Machine Learning

Cheat Sheet

Thomas Karl

July 28, 2021

Contents

Part I.

Classical ML

1. Estimation Theory

1.1. Maximum Likelihood

data matrix $X = (\vec{x}_1, \dots, \vec{x}_n)$, model M with parameter vector $\vec{\Theta}$

probability density $p(X|\vec{\Theta}, M)$, *e. g.* Gauss $p(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$

statistically independent samples: $p(X|\vec{\Theta}) = \prod_{i=1}^{n} p(\vec{x}_i|\vec{\Theta})$ joint probability density

$p(X|\vec{\Theta}) = p(X)$ density, $p(X|\vec{\Theta}) = p(\vec{\Theta}) = L(\Theta)$ likelihood

Task: $L(\vec{\Theta}_{\mathsf{ML}}) = \max_{\vec{\Theta}} L(\vec{\Theta})$

neg-log-likelihood $\mathcal{L}(\vec{\Theta}) = -\ln L(\vec{\Theta}) = -\sum_{i=1}^{n} \ln p(\vec{x}_n|\vec{\Theta})$

Theorem 1 (Maximum-Likelihood Method)

$$\frac{\partial \mathcal{L}(\vec{\Theta})}{\partial \vec{\Theta}} = \sum_{i=1}^{n} \frac{1}{p(\vec{x}_i|\vec{\Theta})} \frac{\partial p(\vec{x}_i|\vec{\Theta})}{\partial \vec{\Theta}} = \vec{0} \tag{1.1}$$

Properties:

- asymptotically unbiased: $\lim_{n\to\infty} E\left[\vec{\Theta}_{\mathsf{ML}}\right] = \vec{\Theta}_0$
- asymptotically consistent:
  $\lim_{n\to\infty} \text{prob}\left\{\|\vec{\Theta}_{\mathsf{ML}} - \vec{\Theta}_0\| \le \varepsilon\right\} = 1$, $\lim_{n\to\infty} E\left[\|\vec{\Theta}_{\mathsf{ML}} - \vec{\Theta}_0\|^2\right] = 0$
- asymptotically efficient, estimation reaches Cramer-Rao Bound, expectation is restricted by

the inverse Fisher information matrix J: $E\{\text{Cov}|\Theta\} \geq J^{-1}$ with

$\text{Cov} = (\vec{\Theta} - \vec{\Theta}_{\text{ML}})(\vec{\Theta} - \vec{\Theta}_{\text{ML}})^T$ error covariance matrix and

$J = E\left\{\left[\frac{\partial}{\partial\vec{\Theta}} \ln p(\vec{x}|\vec{\Theta})\right]\left[\frac{\partial}{\partial\vec{\Theta}} \ln p(\vec{x}|\vec{\Theta})\right]^T |\vec{\Theta}\right\}$

probability density function converges to gaussian for $n \to \infty$ with $\mu = \vec{\Theta}_0$ (central limit theorem)

Theorem 2 (Bayes)

$$P(\vec{\Theta}|X) = \frac{P(X|\vec{\Theta})P(\vec{\Theta})}{P(X)} \tag{1.2}$$

with $P(\vec{\Theta}|X)$ a posterior probability, $P(X|\vec{\Theta})$ likelihood, $P(\vec{\Theta})$ a priori probability of the model, $P(X)$ a priori probability of the data

maximum a posterior $\vec{\Theta}_{\text{MAP}} = \text{argmax}_{\vec{\Theta}} \left(p(X|\vec{\Theta}, M)p(\vec{\Theta}|M)\right)$

advantages:

- accounts for objective criteria and a priori knowledge
- priors get explicit
- comparison of priors and models
- prior $\to 0$ for $n \to \infty$

1.2. Expectation Maximization Algorithm

latent variables Z, complete likelihood $L_C(\vec{\Theta}|X, Z)$

E-step: expectation of L_C $Q(\vec{\Theta}|\vec{\Theta^l}) = E[L_C(\vec{\Theta}|X, Z)|X, \vec{\Theta^l}]$

M-step: $\Theta^{\vec{l+1}} = \text{argmax}_{\vec{\Theta}}(Q(\vec{\Theta}|\vec{\Theta^l}))$

