Optimization for Machine Learning

Part II : Introduction to SGD

Lionel Tondji

African Master's in Machine Intelligence

July 16, 2024

Structure of Optimization Problems Arising in Training Supervised machine Learning Models

Optimization Problems Arising in Machine Learning

In this course, we are interested in the optimization problem

$$\min_{w \in \mathbb{R}^d} f(w) + R(w) \tag{1}$$

Optimization Problems Arising in Machine Learning

In this course, we are interested in the optimization problem

$$\min_{w \in \mathbb{R}^d} f(w) + R(w) \tag{1}$$

Typical structure of f:

- Infinite sum

$$f(w) = E_{\zeta \sim \mathcal{D}}[f_\zeta(w)] \tag{2}$$

Optimization Problems Arising in Machine Learning

In this course, we are interested in the optimization problem

$$\min_{w \in \mathbb{R}^d} f(w) + R(w) \tag{1}$$

Typical structure of f:

- Infinite sum

$$f(w) = E_{\zeta \sim \mathcal{D}}[f_\zeta(w)] \tag{2}$$

- Finite sum :

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w) \tag{3}$$

Optimization Problems Arising in Machine Learning

In this course, we are interested in the optimization problem

$$\min_{w \in \mathbb{R}^d} f(w) + R(w) \tag{1}$$

Typical structure of f:

- Infinite sum

$$f(w) = E_{\zeta \sim \mathcal{D}}[f_\zeta(w)] \tag{2}$$

- Finite sum :

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w) \tag{3}$$

- Finite sum of Finite Sums :

$$f_i(w) = \frac{1}{m} \sum_{j=1}^{m} f_{ij}(w) \tag{4}$$

Optimization Problems Arising in Machine Learning

These problems are of keys importance in supervised learning theory and pratice.

Common feature: It is prohibitively expensive to compute the gradient of f, while an unbiased estimator of the gradient can be computed efficiently/cheaply.

Stochastic Optimization and Machine Learning

In the stochastic optimization problem

$$f(w) = E_{\zeta \sim \mathcal{D}}[f_\zeta(w)]$$

- w represents a machine learning model described by d parameters/features (e.g logistic regression or a deep neural network).

Stochastic Optimization and Machine Learning

In the stochastic optimization problem

$$f(w) = E_{\zeta \sim \mathcal{D}}[f_\zeta(w)]$$

- w represents a machine learning model described by d parameters/features (e.g logistic regression or a deep neural network).
- $\mathcal{D}$ is an unknown distribution of labelled examples,

Stochastic Optimization and Machine Learning

In the stochastic optimization problem

$$f(w) = E_{\zeta \sim \mathcal{D}}[f_\zeta(w)]$$

- w represents a machine learning model described by d parameters/features (e.g logistic regression or a deep neural network).
- $\mathcal{D}$ is an unknown distribution of labelled examples,
- $f_\zeta(w)$ represents the loss of model w on a data point ζ, and

Stochastic Optimization and Machine Learning

In the stochastic optimization problem

$$f(w) = E_{\zeta \sim \mathcal{D}}[f_\zeta(w)]$$

- w represents a machine learning model described by d parameters/features (e.g logistic regression or a deep neural network).
- $\mathcal{D}$ is an unknown distribution of labelled examples,
- $f_\zeta(w)$ represents the loss of model w on a data point ζ, and
- f is the generalization error.

Problem (1) seeks to find the model w minimizing the generalization error

1. In statistical learning theory one assumes that while $\mathcal{D}$ is not known, samples $\zeta \sim \mathcal{D}$ are available.
2. In such case, $\nabla f(w)$ is not computable, while $\nabla f_\zeta(w)$, which is an unbiased estimator of the gradient of f at w, is easily computable

Finite Sum Problems

In this course we will focus on functions f which arise as averages of very large number of (smooth) functions:

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

Finite Sum Problems

In this course we will focus on functions f which arise as averages of very large number of (smooth) functions:

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

- This problem often arises by approximation of the stochastic optimization loss function (2) via Monte Carlo Integration.
- Known as the empirical risk minimization (ERM) problem.
- ERM is currently the dominant paradigm for solving supervised learning problems.

Finite Sum Problems

In this course we will focus on functions f which arise as averages of very large number of (smooth) functions:

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

- This problem often arises by approximation of the stochastic optimization loss function (2) via Monte Carlo Integration.
- Known as the empirical risk minimization (ERM) problem.
- ERM is currently the dominant paradigm for solving supervised learning problems.
- If index i is chosen uniformly at random from $[n] = \{1, 2, \ldots, n\}$, $\nabla f_i(w)$ is an unbiased estimator of $\nabla f(w)$.
- Typically, $\nabla f_i(w)$ is about n times less expensive to compute than $\nabla f(w)$.

Distributed Training

In distributed of supervised models, one considers the finite sum problem (3), with n being the number of machines, and each f_i

- also having a finite sum structure, i.e,

$$f_i(w) = \frac{1}{m} \sum_{j=1}^{m} f_{ij}(w) \tag{5}$$

where m corresponds to the number of training examples stored on machine i.

Distributed Training

In distributed of supervised models, one considers the finite sum problem (3), with n being the number of machines, and each f_i

- also having a finite sum structure, i.e,

$$f_i(w) = \frac{1}{m} \sum_{j=1}^{m} f_{ij}(w) \tag{5}$$

  where m corresponds to the number of training examples stored on machine i.

- or an infinite-sum structure, i.e,

$$f_i(w) = E_{\zeta_i \sim \mathcal{D}_i}[f_{i\zeta_i}(w)] \tag{6}$$

  where $\mathcal{D}_i$ is the distribution of data stored on machine i.

