Machine Learning Cheat Sheet

Based on course materials: Linear Models, Optimization, PCA, Bayesian Learning, Clustering, NLP

1. Basics & Notation

Supervised Learning: Learning $f(\vec{x}) \approx y$ from labeled pairs $(\vec{x}, y)$ (Ground Truth).

- **Regression:** Target $y \in \mathbb{R}$ (continuous).
- **Classification:** Target y is discrete class label.

Notation:

- Input: $X = \{\vec{x}^{(n)}\}_{n=1}^N$ where $\vec{x}^{(n)} \in \mathbb{R}^D$.
- Parameters: Θ or $\vec{\theta}$ (e.g., weights $\vec{a}$, bias b).
- Prediction: $\hat{y} = f_\theta(\vec{x})$.

2. Linear Regression

Model: Linear combination of inputs.

$$f_\theta(\vec{x}) = \vec{a} \cdot \vec{x} + b \quad \text{or} \quad f(\vec{x}) = \sum_{p=0}^{P} a_p x^p \text{ (Poly)}$$

.

Cost Function (MSE): Minimizes squared errors.

$$J(\theta) = \frac{1}{N} \sum_{n=1}^{N} (f_\theta(\vec{x}_n) - y_n)^2$$

.

Gradient Descent (Optimization): Iterative update to find θ^* that minimizes J.

$$\vec{\theta} \leftarrow \vec{\theta} - \eta \vec{\nabla}_\theta J$$

where η is the **learning rate**.

Strategies:

- **Full Batch:** Uses all N examples for gradient ∇J_N.
- **Stochastic (SGD):** Uses 1 random example n.
- **Mini-Batch:** Uses subset of m examples.

3. Linear Classification

The Perceptron

Binary classifier ($y \in \{-1, +1\}$). **Model:** Linear separator (hyperplane).

$$\hat{y} = \text{sign}(\vec{w} \cdot \vec{x})$$

.

Cost Function: Sum of distances of misclassified points.

$$J = \frac{1}{N} \sum_{n=1}^{N} \max(0, -(\vec{w} \cdot \vec{x}_n) t_n)$$

.

Update Rule: For misclassified points only.

$$\vec{w} \leftarrow \vec{w} + \eta \frac{1}{N} \sum_{n \in \text{mis}} \vec{x}_n t_n$$

.

Multi-class (Softmax / Logistic)

For $K > 2$ classes. Replaces sign with Softmax probability.
Model (Softmax): Probability of class k.

$$P(y = k | \vec{x}) = \frac{\exp(\vec{w}_k \cdot \vec{x})}{\sum_j \exp(\vec{w}_j \cdot \vec{x})}$$

.

Cost Function (Cross-Entropy):

$$J = \sum_l t_l \log(\hat{y}_l)$$

where t_l is the one-hot truth vector.

SVM (Support Vector Machine)

Concept: Maximizes the **margin** between classes. Only *support vectors* (points closest to boundary or misclassified) affect the boundary.

4. Preprocessing

Standardization: Centers data on 0 with variance 1. Important when features have different units.

$$x'_{n,d} = \frac{x_{n,d} - \langle x_d \rangle}{\sigma_d}$$

.

Min-Max Scaling: Maps data to $[0, 1]$.

$$x'_{n,d} = \frac{x_{n,d} - \min(x_d)}{\max(x_d) - \min(x_d)}$$

.

One-Hot Encoding: For categorical data (no order). Class $k \to (0, ..., 1, ..., 0)$ (1 at index k).

5. PCA (Principal Component Analysis)

Goal: Project data from D to D' dimensions ($D' < D$) while maximizing variance (information).

1. Covariance Matrix:

$$C = \frac{1}{N} \sum_{n=1}^{N} (\vec{x}_n - \langle \vec{x} \rangle)(\vec{x}_n - \langle \vec{x} \rangle)^T$$

.

2. Eigen Decomposition: Find directions of max variance.

$$C \vec{u}_i = \lambda_i \vec{u}_i$$

$\vec{u}_i$: Principal direction. λ_i: Variance along $\vec{u}_i$.

3. Projection: Using matrix P of top D' eigenvectors.

$$\vec{x}' = (\vec{x} - \langle \vec{x} \rangle) P$$

.

4. Reconstruction (with loss):

$$\vec{x}_{rec} = \vec{x}' P^T + \langle \vec{x} \rangle$$

.

