Gradient Descent

Preliminary (math stuff that will be useful)

1 Convexity

Definition 1.1. *A set C is said to be convex if for all $x, y \in C$ and $0 \leq t \leq 1$, the affine combination $(1-t)x + ty$ also belongs to C.*

Definition 1.2. *Let $f : \mathbb{R}^n \to \mathbb{R}$. For $0 \leq t \leq 1$ and $x, y \in \text{dom}(f)$, if*

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$$

then f is a convex function. If $f(tx + (1-t)y) < tf(x) + (1-t)f(y)$, then f is a strictly convex function. Note that f is convex if and only if its epigraph is a convex set.

Definition 1.3. *A matrix A is said to be positive semidefinite if and only if for any $x \in \mathbb{R}^n, x^T Ax \geq 0$. Similarly, A is positive definite if and only if $x^T Ax > 0$.*

For any matrix A, the *spectral* norm, or the induced 2-norm, is defined as

$$\|A\|_2 = \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2}$$

Definition 1.4. *Let A, B be any two matrices. Then, $A \succcurlyeq B$ means that for any vector x we have $x^T Ax \geq x^T Bx$. Notably, if $A \succcurlyeq 0$, then A is positive semidefinite.*

Theorem 1.1. *Let f be a convex function. Then, for any $x, y \in \text{dom}(f)$,*

$$f(y) \geq f(x) + \nabla f(x) \cdot (y - x)$$

Furthermore, for any $x \in \text{dom}(f)$,

$$\nabla^2 f(x) \succcurlyeq 0$$

Unrelated to convexity, but for any $f : \mathbb{R}^n \to \mathbb{R}$, the second order Taylor approximation is

$$f(x) = f(x_0) + (x - x_0)^T \nabla f(x_0) + \frac{1}{2}(x - x_0)^T \nabla^2 f(x_0)(x - x_0)$$

In most models, it is reasonable to assume that the gradient of the loss function is L-Lipschitz. Then, we have the following theorem.

Theorem 1.2. *Let f be a convex function such that $\nabla f(x)$ is L-Lipschitz. Then, $\nabla^2 f(x) \preccurlyeq L$, which means*

$$f(y) \leq f(x) + \nabla f(x) \cdot (y - x) + \frac{L}{2}\|x - y\|^2$$

What is GD?

- GD is x_{t + 1} = x_t - alpha * grad f(x_t).

- The idea of GD is simple – you are at a particular point, find a direction to move in that helps us get one step closer to the optimum (minimum or maximum), use that direction to make your final move (and now you're on a different spot).
- GD is about how we find that direction.

Rough intuition for how GD works

- The following theorem can be shown using some of the properties I gave above:

Theorem 6.1 *Suppose the function $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable, and that its gradient is Lipschitz continuous with constant $L > 0$, i.e. we have that $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2$ for any x, y. Then if we run gradient descent for k iterations with a fixed step size $t \leq 1/L$, it will yield a solution $f^{(k)}$ which satisfies*

$$f(x^{(k)}) - f(x^*) \leq \frac{\|x^{(0)} - x^*\|_2^2}{2tk}, \tag{6.1}$$

where $f(x^*)$ is the optimal value. Intuitively, this means that gradient descent is guaranteed to converge and that it converges with rate $O(1/k)$.

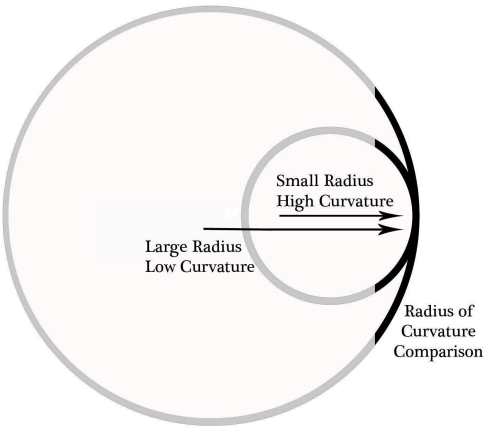
- Important takeaway here is that the convergence is O(1/k). It basically means that the f(x_k) - f(x*) follows the 1/k graph. Key thing to understand here is that increases in k in when k is small lead to significant decrease in the difference than when k is already large
- Below we connect gradient descent to curvature. This will set up discussion for preconditioning, adagrad, etc

$$\mathcal{L}(\theta_{t+1}) \leq \mathcal{L}(\theta_t) - \eta\|g_t\|^2 + \frac{1}{2}\eta^2\|g_t\|^2 L$$

Set $\eta = \frac{1}{L}$. Then

$$\mathcal{L}(\theta_{t+1}) \leq \mathcal{L}(\theta_t) - \frac{1}{L}\|g_t\|^2 + \frac{1}{2L}\|g_t\|^2 \implies \mathcal{L}(\theta_{t+1}) \leq \mathcal{L}(\theta_t) - \frac{\|g_t\|^2}{2L}$$

- This is just using some of the properties you saw above. Important is that last equation on the right side of the second line.
- Basically tells you that you want a large enough gradient relative to your curvature.
    - To review: curvature is how fast the slope is changing (it's the degree to which your function is "curved")
- Now, remember that you never subtract the whole gradient, like you always subtract the learning rate * the gradient.
    - And, so that's how you would "adjust" when you want more of the gradient and when you want less



- The problem is we never know what the local curvature (or the second gradient) is when we are doing gradient descent
- So, we will look into other ways of dealing with this

Preconditioning

- As we discussed earlier, curvature is key to gradient descent
- Now, we talked about one dimensional curvature (just using the second gradient). There is curvature along every dimension in higher dimensions. Furthermore, these can vary as well as you'd expect

  Hence, gradient descent becomes very difficult as certain dimensions require more "push" or "increment" than other dimensions. So, we must somehow account for all this
- When a function has curvature that is different along each dimension, we say that the function is ill-conditioned (we call this a *conditioning problem*)
- To determine whether a function is ill-conditioned, find it's Hessian matrix, then calculate the condition number of the matrix (the ratio of the largest eigenvalue to the smallest eigenvalue) → <u>in a sense, the eigenvalues tell you the dimensions of highest curvature and lowest curvature</u>
  - **Step 2: Compute the Hessian for $f(x, y) = 5x^2 + 13y^2$**

    Let's calculate the second derivatives for our specific function:

    - **First, the partial derivatives:**
      - $\frac{\partial f}{\partial x} = \frac{\partial}{\partial x}(5x^2 + 13y^2) = 10x$
      - $\frac{\partial f}{\partial y} = \frac{\partial}{\partial y}(5x^2 + 13y^2) = 26y$
    - **Now, the second partial derivatives:**
      - $\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}(10x) = 10$
      - $\frac{\partial^2 f}{\partial y^2} = \frac{\partial}{\partial y}(26y) = 26$
      - $\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial}{\partial y}(10x) = 0$ (no y in $10x$)
      - $\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial}{\partial x}(26y) = 0$ (no x in $26y$)

    Since there are no cross terms (like xy) in the function, the mixed partial derivatives are zero. So, the Hessian matrix is:

    $$H = \begin{bmatrix} 10 & 0 \\ 0 & 26 \end{bmatrix}$$

    Step 3: Find the Eigenvalues of the Hessian

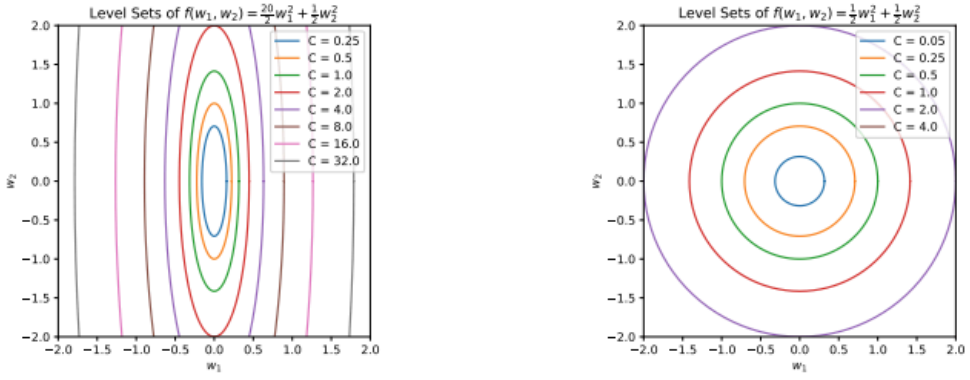
    For a diagonal matrix like this, the eigenvalues are simply the entries on the diagonal: 10 and 26. These eigenvalues represent the curvature of the function along the x- and y-directions, respectively.

