Machine Learning for Physicists: Recitation Notes

Clark Miyamoto (cm6627@nyu.edu)

January 21, 2026

Contents

1 Review of Linear Algebra

1.1 Singular Value Decomposition

Recall the eigen-decomposition of a matrix. Given a symmetric square matrix $A \in \mathbb{R}^{d\times d}$ with eigenvalues $\{\lambda_i\}_i$ and eigenvectors $\{e_i\}_i$. The matrix could be re-expressed as

$$A = U\Lambda U^T \tag{1.1}$$

where $\Lambda = \text{diag}(\lambda_1, ..., \lambda_d) \in \mathbb{R}^{d\times d}$ and $U \in \mathbb{R}^{d\times d}$ is a matrix whose columns are $\{e_i\}_i$.

This decomposition had a lot of nice properties. In particular, Λ is diagonal and U is orthogonal. This allowed us to do all sorts of stuff easily; for example, matrix power.

What happens if we want to do this on non-symmetric, or even non-square matrices? Well we can use the singular value decomposition (SVD).

1.1.1 Definitions

Definition 1 (Singular Values) *Let $A \in \mathbb{R}^{m\times n}$. Now consider $A^TA \in \mathbb{R}^{n\times n}$. This is a symmetric matrix so it has positive eigenvalues $0 \le \lambda_1 \le ... \le \lambda_n$. The singular values σ_i for matrix A are defined as*

$$\sigma_i \equiv \sqrt{\lambda_i},\ \ s.t.\ 0 \le \lambda_1 \le ... \le \lambda_n \tag{1.2}$$

Definition 2 (SVD) *$A \in \mathbb{R}^{m\times n}$ with singular values $0 \le \sigma_1 \le ... \le \sigma_n$. Let r denote the rank, or equivalently the number of singular values of A. The SVD of A is a decomposition*

$$A = U\Sigma V^T \tag{1.3}$$

where

- *$U \in \mathbb{R}^{m\times m}$ orthogonal matrix*
- *$V \in \mathbb{R}^{n\times n}$ orthogonal matrix*
- *$\Sigma \in \mathbb{R}^{m\times n}$ matrix such that $[\Sigma]_{ii} = \sigma_i$ for $i \in [1, ..., r]$ and $[\Sigma]_{ii} = 0$ for $i > r$.*

1.1.2 An Attempt at Intuition

Recall, by 3Blue1Brown,

1.2 Matrix Calculus

1.3 Time Complexity

Since we're talking about these things in the context of a computational class, it'll be good to recap the time complexity of such algorithms. Just keep these in the back of your mind.

- Matrix multiplication: $\mathcal{O}(n^{2.8})$.
- Matrix inverse implemented in `numpy.linalg.solve`: $\mathcal{O}(n^3)$.
- SVD for a $n \times m$ matrix (s.t. $n \le m$) : $\mathcal{O}(mn^2)$.
- Determinant $\mathcal{O}(n^3)$

As a final note, the time complexity of an algorithm doesn't translate to the actual run time of an algorithm.

See https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations#Matrix_algebra for more information.

1.4 References

References

[1] Michael Hutchings, Notes on singular value decomposition for Math 54, https://math.berkeley.edu/~hutching/teach/54-2017/svd-notes.pdf.

[2] Gregory Gundersen, Singular Value Decomposition as Simply as Possible, https://gregorygundersen.com/blog/2018/12/10/svd/

[3] Leslie Lamport (1994) *LaTeX: a document preparation system*, Addison Wesley, Massachusetts, 2nd ed.

2 Review of Probability

Definition 3 (Conditional Probability)

$$\mathbb{P}[A|B] = \frac{\mathbb{P}[A \cap B]}{P[B]} \tag{2.1}$$

Notice that $\mathbb{P}[A \cap B] = \mathbb{P}[B \cap A]$, this allows us to related $\mathbb{P}[A|B]$ and $\mathbb{P}[B|A]$.

$$\mathbb{P}[A|B] = \frac{\mathbb{P}[A \cap B]}{P[B]} \tag{2.2}$$

$$= \frac{\mathbb{P}[B \cap A]}{P[B]} \tag{2.3}$$

$$\boxed{\mathbb{P}[A|B] = \frac{\mathbb{P}[B|A]\,\mathbb{P}[A]}{\mathbb{P}[B]}} \tag{2.4}$$

This is **Bayes' Formula**.

