 \\ -4 & 12 \end{pmatrix}$	$\begin{pmatrix} 0 & -4 \\ -4 & 0 \end{pmatrix}$
Eigenvalues	4,-4	8,16	8,16
Type	saddle point	minimum	minimum

Convexity

Problem

Remember: if F is convex, a local minimum is also a global minimum

Least square problems

Ordinary least square

Classical regression problem, $F(x) = \|Ax - y\|^2$, where

- $x \in \mathbb{R}^n$ (n is the number of parameters)
- $y \in \mathbb{R}^m$ (m is the number of observations)
- $A \in \mathbb{R}^{m \times n}$ is the data observation matrix

An old problem (Gauss and Legendre to predict planets motion, 1805). Example problem:

- We collect biological data on patients (blood pressure, blood analysis, imaging features, etc) as matrix A, as well as an outcome or predicted data y
- x stands for the linear mixtures coefficients of the observations to make the prediction

Two regimes: $m < n$ (under-determined regime, catastrophic) where we have too few observations, and $m \geq n$ (over-determined regime, usual).

Linear inverse problem

Relationship with Linear inverse problems

- Assume that we collect observations y, e.g., blurred images
- The data formation model is modeled through matrix A, e.g., blur matrix
- The inverse problem amounts to computing the *true* image given the observed *blurred* image

Sparse coding is a somehow different problem: suppose that matrix A is an overcomplete dictionary (for instance, wavelets basis functions), one aims at recovering the *sparse* combination of atoms that explain the observed image

Least square exercise

Problem

Exercise: what is the optimal solution of $arg\min_x = \|Ax - y\|^2$?
A is generally not invertible since $n \neq m$

Hints:

1. Express the norm as a scalar product
2. Expand and compute gradient
3. For a scalar quantity a, remember that $a^T = a$

Least square solution

- $f(x) = \|Ax - y\|^2 = (Ax - y)^T(Ax - y) = x^T A^T Ax - 2x^T A^T y + y^T y$
- $\nabla F(x) = 2A^T Ax - 2A^T y$
- $\nabla F(x) = 0 \Leftrightarrow A^T Ax = A^T y$

$A^T A$ invertible $\Leftrightarrow$ ker(A)=$\{0\}$
$(A^T Ax = 0 \Rightarrow \|Ax\|^2 = \langle A^T Ax; x\rangle = 0 \Rightarrow Ax = 0)$

Least square solution

- For the under-determined regime, $A^T A$ is not invertible. See ridge regularization.
- For the over-determined regime, data points are collected with noise, meaning $A^T A$ is invertible with probability 1
- In that case, $x^* = (A^T A)^{-1} A^T y$
- $(A^T A)^{-1} A^T$ is called the Moore-Penrose pseudo-inverse of A
- E. H. Moore in 1920, Arne Bjerhammar in 1951, and Roger Penrose (Physics Nobel prize 2020) in 1955

Compute sparse models

If we gather unnecessary data, inference will cost, and we have too large models. Solutions to limit the number of factors:

- Information criteria AIC and BIC
- Regularized least square

Information criteria: coefficient of determination

Coefficient of determination R^2 or r^2

- Geneticist Sewall Wright, 1921: measure the quality of linear regression
- Formulation:
  - Compute mean of y as $\bar{y}$
  - Compute SST as $SST = \sum(y - \bar{y})^2$. Roughly the variance of the true values
  - Compute SSR as the residuals $mean(\|Ax - y\|^2)$
- Determination coefficient $R^2 = 1 - SSR/SST$
- Perfect fit: $R^2 = 1$
- Intuition $R^2 = 1 - FUV$ (fraction of unexplained variance)

Coefficient of determination and model size

Coefficient of determination is inefficient to compare models

- It will always increase when you add measurements
- Even in case of a infinitely correlated measure, it increases
- It will favor complex models

Akaike information criterion (AIC) and BIC

Based on information theory to measure the loss between perfect model and approximate model

- $AIC = -2\log L + 2K$ or $BIC = -2\log L + 2K\log m$
- where L is the likelihood. The negative LL is the cost function
- K is the number of parameters in the model
- Smaller AIC leads to a better model

Sequential strategies to find the best model:

- Backward. Start with full model and progressively remove factor that leads to decreasing the most AIC
- Ascending or forward.
- Issue: cost, control over complexity..