    Step 4: Calculate the Condition Number

    The **condition number** (κ) of the Hessian is the ratio of the largest eigenvalue to the smallest eigenvalue:

    $$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}} = \frac{26}{10} = 2.6$$
  - Note that this only works for very simple examples. I attached one below. This is NOT to be taken as the norm (meaning that you can always do this). Just a proof of concept.

- Another way to determine the condition number is to multiply the spectral norm of a matrix multiplied by the spectral norm of the inverse of the matrix
  - Few things to remember here.
  - What is the spectral norm? ||A|| = max ||Ax|| / ||x||. Basically, the 2-norm is a quantity that expresses how much a matrix (which is a transformation) can stretch a vector.
    - When we use "||" here, if it's a vector, then it's just the normal euclidean norm.
  - Now, if you replace the max with a min, then actually min ||Ax|| / ||x|| = 1 / ||A^{-1}|| (where this is actually the spectral norm with max).
  - This definition also better helps put into perspective the idea that the condition number of the hessian reflects how much the original function can stretch or shrink a particular vector
  - This is the reason why when you have ill-conditioned functions, gradient descent ends up being zig-zagged, because one dimension is being pulled way more than the other, and so you end up being poorly aligned, and so you "flip-flop"
  - Condition number = 1 → perfectly conditioned. If it's greater than 1, then it gets worse and worse.
- Now, how does it work?



Main idea of preconditioning: rescale the underlying space we're optimizing over to make the level sets more like circles. To do this for an objective function f, let's imagine solving the modified optimization problem where we minimize $g(u) = f(Ru)$ for some fixed matrix R. Gradient descent for this task looks like

$$u_{t+1} = u_t - \alpha \nabla g(u_t) = u_t - \alpha R^T \nabla f(Ru_t).$$

- Idea is you know you have something that is ill-conditioned, so you try to fix it by considering transforming the entire space (that's what R is doing), and that's what g(u) = f(Ru) is. It's basically trying to fix this ill-conditioned space.

If we multiply both sides by R, and let $w_t = Ru_t$, we get

$$w_{t+1} = Ru_{t+1} = Ru_t - \alpha RR^T \nabla f(Ru_t) = w_t - \alpha RR^T \nabla f(w_t).$$

Effectively, we're just running gradient descent with gradients scaled by some positive semidefinite matrix $P = RR^T$ (why is it positive semidefinite?). This method is called *preconditioned gradient descent*, and we can apply the same idea to precondition SGD.

- So, they take what we did above, multiply it by that same R, and get w_t - alpha * RR^T * gradient. Then, it seems like you ignore w_t, and just focus on the RR^T outside of the gradient. This is what's called a **preconditioner**. Basically, you take any positive semidefinite matrix, and use it as a preconditioner.
- Okay so how can we find a preconditioner or what can we use?
    - Information about the loss that you know already. This means we set the preconditioner to what we want. This is a static choice, so it just works for one function, probably "easy" too (toy examples)
    - Statistics of the dataset can be another preconditioner. Variance of the features in your dataset. Maybe you can have Var(X_i), and then have a diagonal matrix.
    - Information about the second derivative. This is what we will discuss next. You can think of these as "Hessian preconditioners"

Hessian Preconditioners

- Idea is to somehow use the hessian matrix as a preconditioner.

The second-order Taylor expansion of a function $f(x)$ around a point x_t is given by

$$f(x) \approx f(x_t) + \langle \nabla f(x_t), x - x_t \rangle + \frac{1}{2} \langle x - x_t, \nabla^2 f(x_t)(x - x_t) \rangle$$

where $\nabla^2 f(x_t)$ is the Hessian matrix of f at x_t. The minimum of $f(x)$ can be found in closed form by setting the gradient (with respect to x) of the above expression to zero, which gives

$$\nabla f(x_t) + \nabla^2 f(x_t)(x - x_t) = 0 \quad \Longrightarrow \quad x = x_t - [\nabla^2 f(x_t)]^{-1} \nabla f(x_t).$$

So, we find that by moving from first-order to second-order Taylor approximation, and assuming that $\nabla^2 f(x_t)$ is invertible, the natural direction of descent changes from

$$\underbrace{d = -\nabla f(x_t)}_{\substack{\text{using first-order} \\ \text{Taylor approximation}}} \quad \text{to} \quad \underbrace{d = -[\nabla^2 f(x_t)]^{-1} \nabla f(x_t)}_{\text{using second-order Taylor approximation}}.$$

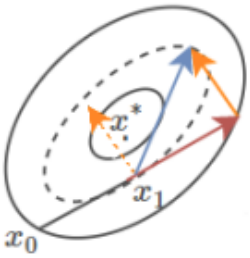
- From the first line to the second line, when they take the gradient, gradient of f(x_t) = 0, gradient of grad f(x_t) dot (x - x_t) is grad f(x_t), and gradient of 1/2 * <x - x_t, hessian f(x_t) * (x - x_t)> is hessian f(x_t) * (x - x_t). That's how they get that term on the left for the second line.
- Then you solve, and so you get x_t - inverse of hessian f(x_t) * grad f(x_t).
- This is probably the most effective way to tackle the conditioning problem. The reason is because the inverse of the hessian matrix will have the inverse eigenvalue across every single dimension, and so you perfectly "scale" the gradients by doing this type of update (as you see in the image above)

- A big issue though is computationally and memory wise - computing a hessian matrix is really difficult when you have millions of parameters. Hessian matrices are order N^2 where N is the number of parameters

Momentum

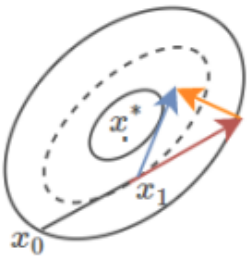
- Momentum is based on a principle we discussed earlier about curvature:
  - Areas of high curvature → let's take smaller steps
  - Areas of lower curvature → let's take larger steps
- This image is really nice to visualize how it works.
- Polyak momentum is actually just an EMA of gradients that have happened up till time stamp t.
  - That is: m_{t + 1} = gamma * m_{t} + (1 - gamma) * gradient of f(x_t), then x_{t + 1} = x_{t} - alpha * m_{t + 1}. It just can be rewritten into what you see below

Polyak's Momentum



$$x_{t+1} = x_t - \alpha \nabla f(x_t) + \mu(x_t - x_{t-1})$$

Nesterov's Momentum



$$x_{t+1} = x_t + \mu(x_t - x_{t-1}) - \gamma \nabla f(x_t + \mu(x_t - x_{t-1}))$$

- Basically, the idea of polyak momentum is to reduce the number of oscillations that happen when we are in areas of high curvature. [This link talks has visuals if you need.]
- Nesterov is just an improvement upon this.
- Polyak is O(1/t) convergence, Nesterov is O(1/t^2) convergence. So, Nesterov is significantly faster.

