Math and machine learning primer

A1.1: Implement MNIST classification (5 points)

Implement the MNIST classification example from the lecture slides. Make sure you get the correct deep CNN model architecture from Lecture 2 (p.18).

(a) *[2 points]* Train the model $\boldsymbol{f}_\theta : \mathbb{R}^{28\times 28} \to \mathbb{R}^{10}$ from the lecture slides with a cross-entropy loss for 10 epochs. Plot the average train/test losses during each epoch (y-axis) over all epochs (x-axis). Do the same with the average train/test accuracies during each epoch. Try to program as modular as possible, as you will re-use the code later.

(b) *[1 point]* Change your optimization criterion to a mean-squared-error loss between the same model architecture $\boldsymbol{f}_\theta : \mathbb{R}^{28\times 28} \to \mathbb{R}^{10}$ you used in (a) and a one-hot encoding ($\boldsymbol{h}_i \in \mathbb{R}^{10}, h_{ij} = 1$ iff $j = y_i$, otherwise $h_{ij} = 0$) of the labels y_i:

$$\mathcal{L} \quad := \quad \tfrac{1}{n}\sum_{i=1}^{n}\Big(\boldsymbol{f}_\theta(\boldsymbol{x}_i) - \boldsymbol{h}_i\Big)^2$$

Plot the same plots as in (a). Try to reuse as much of your old code as possible, e.g., by defining the criterion (which is now different) as external functions that can be overwritten.

(c) *[1 point]* Define a new architecture $f'_\theta : \mathbb{R}^{28\times 28} \to \mathbb{R}$, that is exactly the same as above, but with only one output neuron instead of 10. Train it with a regression mean-squared error loss between the model output and the scalar class identifier.

$$\mathcal{L}' \quad := \quad \tfrac{1}{n}\sum_{i=1}^{n}\Big(f'_\theta(\boldsymbol{x}_i) - y_i\Big)^2$$

Plot the same plots as in (a), but for 50 epochs.

(d) *[1 point]* Learning in (c) should be significantly slower, in terms of accuracy gain per epoch, than in (a) and (b). Use a transformation of your model output (which can be implemented in the functions that compute the criterion and the accuracy, or as an extra module) as $f''_\theta(\boldsymbol{x}_i) := \alpha f'_\theta(\boldsymbol{x}_i) + \beta$, with $\alpha = \beta = 4.5$. Plot the same plots as in (c). Does the learning behavior change? Why?

Bonus-question: Can you come up with an alternative approach to (d) that has the same speed-up effect?

Hint: Evaluate your test loss and accuracy before every training to make sure the accuracy is defined correctly (should be around 0.1 for a model without training). This means that you will always have one test measurement more.

Hint: Try to reuse as much of your old code as possible, e.g., by defining the criterion and the accuracy (which will change for some question) as external functions that can be overwritten later.

Solution A sample implementation can be found in the accompanying Jupyter Notebook.

Bonus-question: Interestingly, a larger learning rate does not help, as the RMSProp seems to automatically compensate for it. However, transforming the output is mathematically equivalent to scaling the last model layer. Both the linear weights `model[-1].weight *= 4.5` and the bias `model[-1].bias[0] = 4.5` of the `torch.nn.Linear` layer need to be adjusted. Note that

PyTorch does not allow you to modify parameters in-place, as this can break auto-differentiation when done during a forward pass. The context `torch.no_grad()` allows to circumvent this security feature. This context is generally helpful when no gradient computation is necessary, as it frees all intermediately computed tensors and can save a lot of memory usage when used correctly.

A1.2: Mean and variance of online estimates **(3 points)**

Let $\{y_t\}_{t=1}^{\infty}$ an infinite training set drawn i.i.d. from the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. At time t, the online estimate f_t of the average over the training set, starting at f_0, is defined as

$$f_t \quad = \quad f_{t-1} \; + \; \alpha \, (y_t - f_{t-1}) \, , \qquad\qquad 0 < \alpha < 1 \, .$$

(a) *[1 point]* Show that for small t the online estimate is *biased*, i.e., $\mathbb{E}[f_t] \neq \mu$.

(b) *[1 point]* Prove that in the limit $t \to \infty$ the online estimate is unbiased, i.e., $\mathbb{E}[f_t] = \mu$.

