

Model Error

Empirical Risk $\hat{R}_D(f) = \frac{1}{n} \sum \ell(y, f(x))$
Population Risk $R(f) = \mathbb{E}_{x,y \sim p} [\ell(y, f(x))]$
It holds that $\mathbb{E}_D[\hat{R}_D(\hat{f})] \leq R(\hat{f})$. We call $R(\hat{f})$ the generalization error.

Bias Variance Tradeoff:

Pred. error = **Bias**² + **Variance** + **Noise**
 $\mathbb{E}_D[R(\hat{f})] = \mathbb{E}_x[f^*(x) - \mathbb{E}_D[\hat{f}_D(x)]]^2 + \mathbb{E}_x[\mathbb{E}_D[(\hat{f}_D(x) - \mathbb{E}_D[\hat{f}_D(x)])^2]] + \sigma$

Bias: how close \hat{f} can get to f^*

Variance: how much \hat{f} changes with D

Regression

Squared loss (convex)

$$\frac{1}{n} \sum (y_i - f(x_i))^2 = \frac{1}{n} \|y - Xw\|_2^2$$
$$\nabla_w L(w) = 2X^T(Xw - y)$$

Solution: $\hat{w} = (X^T X)^{-1} X^T y$

Regularization

Lasso Regression (sparse)

$$\operatorname{argmin}_{w \in \mathbb{R}^d} \|y - \Phi w\|_2^2 + \lambda \|w\|_1$$

Ridge Regression

$$\operatorname{argmin}_{w \in \mathbb{R}^d} \|y - \Phi w\|_2^2 + \lambda \|w\|_2^2$$

$$\nabla_w L(w) = 2X^T(Xw - y) + 2\lambda w$$

Solution: $\hat{w} = (X^T X + \lambda I)^{-1} X^T y$

large $\lambda \Rightarrow$ larger bias but smaller variance

Cross-Validation

- For all folds $i = 1, \dots, k$:
 - Train \hat{f}_i on $D' - D'_i$
 - Val. error $R_i = \frac{1}{|D'_i|} \sum \ell(\hat{f}_i(x), y)$

- Compute CV error $\frac{1}{k} \sum_{i=1}^k R_i$
- Pick model with lowest CV error

Gradient Descent

Converges only for convex case.

$$w^{t+1} = w^t - \eta_t \cdot \nabla \ell(w^t)$$

For linear regression:

$$\|w^t - w^*\|_2 \leq \|I - \eta X^T X\|_{op}^t \|w^0 - w^*\|_2$$

$\rho = \|I - \eta X^T X\|_{op}$ conv. speed for const. η .

Opt. fixed $\eta = \frac{2}{\lambda_{\min} + \lambda_{\max}}$ and max. $\eta \leq \frac{2}{\lambda_{\max}}$.

Momentum: $w^{t+1} = w^t + \gamma \Delta w^{t-1} - \eta_t \nabla \ell(w^t)$

Classification

Zero-One loss not convex or continuous

$$\ell_{0-1}(\hat{f}(x), y) = \mathbb{I}_{y \neq \operatorname{sgn} \hat{f}(x)}$$

Logistic loss $\log(1 + e^{-y \hat{f}(x)})$

$$\nabla \ell(\hat{f}(x), y) = \frac{-y_i x_i}{1 + e^{y_i \hat{f}(x)}}$$

Hinge loss $\max(0, 1 - y \hat{f}(x))$

Softmax $p(1|x) = \frac{1}{1 + e^{-f(x)}}, p(-1|x) = \frac{1}{1 + e^{f(x)}}$

Multi-Class $\hat{p}_k = e^{f_k(x)} / \sum_{i=1}^K e^{f_i(x)}$

Linear Classifiers

$f(x) = w^T x$, the decision boundary $f(x) = 0$.

If data is lin. sep., grad. desc. converges to

Maximum-Margin Solution:

$w_{MM} = \operatorname{argmax} \operatorname{margin}(w)$ with $\|w\|_2 = 1$

Where $\operatorname{margin}(w) = \min_i y_i w^T x_i$.