AdaGrad

The general AdaGrad update rule is given by:

$$x_{t+1} = x_t - \eta G_t^{-1/2} g_t$$

where $G_t^{-1/2}$ is the inverse of the square root of G_t. A simplified version of the update rule takes the diagonal elements of G_t instead of the whole matrix:

$$x_{t+1} = x_t - \eta \text{diag}(G_t)^{-1/2} g_t$$

which can be computed in linear time. In practice, a small quantity ϵ is added to each diagonal element in G_t to avoid singularity problems, the resulting update rule is given by:

$$x_{t+1} = x_t - \eta \text{diag}(\epsilon I + G_t)^{-1/2} g_t$$

where I denotes the identity matrix. An expanded form of the previous update is presented below,

$$\begin{bmatrix} x_{t+1}^{(2)} \\ x_{t+1}^{(2)} \\ \vdots \\ x_{t+1}^{(m)} \end{bmatrix} = \begin{bmatrix} x_t^{(2)} \\ x_t^{(2)} \\ \vdots \\ x_t^{(m)} \end{bmatrix} - \begin{bmatrix} \eta \frac{1}{\sqrt{\epsilon + G_t^{(1,1)}}} \\ \eta \frac{1}{\sqrt{\epsilon + G_t^{(2,2)}}} \\ \vdots \\ \eta \frac{1}{\sqrt{\epsilon + G_t^{(m,m)}}} \end{bmatrix} \odot \begin{bmatrix} g_t^{(2)} \\ g_t^{(2)} \\ \vdots \\ g_t^{(m)} \end{bmatrix}$$

- AdaGrad stands for Adaptive Gradient.
- Here, G_t refers to sum_{tau = 1}^t g_{tau} * g_{tau}^T (so remember G_t) is a large matrix with dimension of N^2 (where N is the dimension of the gradient)
- Now, as you can imagine (and we discussed earlier), this is bad for memory, so we sometimes just take the diagonal of the matrix, and so that's what you see at the very end
    - Also, remember that the inverse of a diagonal matrix is the reciprocal entry in the diagonal. So, everything should make sense here
- Notice how Adagrad is just another preconditioning matrix as we discussed earlier
    - Basically, it's just accumulating past gradients, and then making an assumption that if you have more gradient along one dimension, so then don't apply as much to that one dimension (that's why it's 1 / sqrt of accumulated gradient)
- A key thing to note is that the convergence bound for Adagrad is 1/sqrt(t). Nice way to remember this is that you have a square root of accumulated gradient. 1/sqrt(t) is slower than before though
    - However, idea is that Adagrad works better in ill-conditioned environments while normal gradient descent doesn't really work in those environments

RMSProp

- RMSProp is Adagrad, but you do an EMA approximation for the squared mean of the gradients (which in other words, is also the second moment of the gradients around 0)
- This is what RMSProp looks like:
    - v_t = beta * v_{t - 1} + (1 - beta) * g_t^2
    - theta_{t + 1} = theta_t - learning rate / sqrt(v_t + epsilon) * g_t
- Again, it's the preconditioning idea, but notice it's literally the same thing as Adagrad, but we use an EMA instead of accumulating
- Why RMSProp? What issue does it solve?
    - Adagrad has an issue where the accumulating gradient becomes large, leading to a vanishing gradient update
    - Hence, rather than accumulating, we take the average. This is much more stable.

- In my opinion, it's a very small improvement, but it makes sense why they did it though

ADAM

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

- Okay let's ignore the third and fourth equation. It's basically the same thing as RMSProp, but we include momentum's EMA as well for the gradient update.
- Now, the third and fourth equation is called the bias correction. Look at this below ChatGPT response:

Concretely, if g_i^2 has a mean σ^2, then

$$E[v_t] = E\left[(1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} g_i^2\right] = (1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} E[g_i^2] = \sigma^2 (1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i}.$$

Since $\sum_{i=1}^{t} \beta_2^{t-i} = \sum_{k=0}^{t-1} \beta_2^k = \frac{1-\beta_2^t}{1-\beta_2}$, we get

$$E[v_t] = \sigma^2 (1-\beta_2)\frac{1-\beta_2^t}{1-\beta_2} = \sigma^2 \left(1-\beta_2^t\right).$$

Notice that for small t, $1-\beta_2^t \approx 0$. So $E[v_t]$ is **biased low** in the early steps (it starts at zero and ramps up).

- So, one general thing to note is that EMAs are underestimates for the first/second moment.

Natural Gradient Descent

- NGD is just gradient descent but generalized to different "spaces". We have seen gradient descent being used to optimize functions, but as we'll see in reinforcement learning, gradient descent is also used to optimize probability distributions

$$x_{t+1} = \arg\min_x f(x_t) + \nabla f(x_t)^\top (x - x_t) + D(x, x_t)$$

- For natural gradient descent, D(x, x_t) = ½ * ||x - x_t||, so gradient descent on the euclidean space
    - To see why, take the gradient of f(x_t) + grad f(x_t) * (x - x_t) + ½ ||x - x_t||
    - This is grad f(x_t) + x - x_t = 0. Solving, you get x = x_t - grad f(x_t), so x_{t + 1} = x_t - grad f(x)
    - And, this is the typical gradient descent computation!
    - So, gradient descent is basically the optimal value of the first order approximation of a function

As before, suppose θ_t is a vector of the current parameter values, and we'd like to find a new set of parameters θ_{t+1}. To do so under the natural gradient framework, we solve the following problem.

$$\theta_{t+1} = \arg\min_{\theta} f(\theta_t) + \nabla f(\theta_t)^\top (\theta - \theta_t) + D_{KL}(p(x|\theta) \| p(x|\theta_t)).$$

Unfortunately this minimization is intractable in general. However, we can approximate the KL divergence using a second-order Taylor expansion, which turns out to be the Fisher information matrix F (see Appendix for a derivation). This means that locally around θ_t, we have

$$D_{KL}(p(x|\theta) \| p(x|\theta_t) \approx F.$$

where

$$F = \mathbb{E}_{x \sim p} \left[(\nabla_\theta \log p(x|\theta)) (\nabla_\theta \log p(x|\theta))^\top \right].$$

Thus, the Fisher information matrix contains all the information about the curvature in our likelihood-based loss function. Our update for the natural gradient in this setting is then

$$x_{t+1} = x_t - \gamma F^{-1} \nabla f(\theta_t).$$

- Some other D(x, x_t):
    - Mahalanobis metric → D(x, x_t) = ½ xAx (A is the matrix you pick)
        - This is actually similar to gradient descent with preconditioning matrix
    - Fisher information metric: D(theta, theta_t) = D_{kl}(p(x | theta), p(x | theta_t))
        - So, the distance between two distributions. Theta_t is the previous, theta is the one that we are trying to pick
- For the fisher information metric, the KL divergence between the two distributions is usually approximated as the fisher information of p(x | theta).
    - So, the gradient descent is update is just F^{-1} (where F is the fisher information matrix) as a preconditioner

Fisher information approximates the KL divergence

For notational simplicity, let $D(\theta, \theta_t) = D_{KL}(p_\theta(x)|p_{\theta_t}(x))$. Consider a second-order Taylor approximation to the KL divergence around θ_t,

$$D(\theta, \theta_t) \approx D(\theta_t, \theta_t) + \left(\nabla_\theta D(\theta, \theta_t)\big|_{\theta=\theta_t}\right)^\top (\theta - \theta_t) + (\theta - \theta_t)^\top H_t (\theta - \theta_t)$$

where H_t is the Hessian of $D(\theta_t, \theta_t)$ at θ_t.