(c) *[1 point]* Prove that in the limit $t \to \infty$ the variance of the online estimate is $\mathbb{E}[f_t^2] - \mathbb{E}[f_t]^2 = \frac{\alpha \, \sigma^2}{2-\alpha}$.

Hint: You can use the geometric series $\sum\limits_{k=0}^{t-1} r^k = \frac{1-r^t}{1-r}, \; \forall |r| < 1$.

Bonus-question: Prove that for the decaying learning rate $\alpha_t = \frac{\alpha}{1-(1-\alpha)^t}$ holds $\lim\limits_{\alpha \to 0} \alpha_t = \frac{1}{t}$.

Hint: You can also use the binomial identity $(x+y)^t = \sum_{k=0}^{t} \binom{t}{k} x^k \, y^{t-k}$.

Solution

(a) Note that by recursion $f_t = (1-\alpha)^t \, f_0 + \sum_{i=0}^{t-1} \alpha \, (1-\alpha)^i \, y_{t-i}$.

$$\mathbb{E}[f_t] \quad = \quad (1-\alpha)^t \, f_0 \; + \; \alpha \underbrace{\sum_{i=0}^{t-1} (1-\alpha)^i}_{\frac{1-(1-\alpha)^t}{1-(1-\alpha)}} \underbrace{\mathbb{E}[y_{t-i}]}_{\mu} \quad = \quad \mu \; + \; (1-\alpha)^t (f_0 - \mu) \quad \neq \quad \mu \, .$$

(b) In limit of (a), the term $(1-\alpha)^t$ goes against 0.

$$\lim_{t \to \infty} \mathbb{E}[f_t] \quad = \quad \mu \; + \; \underbrace{\lim_{t \to \infty} (1-\alpha)^t (f_0 - \mu)}_{0} \quad = \quad \mu \, .$$

(c) Note that due to the assumption of i.i.d. sampling $\mathbb{E}[y_i y_j] = \mu^2 + \sigma^2 \delta_{ij}$.

$$
\begin{aligned}
\lim_{t\to\infty} \mathbb{E}[f_t^2] &= \lim_{t\to\infty} \mathbb{E}\Big[\Big((1-\alpha)^t f_0 + \sum_{i=0}^{t-1} \alpha(1-\alpha)^i y_{t-i}\Big)^2\Big] \\
&= \lim_{t\to\infty} \mathbb{E}\Big[\Big((1-\alpha)^t f_0\Big)^2 + 2(1-\alpha)^t f_0 \sum_{i=0}^{t-1} \alpha(1-\alpha)^i y_{t-i} \\
&\qquad + \sum_{i=0}^{t-1} \alpha(1-\alpha)^i y_{t-i} \sum_{j=0}^{t-1} \alpha(1-\alpha)^j y_{t-j}\Big] \\
&= \lim_{t\to\infty} \Big(\underbrace{(1-\alpha)^t}_{\to 0} f_0\Big)^2 + \lim_{t\to\infty} 2\alpha\, f_0\, \mu\, \underbrace{\tfrac{(1-\alpha)^t - (1-\alpha)^{2t}}{1-(1-\alpha)}}_{\to 0} \\
&\qquad + \lim_{t\to\infty} \mathbb{E}\Big[\sum_{i=0}^{t-1} \alpha(1-\alpha)^i y_{t-i} \sum_{j=0}^{t-1} \alpha(1-\alpha)^j y_{t-j}\Big] \\
&= \alpha^2 \lim_{t\to\infty} \sum_{i,j=0}^{t-1} (1-\alpha)^{i+j}\, \mathbb{E}\big[y_{t-i} y_{t-j}\big] \\
&= \mu^2 \underbrace{\Big(\alpha \sum_{i=0}^{\infty} (1-\alpha)^i\Big)^2}_{=1} \;+\; \alpha^2\sigma^2 \lim_{t\to\infty} \sum_{i=0}^{t-1} \big((1-\alpha)^2\big)^i \\
&= \mu^2 + \frac{\alpha^2\,\sigma^2}{1-(1-\alpha)^2} \;=\; \mu^2 + \frac{\alpha\,\sigma^2}{2-\alpha}\,.
\end{aligned}
$$

Subtracting $\lim\limits_{t\to\infty} \mathbb{E}[f_t]^2 = \mu^2$ yields the variance.

