MSc Big Data for Business - MAP 534
Introduction to machine learning

Gradient based optimization

Naive gradient, stochastic gradient & Accelerated gradient
Motivation in Machine Learning

Logistic regression

Feed forward neural networks

General formulation

Gradient descent procedures

Gradient Descent

Stochastic Gradient Descent

Momentum

Coordinate Gradient Descent
Motivation in Machine Learning

Logistic regression

Feed forward neural networks

General formulation

Gradient descent procedures

Gradient Descent

Stochastic Gradient Descent

Momentum

Coordinate Gradient Descent
The objective is to predict the label \( Y \in \{0, 1\} \) based on \( X \in \mathbb{R}^d \).

Logistic regression models the distribution of \( Y \) given \( X \).

\[
\mathbb{P}(Y = 1|X) = \sigma(\langle w, X \rangle + b),
\]

where \( w \in \mathbb{R}^d \) is a vector of model weights and \( b \in \mathbb{R} \) is the intercept, and where \( \sigma \) is the sigmoid function.

\[
\sigma : z \mapsto \frac{1}{1 + e^{-z}}.
\]

The sigmoid function is a model choice to map \( \mathbb{R} \) into \((0, 1)\).

Another widespread solution for \( \sigma \) is \( \sigma : z \mapsto \mathbb{P}(Z \leq z) \) where \( Z \sim \mathcal{N}(0, 1) \), which leads to a probit regression model.
Logistic regression

Log-odd ratio

\[ \log \left( \frac{P(Y = 1|X)}{P(Y = 0|X)} \right) = \langle w, X \rangle + b. \]

Classification rule

Note that

\[ P(Y = 1|X) \geq P(Y = 0|X) \]

if and only if

\[ \langle w, x \rangle + b \geq 0. \]

\[ \rightarrow \] This is a linear classification rule.

\[ \rightarrow \] This classifier requires to estimate \( w \) and \( b \).
\[ \{(X_i, Y_i)\}_{1 \leq i \leq n} \text{ are i.i.d. with the same distribution as } (X, Y). \]

**Likelihood**

\[
\prod_{i=1}^{n} P(Y_i | X_i) = \prod_{i=1}^{n} \sigma(\langle w, X_i \rangle + b)^{Y_i} (1 - \sigma(\langle w, X_i \rangle + b))^{1-Y_i},
\]

\[
= \prod_{i=1}^{n} \sigma(\langle w, x_i \rangle + b)^{Y_i} \sigma(-\langle w, X_i \rangle - b)^{1-Y_i}
\]

and the **normalized negative loglikelihood** is

\[
f(w, b) = \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, \langle w, X_i \rangle + b).
\]
Logistic regression

Compute $\hat{w}_n$ and $\hat{b}_n$ as follows:

$$(\hat{w}_n, \hat{b}_n) \in \arg\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} \left( -Y_i (X_i^T w + b) + \log(1 + e^{X_i^T w + b}) \right).$$

→ It is an **average of losses**, one for each sample point.

→ It is a **convex and smooth problem**.

Using the **logistic loss** function

$$\ell : (y, y') \mapsto \log(1 + e^{-yy'})$$

yields

$$(\hat{w}_n, \hat{b}_n) \in \arg\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, \langle w, X_i \rangle + b).$$
Assume for now that the intercept is 0. Then, the likelihood is,

\[ L_n(w) = \prod_{i=1}^{n} \left( \frac{e^{X_i^T w}}{1 + e^{X_i^T w}} \right)^{Y_i} \left( \frac{1}{1 + e^{X_i^T w}} \right)^{1-Y_i} = \prod_{i=1}^{n} \left( \frac{e^{X_i^T w Y_i}}{1 + e^{X_i^T w}} \right). \]

And the \textbf{negative log-likelihood} is

\[ \ell_n(w) = -\log(L_n(w)) = \sum_{i=1}^{n} \left( -Y_i X_i^T w + \log(1 + e^{X_i^T w}) \right). \]