SGD

Stochastic gradient descent (SGD) is a state-of-the-art algorithmic paradigm for solving optimization problem (1) in situations where f is either of structure (2) or (3).
In its generic form (proximal) SGD defines the new iterate by subtracting a multiple of a stochastic gradient from the current iterate, and subsequently applying the proximal operator of R:

$$w_{k+1} = prox_{\gamma R}(w_k - \gamma g^k) \tag{7}$$

SGD

Stochastic gradient descent (SGD) is a state-of-the-art algorithmic paradigm for solving optimization problem (1) in situations where f is either of structure (2) or (3).
In its generic form (proximal) SGD defines the new iterate by subtracting a multiple of a stochastic gradient from the current iterate, and subsequently applying the proximal operator of R:

$$w_{k+1} = prox_{\gamma R}(w_k - \gamma g^k) \tag{7}$$

- g^k is an unbiased estimator of the gradient (i.e a "stochastic gradient"):

$$E[g^k / w_k] = \nabla f(w_k) \tag{8}$$

SGD

Stochastic gradient descent (SGD) is a state-of-the-art algorithmic paradigm for solving optimization problem (1) in situations where f is either of structure (2) or (3).
In its generic form (proximal) SGD defines the new iterate by subtracting a multiple of a stochastic gradient from the current iterate, and subsequently applying the proximal operator of R:

$$w_{k+1} = prox_{\gamma R}(w_k - \gamma g^k) \tag{7}$$

- g^k is an unbiased estimator of the gradient (i.e a "stochastic gradient"):

$$E[g^k / w_k] = \nabla f(w_k) \tag{8}$$

- $$prox_R(x) = arg\min_u \left\{ R(u) + \frac{1}{2}\|u - x\|^2 \right\}$$

The Prox Operator

Some facts about the prox operator[1]:

1. single-valuedness: $x \mapsto prox_R(x)$ is a function
2. non-expansiveness:

$$\|prox_R(x) - prox_R(y)\| \leq \|x - y\|, \quad \forall x, y \in \mathbb{R}^d$$

3. Moreau decomposition:

$$prox_R(x) - prox_{R^*}(x) = x, \quad \forall x \in \mathbb{R}^d$$

Here R^* is the Fenchel conjugate[2] of R.

[1] Assume $R : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is proper, closed and convex.

[2] $R^*(x) = sup_{y \in \mathbb{R}^d}\{\langle x, y \rangle - R(y)\}$

Stochastic Gradient

There are infinitely many ways of obtaining a random vector g^k satisfying (8)

- Prox: flexibility to construct stochastic gradients in various ways based on problem structure , and in order to target desirable properties such as:
  - convergence speed
  - iteration cost
  - overall complexity
  - parallelizability
  - suitability for given computing architecture
  - communication cost
  - generalization properties

Stochastic Gradient

There are infinitely many ways of obtaining a random vector g^k satisfying (8)

- Cons: A crazy ZOO of methods
  - Little hard to get into the fields, hard to keep up with new results
  - Considerable challenges in terms of convergence analysis. Indeed, if one aims to, as one should, obtain the sharpest bounds possible, dedicated analyses are needed to handle each of the particular variants of SGD.

Batch SGD = Gradient Descent

Gradient Descent

We first describe the (proximal) gradient descent (GD) method for solving the regularized convex optimization problem

$$\min_{w \in \mathbb{R}^d} f(w) + R(w) \tag{9}$$

This is the most basic of all SGD methods, and a starting point for the development of more elaborate variants.

Algorithm GD

starting points $x_0 \in \mathbb{R}^d$, **learning rate** $\gamma > 0$
for $k = 0, 1, 2, \cdots$ **do**
  Set $g^k = \nabla f(w_k)$
  $w_{k+1} = prox_{\gamma R}(w_k - \gamma g^k)$
end for

The idea is f might be something complicated but the linear approximation is simple.

$$\begin{aligned} f(w) &\simeq \text{linear function} \\ &= f(w_0) + \langle \nabla f(w_0), w - w_0 \rangle \end{aligned}$$

The idea is f might be something complicated but the linear approximation is simple.

$$\begin{aligned} f(w) &\simeq \text{linear function} \\ &= f(w_0) + \langle \nabla f(w_0), w - w_0 \rangle \end{aligned}$$

GD is an iterative algorithm where at each step we minimize a linear approximation but plus an additional term that makes sure we do not roll all the way down hill.

The idea is f might be something complicated but the linear approximation is simple.

$$\begin{aligned} f(w) &\simeq \text{linear function} \\ &= f(w_0) + \langle \nabla f(w_0), w - w_0 \rangle \end{aligned}$$

GD is an iterative algorithm where at each step we minimize a linear approximation but plus an additional term that makes sure we do not roll all the way down hill.

Idea of GD:

- start w_0
- $w_{k+1} = arg\min_w f(w_k) + \langle \nabla f(w_k), w - w_k \rangle + \frac{1}{2\gamma}\|w - w_k\|_2^2$

* 1^{st} term: linear function
* 2^{nd} term : quadratic term that penalize w being very far from w_k.