Bonus-question: One can use the binomial identity to reformulate the $(1-\alpha)^t$:

$$
\begin{aligned}
\lim_{\alpha\to 0} \alpha_t &= \lim_{\alpha\to 0} \tfrac{\alpha}{1-(1-\alpha)^t} \;=\; \lim_{\alpha\to 0} \Big[\alpha^{-1} - \big(\alpha^{-\frac{1}{t}} - \alpha^{\frac{t-1}{t}}\big)^t\Big]^{-1} \\
&= \lim_{\alpha\to 0} \Big[\alpha^{-1} - \sum_{k=0}^{t} \tbinom{t}{k} (-1)^{t-k}\, \underbrace{\alpha^{-\frac{k}{t}} \alpha^{\frac{(t-k)(t-1)}{t}}}_{\alpha^{t-k-1}}\Big]^{-1} \\
&= \lim_{\alpha\to 0} \Big[\alpha^{-1} - \sum_{k=0}^{t-2} \tbinom{t}{k} (-1)^{t-k}\, \alpha^{t-k-1} + \underbrace{\tbinom{t}{t-1}}_{t} \alpha^0 - \underbrace{\tbinom{t}{t}}_{1} \alpha^{-1}\Big]^{-1} \\
&= \lim_{\alpha\to 0} \Big[t - \sum_{k=0}^{t-2} \tbinom{t}{k} (-1)^{t-k} \alpha^{t-k-1}\Big]^{-1} \;=\; \frac{1}{t}\,.
\end{aligned}
$$

A1.3: Noise in linear functions **(4 points)**

Let $\{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^m$ denote a set of training samples and $\{y_i\}_{i=1}^n \subset \mathbb{R}$ the set of corresponding training labels. We will use the mean squared loss $\mathcal{L} := \frac{1}{n}\sum_i (f(\boldsymbol{x}_i) - y_i)^2$ to learn a function $f(\boldsymbol{x}_i) \approx y_i, \forall i$.

(a) *[1 point]* Derive the analytical solution for parameter $\boldsymbol{a} \in \mathbb{R}^m$ of a linear function $f(\boldsymbol{x}) := \boldsymbol{a}^\top \boldsymbol{x}$.

(b) *[1 point]* We will now augment the training data by adding i.i.d. noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2) \in \mathbb{R}$ to the training labels, i.e. $\tilde{y}_i := y_i + \epsilon_i$. Show that this does not change the analytical solution of the expected loss $\mathbb{E}[\mathcal{L}]$.

(c) *[1 point]* Let f denote the function that minimizes $\mathcal{L}$ *without* label-noise, and let $\tilde{f}$ denote the function that minimizes $\mathcal{L}$ *with* a random noise ϵ_i added to labels y_i (but *not* the solution of the expected loss $\mathbb{E}[\mathcal{L}]$). Derive the analytical variance $\mathbb{E}[(\tilde{f}(\boldsymbol{x}) - f(\boldsymbol{x}))^2]$ of the noisy solution $\tilde{f}$.

(d) *[1 point]* We will now augment the training data by adding i.i.d. noise $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \in \mathbb{R}^m$ to the training samples: $\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + \boldsymbol{\epsilon}_i$. Derive the analytical solution for parameter $\boldsymbol{a} \in \mathbb{R}^m$ that minimizes the expected loss $\mathbb{E}[\mathcal{L}]$.

Bonus-question: Which popular regularization method is equivalent to (d) and what problem is solved?

Hint: Summarize all training samples into matrix $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^\top \in \mathbb{R}^{n\times m}$, all training labels into vector $\boldsymbol{y} = [y_1, \ldots, y_n]^\top \in \mathbb{R}^n$, and denote the noisy versions $\tilde{\boldsymbol{y}} \in \mathbb{R}^n$ and $\tilde{\mathbf{X}} \in \mathbb{R}^{n\times m}$.

Solution

(a) Setting the gradient to zero: $\nabla_{\boldsymbol{a}}\mathcal{L} = \frac{2}{n}\sum_i (\boldsymbol{a}^\top \boldsymbol{x}_i - y_i)\boldsymbol{x}_i = \frac{2}{n}\mathbf{X}^\top\mathbf{X}\boldsymbol{a} - \frac{2}{n}\mathbf{X}^\top\boldsymbol{y} \overset{!}{=} 0$ allows us to derive the analytic solution for $\boldsymbol{a}$ if the matrix $\mathbf{X}^\top\mathbf{X}$ is invertible: $\boldsymbol{a} \overset{!}{=} (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{y}$.

