# Discussion 4:

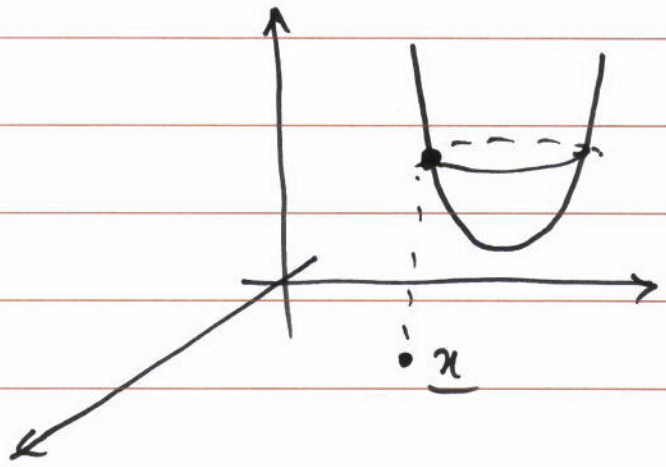
- Briefly talk about Iterative Descent Methods
- preprocessing data
- some tips for HW:

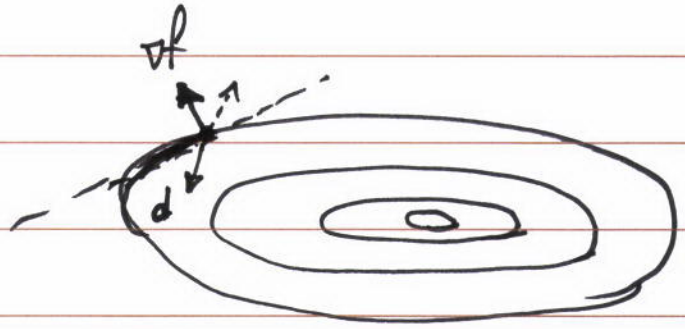
---

Q: How are we gonna achieve opt. point?

A: Iterative Descent Methods

- $\nabla f(x) = 0 \Rightarrow$ we have candidate
- $\nabla f(x) \neq 0 \Rightarrow$ NOT a candidate






if angle $\Delta$ (between $\nabla f$ and $\underline{d}$ $> 90°$)
then direction $\underline{d}$ results in a decrease in $f$

if $\nabla f^T \underline{d} < 0$

$\exists \delta > 0$ s.t. $f(\underline{x} + \alpha \underline{d}) < f(\underline{x})$ $\forall \alpha \in (0, \delta)$

Iterative Descent Methods:

$$\underline{x}^{r+1} = \underline{x}^r + \alpha^r \underline{d}^r$$

$\alpha^r$: step size; $\underline{d}^r$: direction

Choices of direction $\underline{d}^r$:

- Steepest / Gradient Descent: $\underline{d}^r = -\nabla f(\underline{x}^r)$
- Diagonal scaled Gradient Descent:

$$\underline{d}^r = -D^r \nabla f(\underline{x}^r)$$

$D^r$: diagonal, $D^r > 0$

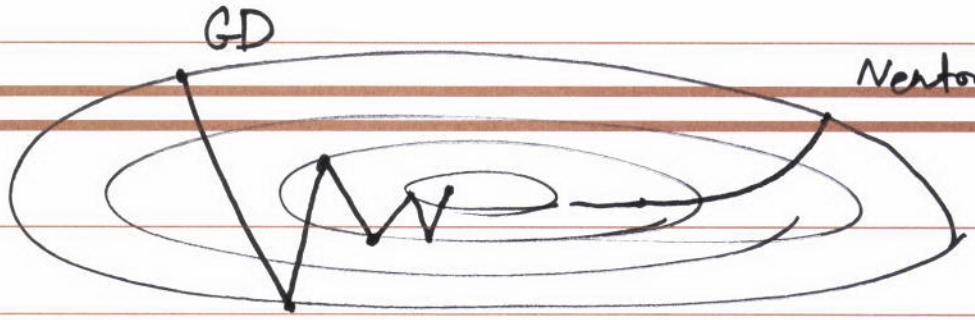
- Newton Direction $\underline{d}^r = -\left(\nabla^2 f(\underline{x}^r)\right)^{-1} \nabla f(\underline{x}^r)$