Definition 4 (Probability Density Function) *A function with the following properties is a **probability density***

- *Positive:* $p : \mathcal{X} \to \mathbb{R}_{\geq 0}$
- *Normalized:* $\int_{\mathcal{X}} p(x)\, dx = 1$

It is interpreted as the probability of observing an event $A \subset \mathcal{X}$ *as*

$$\mathbb{P}[x \in A] = \int_{A \subset \mathcal{X}} p(x)\, dx \tag{2.5}$$

The nice part of densities is that you can compute statistics with that. I.e. what's the mean, variance.

$$\mathbb{E}_{x \sim p}[f(x)] = \int_{\mathcal{X}} f(x)\, p(x)\, dx \tag{2.6}$$

Definition 5 (Characteristic Function) *Consider the probability distribution* p_X. *It has an associated **characteristic function*** φ_X *which is it's Fourier Transform*

$$\varphi_X(k) = \int_{\mathbb{R}} e^{ikx} p(x)\, dx = \mathbb{E}_{x \sim p}[e^{ikx}] \tag{2.7}$$

3 Review of Statistics & Loss Functions

In machine learning, we adjust a model's parameters θ to minimize a loss function $\mathcal{L}(\theta)$. There's a bunch so I think it's nice to hear where they come from. We'll cover

- Mean squared error (MSE)

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \|y_i - f_\theta(x_i)\|^2$$

- Cross entropy

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \|$$

- MSE + L2 Regularization (Ridge)

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \|y_i - f_\theta(x_i)\|_2^2 + \lambda \|\theta\|_2^2$$

3.1 Maximum Likelihood Inference & Mean Squared Error

Say you have the dataset $\mathcal{D} = \{(y_i, x_i)\}_{i=1}^n$ (which we assume you observed in an iid way). You believe that y_i is a noisy observation of some model $f_\theta(x_i)$. Your objective is to come up with the "best" estimate of the parameter θ which matches the data $\mathcal{D}$... You think about it for some while, and realize you maximize the probability of seeing the data for a given θ. This is **maximum likelihood estimation (MLE)**.

To illustrate this method (and all others), we have to assume a particular model. So let's say you believe the noise is additive & gaussian:

$$y_i = f_\theta(x_i) + \epsilon_i, \quad \text{where } \epsilon_i \sim_{iid} \mathcal{N}(0, \mathbb{I}) \tag{3.1}$$

Since ϵ_i is a random variable, you can interpret y_i as a random variable as well.

$$y_i \sim \mathcal{N}(f_\theta(x_i), \mathbb{I}) \tag{3.2}$$

$$p(y_i|\theta) \propto \exp\left(-\frac{1}{2}(y_i - f_\theta(x_i))^2\right) \tag{3.3}$$

$$\log p(y_i|\theta) = -\frac{1}{2}(y_i - f_\theta(x_i))^2 + \text{Constant w.r.t. } \theta \tag{3.4}$$

I've wrote the log prob for reasons that will become clear in a moment.

Note you have more data $\{(y_i, x_i)\}_{i=1}^n$ (which is all iid), so you actually have a joint distribution.

$$p(y_1, ..., y_n|\theta) = \prod_i p(y_i|\theta) \tag{3.5}$$

We'll call this the **likelihood** $L(\theta)$ (that is the likeliness / probability of seeing the data given a configuration of model parameters). For MLE, you choose $\hat{\theta}$ which maximizes the likelihood. However $\arg\max$ of a product of functions is quite difficult, we can compose the function w/ a monontonic function, and that leaves the $\arg\max$ invariant.

$$\log L(\theta) = \log p(y_1, ..., y_n|\theta) \tag{3.6}$$

$$= \sum_i \log p(y_i|\theta) \tag{3.7}$$

$$\propto \sum_i (y_i - f_\theta(x_i))^2 \tag{3.8}$$

This recovers the MSE loss.

3.2 Cross Entropy & Another MLE

3.3 L2 Regularization

In Bayesian statistics, instead of asking what's the probability of seeing the data given a model parameter, we ask *what's the probability of seeing a model parameter given the data?* We can formalize the inverse question using Bayes' theorem

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \tag{3.9}$$

- $p(\theta)$ is your prior. It encodes your prior beliefs into the distribution.