$L_C(\Theta^{\vec{l}+1}|X) \geq L_C(\Theta^{\vec{l}+1}|X)$

Mixtures

gaussian mixture $p(x) = \sum_{k=1}^{K} p(x|G_k)P(G_k)$ with density p and a priori P

k-Means Clustering

codebook vectors $m_k \in \mathbb{R}^M$

- $\hat{b}_k^n \leftarrow 1 \text{ if } \|x^n - m_k\| = \min_l \|x^n - m_l\|$ else 0
- $\hat{m}_k \leftarrow \sum_n \frac{\hat{b}_k x^n}{\sum_n \hat{b}_k^n}$

2. Classification and Regression

2.1. Decision Trees and Random Forests

parameter: function of probability distribution

statistic: function of sample x

B bootstrap samples, replication $\hat{\vec{\Theta}}^*(b) = s(x_b^*)$, $b = 1 \dots B$

standard error $\hat{e}_B = \sqrt{\mathrm{Var}(\vec{\Theta})}$

CART: s_{opt} best split for feature x_m

N samples, M features

$m << M$ variables selected at random at node t

score criterion $L(y, \hat{y}) = (y - \hat{y})^2$

regression: $\hat{c}_m = \frac{1}{N_m} \sum\limits_{x_n \in \hat{R}_m} y_n$, $Q_m(T) = \frac{1}{N_m} \sum\limits_{x_n \in \hat{R}_m} (y_n - \hat{c}_m)^2$

classification: $\phi(\vec{p}) = \sum\limits_j p_j(1 - p_j)$ Gini impurity $m = \sqrt{M}$, $\phi(\vec{p}) = -\sum\limits_j p_j \log p_j$ entropy $m = \frac{M}{3}$

bagging: random forest with $m = M$

2.2. Support Vector Machines

N dimensional data in two classes linear separated by $N-1$ dimensional hyperplane h with normal vector w

$$\langle w,x\rangle + b \begin{cases} >0 & x \in \text{pos. halfspace} \\ =0 & x \in h \\ <0 & x \in \text{neg. halfspace} \end{cases}$$

h does not contain origin: $\langle w,x\rangle = \text{const.}$

for closest points x_1 and x_2 $\langle w, x_1 - x_2\rangle = 2 \qquad /\|w\|$

$\|w\|^{-1}$ margin $\Rightarrow$minimize $\|w\|$

cost function $\tau(w) = \frac{1}{2}\|w\|^2$

lagrangian $L(w,b,\alpha_{1\ldots m}) = \frac{1}{2}\|w\|^2 - \sum\limits_{i=1}^{m} \alpha_i(y_i(\langle x_i, w\rangle + b) - 1)$, labels $y_i = \pm 1$

task: maximize with respect to dual variables (lagrange multiplier) $\alpha_i \geq 0$, minimize with respect to primal variables w, b

$\Rightarrow \sum\limits_{i=1}^{m} \alpha_i y_i = 0 \neq 0 \text{ for } \alpha_i > 0$, $w = \sum\limits_{i=1}^{m} \alpha_i y_i x_i$ (dual optimization problem) $\Rightarrow$quadratic optimization

$\Rightarrow$only support vectors count

data not linear separable: slack variable $y_i(\langle x_i, w\rangle + b) \geq 1 - \varepsilon_i$

$\tau(w,\varepsilon) = \frac{1}{2}\|w\|^2 + C\sum\limits_{i=1}^{m} \varepsilon_i^k$, $0 \leq \alpha_i \leq \frac{C}{m}$