The first two terms are zero. The first term is a divergence between two equal distributions, which makes the divergence zero. For the second term, we can see that

$$\begin{aligned}
\nabla_\theta D(\theta, \theta_t) &= \nabla_\theta \mathbb{E}_{p(x|\theta)}\left[\log \frac{p(x|\theta)}{p(x|\theta_t)}\right] \\
&= \mathbb{E}_{p(x|\theta)}\left[\nabla_\theta \log \frac{p(x|\theta)}{p(x|\theta_t)}\right] && (\text{Swap } \nabla \text{ and } \mathbb{E}) \\
&= \mathbb{E}_{p(x|\theta)}\left[\nabla_\theta \log p(x|\theta)\right] && (\text{Grad. doesn't depend on } \theta_t) \\
&= 0.
\end{aligned}$$

The final line comes from the fact that the expectation of the score is always 0.

Tabular TD(0)

- This is for policy evaluation.
  - → meaning I find a policy and I ask "how good is this policy"?
  - → Then, I use TD(0) to get a "metric" to compare.

- You can use policy evaluation to do iteration.

For example:

→ Do Policy evaluation (what is done on the page on the left)

$$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r'} p(s',r'|s,a)\left[r + V(s')\right]$$

Taken from page 102 of the pdf.

→ For each $s \in S$:

$$\pi(s) \leftarrow \arg\max_a \sum_{s',r'} p(s',r'|s,a)(r+\gamma V(s'))$$

If any old $\pi(s) \neq$ new $\pi(s)$, then continue.

- Idea is policy iteration.

$$\pi_0 \rightarrow V_{\pi_0} \rightarrow \pi_1 \rightarrow V_{\pi_1} \rightarrow \cdots \rightarrow \pi_*$$

- Can be done in "one pass" with Value Iteration.

For each $s \in S$:

$$V \leftarrow V(s)$$

$$V(s) \leftarrow \max_a \sum_{s',r'} p(s',r'|s,a)\left[r + \gamma V(s')\right]$$

Then, $\pi(s) = \arg\max_a \sum_{s',r'} p(s',r'|s,a)\left[r + \gamma V(s')\right]$

(at the very end, after $V(s)$ is computed).

"model-dependant"

DP (Classical)

→ Requires knowledge of transition probabilities the model environment (Reward function, State space, action space, etc)

model-free

TD Methods

→ Fully online.
→ No required knowledge of anything.
→ As data comes, estimates improve!
→ Converge faster than ~~DP~~ MC.
→ A_T in V_T converges faster with TD(0)

TD update: $V(s) \leftarrow V(s) + \alpha[R + \gamma V(s') - V(s)]$

MC update: $V(s) \leftarrow V(s) + \frac{1}{N(s)}[G - V(s)]$

Monte Carlo (model free)

→ Similar to TD Methods, but must wait till end of episode.

Called First-Visit MC:

While True:
  Generate an episode $S_0, A_0, R_0 \ldots$
  $G \leftarrow 0$
  ~~For~~ In Iterate $T-1$ to 0:
    $G \leftarrow \gamma G + R_{t+1}$
    If S_t doesn't appear in $S_0 \ldots S_{t-1}$:
      Append G to Returns(S_t)
      $V(S_t) \leftarrow$ Avg(Returns(S_t))

Note that $V_{\pi_2}(s) = E[G_t | S_t = s]$

S_0, basically, you are using MC methods to average estimates of G_t for a particular State S.

$\left[R + \gamma \max_{a'} Q(s', a')\right]$ vs (flip) $R + \gamma Q(s', a')$

SARSA vs Q-learning

→ Just a MC $Q(s,a)$ approximation but with TD methods now.

→ On policy: Behavior policy = target policy. The behavior policy used by both is the ϵ-greedy one, to be a little exploratory.

→ Learns a policy where it must tolerate some level of randomness due to ϵ-greedy exploration. Hence, it is much safer in practice hence, it comes with more "safe" policies (not as risky) compared to Q-learning.

→ Assumes we still use ϵ-greedy even after training

→ Off-policy: behavior policy ≠ target policy.

Target policy → policy that is being used when evaluating the model

Behavior policy → policy that interacts with the environment

Q-learning uses a target policy where you pick $\text{argmax}_a Q(s,a)$.

→ Learns a policy where we assume that after learning from the episodes, we use a policy that is pure exploitation ($\epsilon = 0$) which means always picking $\text{argmax}_a Q(s,a)$.

Policy Gradient Theorem

- $J(\theta) = V_\pi(s_0)$
- Solve: finding the gradient, you get something like:

$$\nabla_\theta J(\theta) = \sum_{s'} \sum_{k=0}^{\infty} Pr(s \to s', k, \pi) \sum_a \nabla\pi(a|x) Q_\pi(s,a)$$

This actually is the expected number of arrivals to s' if you treat this as a Markov Chain.

So, $N(s') = \sum_{k=0} Pr(s \to s', k, \pi)$,

and normalizing: $\mu(s') = \frac{N(s')}{\sum_x N(x)}$ gives a stationary distribution.

- Note that the stationary distribution is just the on average, what is the fraction of visits to a particular state in the MC?

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a \nabla\pi(a|s) Q_\pi(s,a)$$

$$= \sum_s \mu(s) \sum_a Q_\pi(s,a) \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)$$

$$= E_{\mu(s)\pi_\theta(a|s)} \left[Q_\pi(s,a) \nabla_\theta \log \pi_\theta(a|s) \right]$$

- $N(s)$ that we defined is sometimes defined another way: called discounted state visitation

$$\rho_\pi^\gamma(s) = \sum_{t=0}^{\infty} \gamma^t \sum_{s_0} P_0(s_0) p^\pi(s_0 \to s, t, \pi)$$

Normalized discounted state visitation:

$$p_\pi^\gamma(s) = (1-\gamma)\, \underbrace{\rho_\pi^\gamma(s)}$$

$(\gamma < 1)$ previous page.

REINFORCE

- Go back to the policy gradient theorem:

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a Q_\pi(a|s) \nabla_\theta \pi_\theta(a|s)$$

$$= E_\pi \left[\sum_a Q_\pi(a|s) \nabla_\theta \pi_\theta(a|s) \right]$$

- Note that the expectation is technically under the preparation of states visited, guided by the target policy.
- Can use this as the gradient in our gradient ascent algo.
- It's the "all-actions" method

$$E_\pi \left[\sum_a \pi_\theta(a|s_t) Q_\pi(s_t, a) \frac{\nabla \pi_\theta(a|s_t)}{\pi_\theta(a|s_t)} \right]$$

As the previous notes:

$$\nabla J(\theta) = E_{s \sim \mu(s), \pi_\theta(a|s)} \left[Q_\pi(S_t, A_t) \nabla_\theta \log \pi_\theta(A_t|S_t) \right]$$

So, you sample $S_t \sim \mu(\cdot)$, $A_t \sim \pi_\theta(\cdot|S_t)$.