\textbf{Derivatives}

\[ \frac{\partial \left( \log(L_n(w)) \right)}{\partial w_j} = \sum_{i=1}^{n} \left( Y_i X_{ij} - \frac{x_{ij} e^{X_i^T w}}{1 + e^{X_i^T w}} \right) = \sum_{i=1}^{n} X_{ij} (Y_i - \sigma(\langle w, X_i \rangle)). \]

\rightarrow \textbf{No explicit solution} for the maximizer of the loglikelihood... Parameter estimate obtained using \textbf{gradient based optimization}. 
Motivation in Machine Learning

Logistic regression

Feed forward neural networks

General formulation

Gradient descent procedures

Gradient Descent

Stochastic Gradient Descent

Momentum

Coordinate Gradient Descent
Feed Forward Network

$X$ input in $\mathbb{R}^d$.

$z^h(X)$ pre-activation in $\mathbb{R}^H$, with weight $W^h \in \mathbb{R}^{d \times H}$ and bias $b^h \in \mathbb{R}^H$.

$g$ any activation function to produce $h \in \mathbb{R}^H$.

$z^o(X)$ pre-activation in $\mathbb{R}^M$, with weight $W^o \in \mathbb{R}^{H \times M}$ and bias $b^o \in \mathbb{R}^M$.

Apply the softmax function to produce the output, i.e. $P(Y = m | X)$ for $1 \leq m \leq M$. 
Motivation in Machine Learning

- Logistic regression
- Feed forward neural networks

General formulation

Gradient descent procedures

- Gradient Descent
- Stochastic Gradient Descent
- Momentum
- Coordinate Gradient Descent
Parameter inference in machine learning often boils down to solving

$$\arg\min_{w \in \mathbb{R}^d} f(w) + g(w),$$

with \( f \) a goodness-of-fit function based on a loss \( \ell \),

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle w, x_i \rangle)$$

and

$$g(w) = \lambda \text{pen}(w),$$

where \( \lambda > 0 \) and \( \text{pen}(\cdot) \) is some penalization function.

- \( \text{pen}(w) = \|w\|^2_2 \) (Ridge).
- \( \text{pen}(w) = \|w\|_1 \) (Lasso).
Outline

Motivation in Machine Learning
- Logistic regression
- Feed forward neural networks
- General formulation

Gradient descent procedures
- Gradient Descent
- Stochastic Gradient Descent
- Momentum
- Coordinate Gradient Descent
Motivation in Machine Learning

Logistic regression

Feed forward neural networks

General formulation

Gradient descent procedures

Gradient Descent

Stochastic Gradient Descent

Momentum

Coordinate Gradient Descent
Exhaustive search

\[ w^* \in \arg\min_{w \in [0,1]^d} f(w). \]

Optimizing on a grid of \([0,1]^d\), when \(f\) is regular enough, requires \([1/\varepsilon]^d\) evaluations to achieve a precision of order \(\varepsilon\).

Evaluating the expression

\[ f : x \mapsto \sum_{i=1}^{d} x_i^2, \]

to obtain a precision of \(\varepsilon = 10^{-2}\) requires \(1.75 \times 10^{-3}\) seconds in dimension 1 and \(1.75 \times 10^{15}\) seconds in dimension 10, i.e., nearly 32 millions years.

→ Prohibitive in high dimensions (curse of dimensionality).
First order necessary condition

→ **In dimension one.**

Let \( f : \mathbb{R} \rightarrow \mathbb{R} \) be a differentiable function. If \( x^* \) is a local extremum (minimum/maximum) then \( f'(x^*) = 0 \).

→ **Generalization for \( d > 1 \).**

Let \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) be a differentiable function. If \( x^* \) is a local extremum then \( \nabla f(x^*) = 0 \).

Points such that \( \nabla f(x^*) = 0 \) are called **critical points**.