Regularized least square

Regression problem with regularization,
$F(x) = \|Ax - y\|^2 + \lambda R(x)$, where

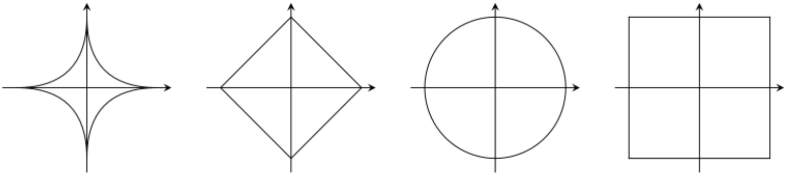
- $x \in \mathbb{R}^n$ (n is the number of parameters)
- $y \in \mathbb{R}^m$ (m is the number of observations)
- $A \in \mathbb{R}^{m \times n}$ is the data observation matrix
- R is a regularization term, enforcing prior on x
- λ is a weighting term

Regularization techniques

- Ridge regularization $R(x) = \|x\|_2^2$
- Lasso regularization $R(x) = \|x\|_1$

p-norms

The penalization can be generally expressed with p-norm:
$\|x\|_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$



$p = \frac{1}{2}$ $p = 1$ $p = 2$ $p = \infty$

Regularized least square exercise

Problem

Exercise: what is the optimal solution for regularized least square with L_2 penalization (ridge)

Hints:

1. Express the norm as a scalar product
2. Expand and compute gradient
3. For a scalar quantity a, remember that $a^T = a$

Regularized least square solution

- $f(x) = \|Ax - y\|^2 + \lambda\|x\|^2 = (Ax - y)^T(Ax - y) + \lambda x^T x = x^T A^T Ax - 2x^T A^T y + y^T y + \lambda x^T x$
- $\nabla F(x) = 2A^T Ax - 2A^T y + 2\lambda x$
- $\nabla F(x) = 0 \Leftrightarrow (A^T A + \lambda I)x = A^T y$

Remark: the λI term helps to invert the matrix. Can be used for under-determined regime.

Regularized least square with Lasso

- Lasso regularization $= L_1$ norm for R
- Main application: project signal on ovecomplete basis A with minimal number of coefficients:
  - Sin-waves for temporal signal
  - DCT basis for images
- Non-smooth optimization since R s not differentiable
- To be studied in a following section...

Practical session

Linear regression notebook

Descent algorithms

Main principle of descent algorithm

- Direction of descent. the vector $d \in \mathbb{R}^n$ is called a descent direction in x if $\exists \alpha > 0 | L(x + \alpha d) < L(x)$
- Algorithms:
  - Start from initial solution x_0 with $k = 0$
  - Compute direction of descent d_k
  - Compute optimal step size α_k with line search such that $L(x_k + \alpha_k d_k)$ decreases *enough*
  - Update $x_{k+1} = x_k + \alpha_k d_k$
  - Repeat until convergence to stationary point $\|\nabla L(x_k)\|^2 \leq \epsilon$
- Methods differ in the choice of d: gradient, stochastic gradient, newton, quasi newton
- Deep learning terminology for α: learning rate.

Gradient descent

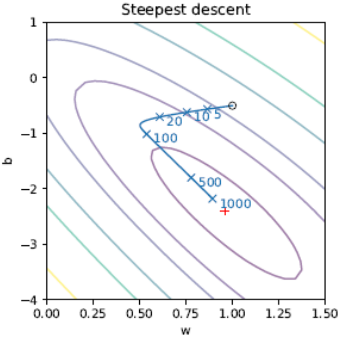
- If L is a differentiable function, then direction $d = -\nabla L \in \mathbb{R}^n$ is a descent direction
- Proof: $L(x + td) = L(x) + t\nabla L^T d + o(td)$. By choosing $d = -\nabla L$, one obtains

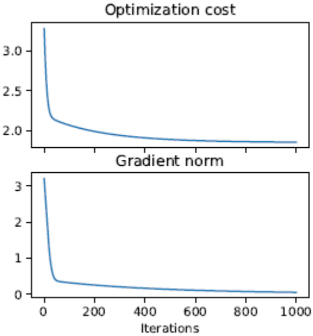
$$L(x + td) - L(x) = -t\|\nabla L(x)\|^2 + o(td) \leq 0$$

  for small enough t
- Complexity of the update step $x_{k+1} = x_k - \alpha_k \nabla L(x_k)$ is $\mathcal{O}(n)$ flops