- $p(y|\theta)$ is the likelihood (from the previous sections)
- $p(\theta|y)$ is the posterior. It accounts for your prior beliefs & what the data says (likelihood).
- $p(y)$ is the evidence. I won't say much about it today.

If you ask, what's the parameter maximizes the posterior (probability of seeing a parameter given the data), this is called **maximum aposteriori estimation (MAP)**.

$$\hat{\theta}_{MAP} = \arg\max_{\theta} p(\theta|y) \tag{3.10}$$

For an example, let's assume we have the additive noise model

$$y_i = f_\theta(x_i) + \epsilon_i \tag{3.11}$$

and that you believe the weights should look distributed according to a Gaussian

$$p(\theta) = \mathcal{N}(0, \lambda^{-1}\mathbb{I}) \tag{3.12}$$

You can see that log posterior has the form

$$\log p(\theta|y) = \sum_i \|y_i - f_\theta(x_i)\|^2 + \lambda\|\theta\|^2 \tag{3.13}$$

3.4 Minimizing the loss function

- The value of the MSE, in a traditional statistics setting, tells you about the uncertainity quantification of the model. However ML models tend to not obey this.
- Difficulty of optimizing via oracle access.
- However! Do you even want to perfectly minimize the loss function? Memorization.

4 Linear Regression

Consider making iid noisy observations of data $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$. We'll assume that the noise is additive, that is

$$y_i = f(x_i) + \epsilon_i \tag{4.1}$$

where $f(x) = \beta^T x$ is the model (we've assumed it's linear for this discussion) and the noise is gaussian $\epsilon_i = \mathcal{N}(0, \sigma_i^2)$ (which is another assumption for this discussion). Since ϵ_i is a random variable, this implies that y_i is also a random variable

$$y_i|\beta = \mathcal{N}(y_i; \beta^T x_i, \sigma_i^2) \tag{4.2}$$

This is just one observation, but in fact, we have a joint distribution $p(y|x) \equiv p(y_1, ..., y_N|x_1, ..., x_N)$ over all observations, which we'll call the **likelihood**. Since observations are iid, it factorizes.

$$p(y|\beta) = \prod_i p(y_i|x_i) \tag{4.3}$$

Your task is to find the β which "best" describes the data. I'll note that "best" is subjective and we'll discuss consequences of this later.

4.1 Frequentist, Maximum Likelihood Estimator

One method is **maximum likelihood estimation**, that is you selct the parameters which is the global maximizer of the likelihood. Why? Just read off what you're doing: adjust β s.t. the probability of having this combination of y's (given x's) is highest.

Apart from being very intuitive, there are also strong theoretical guaranties (which I won't have time to prove) (Notation: when I generically talk about model parameters, we use θ)

- Consistency: $\lim_{n\to\infty} \hat{\theta}_n = \theta$
- Normality: $\hat{\theta}_n \sim \mathcal{N}(0, \mathcal{I})$ (where $\mathcal{I}$ is the fisher information matrix)
- Efficiency: $\mathrm{Var}(\hat{\theta}) \geq 1/\mathcal{I}(\theta)$.

Since arg max is invariant under compositions of monotonic functions, we can maximize the log-likelihood which emits a nicer function

$$\begin{aligned}\log p(y|x) &= \sum_i \log p(y_i|x_i) &(4.4)\\ &= \sum_i \log \mathcal{N}(y_i; \beta^T x_i, \sigma_i^2) &(4.5)\\ &= \sum_i -\frac{1}{2}\frac{\left(y_i - \beta^T x_i\right)^2}{\sigma_i^2} + \text{Constant} &(4.6)\end{aligned}$$