Support Vector Machines

Hard SVM

$$\hat{w} = \min_w \|w\|_2 \text{ s.t. } \forall_i y_i w^T x_i \geq 1$$

Soft SVM allow "slack" in the constraints

$$\hat{w} = \min_{w, \xi} \frac{1}{2} \|w\|_2^2 + \lambda \sum_{i=1}^n \underbrace{\max(0, 1 - y_i w^T x_i)}_{\text{hinge loss}}$$

Choose +1 as the more important class.

True Class			
		$y=+1$	$y=-1$
Prediction	$\hat{y}=+1$	TP	FP
	$\hat{y}=-1$	FN	TN
		error ₁ /FPR : $\frac{FP}{TN+FP}$	error ₂ /FNR : $\frac{FN}{TP+FN}$
		Precision : $\frac{TP}{TP+FP}$	TPR / Recall : $\frac{TP}{TP+FN}$

AUROC: Plot TPR vs. FPR and compare different ROC's with area under the curve.

F1-Score: $\frac{2TP}{2TP + FP + FN}$, Accuracy : $\frac{TP+TN}{P+N}$

Goal: large recall and small FPR.

Kernels

Parameterize: $w = \Phi^T \alpha, K = \Phi \Phi^T$

A kernel is **valid** if K is sym.: $k(x, z) = k(z, x)$

and psd: $z^T K z \geq 0$

lin.: $k(x, z) = x^T z$, **poly.:** $k(x, z) = (x^T z + 1)^m$

rbf: $k(x, z) = \exp(-\frac{\|x-z\|_2}{\tau})$

$\alpha = 1 \Rightarrow$ laplacian kernel

$\alpha = 2 \Rightarrow$ gaussian kernel

Kernel composition rules

$k = k_1 + k_2, \quad k = k_1 \cdot k_2 \quad \forall c > 0. k = c \cdot$

$k_1, \forall f$ convex. $k = f(k_1)$, holds for polynoms

with pos. coefficients or exp function.

$\forall f. k(x, y) = f(x) k_1(x, y) f(y)$

Mercers Theorem: Valid kernels can be decomposed into a lin. comb. of inner products.

Kern. Ridge Reg. $\frac{1}{n} \|y - K\alpha\|_2^2 + \lambda \alpha^T K \alpha$

KNN Classification

- Pick k and distance metric d
- For given x , find among $x_1, \dots, x_n \in D$ the k closest to $x \rightarrow x_{i_1}, \dots, x_{i_k}$
- Output the majority vote of labels

Neural Networks

w are the weights and $\phi: \mathbb{R} \mapsto \mathbb{R}$ is a nonlinear

activation function: $\phi(x, w) = \phi(w^T x)$

ReLU: $\max(0, z)$, **Tanh:** $\frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$

Sigmoid: $\frac{1}{1 + \exp(-z)}$

Universal Approximation Theorem: We can approximate any arbitrary smooth target function, with 1+ layer with sufficient width.

Forward Propagation

Input: $v^{(0)} = [x; 1]$ Output: $f = W^{(L)} v^{(L-1)}$

Hidden: $z^{(l)} = W^{(l)} v^{(l-1)}, v^{(l)} = [\phi(z^{(l)}); 1]$

Backpropagation

Non-convex optimization problem:

$$(\nabla_{w^{(L)}} \ell)^T = \frac{\partial \ell}{\partial W^{(L)}} = \frac{\partial \ell}{\partial f} \frac{\partial f}{\partial W^{(L)}}$$

$$(\nabla_{w^{(L-1)}} \ell)^T = \frac{\partial \ell}{\partial W^{(L-1)}} = \frac{\partial \ell}{\partial f} \frac{\partial f}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial W^{(L-1)}}$$

$$(\nabla_{w^{(L-2)}} \ell)^T = \frac{\partial \ell}{\partial W^{(L-2)}} = \frac{\partial \ell}{\partial f} \frac{\partial f}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial W^{(L-2)}}$$

Only compute **the gradient**. Rand. init.

weights by distr. assumption for ϕ . ($2/n_{in}$ for ReLu and $1/n_{in}$ or $1/(n_{in} + n_{out})$ for Tanh)

Overfitting

Regularization; Early Stopping; Dropout: ignore hidden units with prob. p , after training use all units and scale weights by p ;
Batch Normalization: normalize the input data (mean 0, variance 1) in each layer

CNN $\phi(W * v^{(l)})$

The output dimension when applying m different $f \times f$ filters to an $n \times n$ image with padding p and stride s is: $l = \frac{n+2p-f}{s} + 1$
For each channel there is a separate filter.

