Optimization Techniques for Machine Learning

AMLZC326 · #16 Recap

Anshid Aboobacker

Course at a Glance

M 1: Analytic Geometry (L1–2)
Vector spaces, norms, inner products, orthogonality, Gram-Schmidt, spectral theorem

M 2: Linear Optimization (L3–4)
LPP, simplex method, duality, sensitivity analysis

M 3: Vector Calculus (L5–6)
Gradients, Jacobian, Hessian, Taylor expansion, critical points

M 4: Continuous Optimization (L7–10)
Gradient descent, SGD, AdaGrad, RMSProp, Adam, Lagrangians, KKT

M 5: ML Models (L11–15)
PCA, SVD, Gaussian mixture models, EM algorithm, SVM, ARIMA, Kalman filter

Learning Objectives

By the end of this recap lecture you should be able to:

- Recall the key definitions, theorems, and algorithms from each of the 5 modules
- Connect the mathematical tools (analytic geometry, calculus, duality) to the ML models they enable
- Identify which optimisation technique applies to which ML problem
- See the through-line: how every module builds on the previous one toward EM, SVM, and Kalman

Table of Contents

1. Analytic Geometry
2. Linear Optimization
3. Vector Calculus
4. Continuous Optimization & Duality
5. Principal Component Analysis
6. Machine Learning Models

Vector Spaces & Norms

Vector Space $(V, +, \cdot)$

A set V closed under vector addition and scalar multiplication, satisfying 8 axioms (commutativity, associativity, distributivity, existence of zero and additive inverse, etc.)

Norm $\|\cdot\| : V \to \mathbb{R}$

- **Positivity:** $\|\boldsymbol{x}\| \geq 0; \quad \|\boldsymbol{x}\| = 0 \Leftrightarrow \boldsymbol{x} = \mathbf{0}$
- **Homogeneity:** $\|\alpha \boldsymbol{x}\| = |\alpha| \, \|\boldsymbol{x}\|$
- **Triangle inequality:** $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$

Euclidean Norm

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Inner Products & Orthogonality

Inner Product $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$

Symmetric, bilinear, positive definite. Standard: $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^\top \boldsymbol{y}$

Cauchy-Schwarz Inequality

$$|\langle \boldsymbol{u}, \boldsymbol{v} \rangle| \leq \|\boldsymbol{u}\| \, \|\boldsymbol{v}\|$$

Angle Between Vectors

$$\cos\theta = \frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{\|\boldsymbol{u}\| \, \|\boldsymbol{v}\|}$$

$\boldsymbol{u} \perp \boldsymbol{v} \Leftrightarrow \langle \boldsymbol{u}, \boldsymbol{v} \rangle = 0$

Gram-Schmidt Process

$$\boldsymbol{v}_k = \boldsymbol{u}_k - \sum_{j=1}^{k-1} \frac{\langle \boldsymbol{u}_k, \boldsymbol{v}_j \rangle}{\|\boldsymbol{v}_j\|^2} \boldsymbol{v}_j$$

Gives orthogonal basis; normalise $\hat{\boldsymbol{v}}_k = \boldsymbol{v}_k / \|\boldsymbol{v}_k\|$ for orthonormal basis.

Spectral Theorem

Eigenvalue Problem

$A\boldsymbol{x} = \lambda\boldsymbol{x}$, where λ is an eigenvalue and $\boldsymbol{x} \neq \mathbf{0}$ is the eigenvector.
Characteristic polynomial: $\det(A - \lambda I) = 0$

Spectral Theorem

Every real **symmetric** matrix $A \in \mathbb{R}^{n \times n}$ has real eigenvalues and mutually orthogonal eigenvectors:

$$A = Q\Lambda Q^\top, \qquad Q^\top Q = I, \qquad \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$$

- $\operatorname{tr}(A) = \sum_i \lambda_i \qquad \det(A) = \prod_i \lambda_i$
- $A \succ 0$ (positive definite) $\Leftrightarrow$ all $\lambda_i > 0$