Taking deriv (grad) set to zero:

$$0 + \nabla f(w_k) + \frac{1}{\gamma}(w - w_k) = 0$$

$$\Rightarrow w_{k+1} = w_k - \gamma \nabla f(w_k)$$

Example: $f(x) = 3x^2 + 4x - 2$

Taking deriv (grad) set to zero:

$$0 + \nabla f(w_k) + \frac{1}{\gamma}(w - w_k) = 0$$

$$\Rightarrow w_{k+1} = w_k - \gamma \nabla f(w_k)$$

Example: $f(x) = 3x^2 + 4x - 2$

1. Can solve this directly: $6x + 4 = 0 \Rightarrow x^* = -\frac{2}{3}$

Taking deriv (grad) set to zero:

$$0 + \nabla f(w_k) + \frac{1}{\gamma}(w - w_k) = 0$$

$$\Rightarrow w_{k+1} = w_k - \gamma \nabla f(w_k)$$

Example: $f(x) = 3x^2 + 4x - 2$

1. Can solve this directly: $6x + 4 = 0 \Rightarrow x^* = -\frac{2}{3}$
2. Apply gradient descent. Initialize at x_1, take step size γ.
   - GD iteration: $x^+ = x - \gamma \nabla f(x) = x - \gamma(6x + 4)$

Taking deriv (grad) set to zero:

$$0 + \nabla f(w_k) + \frac{1}{\gamma}(w - w_k) = 0$$

$$\Rightarrow w_{k+1} = w_k - \gamma \nabla f(w_k)$$

Example: $f(x) = 3x^2 + 4x - 2$

1. Can solve this directly: $6x + 4 = 0 \Rightarrow x^* = -\frac{2}{3}$
2. Apply gradient descent. Initialize at x_1, take step size γ.
   - GD iteration: $x^+ = x - \gamma \nabla f(x) = x - \gamma(6x + 4)$

$$\begin{aligned}
\Rightarrow x_{k+1} &= x_k - \gamma 6 x_k - 4\gamma \\
&= (1 - 6\gamma) x_k - 4\gamma \\
&\vdots \\
&= (1 - 6\gamma)^k x_1 - \big((1 - 6\gamma)^{k-1} + (1 - 6\gamma)^{k-2} + \cdots + 1\big) 4\gamma \\
&= (1 - 6\gamma)^k x_1 - \frac{1 - (1 - 6\gamma)^k}{1 - (1 - 6\gamma)} 4\gamma
\end{aligned}$$

Need $|1-6\gamma| < 1 \Rightarrow (1-6\gamma)^k \to 0 \quad \text{as } k \to \infty.$

$$
\begin{aligned}
x_{k+1} &= (1-6\gamma)^k x_1 + \frac{(1-6\gamma)^k}{6\gamma} - \frac{1}{6\gamma} 4\gamma \\
&= (1-6\gamma)^k \left[x_1 + \frac{2}{3} \right] - \frac{2}{3} \to -\frac{2}{3} \text{as } k \text{ grows}
\end{aligned}
$$

$x_k \to -\frac{2}{3}$ very quickly with linear rate.

Need $|1-6\gamma| < 1 \Rightarrow (1-6\gamma)^k \to 0 \quad \text{as } k \to \infty.$

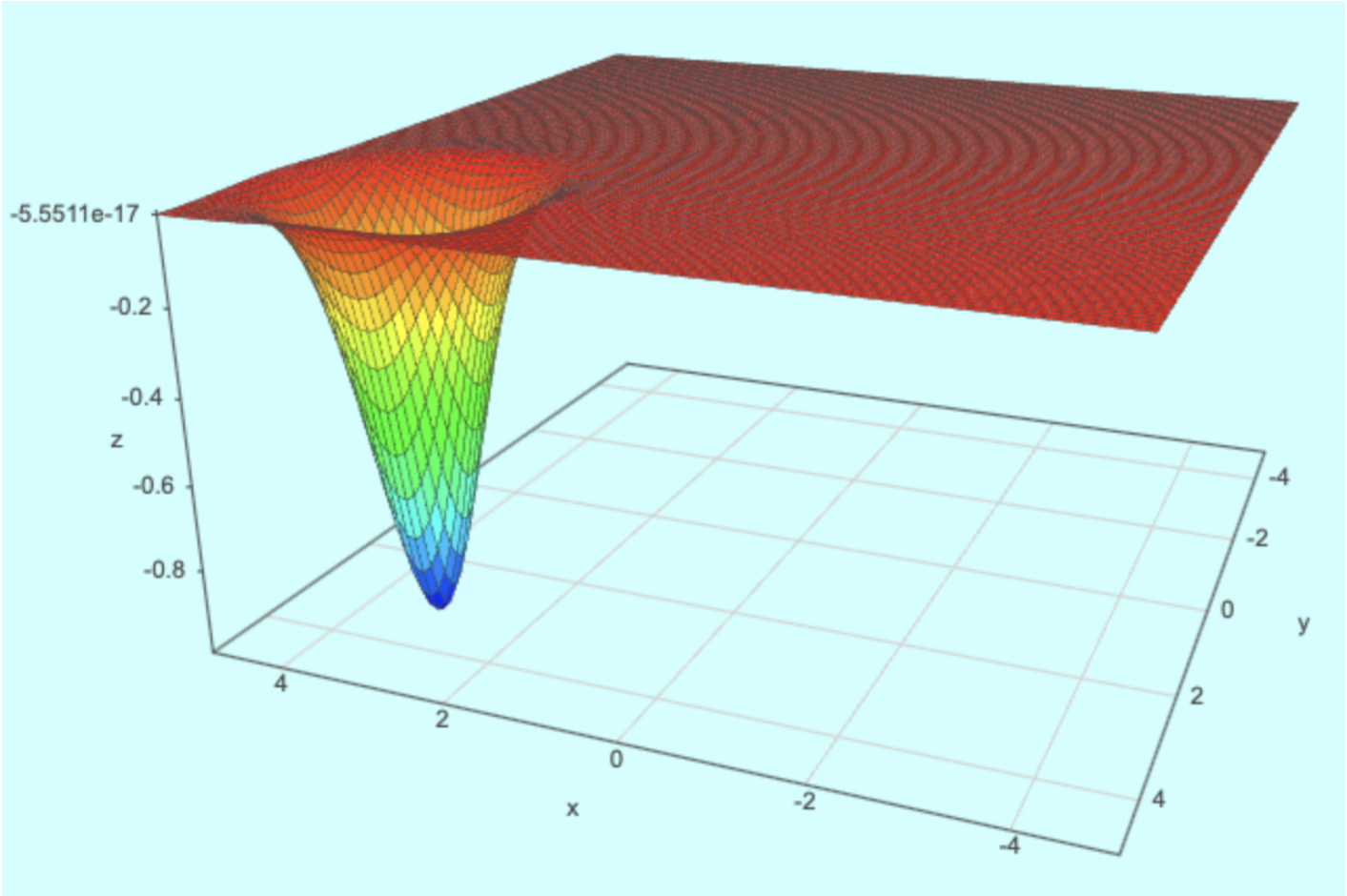
$$x_{k+1} = (1-6\gamma)^k x_1 + \frac{(1-6\gamma)^k}{6\gamma} - \frac{1}{6\gamma} 4\gamma$$
$$= (1-6\gamma)^k \left[x_1 + \frac{2}{3} \right] - \frac{2}{3} \to -\frac{2}{3} \text{as } k \text{ grows}$$