6. Bayesian Learning

MLE (Maximum Likelihood Estimation): Find parameters Θ that make data most probable.

$$\Theta^* = \text{argmax}_\Theta \sum_{n=1}^{N} \log P(\vec{x}_n | \Theta)$$

.

Algorithm: Finding MLE Parameters

To find the parameters Θ^* that maximize the likelihood of the data:

1. **Define Likelihood:** Formulate the likelihood function $\mathcal{L}(\Theta) = P(X|\Theta)$. Assuming N independent and identically distributed (i.i.d.) samples:

$$\mathcal{L}(\Theta) = \prod_{n=1}^{N} P(\vec{x}_n|\Theta)$$

2. **Log-Likelihood:** Take the natural logarithm to convert the product into a sum (easier to differentiate and numerically stable):

$$l(\Theta) = \log \mathcal{L}(\Theta) = \sum_{n=1}^{N} \log P(\vec{x}_n|\Theta)$$

3. **Differentiate:** Calculate the gradient of the log-likelihood with respect to the parameters Θ:

$$\nabla_\Theta l(\Theta) = \frac{\partial}{\partial \Theta} \sum_{n=1}^{N} \log P(\vec{x}_n|\Theta)$$

4. **Solve:** Set the gradient to zero to find critical points (analytical solution):

$$\nabla_\Theta l(\Theta) = 0$$

Note: If an analytical solution is impossible, use iterative optimization methods like Gradient Ascent.

5. **Verify:** Ensure the critical point is a maximum (e.g., check that the second derivative/Hessian is negative definite).

Example: Gaussian MLE

For data following $\mathcal{N}(\mu, \sigma^2)$:

- $\hat{\mu} = \frac{1}{N} \sum x_n$ (Empirical Mean).
- $\hat{\sigma}^2 = \frac{1}{N} \sum (x_n - \mu)^2$ (Empirical Variance).

Naive Bayes Classifier

Assumption: Features x_d are **independent** given class k.

$$P(\vec{x}|y = k) = \prod_{d=1}^{D} P(x_d|y = k)$$

.

Training (Bernoulli): Learn prob. of feature d in class k.

$$p_{k,d} = \frac{\sum_n x_{n,d} \mathbb{I}(y_n = k)}{\sum_n \mathbb{I}(y_n = k)}$$

Also learn class priors $\pi_k = \frac{1}{N} \sum \mathbb{I}(y_n = k)$.

Decision (MAP): Predict class k maximizing posterior.

$$k^* = \text{argmax}_k [\log \pi_k + \sum_d \log P(x_d|p_{k,d})]$$

.

7. Clustering (K-Means)

Unsupervised grouping into K clusters with centers $\vec{\mu}_k$. **Objective:** Minimize intra-cluster variance (Inertia).

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} a_{nk} ||\vec{x}_n - \vec{\mu}_k||^2$$

where $a_{nk} = 1$ if $x_n \in k$, else 0.

Algorithm: Iterate until convergence. 1. **Assignment:** Assign point to closest center.

$$k^* = \text{argmin}_k ||\vec{x}_n - \vec{\mu}_k||$$

2. **Update:** Move center to mean of its points.

$$\vec{\mu}_k = \frac{\sum_n a_{nk} \vec{x}_n}{\sum_n a_{nk}}$$

.

8. NLP (Natural Language Proc.)

Bag of Words: Vector of word counts. **TF-IDF:** Weighs words by importance (rare words > common words).

- **TF (Term Freq):** Count of term t in doc d.

$$\text{TF}(t, d) = \frac{\text{number of times term } t \text{ appears in } d}{\text{total number of words in } d}$$

- **IDF (Inv Doc Freq):** Penalizes common words.

$$IDF(t) = \log\left(\frac{1 + N_{docs}}{1 + DF(t)}\right) + 1$$

  where
  - N_{docs} is a total number of documents
  - $DF(t)$ number of documents where the term t appears

  So if a word is very common like "the", then it appears in almost every document and its IDF would be small but if the word is rare like "Cauchy inequality" it is more informative and its IDF would be higher.

- **Score:** $TF\text{-}IDF = TF(t, d) \times IDF(t)$

.