Illustration of descent algorithm

- Gradient descent with fixed step size (see after adaptive step size)
- Slow convergence, not reached after 1000 iterations





Recall: Lispschitz functions

- Rudolf Lipschitz, german mathematician (1832-1903)
- An real application f from vector space is k-Lispschitz if $\forall x, \forall y, |f(x) - f(y)| \leq k||x - y||$
- property of regularity that is much stronger than continuity
- f is said contractive if $k < 1$

Gradient descent as majoration algorithm

- Definition of Gradient Lispschitz function:

$$\forall d, \forall x, \|\nabla L(x+d) - \nabla L(x)\| \leq K\|d\|$$

- The constant K is called the Lispschitz constant of ∇L
- Descent lemma:

$$\forall d, \forall x, L(x+d) \leq L(x) + \nabla L(x)^T d + \frac{K}{2}\|d\|^2$$

- At iteration k, minimizing the majorant w.r.t. d around x leads to $d^* = -\frac{1}{K}\nabla L(x)$
- Corresponds exactly to gradient descent with step $\alpha = \frac{1}{K}$

Sufficient conditions for convergence

Known as the **Wolfe conditions**:

- First condition (Armijo rule):

$$L(x_k + \alpha d_k) \leq L(x_k) + c_1 \alpha \nabla L(x_k)^T d_k$$

  In plain words, α should decrease enough the function.

- Second condition (curvature rule):

$$\nabla L(x_k + \alpha d_k)^T d_k \geq c_2 \nabla L(x_k)^T d_k$$

- With $0 < c_1 < c_2 < 1$, in practice c_1 is very small $\approx 1e^{-4}$ and $c_2 \approx 0.9$

Additional resources for convergence

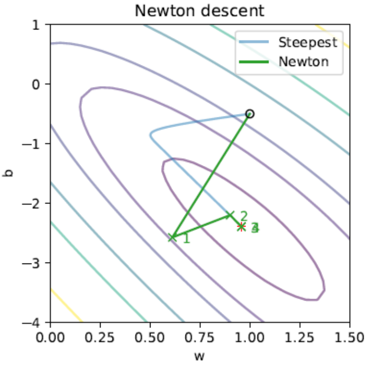
- Bertsekas, 1999, Nonlinear programming https://mcube.lab.nycu.edu.tw/~cfung/docs/books/bertsekas1999nonlinear_programming.pdf.
- Nocedal and Wright, 2006. Numerical optimization (chapter 3) https://www.math.uci.edu/~qnie/Publications/NumericalOptimization.pdf

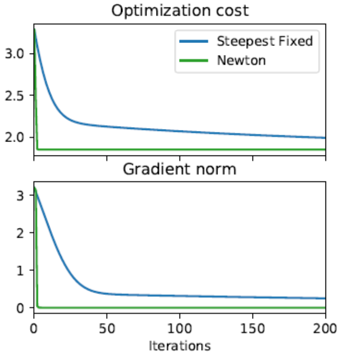
Newton algorithm

- If L is a twice differentiable function with Hessian matrix H
- $$L(x+d) = L(x) + \nabla L(x)^T d + \frac{1}{2} d^T H(x) d$$
- The optimal direction is obtained as
$$\nabla L(x+d) = 0 \Rightarrow d = -H(x)^{-1} \nabla L(x)$$
- If L is quadratic, convergence in one iteration
- Complexity of the update step $x_{k+1} = x_k - \alpha_k H(x_k)^{-1} \nabla L(x_k)$ is $\mathcal{O}(n^3)$ flops
- Be careful! H is not guaranteed to be PSD and d_k could not be a descent direction...
- Levenberg-Marquardt modification, use $\tilde{H} = H + \lambda I$: interpolate between Newton ($\lambda = 0$) and GD (large λ)