$x_k \to -\frac{2}{3}$ very quickly with linear rate.

1. **Good News**: GD for this function, converges very quickly. Error goes down with $(1-6\gamma)^k$ fast when $\gamma < \frac{1}{6}$.

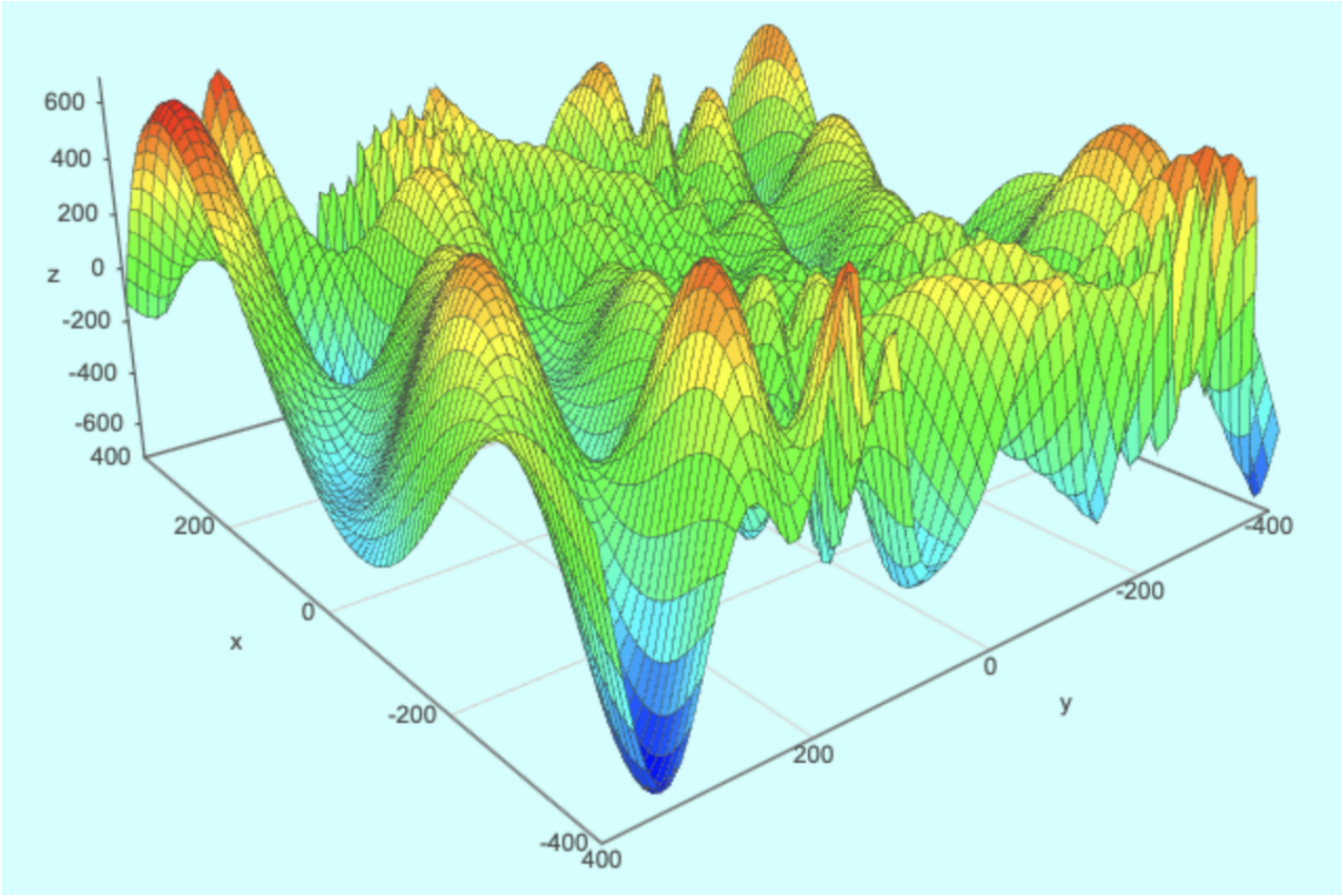
2. **Step size**: small enough so that $|1-6\gamma| < 1$
3. **Improvement at every iteration**: Exercise: check that $f(w_{k+1}) \leq f(w_k)$