**Critical points** are not always extrema (consider \( x \mapsto x^3 \)).
The gradient of a function \( f : \mathbb{R}^d \to \mathbb{R} \) in \( x \in \mathbb{R}^d \), denoted by \( \nabla f(x) \), is the vector of partial derivatives:

\[
\nabla f(x) = \left( \begin{array}{c} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{array} \right).
\]

**Some useful gradients**

- If \( f : \mathbb{R} \to \mathbb{R} \), \( \nabla f(x) = f'(x) \).
- \( f : x \mapsto \langle a, x \rangle \): \( \nabla f(x) = a \).
- \( f : x \mapsto x^T Ax \): \( \nabla f(x) = (A + A^T)x \).
- Particular case: \( f : x \mapsto \|x\|^2 \), \( \nabla f(x) = 2x \).
Heuristic: why gradient descent works?

For a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$, define the **level sets**:

$$C_c = \{ x \in \mathbb{R}^d, f(x) = c \}.$$

**Figure 1:** Gradient descent for function $f : (x, y) \mapsto x^2 + 2y^2$

$\rightarrow$ The gradient is **orthogonal to level sets**.
Convexity

Convexity - Definition

A function \( f : \mathbb{R}^d \to \mathbb{R} \) is **convex** on \( \mathbb{R}^d \) if, for all \( x, y \in \mathbb{R}^d \), for all \( \lambda \in [0, 1] \),
\[
f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).
\]

Convexity - First derivative

A **differentiable function** \( f : \mathbb{R}^d \to \mathbb{R} \) is convex if and only if, for all \( x, y \in \mathbb{R}^d \),
\[
f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle.
\]
If $f : \mathbb{R}^d \to \mathbb{R}$ is \textbf{twice differentiable}, the \textbf{Hessian matrix} in $x \in \mathbb{R}^d$ denoted by $\nabla^2 f(x)$ is given by

$$
\nabla^2 f(x) = \begin{pmatrix}
\frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d}(x) \\
\frac{\partial^2 f}{\partial x_2 \partial x_1}(x) & \frac{\partial^2 f}{\partial x_2^2}(x) & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_d}(x) \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 f}{\partial x_d \partial x_1}(x) & \frac{\partial^2 f}{\partial x_d \partial x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_d^2}(x)
\end{pmatrix}.
$$

The \textbf{Hessian matrix is symmetric if $f$ is twice continuously differentiable}. 
Convexity - Hessian

A **twice differentiable function** $f : \mathbb{R}^d \to \mathbb{R}$ is convex if and only if, for all $x \in \mathbb{R}^d$,

$$\nabla^2 f(x) \geq 0,$$

that is $h^T \nabla^2 f(x) h \geq 0$, for all $h \in \mathbb{R}^d$. 

$x_1 \leq x_2 \implies f'(x_1) \leq f'(x_2)$
Assume that $f$ is twice continuously differentiable.

**Necessary condition**

If $x^*$ is a local minimum, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semi-definite.

**Sufficient condition**

If $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite then $x^*$ is a strict local optimum.

For $d = 1$, this condition boils down to $f'(x^*) = 0$ and $f''(x^*) > 0$. 
Gradient descent algorithms are **iterative procedures**. There are two classes of such algorithms, depending on the information that is used to compute the next iteration.

**First-order algorithms** that use $f$ and $\nabla f$. Standard algorithms when $f$ is differentiable and convex.

**Second-order algorithms** that use $f$, $\nabla f$ and $\nabla^2 f$. They are useful when computing the Hessian matrix is not too costly.
Gradient descent algorithm

**Gradient descent**

**Input:** Function $f$ to minimize, initial vector $w^{(0)}$, $k = 0$.

**Parameters:** step size $\eta > 0$.

While *not converge* do

\[- w^{(k+1)} = w^{(k)} - \eta_{k+1} \nabla f(w^{(k)}).\]

\[- k = k + 1. \]

**Output:** $w^{(n*)}$ where $n*$ is the last iteration.
Gradient descent in practice

![Gradient descent graph](image)

- **Negative log likelihood** vs **Number of iterations**
- Lines represent different step sizes:
  - Blue: Step size 0.010000
  - Orange: Step size 0.100000
  - Green: Step size 0.500000
  - Red: Step size 1.000000
  - Purple: Step size 2.000000

The graph illustrates how different step sizes affect the convergence of gradient descent.
When does gradient descent converge?