9. Pipeline & Evaluation

Data Splitting Strategy

To properly evaluate models, split data into three sets:

- **Train Set:** Used to learn model parameters θ (fitting).
- **Validation Set:** Used to optimize hyper-parameters μ (e.g., regularization coeff., number of clusters). Parameters θ are fixed here.
- **Test Set:** Used for final performance assessment. Neither parameters nor hyper-parameters are tuned here.

Fitting Issues (Bias-Variance Tradeoff)

Diagnosed using **Learning Curves** (Error vs. Training Size/Complexity). **1. Underfitting (High Bias)**

- **Characterization:** High error on **both** Training and Validation sets.
- **Cause:** Model is too simple (under-parametrized) or lacks expressiveness to capture data patterns.
- **Fixes:** Increase model complexity (e.g., add polynomial features), reduce regularization.

2. Overfitting (High Variance)

- **Characterization:** Low Training error but High Validation error (large gap between curves).
- **Cause:** Model is too complex (over-parametrized), learning noise/outliers instead of the signal.
- **Fixes:**
  - **Regularization:** Penalize large weights.
  - **More Data:** Increase training set size.
  - **Dimensionality Reduction:** Remove irrelevant features (PCA).

Handling Class Imbalance

When classes are not represented equally (e.g., rare diseases, fraud), accuracy is a misleading metric. We use rebalancing techniques:

- **Oversampling:** Increasing the number of instances in the minority class (e.g., duplicating examples).
- **Undersampling:** Decreasing the number of instances in the majority class.
- **Application:** Apply during the pre-processing phase to ensure the model does not become biased toward the majority class.

Support Vector Machines (SVM)

1. Geometric Basics

- **Hyperplane:** $w^T x + b = 0$
- **Decision Rule:** $f(x) = \text{sign}(w^T x + b)$
- **Geometric Margin:** $\gamma = \frac{2}{||w||}$

2. Primal Optimization

Hard Margin (Linearly Separable):

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1, \quad \forall i \end{aligned}$$

Soft Margin (With Slack ξ_i):

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned}$$

Note: C controls bias-variance trade-off (Large C = Harder margin).

3. Dual Formulation

Maximize w.r.t Lagrange multipliers α_i:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned}$$

Prediction: $f(x) = \text{sign}(\sum_{i \in SV} \alpha_i y_i K(x_i, x) + b)$

4. Common Kernels $K(x, x')$

- **Linear:** $x^T x'$
- **Polynomial:** $(\gamma x^T x' + r)^d$
- **RBF (Gaussian):** $\exp(-\gamma||x - x'||^2)$

5. Hinge Loss

Used for Gradient Descent optimization:

$$L(y, f(x)) = \max(0, 1 - y \cdot f(x))$$

10. KNN

We are already given points and we need to classify a new point, we just look at the points closest to the new one, look at their classes and take the class of majority.

11. Models

Model	Type of Problem it Resolves
Linear Regression	Regression (Supervised); Optimization and Gradient Descent explanation
Polynomial Regression	Regression (Supervised); Modeling non-linear relationships
Perceptron	Binary Classification (Supervised)
Linear Classifiers	Classification (Supervised); Separating hyperplanes
Support Vector Machine (SVM)	Classification (Supervised); Handles linear and non-linearly separable data
Naïve Bayes	Classification (Supervised); Assumes feature independence
Gaussian Naïve Bayes	Classification (Supervised); Features modelled with Gaussian distributions
Gaussian Model (Non-Naïve)	Classification (Supervised); Modeling correlated dimensions
Minimum Distance Classifier	Classification (Supervised); Assigns points to closest class representative
Decision Trees	Classification (Supervised); Sequence of threshold-based questions
Random Forest Regressor	Regression (Supervised); Learning curve analysis
k-Nearest Neighbours (k-NN)	Classification (Supervised)
Principal Component Analysis (PCA)	Dimensionality Reduction, Visualisation, Data Compression (Unsupervised)
K-Means	Clustering (Unsupervised)
Gaussian Mixture Models	Clustering / Density Estimation (Unsupervised)
Density Estimation (Histograms, KDE)	Density Estimation (Unsupervised); Modeling probability density
Bag of Words (BOW)	Text Representation (NLP); Based on word frequency
TF-IDF	Text Representation / Tokenisation (NLP)
Word Embeddings	Text Representation (NLP); Semantic proximity mapping
Word2Vec	Text Embedding Architecture (NLP)
Skip-gram	Context Word Prediction (NLP)
Continuous Bag-Of-Words (CBOW)	Target Word Prediction (NLP)