$$\underbrace{f(x^r + d)}_{} \simeq f(x^r) + \underbrace{\langle d, \nabla f(x^r)\rangle}_{\triangleq \nabla^T f(x^r)\, \underline{d}} + \frac{1}{2} d^T \underbrace{\nabla^2 f(x^r)}_{} d$$

$$\min_{d} \quad f(x^r + d)$$

$$\frac{\partial}{\partial \underline{d}} = 0 \Rightarrow \quad \nabla f(x^r) + \nabla^2 f(x^r)\, d^* = 0$$

$$\Rightarrow d^* = -\left(\nabla^2 f(x^r)\right)^{-1} \nabla f(x^r)$$




- Benefit : fast convergence
- Drawback : compute inverse

Diagonalized version of Newton:

$$D^r = \text{diag}\left(\nabla^2 f(x^r)\right)^{-1}$$

$$d^r = -D^r \nabla f(x^r)$$

Choice of step size:

- Constant $\alpha^r = \alpha \quad \forall r = 0, 1, \ldots$ (prob: we don't exact const.)

- Exact Minimization $\alpha^r = \underset{\alpha \geq 0}{\operatorname{argmin}} \; f(\underline{x}^r + \alpha \underline{d}^r)$ (1)

- Limited Minimization $\alpha^r = \underset{\alpha \in (0, \bar{\alpha})}{\operatorname{argmin}} \; f(\underline{x}^r + \alpha \underline{d}^r)$

Adv.: — less computation compared to (1)

— less amount time compared to (1)

— we are NOT sure about opt. direction, so don't search entire space

- Diminishing $\alpha^r \downarrow 0$ with $\sum \alpha^r = \infty$

↓

why? need to search the entire space to reach to the opt.

$\alpha^r = \frac{1}{r}$ ✓ $\sum \frac{1}{r} \to \infty$

Convergence Rate:

- Asymptotic behavior of function $f$:

$$e(x) = \|x - x^*\| \quad \text{or} \quad e(x) = |f(x) - f(x^*)|$$

-

$$\lim_{r \to \infty} \sup \frac{e(x^{r+1})}{e(x^r)} = \beta$$

$$\frac{\|x^{r+1} - x^*\|}{\|x^r - x^*\|}$$

$$\beta = \begin{cases} \beta \in (0,1) & \text{linear} \rightarrow \frac{\|x^{r+1} - x^*\|}{\|x^r - x^*\|} = \frac{1}{2} \\ 1 & \text{sublinear} \\ 0 & \text{superlinear} \end{cases}$$

Example:

(for linear, $\beta = \frac{1}{2}$) $\Rightarrow \|x^{r+1} - x^*\| = \left(\frac{1}{2}\right)^r \|x^0 - x^*\|$

* The smaller the $\beta$ is, the faster convergence rate is.

- So far we saw GD alg.
- { Batch GD
  { Stochastic GD

Let's consider the following example to see these two GD (Batch & Stoch.):

Perceptron Problem:

- Given $\{(x_i, y_i)\}_{i=1}^{N}$ and having a $y \in \{\pm 1\}$

linear model $g(w,x) = w^T x$ with hypothesis

$h(x) = \text{sign}(w^T x)$

- A prediction is correct if $y\, w^T x > 0$

Goal: try to find a "good" model s.t. $h(x)$ makes few mis-prediction.

Formalize as optimization prob. { - training data
- loss function

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} L(w^T x_i, y_i)$$

Algorithm to find opt. $w$:

- initialize $w$
- Compute $\nabla J(w)$
- update $w^{(t+1)} = w^{(t)} - \eta \nabla J(w)$ → $\eta$: step size
- repeat this process until convergence

⇒ Batch GD (use all data to compute)

---

what loss function?