Kernel Methods

$\vec{\phi} = \phi(\vec{x})$ non-linear mapping from input to feature space

kernel function $k(\vec{x}, \vec{x}') = \langle \phi(\vec{x}), \phi(\vec{x}')\rangle = \phi^T(\vec{x})\phi(\vec{x}') = \vec{\phi}^T\vec{\phi}' = k(\vec{x}', \vec{x})$

stationary: $k(\vec{x}, \vec{x}') = k(\vec{x} - \vec{x}')$ translation invariant space

homogeneous: $k(\vec{x}, \vec{x}') = k(\|\vec{x} - \vec{x}'\|)$

3. Subspace Methods and Exploratory Matrix Factorization

de-mixing basis vectors W, sources, components, projections $Z = WX$

linear models: algebraic approach

PCA $\Rightarrow Z = U^T X$

BSS, GEVD, ICA, NMF

non-linear: kernels

feature space $Z = W\phi^T(X_B)\phi(X)$

PCA in feature space: KPCA, kernel trick

3.1. Singular Value Decomposition

task: decompose $N \times M$ data matrix $X = U\Sigma V^T$

U: $M \times M$ orthogonal

V: $N \times N$ orthogonal

Σ: $\mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ contains $r = \min(N, M)$ singular values

covariance matrix $\mathrm{Cov} = XX^T = U\Sigma\Sigma^T U^T = UDU^T$ eigenvalue decomposition kernel matrix $K = X^T X = VDV^T$, number of eigenvalues > 0 is $\min(N, M)$

$D = \Sigma\Sigma^T = \Sigma^T\Sigma$, $\sigma_i = \sqrt{\lambda_i}$ eigenvalues

$\sigma_1 > \sigma_2 > \ldots$, $X = \sigma_1 u_1 v_1^T + \ldots$

3.2. Principal Component Analysis

mean centered data: covariance matrix equals correlation matrix

u_i is associated with λ_i, $z = u^T x$, u_1 with maximum variance? $\Rightarrow u_1^T u_1 = 1$

$\Rightarrow \mathrm{Cor} \cdot u_1 = \alpha_1 u_1$

$\Rightarrow$component with largest eigenvalue $\frac{\partial}{\partial u_1}\left\{u_1^T c u_1 - \alpha_1(u_1^T u_1 - 1)\right\} = 0$

PCA: $u_m^T = \tilde{z}_m \longleftrightarrow u_m u_m^T X = u_m \tilde{z}_m$

$Z = U_L^T X = U_L^T U \Sigma V^T = \Sigma_L V_L^T = D_L^{\frac{1}{2}} V_L^T$, $ZZ^T = D$, $U_L^T U = [1\ 0]$, projections are related with eigenvectors of V $\Rightarrow$non-correlated data set

whitening transformation: $YY^T = \mathbb{1}$

$\Rightarrow D^{-\frac{1}{2}} Z Z^T D^{-\frac{1}{2}} = \mathbb{1}$

$\Rightarrow Y = D_L^{\frac{1}{2}} U_L^T X = B^T X$, $B = U_L U_L^{-\frac{1}{2}}$

kernel trick: $U = XVD^{-\frac{1}{2}} = XA$ linear combination

$\Phi = [\phi(x_1), \phi(x_2), ...]$, $U = \Phi V D^{-\frac{1}{2}}$

$Z = U^T \Phi = D^{-\frac{1}{2}} V^T \Phi^T \Phi = D^{-\frac{1}{2}} V^T k(x, y)$

3.3. Blind Signal Separation

task: separate mixed signals from from independent sources

$X \in \mathbb{R}^{N \times M}$ *e. g.* time series of N images containing M pixels

$X \approx WH$, $W : N \times K$, $H : K \times M$

$K = N \Rightarrow \tilde{x}_n = W\tilde{h}_m$ time series for n-th pixel

$X^T \approx WH$, $W : M \times L$, $H : L \times N$

$K = N \Rightarrow \tilde{x}_m = H\tilde{s}_n$ m-th statistically independent image mode

3.4. Independent Component Analysis

mixing matrix W, non-linear activation $\vec{f}(\dots)$, source $\vec{x}$, statistically independent noise $\vec{n}$

output signal $\vec{z} = \vec{f}(W\vec{x}) + \vec{n}$

task:

- maximize mutual information of input and output
- minimize mutual information between output channels
- minimize redundancy of output channels

statistically independent but number of signals not known: ICA $\rightarrow$ decorrelates higer-order statistics, non-orthogonal system

occurrence probability $p(\vec{x})$, $\vec{x}$ in alphabet M_X

Shannon information $I(\vec{x}) = -\log(p(\vec{x}))$

information entropy (average information) $H(\vec{x}) = E[I(\vec{x})] = -\sum_{x \in M_x} p(\vec{x}) \log(p(\vec{x}))$

information that $\vec{z}$ conveys about $\vec{x}$: $I(\vec{x}) - I(\vec{x}|\vec{z})$

mutual information $MI(\vec{x}|\vec{z}) = H(\vec{x}) - H(\vec{x}|\vec{z}) = MI(\vec{z}|\vec{x})$ average information of $\vec{x}$ when observing $\vec{z}$