Table of Contents

1 Analytic Geometry

2 Linear Optimization

3 Vector Calculus

4 Continuous Optimization & Duality

5 Principal Component Analysis

6 Machine Learning Models

Linear Programming Problem

Standard Form $\min_{\boldsymbol{x}} \ \boldsymbol{c}^\top \boldsymbol{x} \quad \text{s.t.} \quad A\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} \geq \boldsymbol{0}$

Inequality $\boldsymbol{a}^\top \boldsymbol{x} \leq b$ converted via slack variable $s \geq 0$: $\boldsymbol{a}^\top \boldsymbol{x} + s = b$

- Feasible region is a **convex polytope**
- **Corner-point theorem:** optimal solution occurs at a vertex (BFS)
- **BFS:** set $n-m$ non-basic variables to zero, solve for m basic variables

Simplex Method

Start at a BFS; check reduced costs $\bar{c}_j$. If all $\bar{c}_j \geq 0$: optimal (minimisation). Otherwise pivot via **minimum ratio test** (leaving variable) and most negative $\bar{c}_j$ (entering variable).

Artificial Variables & Duality

Finding an Initial BFS

Big-M method: add artificial variable a_i with penalty M to objective.

Two-phase method:
Phase I — minimise $\sum a_i$ to find a BFS.
Phase II — optimise original objective.

Primal & Dual

(P) $\min\ \boldsymbol{c}^\top \boldsymbol{x},\ A\boldsymbol{x} \geq \boldsymbol{b},\ \boldsymbol{x} \geq \boldsymbol{0}$

(D) $\max\ \boldsymbol{b}^\top \boldsymbol{y},\ A^\top \boldsymbol{y} \leq \boldsymbol{c},\ \boldsymbol{y} \geq \boldsymbol{0}$

Complementary Slackness:
At optimality: $x_i s_i = 0$,
$y_j t_j = 0$

Shadow price y_i^*: rate of change of optimal objective per unit increase in b_i.

Table of Contents

1 Analytic Geometry

2 Linear Optimization

3 Vector Calculus

4 Continuous Optimization & Duality

5 Principal Component Analysis

6 Machine Learning Models

Gradients & Taylor Expansion

Gradient

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \partial f/\partial x_1 \\ \vdots \\ \partial f/\partial x_n \end{bmatrix} \in \mathbb{R}^n$$

Points in direction of steepest ascent.

Taylor Expansion (2nd order)

Univariate:

$$f(x+h) \approx f(x) + f'(x)h + \tfrac{1}{2}f''(x)h^2$$

Multivariate:

$$f(\boldsymbol{x}+\boldsymbol{h}) \approx f(\boldsymbol{x}) + \nabla f^\top \boldsymbol{h} + \tfrac{1}{2}\boldsymbol{h}^\top H \boldsymbol{h}$$

Directional Derivative: $D_{\boldsymbol{u}} f(\boldsymbol{x}) = \nabla f(\boldsymbol{x})^\top \boldsymbol{u} = \|\nabla f\| \cos\theta$

Jacobian: $[J_f]_{ij} = \partial f_i/\partial x_j$ for $f : \mathbb{R}^n \to \mathbb{R}^m$

Hessian: $H_{ij} = \partial^2 f/\partial x_i \partial x_j$ (symmetric for smooth f)

Critical Points & Extrema Classification

First-Order Necessary Condition

$\boldsymbol{x}^*$ is a local minimum $\Rightarrow \nabla f(\boldsymbol{x}^*) = \mathbf{0}$

Second-Order Conditions (via Hessian $H(\boldsymbol{x}^*)$)

$H \succ 0$ (positive definite) $\Rightarrow$ local **minimum**

$H \prec 0$ (negative definite) $\Rightarrow$ local **maximum**

H indefinite $\Rightarrow$ **saddle point**

Multivariate Chain Rule

For $f(\boldsymbol{g}(t))$ with $\boldsymbol{g} : \mathbb{R} \to \mathbb{R}^n$:

$$\frac{d}{dt} f(\boldsymbol{g}(t)) = \nabla f(\boldsymbol{g}(t))^\top \frac{d\boldsymbol{g}}{dt} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \frac{dg_i}{dt}$$

Table of Contents

1 Analytic Geometry

2 Linear Optimization

3 Vector Calculus

4 Continuous Optimization & Duality

5 Principal Component Analysis

6 Machine Learning Models

Gradient Descent & Lagrange Multipliers

Gradient Descent (Unconstrained)

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta_k \nabla f(\boldsymbol{x}_k)$$

Moves in direction of steepest descent; η_k is the learning rate.