Unsupervised Learning

k-Means Clustering

Optimization Goal (non-convex):

$$\hat{R}(\mu) = \sum_{i=1}^n \min_{j \in \{1, \dots, k\}} \|x_i - \mu_j\|_2^2$$

Lloyd's heuristics: Init. cluster centers $\mu^{(0)}$:

- Assign points to closest center
- Update μ_i as mean of assigned points

Converges in exponential time.

Initialize with **k-Means++**:

- Random data point $\mu_1 = x_i$
- Add μ_2, \dots, μ_k rand., with prob: given $\mu_{1:j}$ pick $\mu_{j+1} = x_i$

where $p(i) = \frac{1}{z} \min_{l \in \{1, \dots, j\}} \|x_i - \mu_l\|_2^2$

Converges in expectation $\mathcal{O}(\log k)$ *
opt. solution. Find k by negligible loss de-

crease or reg.

Principal Component Analysis

Optimization goal: $\operatorname{argmin}_{\sum_{i=1}^n \|x_i - z_i w\|_2^2, \|w\|_2=1, z}$

The optimal solution is given by $z_i = w^T x_i$.
Substituting gives us:

$$\hat{w} = \operatorname{argmax}_{\|w\|_2=1} w^T \Sigma w$$

Where $\Sigma = \frac{1}{n} \sum_{i=1}^n x_i x_i^T$ is the empirical covariance. Closed form solution given by the principal eigenvector of Σ , i.e. $w = v_1$ for $\lambda_1 \geq \dots \geq \lambda_d \geq 0$: $\Sigma = \sum_{i=1}^d \lambda_i v_i v_i^T$
For $k > 1$ we have to change the normalization to $W^T W = I$ then we just take the first k principal eigenvectors so that $W = [v_1, \dots, v_k]$.

PCA through SVD

The first k columns of V where $X = U S V^T$.

Kernel PCA

$\Sigma = \frac{1}{n} \sum_{i=1}^n x_i x_i^T = X^T X \Rightarrow$ kernel trick:

$$\hat{\alpha} = \operatorname{argmax}_{\alpha} \frac{\alpha^T K^T K \alpha}{\alpha^T K \alpha}$$

Closed form solution:

$$\alpha^{(i)} = \frac{1}{\sqrt{\lambda_i}} v_i \quad K = \sum_{i=1}^n \lambda_i v_i v_i^T, \lambda_1 \geq \dots \geq 0$$

A point x is projected as: $z_i = \sum_{j=1}^n \alpha_j^{(i)} k(x_j, x)$

Autoencoders

We want to minimize $\frac{1}{n} \sum_{i=1}^n \|x_i - \hat{x}_i\|_2^2$.

$$\hat{x} = f_{dec}(f_{enc}(x, \theta_{enc}); \theta_{dec})$$

Lin. activation func. & square loss \Rightarrow PCA

Statistical Perspective

Assume that data is generated iid. by some $p(x, y)$. We want to find $f: X \mapsto Y$ that minimizes the **population risk**.

Opt. Predictor for the Squared Loss

f minimizing the population risk:

$$f^*(x) = \mathbb{E}[y | X = x] = \int y \cdot p(y | x) dy$$

Estimate $\hat{p}(y | x)$ with MLE:

$$\theta^* = \operatorname{argmax}_{\theta} \hat{p}(y_1, \dots, y_n | x_1, \dots, x_n, \theta)$$

$$= \operatorname{argmin}_{\theta} - \sum_{i=1}^n \log p(y_i | x, \theta)$$

The MLE for linear regression is unbiased and has minimum variance among all unbiased estimators. However, it can overfit.