Then,

$$\theta_{t+1} = \theta_t + \alpha G_t \nabla_\theta \log \pi_\theta(A_t|s_t)$$

Note that $G_t = Q_\pi(s_t, A_t)$ here.

Reminder:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- REINFORCE has high variance, low bias due to the MC nature of the algorithm.
  → Low bias is due to the fact that we use G_t and directly compute it. We don't bootstrap.
  Note that $E(G_t | s = S_t, a = A_t) = Q_\pi(s, a)$.
  So, G_t is an unbiased estimator of $Q_\pi(s,a)$. But, it has high variance!
- To reduce variance, we introduce a baseline function. This is called the advantage function,

$$A^\pi(S_t, A_t) = G_t - \underbrace{V_w(S_t)}$$

Value function, parameterized by w.

Actor Critic

$$\theta_{t+1} = \theta_t + \alpha \underbrace{\left(G_t - \hat{V}_w(S_t)\right)} \nabla_\theta \log \pi_\theta(A_t|S_t)$$

$$\downarrow$$

$$\left(R_{t+1} + \gamma \hat{V}_w(S_{t+1}) - \hat{V}(S_t)\right)$$

∴ This is Actor-Critic (One Step).

$$S_0, \quad \delta = R + V_w(S') - V_w(S)$$

Then, $w = w + \alpha^w \delta \nabla V_w(S)$

$\theta = \theta + \alpha^\theta I \delta \nabla \ln \pi_\theta(A|S)$

$I = \gamma I$

These are the A2C updates.

Convergence

In distribution

If you have CDFs $F_1, F_2, \ldots F_n$

then

$$\lim_{n \to \infty} F_n(x) = F(x) \text{ is convergence in distribution,}$$

This means that X_1 follows F_1, X_2 follows F_2, and so on.
Written as $X_n \xrightarrow{D} X$.
Note that convergence of the CDFs does not imply convergence of the PDFs.

Continuous Mapping Theorem

If $X_n \xrightarrow{D} X$, then $g(X_n) \xrightarrow{D} g(x)$.

In probability

$X_n \xrightarrow{P} X$ if $\forall \epsilon > 0$, $\lim_{n \to \infty} \mathbb{P}(|X_n - X| > \epsilon) = 0$.

This applies to this too.

Almost sure convergence

$X_n \xrightarrow{a.s.} X$ if $\mathbb{P}\left(\lim_{n \to \infty} X_n = X\right) = 1$

Common usecases

Law of Large Numbers

Strong: $\overline{X_n} \xrightarrow{a.s.} \mu$ as $n \to \infty$.

Weak: $\overline{X_n} \xrightarrow{P} \mu$ as $n \to \infty$.

Central Limit Theorem

For i.i.d $X_1 \dots X_n$ with $E(X_i) = \mu$ and $Var(X_i) = \sigma^2$

$$\frac{\overline{X_n} - \mu}{\frac{\sigma}{\sqrt{n}}} \xrightarrow{D} N(0,1)$$

(can be applied when $n \geq 30$)

Asymptotic Normality (Unbiased)

$$\sqrt{n}(\hat{\theta} - \theta) \longrightarrow \text{normal distribution.}$$

In other words,

$$\hat{\theta}_N \xrightarrow{D} N(\theta_0, \underbrace{I_N(\theta_0)^{-1}}_{\text{The Fischer Information matrix.}})$$

Where, $I_N(\theta_0) = N\ I(\theta_0)$.

Consistency of an estimator

With $n \to \infty$, $X_1 \dots X_n$, then

$$\hat{\theta}_n \to \theta$$

Lecture 14 — Consistency and asymptotic normality of the MLE

14.1 Consistency and asymptotic normality

We showed last lecture that given data $X_1, \ldots, X_n \overset{IID}{\sim} \text{Poisson}(\lambda)$, the maximum likelihood estimator for λ is simply $\hat{\lambda} = \bar{X}$. How accurate is $\hat{\lambda}$ for λ? Recall from Lecture 12 the following computations:

$$\mathbb{E}_\lambda[\bar{X}] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[X_i] = \lambda,$$

$$\text{Var}_\lambda[\bar{X}] = \frac{1}{n^2}\sum_{i=1}^{n} \text{Var}[X_i] = \frac{\lambda}{n}.$$

So $\hat{\lambda}$ is unbiased, with variance λ/n.

When n is large, asymptotic theory provides us with a more complete picture of the "accuracy" of $\hat{\lambda}$: By the Law of Large Numbers, $\bar{X}$ converges to λ in probability as $n \to \infty$. Furthermore, by the Central Limit Theorem,

$$\sqrt{n}(\bar{X} - \lambda) \to \mathcal{N}(0, \text{Var}[X_i]) = \mathcal{N}(0, \lambda)$$

in distribution as $n \to \infty$. So for large n, we expect $\hat{\lambda}$ to be close to λ, and the sampling distribution of $\hat{\lambda}$ is approximately $\mathcal{N}(\lambda, \frac{\lambda}{n})$. This normal approximation is useful for many reasons—for example, it allows us to understand other measures of error (such as $\mathbb{E}[|\hat{\lambda} - \lambda|]$ or $\mathbb{P}[|\hat{\lambda} - \lambda| > 0.01]$), and (later in the course) will allow us to obtain a confidence interval for $\hat{\lambda}$.

In a parametric model, we say that an estimator $\hat{\theta}$ based on $X_1, \ldots, X_n$ is **consistent** if $\hat{\theta} \to \theta$ in probability as $n \to \infty$. We say that it is **asymptotically normal** if $\sqrt{n}(\hat{\theta} - \theta)$ converges in distribution to a normal distribution (or a multivariate normal distribution, if θ has more than 1 parameter). So $\hat{\lambda}$ above is consistent and asymptotically normal.

The goal of this lecture is to explain why, rather than being a curiosity of this Poisson example, consistency and asymptotic normality of the MLE hold quite generally for many "typical" parametric models, and there is a general formula for its asymptotic variance. The following is one statement of such a result:

Theorem 14.1. *Let $\{f(x|\theta) : \theta \in \Omega\}$ be a parametric model, where $\theta \in \mathbb{R}$ is a single parameter. Let $X_1, \ldots, X_n \overset{IID}{\sim} f(x|\theta_0)$ for $\theta_0 \in \Omega$, and let $\hat{\theta}$ be the MLE based on $X_1, \ldots, X_n$. Suppose certain regularity conditions hold, including:*[1]

[1] Some technical conditions in addition to the ones stated are required to make this theorem rigorously true; these additional conditions will hold for the examples we discuss, and we won't worry about them in this class.