Optimization is hard (in general)



$$f(x,y) = -cos(x)cos(y)exp(-(x-\pi)^2 - (y-\pi)^2) \quad \text{in } [-5,5]^2$$

Optimization is hard (in general)

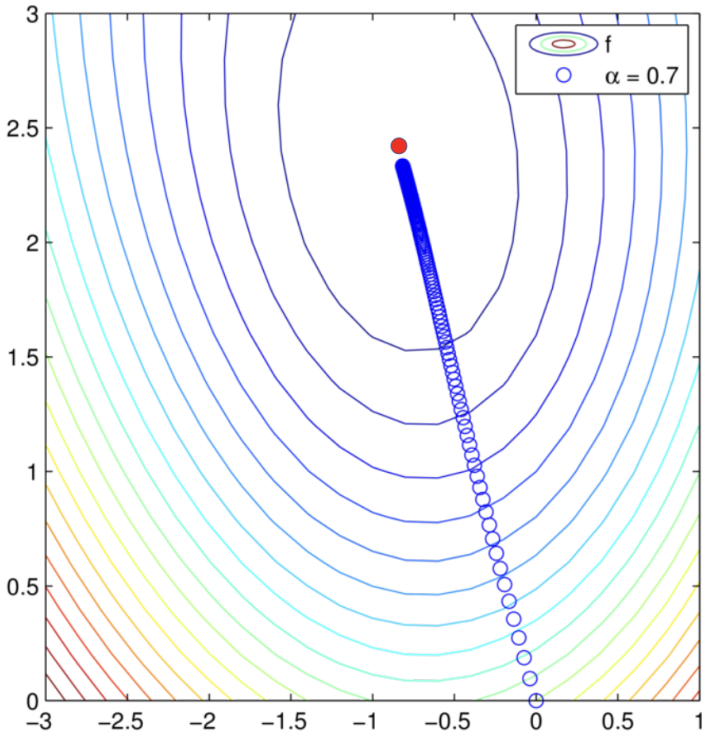


$$f(x,y) = -(y+47)sin\sqrt{|\frac{x}{2}+(y+47)|} - xsin\sqrt{|\frac{x}{2}-(y+47)|} \quad \text{in } [-400,400]^2$$

Gradient Descent Example

A Logistic Regression problem using the fourclass labelled data from LIBSVM $(n, d) = (862, 2)$.
Logistic Regression :

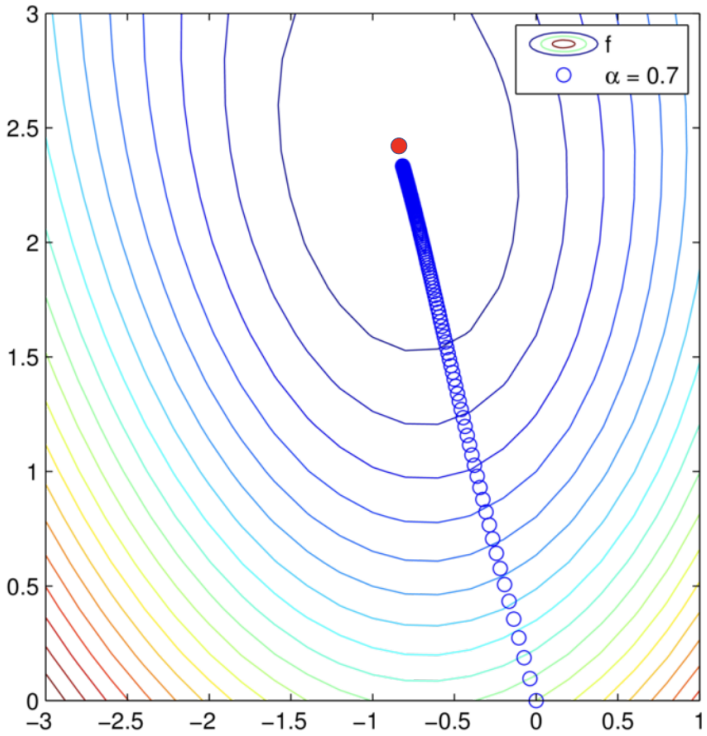
$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} ln(1 + e^{-y^i \langle w, x^i \rangle}) + \lambda \|w\|_2^2$$



Gradient Descent Example

A Logistic Regression problem using the fourclass labelled data from LIBSVM $(n, d) = (862, 2)$.
Logistic Regression :

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} ln(1 + e^{-y^i \langle w, x^i \rangle}) + \lambda \|w\|_2^2$$