Ex. Conditional Linear Gaussian

Assume Gaussian noise $y = f(x) + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ and $f(x) = w^T x$:

$$\hat{p}(y | x, \theta) = \mathcal{N}(y; w^T x, \sigma^2)$$

The optimal \hat{w} can be found using MLE:

$$\hat{w} = \operatorname{argmax}_w p(y | x, \theta) = \operatorname{argmin}_w \sum (y_i - w^T x_i)^2$$

Maximum a Posteriori Estimate

Introduce bias to reduce variance. The small weight assumption is a Gaussian prior $w_i \sim \mathcal{N}(0, \beta^2)$. The posterior distribution of w is given by: $p(w | x, y) = \frac{p(w) \cdot p(y | x, w)}{p(y | x)}$

Now we want to find the MAP for w : $\hat{w} = \operatorname{argmax}_w p(w | \tilde{x}, \tilde{y})$

$$\begin{aligned} &= \operatorname{argmin}_w -\log \frac{p(w) \cdot p(y | x, w)}{p(y | x)} \\ &= \operatorname{argmin}_w \frac{\sigma^2}{\beta^2} \|w\|_2^2 + \sum_{i=1}^n (y_i - w^\top x_i)^2 \end{aligned}$$

Regularization can be understood as MAP inference, with different priors (= regularizers) and likelihoods (= loss functions).

Statistical Models for Classification

f minimizing the population risk:

$$f^*(x) = \operatorname{argmax}_{\hat{y}} p(\hat{y} | x)$$

This is called the Bayes' optimal predictor for the 0-1 loss. Assuming iid. Bernoulli noise, the conditional probability is:

$$p(y | x, w) \sim \operatorname{Ber}(y; \sigma(w^\top x))$$

Where $\sigma(z) = \frac{1}{1 + \exp(-z)}$ is the sigmoid function. Using MLE we get:

$$\hat{w} = \operatorname{argmin}_w \sum_{i=1}^n \log(1 + \exp(-y_i w^\top x_i))$$

Which is the logistic loss. Instead of MLE we can estimate MAP, e.g. with a Gaussian prior:

$$\hat{w} = \operatorname{argmin}_w \|w\|_2^2 + \sum_{i=1}^n \log(1 + e^{-y_i w^\top x_i})$$

Bayesian Decision Theory

Given $p(y | x)$, a set of actions A and a cost $C : Y \times A \mapsto \mathbb{R}$, pick the action with the maximum expected utility.

$$a^* = \operatorname{argmin}_{a \in A} \mathbb{E}_y[C(y, a) | x]$$

Can be used for asymmetric costs or abstention.

Generative Modeling

Aim to estimate $p(x, y)$ for complex situations using Bayes' rule: $p(x, y) = p(x | y) \cdot p(y)$

Naive Bayes Model

GM for classification tasks. Assuming for a class label, each feature is independent. This helps estimating $p(x | y) = \prod_{i=1}^d p(x_i | y_i)$.

Gaussian Naive Bayes Classifier

Naive Bayes Model with Gaussians features. Estimate the parameters via MLE:

MLE for class prior: $p(y) = \hat{p}_y = \frac{\operatorname{Count}(Y=y)}{n}$

MLE for feature distribution:

Where: $p(x_i | y) = \mathcal{N}(x_i; \hat{\mu}_{y,i}, \hat{\sigma}_{y,i}^2)$

$$\hat{\mu}_{y,i} = \frac{1}{\operatorname{Count}(Y=y)} \sum_{j | y_j=y} x_{j,i}$$

$$\hat{\sigma}_{y,i}^2 = \frac{1}{\operatorname{Count}(Y=y)} \sum_{j | y_j=y} (x_{j,i} - \hat{\mu}_{y,i})^2$$

Predictions are made by:

$$y = \operatorname{argmax}_{\hat{y}} p(\hat{y} | x) = \operatorname{argmax}_{\hat{y}} p(\hat{y}) \cdot \prod_{i=1}^d p(x_i | \hat{y})$$

Equivalent to decision rule for bin. class.:

$$y = \operatorname{sgn} \left(\log \frac{p(Y=+1 | x)}{p(Y=-1 | x)} \right)$$

Where $f(x)$ is called the discriminant function. If the conditional independence assumption is violated, the classifier can be overconfident.