$= H(\vec{z}) - H(\vec{N})$, the latter is the information contribution of the noise

Theorem 3 (ICA update rule)

$$\Delta W = \eta \frac{\partial H(\vec{z}|W)}{\partial W} = (W^T)^{-1} - \vec{\phi}\vec{y}\vec{x}^T \qquad \textit{with} \quad \vec{y} = W\vec{x} \tag{3.1}$$
$$\textit{and} \quad \vec{\phi} = -\frac{\mathrm{d}\log(p(y_i))}{\mathrm{d}y_i}$$

4. Dictionary Learning

4.1. Supervised DL

4.2. Convolutional DL

5. Empirical Mode Decomposition

Intrinsic Mode Function

- number of zero crossings and number of extrema differ at most by 1
- mean of envelope of maxima and envelope of minima is 0

properties of $x(t)$

- linearity: $x(t+1)$ depends linear on $x(t)$
- stationary: joint probability density depends only on difference $\tau = t_{n+1} - t_n$
- (weakly stationary: $E[x(t)^2] < \infty$ and $E[x(t)] = 0$, also $\mathrm{Cov}(x(t_1), x(t_2)) = \mathrm{Cov}(t_1 - t_2)$)

Decomposition

$x(t) = \sum_n x_n(t) + r(t)$ achieved through sifting

completeness automatically and proven numerically

IMF local orthogonal $x(t)^2 = \sum_{j=1}^{i+1} x_j(t) + 2Y$ with $Y = \sum_{j=1}^{i+1} \sum_{k=1}^{i+1} x_j x_k$ and $\sum_{t=0}^{T} \frac{Y}{x(t)^2} = 0$

requirements:

- fulfills Nyquist theorem
- digital and analog signal have same number of extrema

- usually > 5 samples per period

Theorem 4 (Nyquist) *A function with no frequencies larger than f_{max} is uniquely determined by an arbitrary series of function values with difference $\tau < \frac{1}{2f_{max}}$.*

Ensemble-EMD

add noise with zero mean and unit variance

averaging cancels noise

Local EMD

sifting in regions with large mean values

Hilbert Transformation

$H\{x(t)\} = \frac{1}{\pi} P \int_{-\infty}^{\infty} \mathrm{d}\tau \, \frac{x(\tau)}{t-\tau}$ with Cauchy prime value P of the singular integral

$z(t) = x(t) + H(x(t)) = a(t) e^{i \int \mathrm{d}t \, \omega(t)}$ with $\omega(t) = -\frac{\mathrm{d}\Theta}{\mathrm{d}t} = -\frac{\mathrm{d}}{\mathrm{d}t} \mathrm{atan}\left(\frac{x(t)}{H(t)}\right)$

Part II.

Neural Networks

6. Single-Layer Perceptron

network output signal $y_i = g(h_i) = g\left(\sum_k w_{ik} y_k\right) = g(W\vec{x})$, h: local field

g: continuous, continuous differentiable, monotonous

linearly independent $\Rightarrow$linearly separable problems, sufficient but not required for convergence

error function $E(\vec{w}) = \frac{1}{2}\sum_{i,\mu}(t_i^\mu - y_i^\mu) = \frac{1}{2}\sum_{i,\mu}(\delta_i^\mu)^2$ with label vector $\vec{t}$ and neuron μ

gradient descent $\delta w_{ik} = -\eta \frac{\partial E}{\partial w_{ik}} = \eta \sum_\mu \delta_i^\mu x_k^\mu$

non-linear: $E = \frac{1}{2}\sum_{i,\mu}\left[t_i^\mu - g\left(\sum_k w_{ik} x_k^\mu\right)\right]^2$