Illustration of Newton algorithm

- Fixed step size
- Fast convergence, 4 iterations



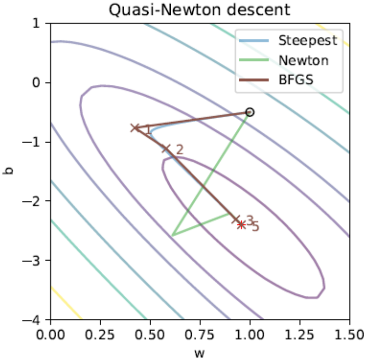


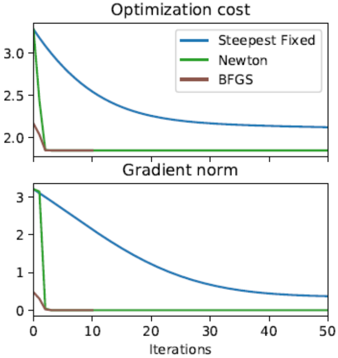
Quasi newton algorithm

- Intuition: replace Hessian matrix with PSD approximation of the inverse Hessian B
- Complexity of the update step $x_{k+1} = x_k - \alpha_k B(x_k)^{-1} \nabla L(x_k)$ is $\mathcal{O}(n^2)$ flops
- Commonly used algorithm is BFGS (Broyden–Fletcher–Goldfarb–Shanno):
  - Initialize B to identity: $B_0 = I$
  - $d_k = -\alpha_k B(x_k)^{-1} \nabla L(x_k)$
  - Update $x_{k+1} = x_k + d_k$
  - Update gradient difference $y_k = \nabla L(x_{k+1}) - \nabla L(x_k)$ and difference $s_k = x_{k+1} x_k$
  - Update B with $B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k^T}{s_k^T B_k s_k}$

Illustration of BFGS algorithm

- Fixed step size
- Fast convergence, 11 iterations
- First step is GD





Summary of descent methods

Method	Descent direction	complexity	convergence
Gradient	$-\nabla L$	$\mathcal{O}(n)$	linear
Quasi-newton	$-B^{-1}\nabla L$	$\mathcal{O}(n^2)$	superlinear
Newton	$-H^{-1}\nabla L$	$\mathcal{O}(^3)$	quadratic

Practical session

Gradient descent notebook

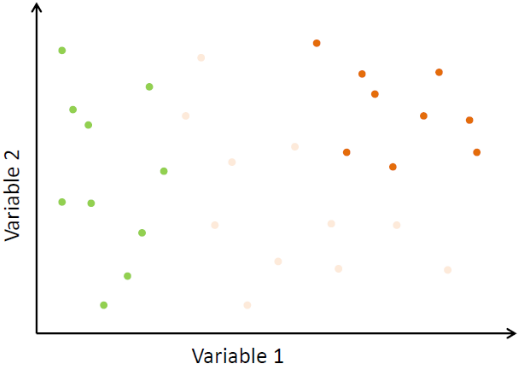
Exercise of gradient descent on logistic regression

Problem

- Binary classification problem
- Logistic regression formulation
- Exercise: compute gradient step

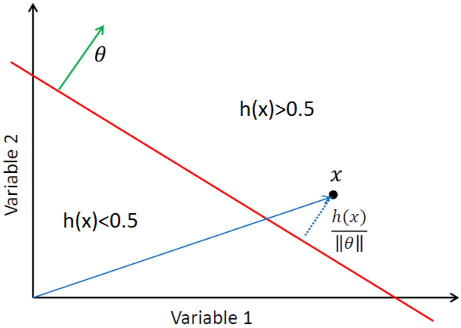
Binary classification

- $y \in \{0, 1\}$
- Find best linear discriminant



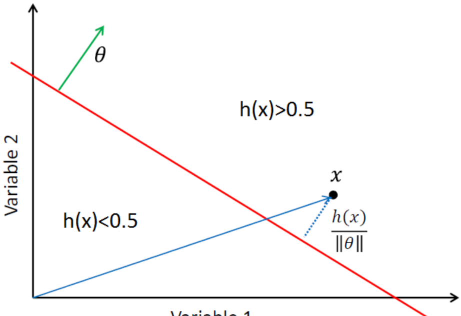
Linear discriminant: intuition

- A discriminant is an hyperplane
- The hyperplane is defined by its orthogonal vector
- The position of a point is defined by the sign of the scalar product with hyperplane vector
- x_i belongs to positive class if $\theta^t x > 0$



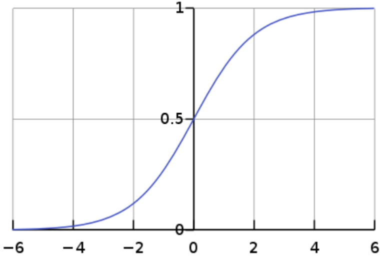
Linear discriminant: technical detail

- $\text{sign}(\theta^t x)$ is not a probability
- Discriminant function $h_\theta(x) = \sigma(\theta^t x)$
- σ is the logistic function
- Hyperplane that separates the space into two classes, according to $sign(h_\theta(x) - 0.5)$
- Probabilistic interpretation $P(y = 1|x) = h_\theta(x) = \sigma(\theta^t x)$