A small comment, this is why you "minimize the squared error" when fitting straight lines in lab, you have been secretly doing maximum likelihood inference this whole time. Notice this is a quadratic form, so you can rewrite it using matrix multiplication

$$\sum_{i=1}^n (y_i - \beta^T x_i)^2/\sigma_i^2 = (y - X\beta)^T \Sigma^{-1}(y - X\beta) \tag{4.7}$$

$$\begin{aligned}\text{where: } y &= \begin{pmatrix} y_1 \\ \vdots \\ v_n \end{pmatrix} \in \mathbb{R}^n, &(4.8)\\ \beta &= \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \in \mathbb{R}^p &(4.9)\\ \Sigma &= \mathrm{diag}(\sigma_1^2, ..., \sigma_n^2) \in \mathbb{R}^{n\times n} &(4.10)\\ X &= \begin{pmatrix} - & x_1^T & - \\ - & x_2^T & - \\ & \vdots & \\ - & x_n^T & - \end{pmatrix} \in \mathbb{R}^{n\times p} &(4.11)\end{aligned}$$

From here we can find the argmax of the quantity

$$\begin{aligned}0 = \frac{\partial \log p(y|x)}{\partial \beta}\Big|_{\beta=\hat{\beta}} &= X^T\Sigma^{-1}(y - X\beta) &(4.12)\\ \implies X^T\Sigma^{-1}y &= X^T\Sigma^{-1}X\hat{\beta} &(4.13)\\ &\boxed{\hat{\beta}_{MLE} = (X^T\Sigma^{-1}X)^{-1}X^T\Sigma^{-1}y} &(4.14)\end{aligned}$$

and find the maximum likelihood estimate for β.

Now we can talk about inference. Say your boss gives you new data X_*, and you're asked what is the corresponding $\hat{y}_*$. You'll report back

$$\boxed{\hat{y}_* = X_* \hat{\beta}_{MLE}} \tag{4.15}$$

4.2 Bayesian Linear Regression

In the Bayesian framework, you're asked what is the probability of seeing the model parameters *given* the data $p(\beta|y)$. You can calculate this using Bayes's formula

$$p(\beta|y) = \frac{p(y|\beta)p(\beta)}{p(y)} \tag{4.16}$$

5 Double Descent

5.1 Soft Inductive Biases

Another way to conceptualize this is **soft inductive biases** (see Andrew Gordon Willson's paper https://arxiv.org/pdf/2503.02113).

6 Training Large Models

6.1 Transformers

6.2 μP Optimizer

7 Geometric Deep Learning

Part I
Generative Models

8 "Old" Generative Models

8.1 Variational Autoencoders

8.2 Generative Adversarial Networks (GANs)

8.3 Denoising Diffusion Probablistic Models (DDPMs)

9 Modern Generative Models

9.1 Review of Non-Equilibrium Statistical Mechanics

Consider a classical particle at position X_t at time t (usually we use the notation $x(t)$, but I'll use X_t as it's the modern stochastic calculus notation) moving in a potential $V(x)$. The weird thing is this particle appears jittery, it feels a bunch of random forces due to thermal fluctuations

$$m\ddot{X}_t = -V(X_t) + \xi_t \tag{9.1}$$

9.2 Measure Transport

Theorem 1 (Fokker-Planck Equation) *Consider a stochastic process*

$$dX_t = \mu_t(X_t)\,dt + \sigma_t(X_t)\,dW_t \tag{9.2}$$

$$X_0 \sim p_{base} \tag{9.3}$$

The stochastic process emits a probability distribution at every points in time (notationally $X_t \sim p_t$), where the distribution p_t satisfies a partial differential equation called the ***Fokker-Planck equation***

$$\partial_t p_t(x) = -\nabla \cdot (\mu_t(x)\,p_t(x)) + \frac{1}{2}\sigma_t^2(x)\Delta(p_t(x)) \tag{9.4}$$

$$p_{t=0}(x) = p_{base} \qquad \textit{(Boundary condition)} \tag{9.5}$$

A small remark, take $\sigma \to 0$ and you recover the transport equation

Theorem 2 (Transport Equation) *Consider the deterministic process*

$$dX_t = \mu(X_t)\,dt \tag{9.6}$$

$$X_0 \sim p_{base} \tag{9.7}$$

The associated probability distribution $X_t \sim p_t$ satisfies the partial differential equation called the ***transport equation***

$$\partial_t p_t(x) = -\nabla \cdot (\mu_t(X_t)\,p_t(x)) \tag{9.8}$$

This increases the design space.

9.3 Score Based Diffusion

9.4 Flow Matching

9.5 Stochastic Interpolants