Constrained Optimisation & Lagrangian

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x}) \quad \text{s.t.} \quad g_i(\boldsymbol{x}) \leq 0, \quad h_j(\boldsymbol{x}) = 0$$

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_i \lambda_i g_i(\boldsymbol{x}) + \sum_j \mu_j h_j(\boldsymbol{x})$$

Stationarity Condition

Unconstrained minimum: $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$.

Constrained optimum: $\nabla_{\boldsymbol{x}} \mathcal{L} = \mathbf{0}$.

Adaptive Gradient Methods

Stochastic Gradient Descent (SGD)

Use mini-batch gradient instead of full dataset: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\, \boldsymbol{g}_t$

AdaGrad

$$G_t = G_{t-1} + g_t^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{G_t + \varepsilon}}\, g_t$$

Adapts per-parameter; shrinks step for frequent features. G_t grows monotonically (drawback).

RMSProp

$$E[g^2]_t = \rho\, E[g^2]_{t-1} + (1-\rho)\, g_t^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}}\, g_t$$

Fixes diminishing learning rate via exponential moving average.

Adam Optimizer

Adaptive Moment Estimation

Combines **momentum** (1st moment) with **adaptive learning rate** (2nd moment):

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \qquad \text{(1st moment)}$$

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

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \qquad \text{(bias correction)}$$

$$w_{t+1} = w_t - \frac{\eta\, \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \qquad \text{(update)}$$

Defaults: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$. Bias correction compensates for zero initialisation of m_0, v_0.

Lagrangian Duality & KKT Conditions

Primal & Dual

$$p^* = \min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$$

$$d^* = \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$$

Weak Duality (always holds): $d^* \leq p^*$

Strong Duality: $d^* = p^*$ under **Slater's condition** — $\exists\, \boldsymbol{x}$ strictly feasible ($g_i(\boldsymbol{x}) < 0$), f and g_i convex.

KKT Conditions

(necessary & sufficient for convex)

1. **Stationarity:** $\nabla_{\boldsymbol{x}} \mathcal{L} = \mathbf{0}$
2. **Primal feasibility:** $g_i(\boldsymbol{x}^*) \leq 0$
3. **Dual feasibility:** $\lambda_i^* \geq 0$
4. **Comp. slackness:** $\lambda_i^* g_i(\boldsymbol{x}^*) = 0$

Table of Contents

1 Analytic Geometry

2 Linear Optimization

3 Vector Calculus

4 Continuous Optimization & Duality

5 Principal Component Analysis

6 Machine Learning Models

PCA: Maximum Variance Perspective

Goal: Find M orthogonal directions $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M \in \mathbb{R}^D$ capturing maximum variance in centred data $\{\boldsymbol{x}_n\}_{n=1}^N$.

Covariance Matrix & Eigenvalue Problem

$$S = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top \in \mathbb{R}^{D \times D}$$

Maximise $\boldsymbol{b}^\top S \boldsymbol{b}$ subject to $\|\boldsymbol{b}\| = 1 \;\Rightarrow\; S\boldsymbol{b} = \lambda \boldsymbol{b}$

- **Principal components:** eigenvectors of S, sorted by $\lambda_1 \geq \lambda_2 \geq \cdots$
- **Projected data:** $\boldsymbol{y}_n = B^\top \boldsymbol{x}_n, \quad B = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M]$
- **Explained variance** of k-th PC: $\lambda_k \Big/ \sum_j \lambda_j$

PCA via SVD & Low-Rank Approximation

Singular Value Decomposition

$$X = U\Sigma V^\top, \qquad U^\top U = I, \quad V^\top V = I$$
$$\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_p), \qquad \sigma_1 \geq \cdots \geq \sigma_p \geq 0$$

Principal components $\equiv$ columns of V; $\lambda_i \propto \sigma_i^2$.