(b) Using the result from (a): $\mathbb{E}[\nabla_{\boldsymbol{a}}\mathcal{L}] = \frac{2}{n}\mathbf{X}^\top\mathbf{X}\boldsymbol{a} - \frac{2}{n}\mathbf{X}^\top\mathbb{E}[\tilde{\boldsymbol{y}}] = \frac{2}{n}\mathbf{X}^\top\mathbf{X}\boldsymbol{a} - \frac{2}{n}\mathbf{X}^\top\boldsymbol{y}$, because $\mathbb{E}[\tilde{\boldsymbol{y}}] = \boldsymbol{y} + \mathbb{E}[\boldsymbol{\epsilon}] = \boldsymbol{y}$ due to the zero mean noise vector $\boldsymbol{\epsilon} := [\epsilon_1, \ldots, \epsilon_n]^\top$.

(c) First note that $\mathbb{E}[(\tilde{f}(\boldsymbol{x}) - f(\boldsymbol{x}))^2] = \mathbb{E}[\tilde{f}^2(\boldsymbol{x})] - 2f(\boldsymbol{x})\mathbb{E}[\tilde{f}(\boldsymbol{x})] + f^2(\boldsymbol{x}) = \mathbb{E}[\tilde{f}^2(\boldsymbol{x})] - f^2(\boldsymbol{x})$, because $\mathbb{E}[\tilde{\boldsymbol{y}}] = \boldsymbol{y}$. Due to i.i.d. noise we have $\mathbb{E}[\tilde{\boldsymbol{y}}\tilde{\boldsymbol{y}}^\top] = \boldsymbol{y}\boldsymbol{y}^\top + \boldsymbol{y}\mathbb{E}[\boldsymbol{\epsilon}^\top] + \mathbb{E}[\boldsymbol{\epsilon}]\boldsymbol{y}^\top + \mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top] = \boldsymbol{y}\boldsymbol{y}^\top + \sigma^2\mathbf{I}$, and $\mathbb{E}[\tilde{f}^2(\boldsymbol{x})] = \boldsymbol{x}^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbb{E}[\tilde{\boldsymbol{y}}\tilde{\boldsymbol{y}}^\top]\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\boldsymbol{x} = f^2(\boldsymbol{x}) + \sigma^2\boldsymbol{x}^\top(\mathbf{X}^\top\mathbf{X})^{-1}\boldsymbol{x}$. The variance is therefore $\mathbb{E}[(\tilde{f}(\boldsymbol{x}) - f(\boldsymbol{x}))^2] = \sigma^2\,\boldsymbol{x}^\top(\mathbf{X}^\top\mathbf{X})^{-1}\boldsymbol{x}$.

(d) Let $\mathbf{E} := [\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_n]^\top$, where we know from the definition of all $\boldsymbol{\epsilon}_i$ that $\mathbb{E}[\mathbf{E}] = \mathbf{0} \in \mathbb{R}^{n\times m}$ and $\mathbb{E}[\mathbf{E}^\top\mathbf{E}] = \mathbb{E}[\sum_i \boldsymbol{\epsilon}_i\boldsymbol{\epsilon}_i^\top] = n\boldsymbol{\Sigma}$. Gradients pass through sums (and therefore expectations): $\nabla_{\boldsymbol{a}}\mathbb{E}[\mathcal{L}] = \frac{1}{n}\mathbb{E}[\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}]\boldsymbol{a} - \frac{1}{n}\mathbb{E}[\tilde{\mathbf{X}}^\top]\boldsymbol{y} = (\frac{1}{n}\mathbf{X}^\top\mathbf{X} + \boldsymbol{\Sigma})\boldsymbol{a} - \frac{1}{n}\mathbf{X}^\top\boldsymbol{y} \overset{!}{=} 0$. Now the optimal solution for the parameter vector is $\boldsymbol{a} \overset{!}{=} (\mathbf{X}^\top\mathbf{X} + n\boldsymbol{\Sigma})^{-1}\mathbf{X}^\top\boldsymbol{y}$.

