CSE 840: Computational Foundations of Artificial Intelligence March 19, 2025

Higher-Order Derivatives, Minima, Maxima, Saddle Points, Matrix Calculus

Instructor: Vishnu Boddeti *Scribes: Tolu Oshin and Roshan Atluri*

1 Higher Order Derivatives

Consider $f : \mathbb{R}^n \to \mathbb{R}$, assume it is differentiable, so all partial derivatives $\frac{\partial f}{\partial x_i} : \mathbb{R}^n \to \mathbb{R}$ exist.

If this function is differentiable, we can take its derivative:

$$\frac{\partial}{\partial x_i}\left(\frac{\partial f}{\partial x_j}\right) = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

These are called second order partial derivatives.

Warning: In general, we cannot change the order of derivatives:

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \neq \frac{\partial^2 f}{\partial x_j \partial x_i}$$

Example: Let $f(x, y) = \frac{x \cdot y^3}{x^2+y^2}$

$$\nabla f(x, y) = \left(\frac{y^3(y^2 - x^2)}{(x^2 + y^2)^2}, \frac{xy^2(3x^2 + y^2)}{(x^2 + y^2)^2}\right)$$

Have:

$$\frac{\partial f}{\partial x}(0, y) = y \quad \forall y \quad , \quad \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right) = 1$$

$$\frac{\partial f}{\partial y}(x, 0) = 0 \quad \forall x \quad , \quad \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial y}\right) = 0$$

Definition 1 *We say that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable if all partial derivatives exist and are continuous.*

We say that f is twice continuously differentiable if f is continuously differentiable and all its partial derivatives $\frac{\partial f}{\partial x_i}$ are again continuously differentiable.

Analogously: k-times continuously differentiable.

Notation:

$$C^k(\mathbb{R}^n, \mathbb{R}^m) = \{f : \mathbb{R}^n \to \mathbb{R}^m \mid k \text{ times continuously differentiable}\}$$
$$C^\infty(\mathbb{R}^n, \mathbb{R}^m) = \{f : \mathbb{R}^n \to \mathbb{R}^m \mid \infty \text{ often continuously differentiable}\}$$

Theorem 2 *(Schwarz) Assume that f is twice continuously differentiable. Then we can exchange the order in which we take partial derivatives:*

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}$$

Analogously: k-times continuously differentiable implies we can exchange the order of the first k partial derivatives.

Warning: Caution about derivatives:

$$f : \mathbb{R}^n \to \mathbb{R} \quad \text{(function)}$$
$$\nabla f : \mathbb{R}^n \to \mathbb{R}^n \quad \text{(first derivative: } n \text{ partial derivatives }, \frac{\partial f}{\partial x_i})$$
$$Hf : \mathbb{R}^n \to \mathbb{R}^{n \times n} \quad \text{(second derivative: } n^2 \text{ partial derivatives, } \frac{\partial^2 f}{\partial x_i \partial x_j})$$

Definition 3 *Let $f : \mathbb{R}^n \to \mathbb{R}$, then we define the Hessian of f at point x by:*

$$(Hf)_{ij}(x) := \frac{\partial^2 f(x)}{\partial x_i \partial x_j}, \quad i, j = 1, 2, \ldots, n$$

$$Hf(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

2 Minima/Maxima

Definition 4 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable. If $\nabla f(x) = 0$, then we call x a critical point.*