- *All PDFs/PMFs $f(x|\theta)$ in the model have the same support,*
- *θ_0 is an interior point (i.e., not on the boundary) of Ω,*
- *The log-likelihood $l(\theta)$ is differentiable in θ, and*
- *$\hat{\theta}$ is the unique value of $\theta \in \Omega$ that solves the equation $0 = l'(\theta)$.*

Then $\hat{\theta}$ is consistent and asymptotically normal, with $\sqrt{n}(\hat{\theta}-\theta_0) \to \mathcal{N}(0, \frac{1}{I(\theta_0)})$ in distribution. Here, $I(\theta)$ is defined by the two equivalent expressions

$$I(\theta) := \mathrm{Var}_\theta[z(X,\theta)] = -\mathbb{E}_\theta[z'(X,\theta)],$$

where Var_θ and $\mathbb{E}_\theta$ denote variance and expectation with respect to $X \sim f(x|\theta)$, and

$$z(x,\theta) = \frac{\partial}{\partial\theta}\log f(x|\theta), \qquad z'(x,\theta) = \frac{\partial^2}{\partial\theta^2}\log f(x|\theta).$$

$z(x,\theta)$ is called the **score function**, and $I(\theta)$ is called the **Fisher information**. Heuristically for large n, the above theorem tells us the following about the MLE $\hat{\theta}$:

- $\hat{\theta}$ is *asymptotically unbiased*. More precisely, the bias of $\hat{\theta}$ is less than order $1/\sqrt{n}$. (Otherwise $\sqrt{n}(\hat{\theta}-\theta_0)$ should not converge to a distribution with mean 0.)
- The variance of $\hat{\theta}$ is approximately $\frac{1}{nI(\theta_0)}$. In particular, the standard error is of order $1/\sqrt{n}$, and the variance (rather than the squared bias) is the main contributing factor to the mean-squared-error of $\hat{\theta}$.
- If the true parameter is θ_0, the sampling distribution of $\hat{\theta}$ is approximately $\mathcal{N}(\theta_0, \frac{1}{nI(\theta_0)})$.

Example 14.2. Let's verify that this theorem is correct for the above Poisson example. There,

$$\log f(x|\lambda) = \log\frac{\lambda^x e^{-\lambda}}{x!} = x\log\lambda - \lambda - \log(x!),$$

so the score function and its derivative are given by

$$z(x,\lambda) = \frac{\partial}{\partial\lambda}\log f(x|\lambda) = \frac{x}{\lambda} - 1, \qquad z'(x,\lambda) = \frac{\partial^2}{\partial\lambda^2}\log f(x|\lambda) = -\frac{x}{\lambda^2}.$$

We may compute the Fisher information as

$$I(\lambda) = -\mathbb{E}_\lambda[z'(X,\lambda)] = \mathbb{E}_\lambda\left[\frac{X}{\lambda^2}\right] = \frac{1}{\lambda},$$

so $\sqrt{n}(\hat{\lambda}-\lambda) \to \mathcal{N}(0,\lambda)$ in distribution. This is the same result as what we obtained using a direct application of the CLT.

14.2 Proof sketch

We'll sketch heuristically the proof of Theorem 14.1, assuming $f(x|\theta)$ is the PDF of a continuous distribution. (The discrete case is analogous with integrals replaced by sums.)

To see why the MLE $\hat{\theta}$ is consistent, note that $\hat{\theta}$ is the value of θ which maximizes

$$\frac{1}{n} l(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log f(X_i|\theta).$$

Suppose the true parameter is θ_0, i.e. $X_1, \dots, X_n \overset{IID}{\sim} f(x|\theta_0)$. Then for any $\theta \in \Omega$ (not necessarily θ_0), the Law of Large Numbers implies the convergence in probability

$$\frac{1}{n} \sum_{i=1}^{n} \log f(X_i|\theta) \to \mathbb{E}_{\theta_0}[\log f(X|\theta)]. \tag{14.1}$$

Under suitable regularity conditions, this implies that the value of θ maximizing the left side, which is $\hat{\theta}$, converges in probability to the value of θ maximizing the right side, which we claim is θ_0. Indeed, for any $\theta \in \Omega$,

$$\mathbb{E}_{\theta_0}[\log f(X|\theta)] - \mathbb{E}_{\theta_0}[\log f(X|\theta_0)] = \mathbb{E}_{\theta_0}\left[\log \frac{f(X|\theta)}{f(X|\theta_0)}\right].$$

Noting that $x \mapsto \log x$ is concave, Jensen's inequality implies $\mathbb{E}[\log X] \leq \log \mathbb{E}[X]$ for any positive random variable X, so

$$\mathbb{E}_{\theta_0}\left[\log \frac{f(X|\theta)}{f(X|\theta_0)}\right] \leq \log \mathbb{E}_{\theta_0}\left[\frac{f(X|\theta)}{f(X|\theta_0)}\right] = \log \int \frac{f(x|\theta)}{f(x|\theta_0)} f(x|\theta_0) dx = \log \int f(x|\theta) dx = 0.$$

So $\theta \mapsto \mathbb{E}_{\theta_0}[\log f(X|\theta)]$ is maximized at $\theta = \theta_0$, which establishes consistency of $\hat{\theta}$.

To show asymptotic normality, we first compute the mean and variance of the score:

Lemma 14.1 (Properties of the score)**.** *For $\theta \in \Omega$,*

$$\mathbb{E}_\theta[z(X,\theta)] = 0, \qquad \mathrm{Var}_\theta[z(X,\theta)] = -\mathbb{E}[z'(X,\theta)].$$

Proof. By the chain rule of differentiation,

$$z(x,\theta) f(x|\theta) = \left(\frac{\partial}{\partial\theta} \log f(x|\theta)\right) f(x|\theta) = \frac{\frac{\partial}{\partial\theta} f(x|\theta)}{f(x|\theta)} f(x|\theta) = \frac{\partial}{\partial\theta} f(x|\theta). \tag{14.2}$$

Then, since $\int f(x|\theta) dx = 1$,

$$\mathbb{E}_\theta[z(X,\theta)] = \int z(x,\theta) f(x|\theta) dx = \int \frac{\partial}{\partial\theta} f(x|\theta) dx = \frac{\partial}{\partial\theta} \int f(x|\theta) dx = 0.$$

Next, we differentiate this identity with respect to θ:

$$\begin{aligned}
0 &= \frac{\partial}{\partial\theta}\mathbb{E}_\theta[z(X,\theta)] \\
&= \frac{\partial}{\partial\theta}\int z(x,\theta)f(x|\theta)dx \\
&= \int \left(z'(x,\theta)f(x|\theta) + z(x,\theta)\left(\frac{\partial}{\partial\theta}f(x|\theta)\right)\right)dx \\
&= \int \Big(z'(x,\theta)f(x|\theta) + z(x,\theta)^2 f(x|\theta)\Big)dx \\
&= \mathbb{E}_\theta[z'(X,\theta)] + \mathbb{E}_\theta[z(X,\theta)^2] \\
&= \mathbb{E}_\theta[z'(X,\theta)] + \mathrm{Var}_\theta[z(X,\theta)],
\end{aligned}$$

where the fourth line above applies (14.2) and the last line uses $\mathbb{E}_\theta[z(X,\theta)] = 0$. □

Since $\hat{\theta}$ maximizes $l(\theta)$, we must have $0 = l'(\hat{\theta})$. Consistency of $\hat{\theta}$ ensures that (when n is large) $\hat{\theta}$ is close to θ_0 with high probability. This allows us to apply a first-order Taylor expansion to the equation $0 = l'(\hat{\theta})$ around $\hat{\theta} = \theta_0$:

$$0 \approx l'(\theta_0) + (\hat{\theta} - \theta_0)l''(\theta_0),$$

so

$$\sqrt{n}(\hat{\theta} - \theta_0) \approx -\sqrt{n}\frac{l'(\theta_0)}{l''(\theta_0)} = -\frac{\frac{1}{\sqrt{n}}l'(\theta_0)}{\frac{1}{n}l''(\theta_0)}. \tag{14.3}$$

For the denominator, by the Law of Large Numbers,

$$\frac{1}{n}l''(\theta_0) = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial^2}{\partial\theta^2}\Big[\log f(X_i|\theta)\Big]_{\theta=\theta_0} = \frac{1}{n}\sum_{i=1}^{n} z'(X_i,\theta_0) \to \mathbb{E}_{\theta_0}[z'(X,\theta_0)] = -I(\theta_0)$$

in probability. For the numerator, recall by Lemma 14.1 that $z(X,\theta_0)$ has mean 0 and variance $I(\theta_0)$ when $X \sim f(x|\theta_0)$. Then by the Central Limit Theorem,

$$\frac{1}{\sqrt{n}}l'(\theta_0) = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}\frac{\partial}{\partial\theta}\Big[\log f(X_i|\theta)\Big]_{\theta=\theta_0} = \frac{1}{\sqrt{n}}\sum_{i=1}^{n} z(X_i,\theta_0) \to \mathcal{N}(0, I(\theta_0))$$

in distribution. Applying these conclusions, the Continuous Mapping Theorem, and Slutsky's Lemma[2] to (14.3),

$$\sqrt{n}(\hat{\theta}_n - \theta_0) \to \frac{1}{I(\theta_0)}\mathcal{N}(0, I(\theta_0)) = \mathcal{N}(0, I(\theta_0)^{-1}),$$

as desired.