The logistic function σ

- $\sigma(x) = \frac{1}{1+e^{-x}}$
- Symmetric property $\sigma(-x) = 1 - \sigma(x)$
- $\frac{d\sigma}{dx}(x) = \frac{d\sigma}{dx}(-x) = \sigma(x)(1-\sigma(x))$
- The logistic function shrinks all data in $[0, 1]$



Linear discriminant and logistic cost

- $P(y_i = 1|x_i) = \sigma(\theta^t x_i) = h_\theta(x_i) = p_i$
- Conditional likelihood function $\mathcal{L}(\theta, X, Y) = p(Y|X)$

$$\mathcal{L} = \prod_i p(y_i|x_i) = \prod_i p_i^{y_i}(1 - p_i)^{1-y_i}$$

- Conditional minus log-likelihood
  $\mathcal{J}(\theta; X, Y) = -\sum_i (y_i \ln(h_\theta(x_i)) + (1 - y_i) \ln(1 - h_\theta(x_i)))$

Exercice

Problem

Compute gradient descent step of

$$\mathcal{J}(\theta; X, Y) = -\sum_i \left(y_i \ln(\sigma(\theta^t x_i)) + (1 - y_i) \ln(1 - \sigma(\theta^t x_i))\right)$$

knowing that

- $\ln'(x) = \frac{1}{x}$
- $(g \circ f)'(x) = f'(x) g'(f(x))$
- $\sigma'(x) = \sigma(x)(1 - \sigma(x))$

Gradient descent

Drop index of sample i and compute partial derivative on index j

$$
\begin{aligned}
\frac{\partial \mathcal{J}(\theta)}{\partial \theta_j} &= \left(\frac{y}{h_\theta(x)} - \frac{1-y}{1-h_\theta(x)} \right) \frac{\partial h_\theta(x)}{\partial \theta_j} \\
&= \left(\frac{y}{h_\theta(x)} - \frac{1-y}{1-h_\theta(x)} \right) \frac{\partial \sigma(\theta^t x)}{\partial \theta_j} \\
&= \left(\frac{y}{\sigma(\theta^t x)} - \frac{1-y}{1-\sigma(\theta^t x)} \right) \sigma(\theta^t x)(1-\sigma(\theta^t x)) \frac{\partial \theta^t x}{\partial \theta_j} \\
&= \left(y(1-\sigma(\theta^t x)) - (1-y)\sigma(\theta^t x) \right) x_j \\
&= (y - h_\theta(x))\, x_j
\end{aligned} \tag{1}
$$

Linear discriminant and logistic cost

- Cost
  $\mathcal{J}(\theta; X, Y) = -\sum_i \left(y_i \ln(h_\theta(x_i)) + (1 - y_i) \ln(1 - h_\theta(x_i))\right)$
- Gradient descent $\nabla_\theta \mathcal{J}(\theta) = \sum_i (h_\theta(x_i) - y_i) x_i$
- Once θ is learned, each new sample is tested: if $\sigma(\theta^t x) > 0.5$, then x belongs to the "1" class

Practical session

Logistic regression notebook

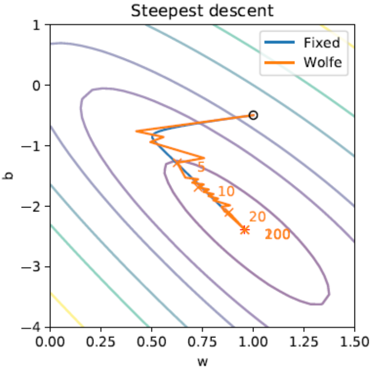
Line search methods

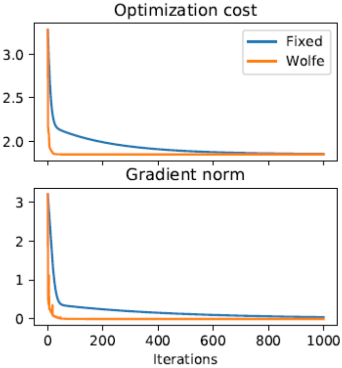
For any direction (GD, newton, quasi newton), a crucial choice is the step size α

