MATH 565: Lecture 6 (01/29/2026)

Today: * 2nd order optimality examples
* gradient descent: line search
* stochastic gradient descent (SGD)

Second order local optimality conditions (from Lecture 4): With $\nabla J(\bar{w}_0)=\bar{0}$,

#4. If $HJ(\bar{w}_0) \succcurlyeq 0$, test is inconclusive.

Consider $f(x,y)=x^2+y^4$
$g(x,y)=x^2-y^4$
$h(x,y)=x^2+y^3$

$\Rightarrow \nabla f=\begin{bmatrix}2x\\4y^3\end{bmatrix}, \nabla g=\begin{bmatrix}2x\\-4y^3\end{bmatrix}, \nabla h=\begin{bmatrix}2x\\3y^2\end{bmatrix}; \Rightarrow \nabla=\bar{0}$ for all three functions at $\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix}$

Also, $Hf=\begin{bmatrix}2&0\\0&12y^2\end{bmatrix}, Hg=\begin{bmatrix}2&0\\0&-12y^2\end{bmatrix}, Hh=\begin{bmatrix}2&0\\0&6y\end{bmatrix}$

$\Rightarrow H=\begin{bmatrix}2&0\\0&0\end{bmatrix}$ at $\bar{0}$ for all cases, and $H \succcurlyeq 0$ (PSD).
→ eigenvalues are 2, 0.

But $\begin{bmatrix}x\\y\end{bmatrix}=\bar{0}$ is a local minimum of f, a saddle point of g, and is neither of h!

Consider the Taylor series expansion:

$$J(\bar{w}) \approx J(\bar{w}_0) + \underbrace{\nabla J^T}_{=0}(\bar{w}-\bar{w}_0) + \underbrace{(\bar{w}-\bar{w}_0)^T HJ(\bar{w}_0)}_{=0?}(\bar{w}-\bar{w}_0) + \text{higher order terms}$$

If $H=0$ at $\bar{w}_0$, then behavior of J depends on the higher order terms, and hence the second order test is inconclusive when determining local optima.

Back to gradient descent

$$\bar{w}_{t+1} = \bar{w}_t - \alpha_t \nabla J(\bar{w}_t)$$

Choosing α_t (continued...)

→ We saw decay strategies for changing α_t last time...

5. **Line search**

Instead of decaying, choose α_t optimally!

Let $\bar{g}_t$ be a descent direction in step t. We can set $\bar{g}_t = -\nabla J(\bar{w}_t)$ by default, but can pick it differently.

In $\bar{w}_{t+1} \leftarrow \bar{w}_t + \alpha_t \bar{g}_t$, we pick α_t as

$$\alpha_t = \underset{\alpha}{\operatorname{argmin}}\ J(\bar{w}_t + \alpha \bar{g}_t) \qquad (\ell)$$

(ℓ for line search)

Since we are trying to minimize J, we may as well go all the way in each iteration. Also, we get the following result.

If we choose α_t by solving (ℓ), we can guarantee that

$$\nabla J(\bar{w}_{t+1}) \perp \bar{g}_t$$

else, α_t is not optimal for (ℓ)!

Consider moving a small amount ($\pm\delta$) further along $\bar{g}_t$ (beyond α_t):

$$J(\bar{w}_t + \alpha_t \bar{g}_t \pm \delta \bar{g}_t) \approx J(\bar{w}_t + \alpha_t \bar{g}_t) \pm \delta \bar{g}_t^T \nabla J(\bar{w}_t + \alpha_t \bar{g}_t)$$

$\neq 0 \Rightarrow$ J can be improved further.

If $\nabla J(\bar{w}_{t+1}) \neq \bar{0}$, we can decrease J further by moving $\pm\delta$ units along $\bar{g}_t$.

Choosing descent directions in this orthogonal fashion could be useful in general...

How to solve (1)?

Pick some α_{max} and consider $\alpha \in [0, \alpha_{max}]$, and search this interval using

- binary search → halve the interval in each iteration, keeping the half that is beneficial
- golden section search
  let current interval is $[a, b]$
  check $a < m_1 < m_2 < b$ → unlike in binary search, evaluate only (up to) 4 points.
  can drop $[a, m_1]$ or $[m_2, b]$ or
  can exclude $[a, m_2)$ or $(m_1, b]$.
- Armijo rule (check L04ML). → another inexact but efficient method

---

Properties of Optimization in ML

* objective functions are loss functions that are usually additively separable — contribution to loss from each sample or point is added up to get the total loss.

e.g., $J = \frac{1}{2}\|D\bar{w} - \bar{y}\|^2 = \frac{1}{2}\sum_{i=1}^{n}(\bar{x}_i^T\bar{w} - y_i)^2$

Here is another common loss function in ML:

- product of probabilities $P(\bar{x}_i, y_i, \bar{w})$ that the prediction for sample i using $\bar{w}$ matches true value y_i for each sample:

$$\prod_{i=1}^{n} P(\bar{x}_i, y_i, \bar{w}).$$

But instead of the product, we use → additively separable

$$J(\bar{w}) = -\log\left(\prod_{i=1}^{n} P(\bar{x}_i, y_i, \bar{w})\right) = -\sum_{i=1}^{n} \log\left(P(\bar{x}_i, y_i, \bar{w})\right).$$

In general, we have

$$J(\bar{w}) = \sum_{i=1}^{n} J_i(\bar{w}).$$

$J_i(\bar{w})$: loss from sample/point i.

i: sample from the n points

This structure makes it possible to use stochastic gradient descent (SGD):

Let S be a sample of the n points (observations) $S \subset \{1,\dots,n\}$. Consider $J_S(\bar{w}) = \sum_{i \in S} J_i(\bar{w})$. For instance, in regression, $J_S(\bar{w}) = \sum_{i \in S} \|\bar{x}_i^T \bar{w} - y_i\|^2$.

S is called a **minibatch**, and the method is called the minibatch SGD, where we compute

$$\bar{w} \leftarrow \bar{w} - \alpha \nabla J_S(\bar{w})$$

J_S instead of J

Often, $|S| \ll n$, so minibatch SGD is typically more efficient than GD.

The special case with $|S|=1$ is called SGD. (without the minibatch qualifier)

→ in each iteration, choose one random observation.

SGD (with $|S|=1$) is used when n is huge (and d also is large).

One typically uses minibatch SGD with $|S| = 2^r$ (e.g., 64, 256, ...) for effective use of GPUs (or other processors/cores).

The minibatch gradients are often close to the (full) gradient at start — when $\bar{w}$ is far from a (local) minimum. But they are not as close when we get closer to the minimum — but regularization usually helps out.

Minibatch SGD usually tries to use as much of the data (points) as the iterations proceed. Here is one approach. We use

$R[n]$ = a random permutation of $\{1,\dots,n\} = \{r_1,\dots,r_n\}$.

(each r_i is one of $\{1,2,\dots,n\}$.)

Let $|S| = k$ (minibatch size).

Set $S_i = \{r_{(i-1)k+1},\dots,r_{ik}\}$, for $i = 1,\dots,\lceil \frac{n}{k} \rceil$.

A single S_i is referred to as an epoch. Each epoch uses k points (observations), except maybe the last epoch. There are a total of $\lceil \frac{n}{k} \rceil$ epochs.

In a typical minibatch SGD run,

- ∇J_S is a good approximation of ∇J
- Choice of S becomes more important as $\bar{w}$ gets closer to a local minimum.
  → (regularization helps).