f has a local minimum at x_0 if there exists $\varepsilon > 0$ such that

$$\forall x \in B_\varepsilon(x_0) : \quad f(x) \geq f(x_0)$$

f has a strict local minimum at x_0 if there exists $\varepsilon > 0$ such that

$$\forall x \in B_\varepsilon(x_0) : \quad f(x) > f(x_0)$$

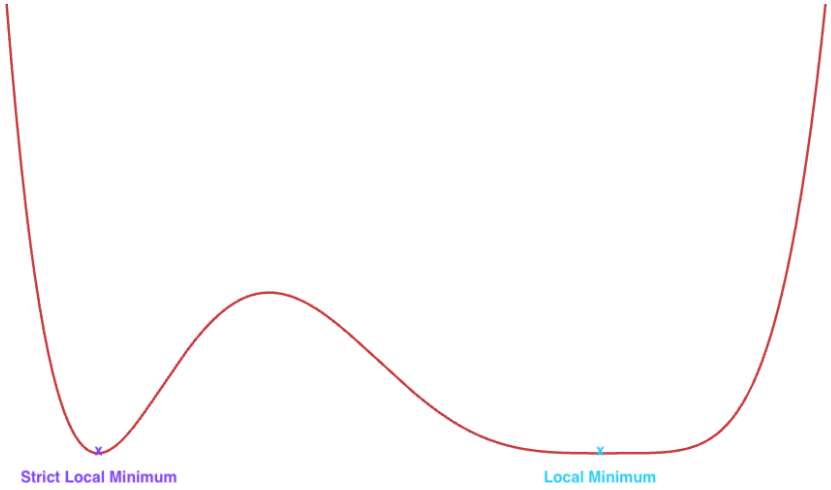


Figure 1: Strict local vs local minimums

f has a local maximum (respectively, a strict local maximum) at x_0 if

$$\forall x \in B_\varepsilon(x_0): \quad f(x) \leq f(x_0) \quad (\textit{respectively } f(x) < f(x_0))$$

If f is differentiable and x_0 is a critical point that is neither a local minimum nor a local maximum, we call it a saddle point.