- For gradient lipschitz functions, $\alpha = 1/K$ ensures convergence, but slowly
- Line search: at each iteration k, tune α so that Wolfe conditions are met
- Most popular technique: backtracking line search:
  - Init α and choose $0 < \tau < 1$
  - repeat $\alpha \leftarrow \alpha\tau$ until $L(x + \alpha d) < L(x) + \alpha c_1 d^T \nabla L(x)$

Impact of line search

- Large impact
- Increases the complexity due to function evaluation





Acceleration of gradient methods

Practical issues: the number of iterations and the complexity of each iteration. Solution proposed:

- Heavy-ball methods (1964)
- Nesterov acceleration (1983 and follow-up papers)
- Conjugate gradient (1952)

Heavy-ball and conjugate gradient methods

Introduce momentum in gradient methods

- Heavy-ball:

$$x_{k+1} = x_k - \alpha \nabla F(x_k) + \beta(x_k - x_{k-1})$$

- Conjugate gradient

$$x_{k+1} = x_k + \alpha p_k$$

with

$$p_k = -\nabla F(x_k) + \beta_k p_{k-1}$$

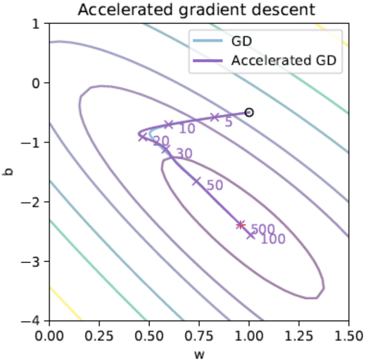
Nesterov accelerated gradient descent

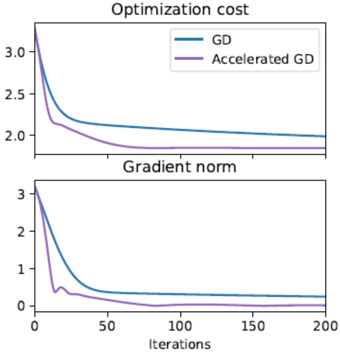
Nesterov (1983) proposes momentum for gradient descent:

- Init x_0, $y_0 = x_0$ and $\alpha < 1/K$ (K is a lispschitz constant of gradient)
- At iteration k:
  - $y_k = x_k + \frac{k-1}{k+2}(x_k - x_{k-1})$
  - $x_{k+1} = y_k - \alpha \nabla L(y_k)$
- Intuition: it anticipates the position of the next point for gradient computation

Impact of Nesterov accelerated

- Here no line search
- Acceleration speedup is important
- The effect of momentum can be seen on the trajectory





Stochastic gradient descent

Reminder, loss for supervised leaning:

$$\hat{\theta} = arg \min_{\theta \in \mathbb{R}^n} \mathcal{J}(\theta; X, Y) = arg \min_{\theta \in \mathbb{R}^n} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(f_\theta(X_i), Y_i)$$

where

- n is the dimension (number of parameters) and m is the number of training points
- In practice, both n and m are large and computation of $\nabla \mathcal{J}$ costs $\mathcal{O}(nm)$
- Dataset might not fit in memory

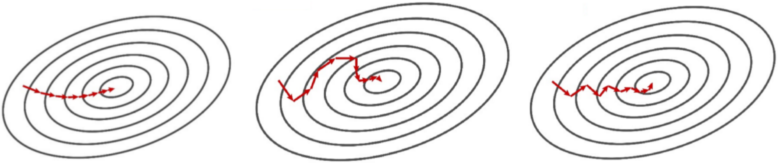
The solution is stochastic and batch gradient descent

Stochastic gradient descent

Algorithm:

- At each iteration k:
  - Pick one sample (X_j, Y_j) (stochastic GD) or a subset of m_B samples $(X_j, Y_j), j \in B$(batch GD)
  - Compute the individual gradient $-\nabla \mathcal{J}(\theta; X_j, Y_j)$ (stochastic GD)or average the gradients over subset B
  - Compute line search and perform descent step
- Variant: keep gradient history in memory and average (viscosity) temporally. Discussed in more details afterwards.