- 0/1 Loss function

$$J_{0/1}(w) = \frac{1}{N} \sum L\left(\text{sgn}(w^T x_i), y_i\right)$$

$$L(y_i', y_i) \triangleq \begin{cases} 0 & \text{if } y_i' = y_i \\ 1 & \text{else} \end{cases}$$

(Not a good loss function because of flat shape of $J(w)$)

- perceptron criterion (hinge loss)

$$J_p(w) = \frac{1}{N} \sum_{i=1}^{N} \max(0, -y_i w^T x_i)$$

→ is 0 if $y_i$ is correctly predicted, otherwise is "confidence" in the misprediction.

In order to update $\underline{w}$, we don't need to have ALL data points, we can update $\underline{w}$ by having every single data point.

for sample $i$: $J_i(w) = \max(0, -y_i w^T x_i)$

$$\frac{\partial J_i}{\partial w_j} = \begin{cases} 0 & \text{if } y_i w^T x_i > 0 \\ -y_i x_{ij} & \text{else} \end{cases}$$

Vector ↓

$$\nabla J_i = \begin{cases} 0 & \text{if } y_i w^T x_i > 0 \\ -y_i \underline{x}_i & \text{else} \end{cases}$$

after observing $(\underline{x}_i, y_i)$, if it is mis predicted, we update $w$ as follows

$$w \leftarrow w + y_i \underline{x}_i$$

Algorithm for Stochastic GD:

- initialize $w$
- Repeat

$$u_i \leftarrow w^T x_i$$
$$\text{if } y_i u_i \leq 0$$
$$w \leftarrow w + y_i x_i$$

* $\eta$: (is also refferred as learning rate)
(As learning continues, $\eta$ should be smaller)

- Preprocessing of Data

  - Represent Data in appropriate way
    - Categorical
    - Real
    - Integer
  - Standizing data
    - subtract by mean
    - divid by standard deviation
  - Hadling missing data

Categorical Data

| index | feature (Grade) | label |
|---|---|---|
| 1 | A | 1 |
| 2 | B | 1 |
| 3 | C | 1 |
| 4 | F | 0 |

(Encoding)

Converting Dummy variable V:

- for D distinct component, we are gonna add D features which are 1 or 0

New form

| index | $f_1$ | $f_2$ | $f_3$ | $f_4$ | label |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 | 0 | 1 |
| 3 | 0 | 0 | 1 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 | 0 |

4 distinct letters

A → 1 0 0 0
B → 0 1 0 0
C → 0 0 1 0
F → 0 0 0 1

drawback is if you have large D, you need to add so many features!!

• Standardizing Data:

(Variance) Assume feature 1 component over all samples are as follows

| $f_1$ |
|---|
| 1 |
| 2 |
| 3 |

| $f_1'$ |
|---|
| 4 |
| 8 |
| 12 |

if we replace $f_1$ by $f_1'$ where

should the model change weights for the 1st feature?

Since values in $f_1'$ are larger, model considers larger weights $w_1'$ compared to weight for $f_1$, i.e. $w_1$.

* To resolve this issue, we need to normalize all features by dividing all features by their standard deviation.

(Mean) Assume $f_1$ as before but $f_1'$ as $f_1'$

1001
1002
1003

if we replace $f_1$ by $f_1'$, should the model change the weights for the 1st feature? Again, since values in $f_1'$ is larger compared to $f_1$, the weights for $f_1'$, i.e. $w_1'$, would be larger compared to the weight for $f_1$.

* To resolve this issue, we need to subtract all features by their corresponding mean.

Therefore, by performing the aforementioned process all features have the same importance.

(Next Session for handling missing data)

Tips for HW:

python users (only use numpy.shuffle to suffle the indices, then code on your own to make cross-validation)