Convex function

A function $f : \mathbb{R}^d \to \mathbb{R}$ is **convex** on $\mathbb{R}^d$ if, for all $x, y \in \mathbb{R}^d$, for all $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).$$

$L$-smooth function

A function $f$ is said to be **$L$-smooth** if $f$ is differentiable and if, for all $x, y \in \mathbb{R}^d$,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

If $f$ is **twice differentiable**, this is equivalent to writing that for all $x \in \mathbb{R}^d$,

$$\lambda_{\text{max}}(\nabla^2 f(x)) \leq L.$$
Theorem

Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be a $L$-smooth convex function. Let $w^*$ be a minimum of $f$ on $\mathbb{R}^d$. Then, Gradient Descent with step size $\eta \leq 1/L$ satisfies

$$f(w^{(k)}) - f(w^*) \leq \frac{\|w^{(0)}-w^*\|_2^2}{2\eta k}.$$ 

In particular, for $\eta = 1/L$, 

$$L\|w^{(0)} - w^*\|_2^2/2$$

iterations are sufficient to get an $\varepsilon$-approximation of the minimal value of $f$. 

27/45
A key point: the descent lemma

If $f$ is $L$-smooth, then for any $w, w' \in \mathbb{R}^d$,

$$f(w') \leq f(w) + \langle \nabla f(w), w' - w \rangle + \frac{L}{2} \|w - w'\|^2_2.$$  

Using the descent Lemma,

$$\arg\min_{w \in \mathbb{R}^d} \left\{ f(w^k) + \langle \nabla f(w^k), w - w^k \rangle + \frac{L}{2} \|w - w^k\|^2_2 \right\}$$

$$= \arg\min_{w \in \mathbb{R}^d} \|w - \left( w^k - \frac{1}{L} \nabla f(w^k) \right) \|^2_2.$$  

Hence, it is natural to choose

$$w^{k+1} = w^k - \frac{1}{L} \nabla f(w^k).$$

This is the most standard gradient descent algorithm.
Faster rate for strongly convex function

**Strong convexity**

A function $f : \mathbb{R}^d \to \mathbb{R}$ is $\mu$-**strongly convex** if
\[ x \mapsto f(x) - \frac{\mu}{2} \|x\|_2^2 \]
is convex.

If $f$ is differentiable it is equivalent to, for all $x \in \mathbb{R}^d$,
\[ \lambda_{\min}(\nabla^2 f(x)) \geq \mu. \]
This is also equivalent to, for all $x, y \in \mathbb{R}^d$,
\[ f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} \|y - x\|_2^2. \]

**Theorem**

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a $L$-**smooth, $\mu$ strongly convex function**. Let $w^*$ be a minimum of $f$ on $\mathbb{R}^d$. Then, Gradient Descent with step size $\eta \leq 1/L$ satisfies

\[ f(w^{(k)}) - f(w^*) \leq (1 - \eta \mu)^k \|f(w^{(0)}) - f(w^*)\|_2^2. \]
How to choose $\eta$?

**Exact line search**

At each step, choose the best $\eta$ by optimizing

$$
\eta^{(k)} = \arg\min_{\eta>0} f(w - \eta \nabla f(w)).
$$

→ **Computationally very intensive...**

**Backtracking line search**

Let $0 < \beta < 1$, then at each iteration, start with $\eta_k = 1$ and while

$$
f(w^{(k)} - \eta_k \nabla f(w^{(k)})) - f(w^{(k)}) > -\frac{\eta_k}{2} \| \nabla f(w^{(k)}) \|^2,
$$

update $\eta_k \leftarrow \beta \eta_k$.