Bonus-question: The popular L_2 regularization, also called *weight decay*, adds the term $\lambda\|\boldsymbol{a}\|^2$ to the loss and yields the analytical solution $\boldsymbol{a} \overset{!}{=} (\mathbf{X}^\top\mathbf{X} + n\lambda\mathbf{I})^{-1}\mathbf{X}\boldsymbol{y}$, which is also called *ridge regression*. This regularization guarantees that the matrix $\mathbf{X}^\top\mathbf{X} + n\lambda\mathbf{I}$ is invertible for all $\lambda > 0$ and yields smoother functions. For $\boldsymbol{\Sigma} = \lambda\mathbf{I}$, which corresponds to noising each input dimension independently with variance λ, the two solutions are the same, which indicates that noising the input smoothes the learned function!

A1.4: Discounted values can be counter intuitive **(3 points)**

When computer scientists design the tasks that deep RL algorithms are supposed to solve, they often assume the algorithm would find the policy that collects the most reward, e.g. maximizes the average reward. However, most deep RL algorithms are based on discounted values and do not optimize this measure directly, which can yield counter-intuitive results. Take the following MDP[1] that has only one decision state s_0 with the choice $\mathcal{A} := \{L, R\}$. If the agent chooses $L \in \mathcal{A}$, it receives the reward r and embarks on a journey of n states $s_1, \ldots, s_n, s_0$, during which it cannot make any decisions (or all decisions have the same outcome). If on the other hand the agent chooses $R \in \mathcal{A}$ in s_0, it receives no immediate reward and embarks on a different journey of n states $s'_1, \ldots, s'_n, s_0$, during which it also cannot make a decision, but which *ends* with a reward of r' for the transition $s'_n \to s_0$. Note that neither choice ends the Markov chain, which has therefore infinite length.

[1]This MDP is based on the counter-example from Naik et al. (2019) ⟨`https://arxiv.org/abs/1910.02140`⟩.

(a) Derive the discounted infinite-horizon value $V^\pi(s_0)$ for one policy that chooses $\pi(L|s_0) = 1$ and for another policy that chooses $\pi'(R|s_0) = 1$. Give a solution without infinite sums if you can.

(b) For $r < r'$, derive the range of $\gamma \in [0, 1)$ for which the optimal policy based on the discounted value does **not** also yield the largest *average reward.*

Solution

(a) Note that due to construction of the MDP, the agent will receive a reward every $n + 1$ steps. $V^\pi(s_0) = \sum_{i=0}^\infty \gamma^{(n+1)i} r = r \sum_{i=0}^\infty (\gamma^{(n+1)})^i = \frac{r}{1-\gamma^{n+1}}$ and $V^{\pi'}(s_0) = \sum_{i=0}^\infty \gamma^{(n+1)i+n} r' = \gamma^n r' \sum_{i=0}^\infty (\gamma^{(n+1)})^i = \frac{\gamma^n r'}{1-\gamma^{n+1}}$ due to the geometric series.

(b) As $r' > r$, the policy π' yields the highest average reward ($\frac{r'}{n+1} > \frac{r}{n+1}$). The optimal policy w.r.t. discounted value will be different if $V^\pi(s_0) > V^{\pi'}(s_0) \Leftrightarrow \frac{r}{1-\gamma^{n+1}} > \frac{\gamma^n r'}{1-\gamma^{n+1}} \Leftrightarrow \frac{r}{r'} > \gamma^n \Leftrightarrow \sqrt[n]{\frac{r}{r'}} > \gamma$. This is therefore the case for all $0 \le \gamma < \sqrt[n]{\frac{r}{r'}}$.

A1.5: Expected training labels minimize classification loss **(2 points)**

In this exercise we would like to correctly predict the probability $\pi_\theta(a|s)$, parameterized by θ, of 'optimal actions' $a \in \mathcal{A} \subset \mathbb{N}$ given some states $s \in \mathcal{S} \subset \mathbb{R}^d$. The training-set is drawn from state distribution $p(s)$ and stochastic training policy $p(a|s)$. As this is essentially a classification task, we use the *negative log-likelihood* as loss function:

$$\mathcal{L}[\theta] \quad := \quad -\int p(s) \sum_{a\in\mathcal{A}} p(a|s) \log \pi_\theta(a|s)\, ds\,.$$

Prove analytically that the policy $\pi_\theta(a|s) = p(a|s), \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$, is an extreme point of this loss.

Bonus-question: Can you also show that it is a minimum?