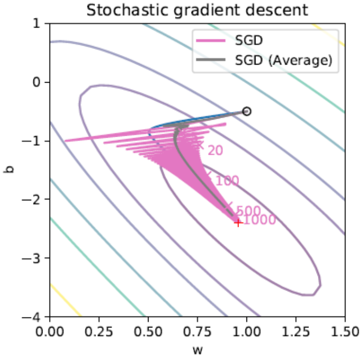
Comparison of GD algorithm



Full gradient SGD (online) SGD (mini-batch)

Illustration of stochastic GD algorithm

- The gradient is a (rough) approximation of the true gradient
- Hence, the observed oscillations
- Iteration complexity is $\mathcal{O}(n)$ for stochastic GD and $\mathcal{O}(nm_B)$ for batch GD



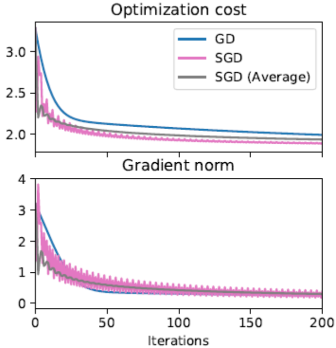
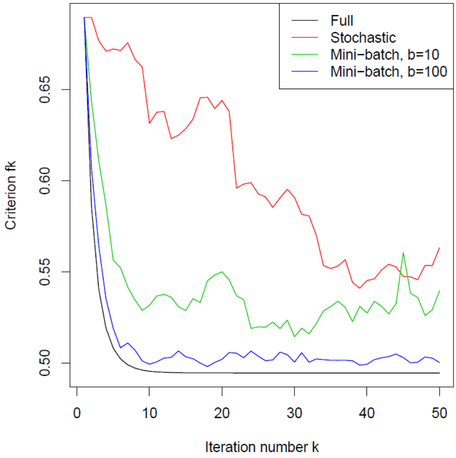


Illustration of batch GD algorithm

- Influence of batch size on the bias towards the true gradient



Non-smooth optimization

Regularized machine learning problem

$$\hat{\theta} = arg \min_{\theta} \mathcal{J}(\theta; X, Y) = arg \min_{\theta} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(f_\theta(X_i), Y_i) + \lambda \mathcal{R}(\theta)$$

where

- $\mathcal{R}$ penalizes large values of θ (ridge) or enforces prior/sparse knowledge (lasso)
- λ is a weighting parameter to be adjusted

If $\mathcal{R}$ is not differentiable (L_1 norm, or lasso regularization), then GD is not applicable

Examples

- Lasso regression and lasso . We collect exhaustive data without prior knownledge, but one aims at recovering the minimal features that explain the observations (either regression or classification)
- Sparse coding: in some transform domains a signal ought to have only a few non-zero coefficients (complex sine-waves in the Fourier domain, natural images in the DCT and Wavelet coefficient domains)
- Low-rank matrix factorization for recommendation systems, aka the Netflix challenge. Fill in data matrix with sparsity constraints.

Fixed point iteration

- For a function $g : \mathbb{R}^n \to \mathbb{R}^n$
- The iterative schema $x_{k+1} = g(x_k)$ converges to a fixed point if and only if g is contractive
- Contractive mapping: g is L-Lipschitz with $L < 1$
- $\|g(x) - g(y)\| \leq L\|x - y\|$

Proximal operator

For a scalar function $g : \mathbb{R}^n \to \mathbb{R}$ and a scalar $\lambda > 0$, the proximity or proximal operator is defined as:

$$\begin{aligned} prox_{\lambda g} :& \mathbb{R}^n \to \mathbb{R}^n \\ & x \rightarrowtail prox_{g,\lambda}(x) = arg \min_y (g(y) + \frac{1}{2\lambda}\|x - y\|^2) \end{aligned}$$

- Returns a vector that minimizes g but close to x
- Building brick of proximal splitting, or forward backward splitting
- x^* minimizes g if and only if it is a fixed point of the proximal operator $x^* = prox_{\lambda g}(x^*)$

Usual proximal operator

- $g(x) = 0,\ prox_{\lambda g}(x) = x$ (identity)
- $g(x) = \|x\|^2,\ prox_{\lambda g}(x) = \frac{1}{1+\lambda}x$ (scaling)
- $g(x) = \|x\|_1,\ prox_{\lambda g}(x) = sign(x) max(0, |x| - \lambda)$ (soft shrinkage)
- $g(x) = \mathcal{X}_C,\ prox_{\lambda g}(x) = arg\min_u \|x - u\|^2$ (orthogonal projection)

