# W2 Discussion

Yijia Zhao

2025-04-10

## Key Takeaways

### 1. Selection of Step Size in Gradient Descent

- Exact Line Search method: $\eta_k = \text{argmin}_{\eta>0} f(x_k - \eta \nabla f(x_k))$ (usually harder than solving the original optimization problem).
- Backtracking line search method: set $\eta_k = \tau \eta_k$ until the Armijo condition holds, i.e., $h(\eta_k) \leq h(0) + \epsilon \eta_k h'(0)$, where $h(\eta_k) = f(x_k - \eta_k \nabla f(x_k))$, $h'(0) = -[\nabla f(x_k)]^\top \nabla f(x_k)$.
- Constant step size $\eta$: for strictly convex functions, if $\eta \in (0, \frac{1}{L})$, then the gradient descent algorithm will converge to the unique global minimum, where $L$ is the Lipschitz constant such that $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2$.
- Diminishing step size along a fixed sequence: $\sum_{k=1}^{\infty} \eta_k = \infty$.

### 2. Momentum-based Variants

Momentum methods aim to implicitly "tell the difference" between the curved and flat directions.

#### 2.1 Polyak's Heavy Ball Method

Gradient step: $x_{k+1} = y_k - \eta_k \nabla f(x_k)$.

Momentum step: $y_k = x_k + \underbrace{\xi (x_k - x_{k-1})}_{\text{momentum}}, \quad \xi \in (0, 1)$.

#### 2.2 Nesterov's Momentum Method

"Look-ahead" gradient step: $x_{k+1} = y_k - \eta_k \nabla f(y_k)$.

Momentum step: $y_k = x_k + \underbrace{\xi_k (x_k - x_{k-1})}_{\text{momentum}}, \quad \xi_k \in (0, 1)$, where the optimal choice of $\xi_k$ for convex function is $\xi_k = \frac{k-1}{k+2}$.

NOTE: Polyak and Nesterov momentum methods do not use a pure descent direction at each iteration.

# HW1 Q&A

## Problem 1 (a)

Solve $\nabla f(x) = 0$ and use $\nabla^2 f(x) > 0$ to show the solution is a global minimum.

## Problem 1 (b)

Go over the gradient descent linear regression back-tracking line search code. Please download the **latest** version!

Revise the code to implement gradient descent for problem 1, mainly the objective function and the gradient.

If you use python, also need to revise the code in order to keep track of $\eta_k$.

## Problem 2 (a)

Selection of constant step size: quadratic form in $\beta$ with $Q = X^\top X/(2n)$. The range of step size can be determined by the largest eigenvalue of $Q$. —> Figure out the singular values of $X$.

In R:

Calculation of Stepsize using singular value:

```
n <- dim(X)[1]
1/(max(svd(X)$d)^2/(2*n))
```

Or using eigenvalue:

```
n <- dim(X)[1]
1/(max(eigen(t(X)%*%X)$values)/(2*n))
```

```
data1 = read.csv("/Users/yijiazhao/Dropbox/MyMacOngoing/TA/102B/HW1/dataset1.csv")
```

```
model1  = lm(Y~.-1, data = data1)
beta = model1$coefficients
```

In python:

Calculation of step size using singular value:

```
svd_decomp = np.linalg.svd(X)
1/(max(svd_decomp.S**2)/(2*n))
```

Or using eigenvalue:

```
eig_decomp = np.linalg.eig(X.T@X)
1/(max(eig_decomp.eigenvalues)/(2*n))
```

```
import pandas as pd
data1 = pd.read_csv("/Users/yijiazhao/Dropbox/MyMacOngoing/TA/102B/HW1/dataset1.csv")
df1 = pd.DataFrame(data1)
X = df1[df1.columns[1:]]
y = df1[df1.columns[0]]
from sklearn.linear_model import LinearRegression
model1 = LinearRegression()
model1.fit(X, y)
beta_fit = model1.coef_
```

## Problem 2(b)

Add an extra step at the beginning of each round to calculate $y_k$ and revise the updating rule accordingly.

What you need to revise:

- A vector to record $x_{k-1}$.
- Rules of momentum step at the beginning of each round of calculating $y_k$.
- Rules of gradient step at each time of calculating $x_{k+1}$.
- The calculation of $h'(0)$ may change to $-\nabla f(y_k)^\top \nabla f(x_k)$ for Polyak's method and $-\nabla f(y_k)^\top \nabla f(y_k)$ for Nesterov's method in Armijo condition.
- Update $x_{k-1}, x_k$ at the end of each round.