Eckart-Young Theorem

Best rank-M approximation: $\tilde{X}_M = \sum_{i=1}^{M} \sigma_i \, \boldsymbol{u}_i \boldsymbol{v}_i^\top$

Power iteration: $\boldsymbol{v}_{k+1} = A\boldsymbol{v}_k / \|A\boldsymbol{v}_k\|$ — finds dominant eigenvector iteratively

Table of Contents

1 Analytic Geometry

2 Linear Optimization

3 Vector Calculus

4 Continuous Optimization & Duality

5 Principal Component Analysis

6 Machine Learning Models

Gaussian Mixture Models

GMM

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \Sigma_k), \qquad \sum_k \pi_k = 1, \quad \pi_k \geq 0$$

Expectation-Maximization (EM)

Iteratively maximise $\sum_n \log p(\boldsymbol{x}_n)$ with latent cluster assignments.

E-step — compute responsibilities:

$$r_{nk} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_j, \Sigma_j)}$$

M-step — update parameters:

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{\sum_n r_{nk} \boldsymbol{x}_n}{\sum_n r_{nk}}, \qquad \pi_k^{\text{new}} = \frac{\sum_n r_{nk}}{N}$$

Support Vector Machine (SVM)

Hard Margin SVM

Linearly separable. Boundary: $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$, margin $= 2/\|\boldsymbol{w}\|$

$$\min_{\boldsymbol{w},b} \ \tfrac{1}{2}\|\boldsymbol{w}\|^2 \quad \text{s.t. } y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1$$

Support vectors: points on $y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) = 1$

Soft Margin SVM

Non-separable. Slack $\xi_i \geq 0$:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \ \tfrac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$

$$\text{s.t. } y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i, \ \xi_i \geq 0$$

C trades margin vs. violations.

The SVM dual (via KKT, Sec. 4) reveals that only support vectors determine the decision boundary.

Hinge Loss: $\ell_i = \max\big(0,\ 1 - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)\big)$ (zero for correctly classified points beyond margin)

Autoregressive & Moving Average Models

Stationarity

Mean, variance, and autocovariance constant over time. Non-stationary series made stationary by **differencing** (d times).

AR(p) Model

$$y_t = \mu + \sum_{i=1}^{p} \phi_i \, y_{t-i} + \varepsilon_t$$

Uses p past observations.

MA(q) Model

$$y_t = \mu + \varepsilon_t + \sum_{j=1}^{q} \theta_j \, \varepsilon_{t-j}$$

Uses q past errors.

- Use **PACF** to identify p (cuts off after lag p); use **ACF** to identify q (cuts off after lag q)
- SMA, WMA, EMA smooth the series and remove noise

ARIMA, SARIMA & Kalman Filter

ARIMA(p, d, q)
Difference d times for stationarity, then fit ARMA(p, q).

SARIMA(p, d, q)$\times$(P, D, Q, s)
Adds seasonal AR and MA terms at period s.

Kalman Filter (state-space model)
Predict:

$$\hat{\boldsymbol{x}}_{t|t-1} = F\hat{\boldsymbol{x}}_{t-1|t-1}$$
$$P_{t|t-1} = FP_{t-1|t-1}F^\top + Q$$

Update:

$$K_t = P_{t|t-1}C^\top(CP_{t|t-1}C^\top + R)^{-1}$$
$$\hat{\boldsymbol{x}}_{t|t} = \hat{\boldsymbol{x}}_{t|t-1} + K_t(\boldsymbol{y}_t - C\hat{\boldsymbol{x}}_{t|t-1})$$

K_t: Kalman gain — optimally weights prediction vs. observation.

Key Takeaways

- **Module thread:** Geometry $\rightarrow$ LP $\rightarrow$ Calculus $\rightarrow$ Optimisation $\rightarrow$ ML Models — each module builds on the previous
- **KKT conditions** (M4) are the bridge: they power the SVM dual, LASSO, and neural network training
- **PCA** (M5) uses both the Spectral Theorem (M1) and SVD — showing the full mathematical circle
- **Iterative optimisers everywhere:** EM (GMM), ARIMA fitting, and Kalman filter are all gradient-descent-style iterative schemes applied to structured models

Thank you :)