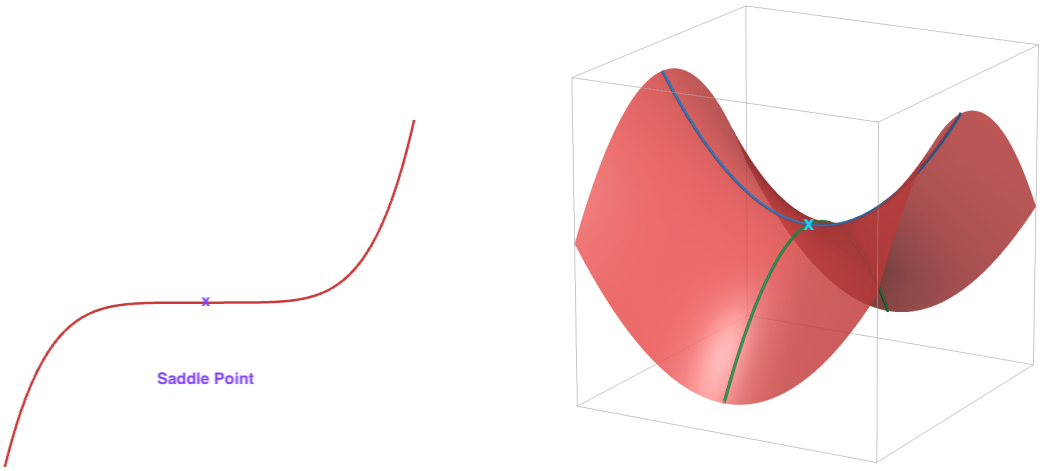


Figure 2: Examples of saddle points in $\mathbb{R}^2$ and $\mathbb{R}^3$

f has a global minimum at x_0 if

$$\forall x: \quad f(x) \geq f(x_0)$$

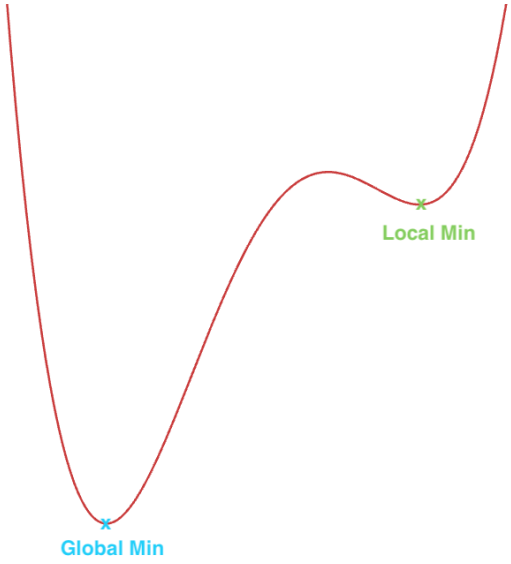
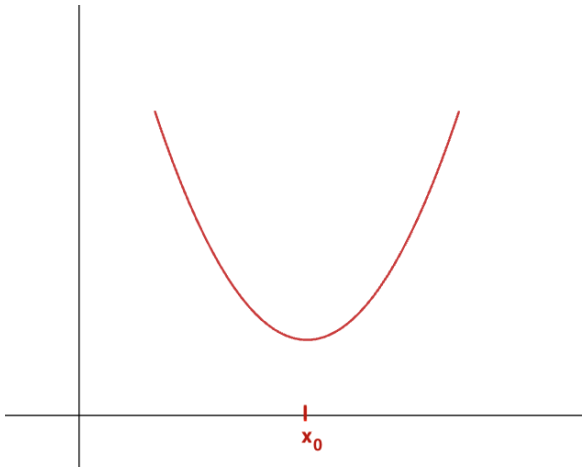


Figure 3: Global vs. local minimums

Intuition: How can we identify which type of point we have? In $\mathbb{R}$:

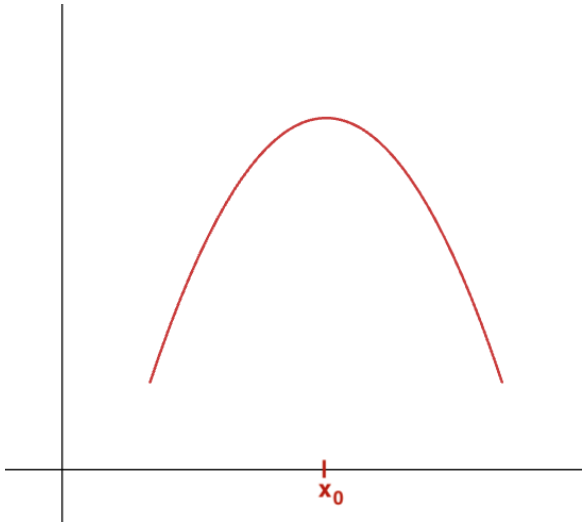
- Local minimum at x_0:

$$f'(x_0) = 0, \quad f''(x_0) > 0$$



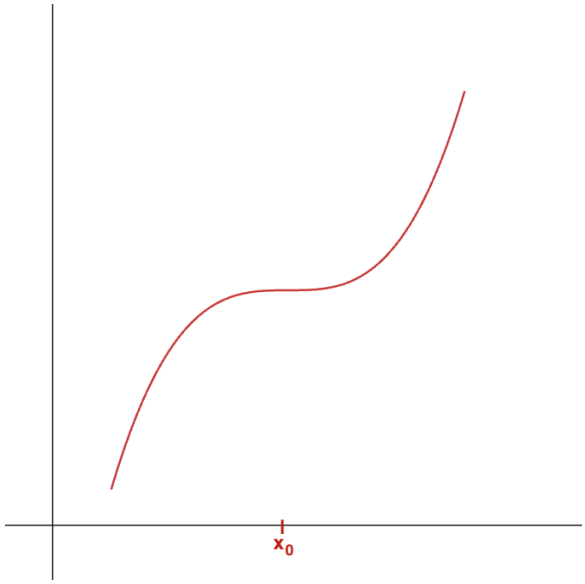
- Local maximum at x_0:

$$f'(x_0) = 0, \quad f''(x_0) < 0$$



- Saddle point at x_0:

$$f'(x_0) = 0, \quad f''(x_0) = 0$$



Theorem 5 *Let $f : \mathbb{R}^n \to \mathbb{R}$, with $f \in C^2(\mathbb{R}^n)$. Assume that x_0 is a critical point, i.e., $\nabla f(x_0) = 0$. Then:*

(i) If x_0 is a local minimum (respectively, maximum), then the Hessian $Hf(x_0)$ is positive semi-definite (respectively, negative semi-definite).