→ **Simple and work pretty well in practice.**

If $f : \mathbb{R}^d \to \mathbb{R}$ is a **$L$-smooth convex function**, then, Gradient Descent with backtracking line search satisfies

$$
f(w^{(k)}) - f(w^*) \leq \frac{\| w^{(0)} - w^* \|^2}{2k \min(1, \beta/L)}.
$$
Motivation in Machine Learning

- Logistic regression
- Feed forward neural networks
- General formulation

Gradient descent procedures

- Gradient Descent
- **Stochastic Gradient Descent**
- Momentum
- Coordinate Gradient Descent
Stochastic Gradient Descent (SGD)

Previous methods are based on **full gradients**, since each iteration requires the computation of

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

which depends on the whole dataset.

If $n$ is large, computing $\nabla f(w)$ is computationally expensive.

If $I$ is chosen uniformly at random in $\{1, \ldots, n\}$, then

$$\mathbb{E}[\nabla f_i(w)] = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w) = \nabla f(w),$$

$\nabla f_i(w)$ is an **unbiased** but very noisy estimate of the full gradient $\nabla f(w)$.

Computation of $\nabla f_i(w)$ only requires the $I$-th observation.
Stochastic Gradient Descent (SGD)

**Stochastic gradient descent algorithm**

**Input**: starting point \( w^{(0)} \), steps (learning rates) \( \eta_k \)

For \( k = 1, 2, \ldots \) until *convergence* do

→ Pick at random (uniformly) \( l_k \) in \( \{1, \ldots, n\} \).

→ compute

\[
  w^{(k)} = w^{(k-1)} - \eta_k \nabla f_{l_k}(w^{(k-1)}).
\]

**Return** last \( w^{(k)} \).

**Remarks**

→ Each iteration **has complexity** \( O(d) \) **instead of** \( O(nd) \) for full gradient methods.

→ Possible to reduce this to \( O(s) \) when features are \( s \)-sparse using **lazy-updates**.
Stochastic gradient descent in practice (I)

![Stochastic gradient descent graph](image)

- **Negative loglikelihood** vs. **Number of iterations**
- Lines for different step sizes:
  - Step size 0.010000
  - Step size 0.100000
  - Step size 0.500000
  - Step size 1.000000
  - Step size 2.000000
Project each estimate into the ball $B(0, R)$ with $R > 0$ fixed.

Let

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

**Theorem**

Assume that $f$ is convex and that there exists $b > 0$ satisfying, for all $x \in B(0, R)$,

$$\|\nabla f_i(x)\| \leq b.$$

Assume also that all minima of $f$ belong to $B(0, R)$. Then, setting $\eta_k = 2R/(b\sqrt{k})$,

$$\mathbb{E} \left[ f \left( \frac{1}{k} \sum_{j=1}^{k} w(j) \right) \right] - f(w^*) \leq \frac{3Rb}{\sqrt{k}}.$$
Motivation in Machine Learning

- Logistic regression
- Feed forward neural networks
- General formulation

Gradient descent procedures

- Gradient Descent
- Stochastic Gradient Descent
- Momentum
- Coordinate Gradient Descent
Improving Polyak’s momentum

Nesterov Accelerated Gradient Descent

Input: starting point \( w^{(0)} \), learning rate \( \eta_k > 0 \), initial velocity \( v^{(0)} = 0 \), momentum \( \beta_k \in [0, 1] \).