[2]Slutsky's Lemma says: If $X_n \to c$ in probability and $Y_n \to Y$ in distribution, then $X_nY_n \to cY$ in distribution.

Lecture 15 — Fisher information and the Cramer-Rao bound

15.1 Fisher information for one or more parameters

For a parametric model $\{f(x|\theta) : \theta \in \Omega\}$ where $\theta \in \mathbb{R}$ is a single parameter, we showed last lecture that the MLE $\hat{\theta}_n$ based on $X_1, \ldots, X_n \overset{IID}{\sim} f(x|\theta)$ is, under certain regularity conditions, asymptotically normal:

$$\sqrt{n}(\hat{\theta}_n - \theta) \to \mathcal{N}\left(0, \frac{1}{I(\theta)}\right)$$

in distribution as $n \to \infty$, where

$$I(\theta) := \mathrm{Var}_\theta\left[\frac{\partial}{\partial\theta}\log f(X|\theta)\right] = -\mathbb{E}_\theta\left[\frac{\partial^2}{\partial\theta^2}\log f(X|\theta)\right]$$

is the **Fisher information**. As an application of this result, let us study the sampling distribution of the MLE in a one-parameter Gamma model:

Example 15.1. Let $X_1, \ldots, X_n \overset{IID}{\sim} \mathrm{Gamma}(\alpha, 1)$. (For this example, we are assuming that we know $\beta = 1$ and only need to estimate α.) Then

$$\log f(x|\alpha) = \log\frac{1}{\Gamma(\alpha)}x^{\alpha-1}e^{-x} = -\log\Gamma(\alpha) + (\alpha-1)\log x - x.$$

The log-likelihood of all observations is then

$$\begin{aligned} l(\alpha) &= \sum_{i=1}^n \left(-\log\Gamma(\alpha) + (\alpha-1)\log X_i - X_i\right) \\ &= -n\log\Gamma(\alpha) + (\alpha-1)\sum_{i=1}^n \log X_i - \sum_{i=1}^n X_i. \end{aligned}$$

Introducing the digamma function $\psi(\alpha) = \frac{\Gamma'(\alpha)}{\Gamma(\alpha)}$, the MLE $\hat{\alpha}$ is obtained by (numerically) solving

$$0 = l'(\alpha) = -n\psi(\alpha) + \sum_{i=1}^n \log X_i.$$

What is the sampling distribution of $\hat{\alpha}$? We compute

$$\frac{\partial^2}{\partial\alpha^2}\log f(x|\alpha) = -\psi'(\alpha).$$

As this does not depend on x, the Fisher information is $I(\alpha) = -\mathbb{E}_\alpha[-\psi'(\alpha)] = \psi'(\alpha)$. Then for large n, $\hat{\alpha}$ is distributed approximately as $\mathcal{N}(\alpha, \frac{1}{n\psi'(\alpha)})$.

Asymptotic normality of the MLE extends naturally to the setting of multiple parameters:

Theorem 15.2. *Let $\{f(x|\theta) : \theta \in \Omega\}$ be a parametric model, where $\theta \in \mathbb{R}^k$ has k parameters. Let $X_1, \ldots, X_n \overset{IID}{\sim} f(x|\theta)$ for $\theta \in \Omega$, and let $\hat{\theta}_n$ be the MLE based on $X_1, \ldots, X_n$. Define the* **Fisher information matrix** *$I(\theta) \in \mathbb{R}^{k \times k}$ as the matrix whose (i, j) entry is given by the equivalent expressions*

$$I(\theta)_{ij} = \mathrm{Cov}_\theta \left[\frac{\partial}{\partial \theta_i} \log f(X|\theta), \frac{\partial}{\partial \theta_j} \log f(X|\theta) \right] = -\mathbb{E}_\theta \left[\frac{\partial^2}{\partial \theta_i \partial \theta_j} \log f(X|\theta) \right]. \tag{15.1}$$

Then under the same conditions as Theorem 14.1,

$$\sqrt{n}(\hat{\theta}_n - \theta) \to \mathcal{N}(0, I(\theta)^{-1}),$$

where $I(\theta)^{-1}$ is the $k \times k$ matrix inverse of $I(\theta)$ (and the distribution on the right is the multivariate normal distribution having this covariance).

(For $k = 1$, this definition of $I(\theta)$ is exactly the same as our previous definition, and $I(\theta)^{-1}$ is just $\frac{1}{I(\theta)}$. The proof of the above result is analogous to the $k = 1$ case from last lecture, employing a multivariate Taylor expansion of the equation $0 = \nabla l(\hat{\theta})$ around $\hat{\theta} = \theta_0$.)

Example 15.3. Consider now the full Gamma model, $X_1, \ldots, X_n \overset{IID}{\sim} \mathrm{Gamma}(\alpha, \beta)$. Numerical computation of the MLEs $\hat{\alpha}$ and $\hat{\beta}$ in this model was discussed in Lecture 13. To approximate their sampling distributions, note

$$\log f(x|\alpha, \beta) = \log \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} = \alpha \log \beta - \log \Gamma(\alpha) + (\alpha - 1) \log x - \beta x,$$

so

$$\frac{\partial^2}{\partial \alpha^2} \log f(x|\alpha, \beta) = -\psi'(\alpha), \quad \frac{\partial^2}{\partial \alpha \partial \beta} \log f(x|\alpha, \beta) = \frac{1}{\beta}, \quad \frac{\partial^2}{\partial \beta^2} \log f(x|\alpha, \beta) = -\frac{\alpha}{\beta^2}.$$

These partial derivatives again do not depend on x, so the Fisher information matrix is

$$I(\alpha, \beta) = \begin{pmatrix} \psi'(\alpha) & -\frac{1}{\beta} \\ -\frac{1}{\beta} & \frac{\alpha}{\beta^2} \end{pmatrix},$$

and its inverse is

$$I(\alpha, \beta)^{-1} = \frac{1}{\psi'(\alpha)\frac{\alpha}{\beta^2} - \frac{1}{\beta^2}} \begin{pmatrix} \frac{\alpha}{\beta^2} & \frac{1}{\beta} \\ \frac{1}{\beta} & \psi'(\alpha) \end{pmatrix}.$$

$(\hat{\alpha}, \hat{\beta})$ is approximately distributed as the bivariate normal distribution $\mathcal{N}\left((\alpha, \beta), \frac{1}{n} I(\alpha, \beta)^{-1}\right)$. In particular, the marginal distribution of $\hat{\alpha}$ is approximately

$$\mathcal{N}\left(\alpha, \frac{1}{n(\psi'(\alpha)\frac{\alpha}{\beta^2} - \frac{1}{\beta^2})} \frac{\alpha}{\beta^2}\right).$$

Suppose, in this example, that in fact the true parameter $\beta = 1$. Then the variance of $\hat{\alpha}$ reduces to $\frac{1}{n(\psi'(\alpha) - 1/\alpha)}$, which is *not* the variance $\frac{1}{n\psi'(\alpha)}$ obtained in Example 15.1—the variance here is larger. The difference is that in this example, we do not assume that we know $\beta = 1$ and instead are estimating β by its MLE $\hat{\beta}$. As a result, the MLEs of α in these two examples are not the same, and here our uncertainty about β is also increasing the variability of our estimate of α.