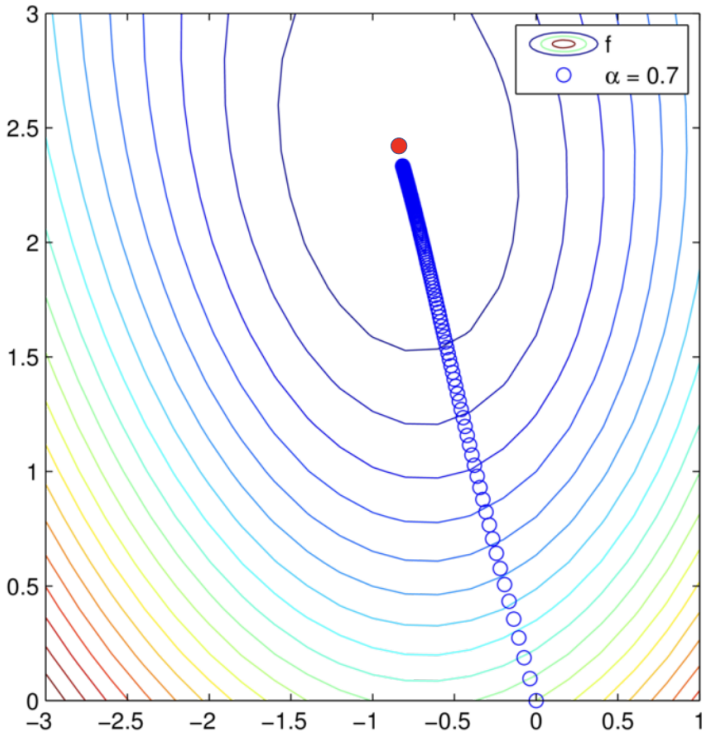


1. Can we prove that this always works?

Gradient Descent Example

A Logistic Regression problem using the fourclass labelled data from LIBSVM $(n, d) = (862, 2)$.
Logistic Regression :

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} ln(1 + e^{-y^i \langle w, x^i \rangle}) + \lambda \|w\|_2^2$$



1. Can we prove that this always works?
2. **No**! There is no universal optimization method. The "no free lunch" of Optimization
3. Need assumptions: **Convex** and **smooth** training problems

Main assumption

Nice property:

$$\text{If } \nabla f(w^*) = 0 \quad \text{then } f(w^*) \leq f(w),\ \forall\ w \in \mathbb{R}^d$$

$\Rightarrow$ All stationary points are global minima.

Main assumption

Nice property:

$$\text{If } \nabla f(w^*) = 0 \quad \text{then } f(w^*) \leq f(w), \ \forall \ w \in \mathbb{R}^d$$

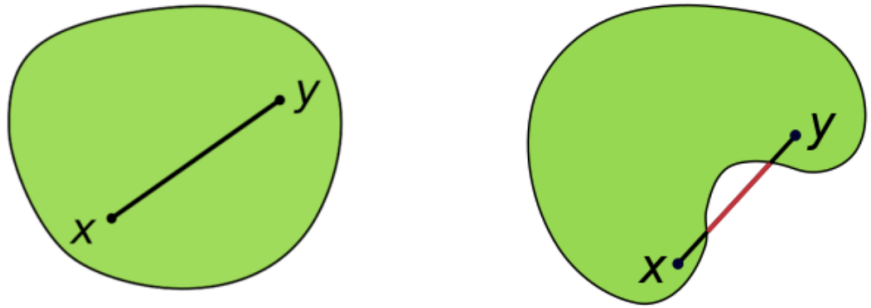
$\Rightarrow$ All stationary points are global minima.

Lemma

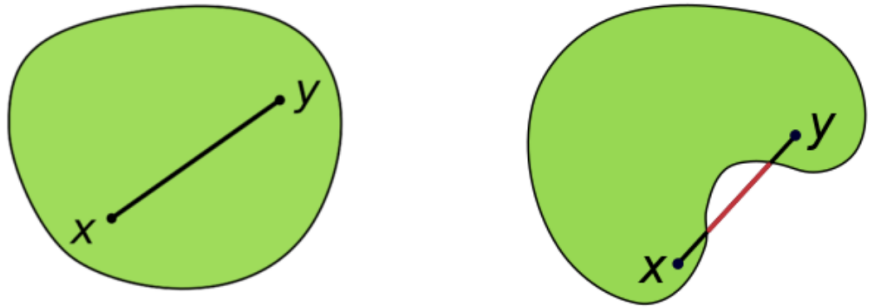
Convexity $\Rightarrow$ ***Nice property***.
If $f(w) \geq f(y) + \langle \nabla f(y), w - y \rangle, \quad \forall \ w, y \in \mathbb{R}^d$ *then Nice property holds.*

PROOF: Choose $y = w^*$.

Convex sets - Definition



Convex sets - Definition



A set $\mathbb{C} \subseteq \mathbb{R}^n$, is convex if

$$\forall\, x, y \in \mathbb{C},\ \forall\ \lambda \in [0,1], \quad \lambda x + (1-\lambda) y \in \mathbb{C}. \tag{10}$$

Why it is important ?

Convex sets

Example

Definition

$M \in \mathbb{R}^{n \times n}$ matrix is p.s.d if:

1. Symmetric
2. $x^\top M x \geq 0, \ \forall \ x \in \mathbb{R}^n$

We denote by $\mathcal{S}_+^n$, the set of symmetric p.s.d matrices.

The set of p.s.d matrices is a convex set.
Let $M_1, M_2 \in \mathcal{S}_+^n$, we want to show that
$M = \lambda M_1 + (1 - \lambda) M_2 \in \mathcal{S}_+^n, \ \lambda \in [0, 1]$

Convex sets

Copositive matrices : a hard convex set

A symmetric matrix M is **copositive** if

$$x^\top Mx \geq 0, \ \forall \ x \in \mathbb{R}^n_+$$

We denote by $\mathcal{C}^n$, the set of symmetric copositive matrices.

1. Q: Is the set of copositives matrices bigger or smaller than $\mathcal{S}^n_+$?
2. In general it is intractable to determine whether a matrix M is copositive.