Gaussian Bayes Classifier

No independence assumption, model the features with a multivariate Gaussian $\mathcal{N}(x; \mu_y, \Sigma_y)$:

$$\begin{aligned} \mu_y &= \frac{1}{\operatorname{Count}(Y=y)} \sum_{j | y_j=y} x_j \\ \Sigma_y &= \frac{1}{\operatorname{Count}(Y=y)} \sum_{j | y_j=y} (x_j - \hat{\mu}_y)(x_j - \hat{\mu}_y)^\top \end{aligned}$$

This is also called the **quadratic discriminant analysis** (QDA). LDA: $\Sigma_+ = \Sigma_-$, Fisher LDA: $p(y) = \frac{1}{2}$, Outlier detection: $p(x) \leq \tau$.

Avoiding Overfitting

MLE is prone to overfitting. Avoid this by restricting model class (fewer parameters, e.g. GNB) or using priors (restrict param. values).

Generative vs. Discriminative

Discriminative models:

$p(y | x)$, can't detect outliers, more robust

Generative models:

$p(x, y)$, can be more powerful (detect outliers, missing values) if assumptions are met, are typically less robust against outliers

Gaussian Mixture Model

Assume that data is generated from a convex combination of Gaussian distributions:

$p(x | \theta) = p(x | \mu, \Sigma, w) = \sum_{j=1}^k w_j \mathcal{N}(x; \mu_j, \Sigma_j)$
We don't have labels and want to cluster this data. The problem is to estimate the param. for the Gaussian distributions.

$$\operatorname{argmin}_{\theta} -\sum_{i=1}^n \log \sum_{j=1}^k w_j \cdot \mathcal{N}(x_i | \mu_j, \Sigma_j)$$

This is a non-convex objective. Similar to training a GBC without labels. Start with guess for our parameters, predict the unknown labels and then impute the missing data. Now we can get a closed form update.

Hard-EM Algorithm

E-Step: predict the most likely class for each data point:

$$\begin{aligned} z_i^{(t)} &= \operatorname{argmax}_z p(z | x_i, \theta^{(t-1)}) \\ &= \operatorname{argmax}_z p(z | \theta^{(t-1)}) \cdot p(x_i | z, \theta^{(t-1)}) \end{aligned}$$

M-Step: compute MLE of $\theta^{(t)}$ as for GBC.

Problems: labels if the model is uncertain, tries to extract too much inf. Works poorly if clusters are overlapping. With uniform weights and spherical covariances is equivalent to k-Means with Lloyd's heuristics.

Soft-EM Algorithm

E-Step: calculate the cluster membership weights for each point ($w_j = \pi_j = p(Z = j)$):

$$\gamma_j^{(t)}(x_i) = p(Z = j | D) = \frac{w_j \cdot p(x_i; \theta_j^{(t-1)})}{\sum_k w_k \cdot p(x_i; \theta_k^{(t-1)})}$$

M-Step: compute MLE with closed form:

$$\begin{aligned} w_j^{(t)} &= \frac{1}{n} \sum_{i=1}^n \gamma_j^{(t)}(x_i) \quad \mu_j^{(t)} = \frac{\sum_{i=1}^n x_i \gamma_j^{(t)}(x_i)}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)} \\ \Sigma_j^{(t)} &= \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i) (x_i - \mu_j^{(t)})(x_i - \mu_j^{(t)})^\top}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)} \end{aligned}$$

Init. the weights as uniformly distributed, rand. or with k-Means++ and for variances use spherical init. or empirical covariance of the data. Select k using cross-validation.

Degeneracy of GMMs

GMMs can overfit with limited data. Avoid this by add $v^2 I$ to variance, so it does not collapse (equiv. to a Wishart prior on the covariance matrix). Choose v by cross-validation.