(ii) If $Hf(x_0)$ is positive definite (respectively, negative definite), then x_0 is a strict local minimum (respectively, maximum). If $Hf(x_0)$ is indefinite, then x_0 is a saddle point.

3 Matrix/Vector Calculus

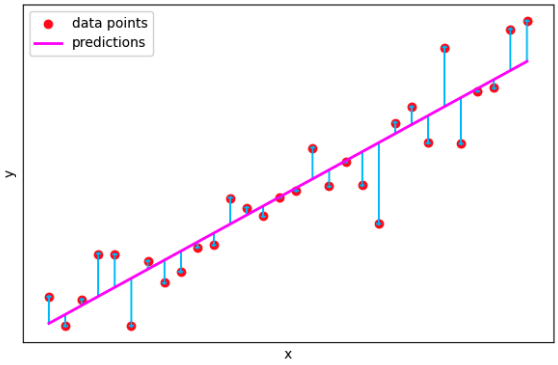


Figure 4: Linear Least Squares in $\mathbb{R}^2$

Example: Linear least squares

$$f : \mathbb{R}^n \to \mathbb{R}, \quad \hat{y}(\omega) = A\omega$$

where:

- $\hat{y}$ is the predicted output
- A is the matrix of input data
- ω is the weight vector (parameters we want to find)

The function:

$$f(\omega) = \|y - \hat{y}(\omega)\|_2^2 = \|y - A\omega\|_2^2$$

measures how good the prediction is using parameters ω. We want to minimize $f(\omega)$, so we need to look at the gradient:

$$\nabla f : \mathbb{R}^n \to \mathbb{R}^n$$

Compute Gradient:

We write out $f(\omega)$ in expanded form:

$$f\left(\begin{pmatrix}\omega_1\\ \vdots \\ \omega_n\end{pmatrix}\right) = \sum_{j=1}^{n}\left(y_j - \sum_{k=1}^{n} a_{jk}\omega_k\right)^2$$

Then the partial derivative with respect to ω_i is:

$$\frac{\partial f}{\partial \omega_i} = \sum_{j=1}^{n}(-a_{ji}) \cdot 2 \cdot \left(y_j - \sum_{k=1}^{n} a_{jk}\omega_k\right)$$

This simplifies to:

$$\frac{\partial f}{\partial \omega_i} = -2\sum_{j=1}^{n} a_{ji}((y - A\omega)_j) = \left(-2A^T(y - A\omega)\right)_i$$

So the full gradient is:

$$\nabla f(\omega) = -2A^T(y - A\omega)$$

Intuition: The matrix case is similar in "syntax" to the 1-dimensional case:

$$f(\omega) = (y - a\omega)^2, \quad f'(\omega) = -2a(y - a\omega)$$

Matrix-vector calculus: Lookup table for gradients of common functions

$f : \mathbb{R}^n \to \mathbb{R}$

- $f(x) = a^\top x =< a, x > \quad (a \in \mathbb{R}^n)$

$$\frac{\partial f}{\partial x} = a \in \mathbb{R}^n$$

- $f(x) = x^\top Ax \quad (A \in \mathbb{R}^{n \times n})$

$$\frac{\partial f}{\partial x} = (A + A^\top)x \in \mathbb{R}^n$$

$f : \mathbb{R}^{n \times m} \to \mathbb{R}$

- $f(X) = a^\top Xb \quad (a \in \mathbb{R}^n,\ b \in \mathbb{R}^m)$

$$\frac{\partial f}{\partial X} = ab^\top \in \mathbb{R}^{n \times m}$$

- $f(X) = a^\top X^\top CXb \quad (a, b \in \mathbb{R}^m,\ C \in \mathbb{R}^{n \times n})$