Example

1. Every matrix with only non negative entries is copositive Hence $M = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ is copositive but not p.s.d due to $det(M) = -1$.
2. Every p.s.d is also copositive but the converse is false i.e. $\mathcal{S}^n_+ \subseteq \mathcal{C}^n$

Exercise

Characterize the triples $(x, y, z) \in \mathbb{R}^3$ for which the matrix $M = \begin{pmatrix} x & z \\ z & y \end{pmatrix}$ is

1. copositive
2. p.s.d

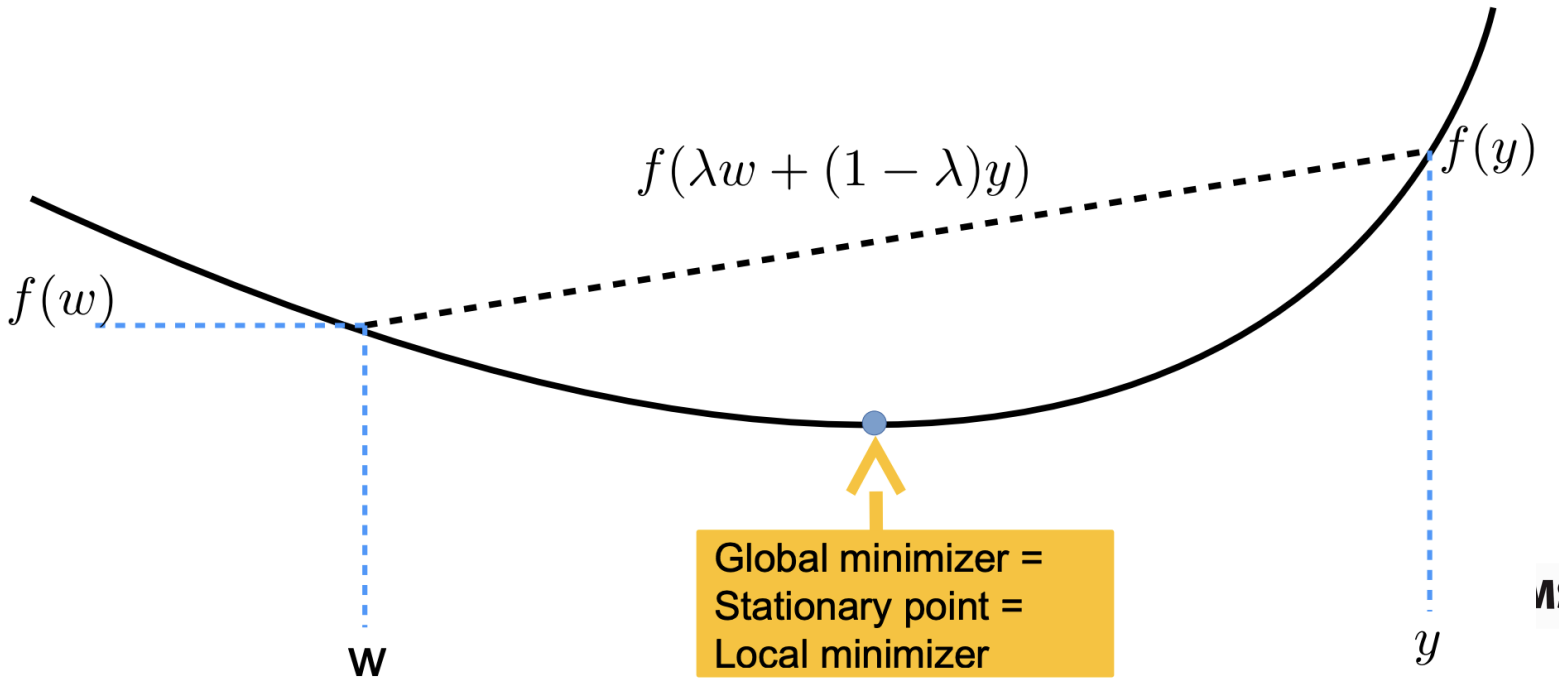
Def 1 : Convexity

Definition

We say that $f : dom(f) \subseteq \mathbb{R}^d \to \mathbb{R}$ is **convex** if $dom(f)$ is **convex** and