Hint: Gradient operators ∇ pass through sums and integrals, e.g. $\nabla \sum_i f_i(x) = \sum_i \nabla f_i(x)$, and the gradient of the logarithm is $\nabla \log f(x) = \frac{\nabla f(x)}{f(x)}$. Minima have a positive definite Hessian matrix. You can look up those terms at `https://en.wikipedia.org/wiki/Definite_matrix` and `https://en.wikipedia.org/wiki/Hessian_matrix`.

Solution The first derivative ∇ (w.r.t. θ) of $\mathcal{L}$ is zero for $\pi_\theta(a|s) = p(a|s)$, which makes this an extreme point:

$$\nabla\mathcal{L}[\theta] \;=\; -\int p(s) \sum_{a\in\mathcal{A}} p(a|s) \frac{\nabla\pi_\theta(a|s)}{\pi_\theta(a|s)}\, ds \overset{\pi_\theta = p}{=} -\nabla\int p(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s)\, ds \;=\; -\nabla 1 \;=\; 0\,.$$

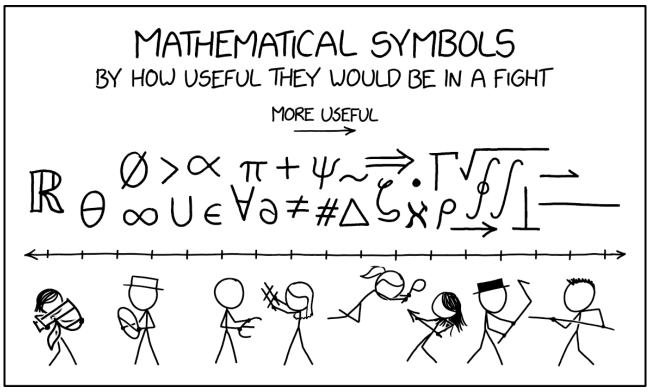
Bonus-question: The second derivative $\frac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}$, which is the entry (i, j) in the Hessian matrix, is

$$\begin{aligned}
\tfrac{\partial^2}{\partial\theta_i\partial\theta_j}\mathcal{L}[\theta] &= -\tfrac{\partial}{\partial\theta_i} \int p(s) \sum_{a\in\mathcal{A}} p(a|s) \frac{\frac{\partial}{\partial\theta_j}\pi_\theta(a|s)}{\pi_\theta(a|s)}\, ds \\
&= -\int p(s) \sum_{a\in\mathcal{A}} p(a|s) \left(\frac{\frac{\partial^2}{\partial\theta_i\partial\theta_j}\pi_\theta(a|s)}{\pi_\theta(a|s)} - \frac{\frac{\partial}{\partial\theta_i}\pi_\theta(a|s)}{\pi_\theta(a|s)} \frac{\frac{\partial}{\partial\theta_j}\pi_\theta(a|s)}{\pi_\theta(a|s)} \right) ds \\
&\overset{\pi_\theta = p}{=} -\tfrac{\partial^2}{\partial\theta_i\partial\theta_j} \int p(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s)\, ds + \int p(s) \sum_{a\in\mathcal{A}} p(a|s) \frac{\frac{\partial}{\partial\theta_i}\pi_\theta(a|s)}{\pi_\theta(a|s)} \frac{\frac{\partial}{\partial\theta_j}\pi_\theta(a|s)}{\pi_\theta(a|s)}\, ds \\
&= \int p(s) \sum_{a\in\mathcal{A}} p(a|s) \left(\nabla \log \pi_\theta(a|s)\right)_i \left(\nabla \log \pi_\theta(a|s)\right)_j ds\,.
\end{aligned}$$

This implies that the Hessian matrix for $\pi_\theta = p$ is

$$\mathbf{H} = \int p(s) \sum_{a\in\mathcal{A}} p(a|s)(\nabla \log \pi_\theta(a|s))(\nabla \log \pi_\theta(a|s))^\top \, ds.$$

As $\mathbf{H}$ is symmetric and for finite state spaces $\mathcal{S}$ can be expressed as product of a matrix (which concatenate all $\nabla \log \pi_\theta(a|s)$) with its transposed self, we can conclude that $\mathbf{H}$ is positive semi-definite (at least for finite $\mathcal{S}$). The policy $\pi_\theta = p$ must thus be a minimum (or at least a saddle point) of the loss function.



https://xkcd.com/2343

Total 17 points.