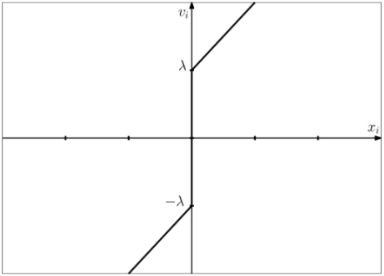
Proximal operator for L_1 norm

Let us derive the proximal operator for the lasso regularization

- $prox_{\lambda f}(v) = arg \min_x (f(x) + \frac{1}{2\lambda}\|x - v\|^2)$
- $\phi(x, v) = f(x) + \frac{1}{2\lambda}\|x - v\|^2 = \sum_i |x_i| + \frac{1}{2\lambda}\sum_i (x_i - v_i)^2$
- $\frac{\partial \phi}{\partial x_i} = sign(x_i) + \frac{1}{\lambda}(x_i - v_i)$
- For argmin x, $\frac{\partial \phi}{\partial x_i} = 0$, leading to $v_i = \lambda sign(x_i) + x_i$

Proximal operator for L_1 norm

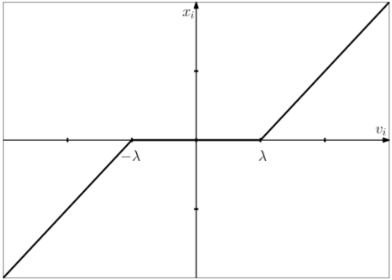
$$v_i = \begin{cases} x_i - \lambda & \text{if } x_i < 0 \\ x_i + \lambda & \text{if } x_i > 0 \end{cases}$$



Proximal operator for L_1 norm

If we reverse and express x_i in terms of v_i

$$x_i = \begin{cases} v_i + \lambda & \text{if } v_i < -\lambda \\ 0 & \text{if } |v_i| \leq \lambda \\ v_i - \lambda & \text{if } v_i > \lambda \end{cases}$$



Soft-threshold operator
$S_\lambda(u) = sign(u)max(|u| - \lambda, 0) = sign(u)Relu(|u| - \lambda)$

Forward backward splitting (FBS)

2009 paper, `https://web.stanford.edu/~jduchi/projects/DuchiSi09b.pdf`:

- Minimize $f(x) + g(x)$ where f is smooth and convex, while g is not
- One gradient step of f and one proximal step of g
- At each iteration k, $x_{k+1} = prox_{\alpha g}(x_k - \alpha \nabla f(x_k))$
- Efficient when the proximal operator can be computed in closed form Solve lasso problem:
  - Ista (Iterative soft thresholding) algorithm to solve $arg \min_x \frac{1}{2}\|Ax - y\|^2 + \lambda \|x\|_1$
  - $x_{k+1} = S_{\frac{\lambda}{L}}(x_k - \frac{1}{L}A^T(Ax - y))$
  - where L is the square of the largest eigenvalue of A, i.e., the lipschitz constant of $A^T A$

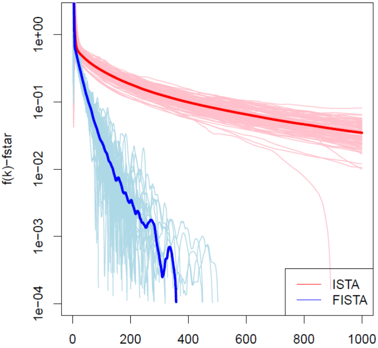
Fista: fast ista

Main idea: use Nesterov acceleration with ISTA.
Proposed in Beck, A., and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM journal on imaging sciences, 2(1), 183-202.
At each iteration k,

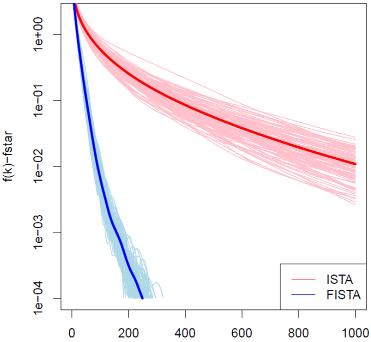
- $x_{k+1} = prox_{\alpha g}(y_k - \alpha \nabla f(y_k))$
- $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
- $y_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1})$

Fista: fast ista

Can be used for lass regression and lasso logistic regression



Lasso regression



Lasso logistic regression

Practical session

Ista notebook