$$f(\lambda w + (1-\lambda)y) \leq \lambda f(w) + (1-\lambda)f(y), \tag{11}$$

$\forall\ w, y \in dom(f),\ \lambda \in [0,1]$

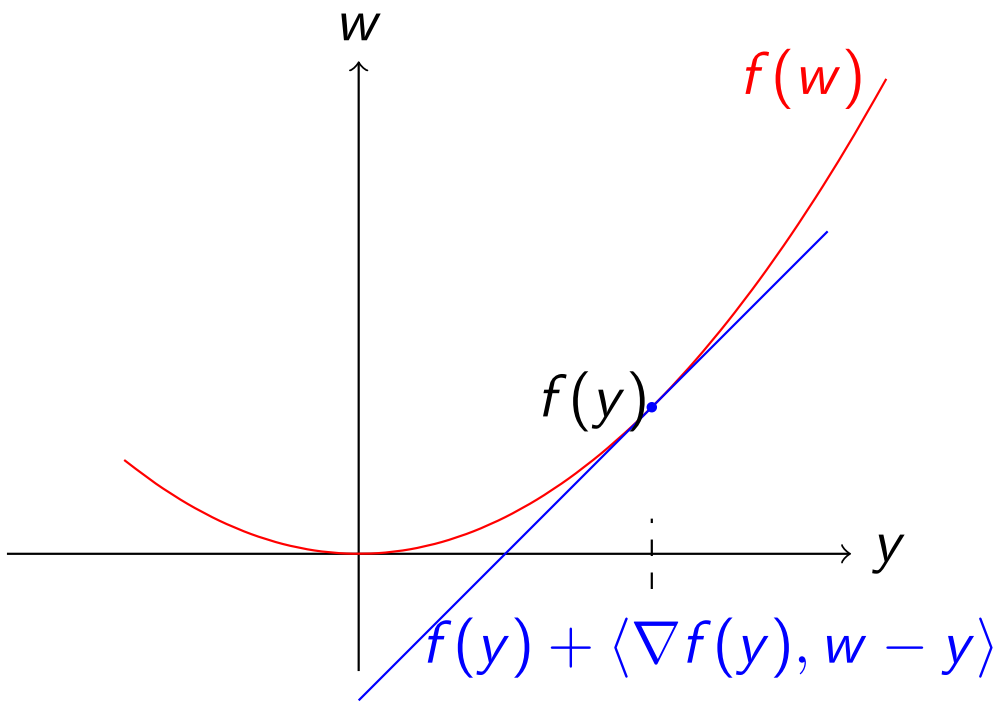


Def 2 : Convexity - First derivative

Definition

A differential function $f : dom(f) \subseteq \mathbb{R}^d \to \mathbb{R}$ is **convex** iff

$$f(w) \geq f(y) + \langle \nabla f(y), w - y \rangle, \quad \forall\ w, y \in dom(f) \tag{12}$$

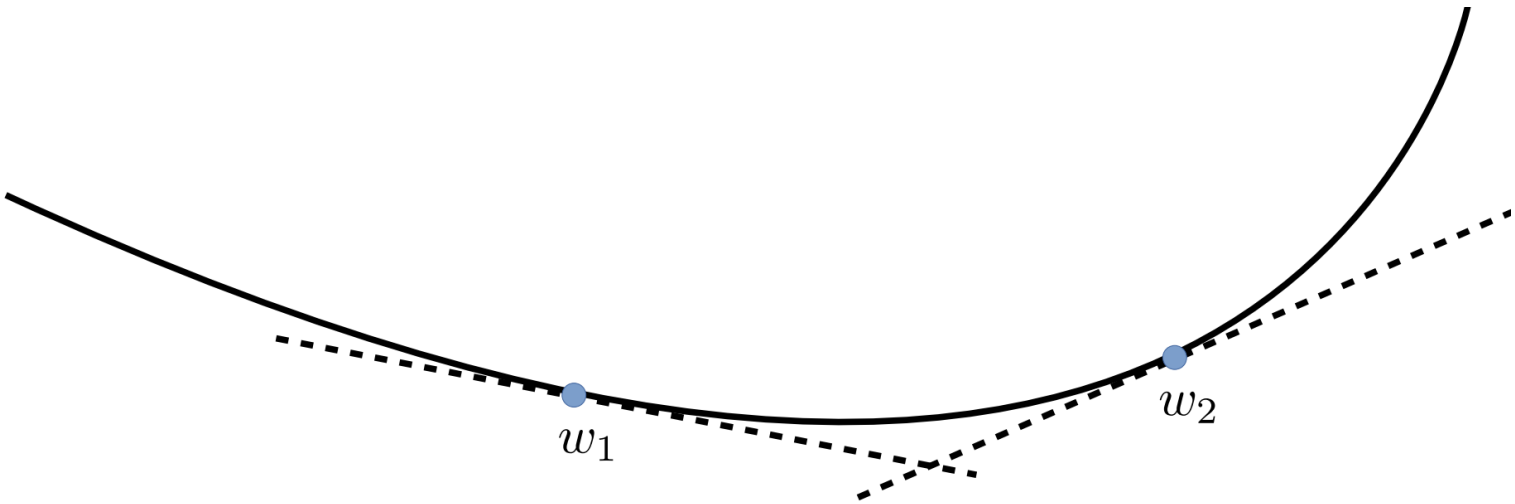


Def 3 : Convexity - Second derivative

Definition

A twice differentiable function $f : dom(f) \subseteq \mathbb{R}^d \to \mathbb{R}$ is **convex** iff

$$\nabla^2 f(w) \succcurlyeq 0 \Leftrightarrow v^\top \nabla^2 f(w) v \geq 0, \quad \forall\ w, v \in dom(f) \tag{13}$$



$w_1 \leq w_2 \Rightarrow f^{'}(w_1) \leq f^{'}(w_2)$.
($\nabla^2 f(w)$ p.s.d i.e. all eigen values of $\nabla^2 f(w)$ are ≥ 0).

Convexity : Examples

1. Norms and squared norms : $x \mapsto \|x\|$, $x \mapsto \|x\|^2$
2. Negative log and logistic : $x \mapsto \log(x)$, $x \mapsto log(1 + e^{-y\langle a,x\rangle})$
3. Hinge Loss : $x \mapsto max\{0, 1 - yx\}$
4. Negatives log determinant, exponentiation ... etc

Def 2' : Convex functions (non-smooth)

Definition

A function f is **convex** if $\forall\, y,\ \exists g$ such that

$$f(w) \geq f(y) + \langle g, w - y \rangle \tag{14}$$

i.e at every point of the function there exists some linear function which touch the function at that point but it is not necessarily unique as a gradient.

- If f is convex and differentiable, then $g = \nabla f(y)$ satisfies this. It is unique.

Subgradient and Subdifferential

Definition

For a convex function f, a vector g such that

$$f(w) \geq f(y) + \langle g, w - y \rangle, \quad \forall y$$

is called a subgradient.

The set of subgradients of f at a point w is called the subdifferential of f at w and denoted by $\partial f(w)$

More examples

1. Applying the three definitions for $f(x) = x^\top Q x, Q \succcurlyeq 0$

Acknowledgements

Thanks for your attention!