More generally, for any 2×2 Fisher information matrix

$$I = \begin{pmatrix} a & b \\ b & c \end{pmatrix},$$

the first definition of equation (15.1) implies that $a, c \geq 0$. The upper-left element of I^{-1} is $\frac{1}{a - b^2/c}$, which is always at least $\frac{1}{a}$. This implies, for any model with a single parameter θ_1 that is contained inside a larger model with parameters (θ_1, θ_2), that the variability of the MLE for θ_1 in the larger model is always at least that of the MLE for θ_1 in the smaller model; they are equal when the off-diagonal entry b is equal to 0. The same observation is true for any number of parameters $k \geq 2$ in the larger model.

This is a simple example of a trade-off between model complexity and accuracy of estimation, which is fundamental to many areas of statistics and machine learning: a complex model with more parameters might better capture the true distribution of data, but these parameters will also be more difficult to estimate than those in a simpler model.

15.2 The Cramer-Rao lower bound

Let's return to the setting of a single parameter $\theta \in \mathbb{R}$. Why is the Fisher information $I(\theta)$ called "information", and why should we choose to estimate θ by the MLE $\hat{\theta}$?

If $X_1, \ldots, X_n \overset{IID}{\sim} f(x|\theta_0)$ for a true parameter θ_0, and $l(\theta) = \sum_{i=1}^n \log f(X_i|\theta)$ is the log-likelihood function, then

$$I(\theta_0) = -\mathbb{E}_{\theta_0}\left[\frac{\partial^2}{\partial \theta^2}\Big[\log f(X|\theta)\Big]_{\theta=\theta_0}\right] = -\frac{1}{n}\mathbb{E}_{\theta_0}[l''(\theta_0)].$$

$I(\theta_0)$ measures the expected curvature of the log-likelihood function $l(\theta)$ around the true parameter $\theta = \theta_0$. If $l(\theta)$ is sharply curved around θ_0—in other words, $I(\theta_0)$ is large—then a small change in θ can lead to a large decrease in the log-likelihood $l(\theta)$, and hence the data provides a lot of "information" that the true value of θ is close to θ_0. Conversely, if $I(\theta_0)$ is small, then a small change in θ does not affect $l(\theta)$ by much, and the data provides less information about θ. In this (heuristic) sense, $I(\theta_0)$ quantifies the amount of information that each observation X_i contains about the unknown parameter.

The Fisher information $I(\theta)$ is an intrinsic property of the model $\{f(x|\theta) : \theta \in \Omega\}$, not of any specific estimator. (We've shown that it is related to the variance of the MLE, but its definition does not involve the MLE.) There are various information-theoretic results stating that $I(\theta)$ describes a fundamental limit to how accurate *any* estimator of θ based on $X_1, \ldots, X_n$ can be. We'll prove one such result, called the **Cramer-Rao lower bound**:

Theorem 15.4. *Consider a parametric model $\{f(x|\theta) : \theta \in \Omega\}$ (satisfying certain mild regularity assumptions) where $\theta \in \mathbb{R}$ is a single parameter. Let T be any unbiased estimator of θ based on data $X_1, \ldots, X_n \overset{IID}{\sim} f(x|\theta)$. Then*

$$\mathrm{Var}_\theta[T] \geq \frac{1}{nI(\theta)}.$$

Proof. Recall the score function

$$z(x,\theta) = \frac{\partial}{\partial\theta}\log f(x|\theta) = \frac{\frac{\partial}{\partial\theta}f(x|\theta)}{f(x|\theta)},$$

and let $Z := Z(X_1,\ldots,X_n,\theta) = \sum_{i=1}^n z(X_i,\theta)$. By the definition of correlation and the fact that the correlation of two random variables is always between -1 and 1,

$$\mathrm{Cov}_\theta[Z,T]^2 \le \mathrm{Var}_\theta[Z]\times\mathrm{Var}_\theta[T].$$

The random variables $z(X_1,\theta),\ldots,z(X_n,\theta)$ are IID, and by Lemma 14.1, they have mean 0 and variance $I(\theta)$. Then

$$\mathrm{Var}_\theta[Z] = n\,\mathrm{Var}_\theta[z(X_1,\theta)] = nI(\theta).$$

Since T is unbiased,

$$\theta = \mathbb{E}_\theta[T] = \int_{\mathbb{R}^n} T(x_1,\ldots,x_n)f(x_1|\theta)\times\ldots\times f(x_n|\theta)dx_1\ldots dx_n.$$

Differentiating both sides with respect to θ and applying the product rule of differentiation,

$$\begin{aligned}
1 &= \int_{\mathbb{R}^n} T(x_1,\ldots,x_n)\bigg(\frac{\partial}{\partial\theta}f(x_1|\theta)\times f(x_2|\theta)\times\ldots\times f(x_n|\theta)\\
&\qquad + f(x_1|\theta)\times\frac{\partial}{\partial\theta}f(x_2|\theta)\times\ldots\times f(x_n|\theta)+\ldots\\
&\qquad + f(x_1|\theta)\times f(x_2|\theta)\times\ldots\times\frac{\partial}{\partial\theta}f(x_n|\theta)\bigg)dx_1\ldots dx_n\\
&= \int_{\mathbb{R}^n} T(x_1,\ldots,x_n)Z(x_1,\ldots,x_n,\theta)f(x_1|\theta)\times\ldots\times f(x_n|\theta)dx_1\ldots dx_n\\
&= \mathbb{E}_\theta[TZ].
\end{aligned}$$

Since $\mathbb{E}_\theta[Z]=0$, this implies $\mathrm{Cov}_\theta[T,Z] = \mathbb{E}_\theta[TZ] = 1$, so $\mathrm{Var}_\theta[T] \ge \frac{1}{nI(\theta)}$ as desired. □

For two unbiased estimators of θ, the ratio of their variances is called their **relative efficiency**. An unbiased estimator is **efficient** if its variance equals the lower bound $\frac{1}{nI(\theta)}$. Since the MLE achieves this lower bound asymptotically, we say it is **asymptotically efficient**.

The Cramer-Rao bound ensures that no unbiased estimator can achieve asymptotically lower variance than the MLE. Stronger results, which we will not prove in this class, in fact show that no estimator, biased or unbiased, can asymptotically achieve lower mean-squared-error than $\frac{1}{nI(\theta)}$, except possibly on a small set of special values $\theta\in\Omega$.[1] In particular, when the method-of-moments estimator differs from the MLE, we expect it to have higher mean-squared-error than the MLE for large n, which explains why the MLE is usually the preferred estimator in simple parametric models.

[1]For example, the constant estimator $\hat\theta = c$ for fixed $c\in\Omega$ achieves 0 mean-squared-error if the true parameter happened to be the special value c, but at all other parameter values is worse than the MLE for sufficiently large n.