While not converge do

\[ v^{(k+1)} = w^{(k)} - \eta \nabla f(w^{(k)}). \]
\[ w^{(k+1)} = v^{(k+1)} + \beta_{k+1}(v^{(k+1)} - v^{(k)}). \]

Return last \( w^{(k+1)} \).
Theorem

Assume that $f$ is a $L$-smooth, convex function whose minimum is reached at $w^*$. Then, if $\beta_{k+1} = k/(k+3)$,

$$f(w^{(k)}) - f(w^*) \leq \frac{2\|w^{(0)} - w^*\|^2}{\eta(k+1)^2}.$$

Theorem

Assume that $f$ is a $L$-smooth, $\mu$ strongly convex function whose minimum is reached at $w^*$. Then, choosing

$$\beta_k = \frac{1 - \sqrt{\mu/L}}{1 + \sqrt{\mu/L}},$$

yields

$$f(w^{(k)}) - f(w^*) \leq \frac{\|w^{(0)} - w^*\|^2}{\eta} \left(1 - \sqrt{\frac{\mu}{L}}\right)^k.$$
Outline

Motivation in Machine Learning
  Logistic regression
  Feed forward neural networks
  General formulation

Gradient descent procedures
  Gradient Descent
  Stochastic Gradient Descent
  Momentum
  Coordinate Gradient Descent
→ Received a lot of attention in machine learning and statistics the last 10 years.
→ It is state-of-the-art on several machine learning problems.
→ This is what is used in many R packages and for scikit-learn Lasso / Elastic-net and LinearSVC.

→ Minimize one coordinate at a time (keeping all others fixed).
**Exact coordinate descent (CD)**

For $k \geq 1$,

→ Choose $j \in \{1, \ldots, d\}$.

→ Compute

$$w_{j}^{k+1} = \arg\min_{z \in \mathbb{R}} f(w_{1}^{k}, \ldots, w_{j-1}^{k}, z, w_{j+1}^{k}, \ldots, w_{d}^{k}),$$

$$w_{j'}^{k+1} = w_{j'}^{k} \quad \text{for} \ j' \neq j.$$ 

**Remarks**

→ **Cycling through the coordinates is arbitrary:** uniform sampling, pick a permutation and cycle over it every each $d$ iterations.

→ Only **1D optimization problems to solve**, but a lot of them.
Theorem - Warga (1963)

If $f$ is continuously differentiable and strictly convex, then exact coordinate descent converges to a minimum.

Remarks

→ A 1D optimization problem to solve at each iteration: cheap for least-squares, but can be expensive for other problems.

→ Replace exact minimization w.r.t. one coordinate by a single gradient step in the 1D problem.
Coordinate gradient descent (CGD)

For $k \geq 1$,

→ Choose $j \in \{1, \ldots, d\}$.

→ Compute

$$w_j^{k+1} = w_j^k - \eta_j \nabla_{w_j} f(w^k),$$
$$w_{j'}^{k+1} = w_{j'}^k \quad \text{for } j' \neq j.$$

$\eta_j =$ the step-size for coordinate $j$, can be taken as $\eta_j = 1/L_j$ where $L_j$ is the Lipchitz constant of

$$f^j(z) = f(w + z e_j) = f(w_1, \ldots, w_{j-1}, z, w_{j+1}, \ldots, w_d).$$
Theorem - Nesterov (2012)

Assume that \( f \) is convex and smooth and that each \( f^j \) is \( L_j \)-smooth.

Consider a sequence \( \{w^k\} \) given by CGD with \( \eta_j = 1/L_j \) and coordinates chosen at random: i.i.d and uniform distribution in \( \{1, \ldots, d\} \). Then,

\[
\mathbb{E}[f(w^{k+1})] - f(w^*) \leq \frac{n}{n+k} \left( (1 - \frac{1}{n})(f(w^0) - f(w^*)) + \frac{1}{2} \|w^0 - w^*\|_L^2 \right),
\]

with \( \|w\|_L^2 = \sum_{j=1}^{d} L_j w_j^2 \).

→ **Bound in expectation**, since coordinates are taken at random.

→ For **cycling coordinates** \( j = (k \mod d) + 1 \) the bound is much worse.