$$\frac{\partial f}{\partial X} = C^\top Xab + CXba^\top$$

- $f(X) = \mathrm{tr}(X)$

$$\frac{\partial f}{\partial X} = I$$

- $f(X) = \mathrm{tr}(AX)$

$$\frac{\partial f}{\partial X} = A$$

- $f(X) = \mathrm{tr}(X^\top AX)$

$$\frac{\partial f}{\partial X} = (A + A^\top)X$$

- $f(X) = \det(X)$

$$\frac{\partial f}{\partial X} = \det(X) \cdot (X^\top)^{-1}$$

$$\frac{\partial \det}{\partial x_{sr}} = \det(X) \cdot \left(X^{-1}\right)_{rs}$$

$f : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m}$ **(inverse)**

- $f(A) = A^{-1}, \quad f_{ij} := \left(A^{-1}\right)_{ij}$

$$\frac{\partial f_{ij}}{\partial a_{uv}} = -\left(a_{iu}\right)^{-1}\left(a_{vj}\right)^{-1}$$

4 Applications of Lecture Concepts in Machine Learning

4.1 Higher-order derivatives

Higher-order derivatives are used in some more advanced machine learning methods. While gradients (first-order derivatives) are used in most optimization algorithms, higher-order derivatives give more information about the shape of the loss function. This can help with better optimization, understanding generalization, and tuning hyperparameters.

4.1.1 Second-Order Optimization

Second-order methods use the Hessian matrix, which contains all the second derivatives of the loss function with respect to the model parameters. One example is Newton's method, which updates parameters like this:

$$x^{(k)} = x^{(k-1)} - \left(\nabla^2 f(x^{(k-1)})\right)^{-1} \nabla f(x^{(k-1)})$$

where, $\nabla^2 f$ is the Hessian and ∇f is the gradient [1]. This update can lead to much faster convergence compared to gradient descent, however, it can diverge [2]. However, computing and inverting the Hessian is expensive for large models, so people use approximations. A common example of this is the BFGS method, which utilizes an approximate inverse Hessian.

4.1.2 Sharpness and Generalization

Higher-order derivatives can also be used to analyze how well a model will generalize. One idea is to look at how "sharp" or "flat" a minimum is. This is done by looking at the eigenvalues of the Hessian:

- Sharp minima (large eigenvalues): usually don't generalize well.
- Flat minima (small eigenvalues): tend to generalize better.

Research shows that models trained with large batch sizes often converge to sharper minima, which may hurt generalization [3]. Techniques like using smaller batch sizes or adding regularization can help find flatter minima and thus improve the performance of the model on new datasets.

4.2 Matrix/Vector Calculus

4.2.1 Backpropogation

Matrix and vector calculus play a central role in many core machine learning algorithms, and backpropogation is an important example of an appplication of this. Backpropagation is the algorithm used to compute gradients of the loss function with respect to each parameter in a neural network. It is an application of the multivariable chain rule in matrix and vector form.

Consider a simple feedforward neural network layer:

$$z = Wx + b, \quad a = \sigma(z)$$

where x is the input vector, W is the weight matrix, b is the bias vector, σ is an activation function, and a is the output of the layer.

To update W during training, we need to compute the partial derivative of the loss L with respect to W:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial W}$$

This is a direct application of the chain rule for functions composed of matrix operations. In practice, modern deep learning libraries like PyTorch and TensorFlow use automatic differentiation, which internally applies these matrix calculus rules to compute gradients efficiently.

Backpropagation generalizes to deep networks by recursively applying the chain rule layer by layer, from the output back to the input.

References

[1] Ryan Tibshirani. *Lecture 14: Newton's Method*. Mar. 2015. URL: https://www.stat.cmu.edu/~ryantibs/convexopt-S15/scribes/14-newton-scribed.pdf.

[2] Kilian Weinberger. *Gradient Descent (and Beyond)*. URL: https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote07.html.

[3] Nitish Shirish Keskar et al. *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*. 2017. arXiv: 1609.04836 [cs.LG]. URL: https://arxiv.org/abs/1609.04836.