$\frac{\partial E}{\partial w_{ik}} = -\sum_\mu \{[t_i^\mu - g(h_i^\mu)]\, g'(h_i^\mu) x_k^\mu\} = -\sum_\mu \delta_i^\mu g'(h_i^\mu) x_k^\mu$

Theorem 5 (SLP update rule)

$$\Delta w_{ik} = -\eta \sum_\mu \delta_i^\mu g'(h_i^\mu) x_k^\mu \qquad \textit{with} \quad \delta_i^\mu = t_i^\mu - g(h_i^\mu) \tag{6.1}$$

7. Multi-Layer Perceptron

number of training samples n, number of input neurons n, number of output neurons m

number of hidden units $k \approx \frac{P}{10(n+m)}$

error output layer: $\delta_i = (t_i - y_i)g'(h_i)$

error hidden layer: $\delta_j = g'(h_j) \sum_i w_{ij}\delta_i$

Theorem 6 (MLP update rule (hidden layer))

$$\Delta w_{ij} = \eta \delta_j y_j \qquad \textit{with} \quad \delta_j = (t_i - y_i)g'(h_i) \tag{7.1}$$

stop criterion: $\|\vec{\nabla}_w E\|$ sufficient small or $\|\vec{w}(t+1) - \vec{w}(t)\| \leq \varepsilon$

for a function $\vec{y}_j = \begin{pmatrix} y_{1j} \\ y_{2j} \\ \vdots \end{pmatrix} = \begin{pmatrix} F_1(\vec{x}_j) \\ F_2(\vec{x}_j) \\ \vdots \end{pmatrix} = \vec{F}(\vec{x}_j)$ that minimizes mean quadratic error

$L(\vec{F}) = \frac{1}{2N} \sum_{j=1}^{N} \|\vec{t}_j - \vec{F}(\vec{x}_j)\|^2$ choose neuron k for which $F_k(\vec{x}) \geq \vec{F}_j(\vec{x})$ holds $\forall j \neq k$

8. Self-Organizing Maps

task: map m-dimensional data onto n dimensions (typically $n \in \{1, 2\}$)

weight vector $\vec{w}$: euclidean point in input space

$h_i = \sum_j w_{ij} x_j = \vec{w}_i \vec{x}$

if $\|w_i\|$: $\Delta w_{i*j} = \eta(x_j^\mu - w_{i*j})$

else: $\Delta w_{i*j} = \eta \left(\frac{x_j^\mu}{\sum_l x_l^\mu} - w_{i*j} \right)$

binary: $\Delta w_{i*j} = \eta(y_i x_j - y_i w_{ij})$, Hebb minus decay term, $y_i = 1$ if $i = i*$ else 0

with normalized vectors $w_{i*}(t+1) = \frac{\vec{w}_{i*}(t) + \eta(\vec{x} - \vec{w}_{i*}(t))}{\|\vec{w}_{i*}(t) + \eta(\vec{x} - \vec{w}_{i*}(t))\|}$

$\eta(t) = \eta_0 t^{-\alpha}$, $\eta(t) = \eta_0(1 - \alpha t)$, $\alpha \leq 1$

stimulus: $y_j(t+1) = f\left(\sum_{l=0}^{p} w_{jl} x_l + \eta \sum_{k=-K}^{k} c_{ik} y_{j+k}(t) \right)$, f Mexican hat

$\Rightarrow g(y_j)$ binary, $h(i, i*)$ neighborhood

$y_j = \begin{cases} 1 & j \in U \\ 0 \end{cases}$, $g(y_j) = \begin{cases} \eta & j \in U \\ 0 \end{cases}$, $h(i, i*) = \exp\left(\frac{|\vec{r}_i - \vec{r}_i^*|^2}{-2\sigma^2} \right)$

Theorem 7 (SOM update rule)

$$\vec{w}_j(t+1) = \vec{w}_j(t) + \eta(t)h(i, i*)(\vec{x} - \vec{w}_j(t))$$
$$\eta(t), \sigma(t) \sim \begin{cases} \exp\left(-\frac{t}{\tau}\right) \\ t^{-\alpha} \end{cases} \qquad \textit{with} \quad 0 \leq \alpha \leq 1 \tag{8.1}$$