Gaussian-Mixture Bayes Classifiers

Assume that $p(x | y)$ for each class can be modelled by a GMM.

$$p(x | y) = \sum_{j=1}^{k_y} w_j^{(y)} \mathcal{N}(x; \mu_j^{(y)}, \Sigma_j^{(y)})$$

Giving highly complex decision boundaries:

$$p(y | x) = \frac{1}{z} p(y) \sum_{j=1}^{k_y} w_j^{(y)} \mathcal{N}(x; \mu_j^{(y)}, \Sigma_j^{(y)})$$

GMMs for Density Estimation

Can be used for anomaly detection or data imputation. Detect outliers, by comparing the estimated density against τ . Allows to control the FP rate. Use ROC curve as evaluation criterion and optimize using CV to find τ .

General EM Algorithm

E-Step: Take the expected value over latent variables z to generate likelihood function Q : $Q(\theta; \theta^{(t-1)}) = \mathbb{E}_Z[\log p(X, Z | \theta) | X, \theta^{(t-1)}]$

$$\begin{aligned} &= \sum_{i=1}^n \sum_{z_i=1}^k \gamma_{z_i}(x_i) \log p(x_i, z_i | \theta) \end{aligned}$$

with $\gamma_{z_i}(x) = p(z | x, \theta^{(t-1)})$

M-Step: Compute MLE / Maximize:

$$\theta^{(t)} = \operatorname{argmax}_{\theta} Q(\theta; \theta^{(t-1)})$$

We have monotonic convergence, each EM-iteration increases the data likelihood.

GANs

Learn f : "simple" distr. \mapsto non linear distr. Computing likelihood of the data becomes hard, therefore we need a different loss.

$$\min_{w_G} \max_{w_D} \mathbb{E}_{x \sim p_{\text{data}}} [\log D(x, w_D)] + \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z, w_G), w_D))]$$

Training requires finding a saddle point, always converges to saddle point with if G, D have enough capacity. For a fixed G , the optimal discriminator is:

$$D_G(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}$$

The prob. of being fake is $1 - D_G$. Too powerful discriminator could lead to memorization of finite data. Other issues are oscillations/divergence or mode collapse.

One possible performance metric:

$$DG = \max_{w_D} M(w_G, w_D') - \min_{w_G'} M(w_G', w_D)$$

Where $M(w_G, w_D)$ is the training objective.

Various

Derivatives:

$$\begin{aligned} \nabla_x x^\top A &= A \quad \nabla_x a^\top x = \nabla_x x^\top a = a \\ \nabla_x b^\top A x &= A^\top b \quad \nabla_x x^\top x = 2x \quad \nabla_x x^\top A x = 2A x \\ \nabla_w \|y - Xw\|_2^2 &= 2X^\top (Xw - y) \end{aligned}$$

Bayes Theorem:

$$p(y | x) = \frac{1}{p(x)} \underbrace{p(y) \cdot p(x | y)}$$

Normal Distribution:

$$\mathcal{N}(x; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{(x - \mu)^\top \Sigma^{-1} (x - \mu)}{2}\right)$$

Other Facts

$\operatorname{Tr}(AB) = \operatorname{Tr}(BA)$, $\operatorname{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$, $X \in \mathbb{R}^{n \times d}$: $X^{-1} \rightarrow \mathcal{O}(d^3)$ $X^\top X \rightarrow \mathcal{O}(nd^2)$,

$$\binom{n}{k} = \frac{n!}{(n-k)!k!}, \|w^\top w\|_2 = \sqrt{w^\top w}$$

$$\operatorname{Cov}[X] = \mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^\top]$$

$$p(z | x, \theta) = \frac{p(x, z | \theta)}{p(x | \theta)}$$

Convexity

0: $L(\lambda w + (1 - \lambda)v) \leq \lambda L(w) + (1 - \lambda)L(v)$

1: $L(w) + \nabla L(w)^\top (v - w) \leq L(v)$

2: Hessian $\nabla^2 L(w) \succeq 0$ (psd)

- $\alpha f + \beta g$, $\alpha, \beta \geq 0$, convex if f, g convex
- $f \circ g$, convex if f convex and g affine or f non-decreasing and g convex
- $\max(f, g)$, convex if f, g convex