9. Deep Learning

10. Recurrent Neural Networks

input layer W	$u_t \leftarrow W_{hx}x_t + U_{hh}h_{t-1}$	
hidden layer U	$h_t \leftarrow g_h(u_t)$	$h_0 = g(Wx_0)$
label V	$o_t \leftarrow V_{oh}h_t$	
	$y_t \leftarrow g_y(o_t)$	$L(y - \hat{y})^2$

back-propagation through time:

- $\mathrm{d}o_t \leftarrow g'(o_t)\frac{\partial L(y_t, x_t)}{\partial y_t}$
- $\mathrm{d}V_{oh} \leftarrow \mathrm{d}V_{oh} + \mathrm{d}o_t h_t^T$
- $\mathrm{d}h_t \leftarrow \mathrm{d}h_t + W_{oh}^t \mathrm{d}o_t$
- $\mathrm{d}h_{t-1} \leftarrow U_{nn}^T \mathrm{d}y_t$
- $\mathrm{d}y_t \leftarrow \mathrm{d}y_t + g_h'(y_t)\mathrm{d}h_t$
- $\mathrm{d}U_{hh} \leftarrow \mathrm{d}U_{hh} + \mathrm{d}y_t h_{t-1}^T$
- $\mathrm{d}W_{hx} \leftarrow \mathrm{d}W_{hx} + \mathrm{d}y_t x_t^T$

11. Autoencoder

$\tilde{x} = UVx$

PCA: $U = V^T$

Error: $\sum_{m=1}^{M} (\tilde{x}^{(m)} - x^{(m)})^2$

11.1. Deep Autoencoder

11.2. Sparse Autoencoder

12. Restricted Boltzmann Machines

Gibbs distribution $p(x) = \frac{\exp(-E(x))}{\sum\limits_{x} \exp(-E(x))}$

energy: $E(v, h) = -(h^T W v + b^T v + c^T h)$

if conditionally independent: $p(v|h) = \prod\limits_{i} p(h_i|v)$

$P(h_i = 1|v) = \sigma(c_i + \sum w_{ij} v_j)$

$P(v_j = 1|h) = \sigma(b_j + \sum w_{ij} h_i)$

positive phase: use probability to sample h_j

$$h_j \leftarrow \begin{cases} 1 & h_j > \text{rand}(0, 1) \\ 0 & \end{cases}$$

negative phase: $\hat{v}_j \to \hat{h}_j$

Theorem 8 (RBM update rule)

$$\begin{aligned} \Delta W &= \eta(v h^T - \hat{v} \hat{h}^T) \\ \Delta b &= \eta(v - \hat{v}) \\ \Delta c &= \eta(h - \hat{h}) \end{aligned} \tag{12.1}$$

Deep Belief Networks

13. Convolutional Neural Networks

Architectures

14. Graph Neural Networks

V vertices $|V| = N$

edges E, $e_{nn'} = (v_n, v_{n'})$

$$A \in \mathbb{R}^{N \times N},\ A_{nn'} = \begin{cases} w_{nn'} > 0 & \text{if} \quad e_{nn'} \in E \\ 0 & \text{else} \end{cases}$$

degree matrix D with $d_{nn'} = d_{nn}$ if $n = n'$ else 0, $d_{nn} = \sum_{nn'} a_{nn'}$

lagrangian $L = D - A$, $L = U \Lambda U^T$, $M = D^{-\frac{1}{2}} U$

$$L_n = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = M \Lambda M^T$$

$$L_{\text{tr}} = D^{-1} A,\ L_{rw} = \mathbb{1} - L_{\text{tr}}$$

graph Fourier transform $\tilde{x} = \text{GFT}(x) = M^T x$

F_Θ kernel filter (diffusion), diagonal feature matrix $\Theta \to$ convolution theorem

$x \otimes F_\Theta = M\left((M^T F_\Theta) \circ (M^T X)\right) = (M \Theta M^T)\tilde{x}$, $\circ$ denotes Hadamard product

$$\vec{\Theta} = \sum_{k=0}^{k-1} \gamma_k \Lambda^k$$

if $k = 2, \gamma = \gamma_0 = -\gamma_1$

Part III.

Reinforcement Learning

15. Basics