Machine Learning Cheat Sheet

By Ben NG Hok Chun

Distributions

Expectation: $\mathbb{E}[x] = \int xp(x)dx$
Mean: $\mathbb{E}[\mathbf{x}]$
Variance: $\sigma^2 = \mathbb{E}[(x-\mu)^2]$
$\text{Cov}(\mathbf{x}) = \mathbb{E}[(\mathbf{x}-\mu)(\mathbf{x}-\mu)^T]$
Normal: $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$
Binomial: $p(x) = \binom{n}{k} p^k (1-p)^{n-k}$
Multinomial: $p(x) = \frac{n!}{x_1! \cdots x_k!} p_1^{x_1} \cdots p_k^{x_k}$
Poisson: $p(x) = \frac{\lambda^k e^{-\lambda}}{k!}$
Multivariate Normal:
$p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu))$

Common Functions

$\varsigma(x) = \frac{1}{1+\exp(-x)}$ (Posterior class probability)
$\varsigma' = \varsigma(1-\varsigma)$
Likelihood: $L(\theta|\mathcal{X}) = p(\mathcal{X}|\theta))$
Cross-entropy: $E(\theta|\mathcal{X}) = -\log L(\theta|\mathcal{X})$
$\sigma_i = \sigma(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$ (Posterior class probability)
$\frac{\partial \sigma_i}{x_i} = \sigma_i(1-\sigma_i)$
$\frac{\partial \sigma_i}{x_j} = -\sigma_i \sigma_j, i \neq j$
Lagrangian: $\mathcal{L} = a - \sum_l \alpha_l b_l, \alpha_l \geq 0 \quad \forall l$
for minimize a subject to $b_l \geq 0 \quad \forall l$
Kernel: $K(a,b) = \phi(a)^T \phi(b)$
Polynomial kernel: $(a^T b + 1)^q$
RBF kernel: $exp(-\frac{||a-b||^2}{2s^2})$ OR
$exp(-\frac{\mathcal{D}(a,b)}{2s^2})$ for some distance function $\mathcal{D}$ Sigmoidal kernel: $\tanh(2a^T b + 1)$
Leaky ReLU: $f(x) = \begin{cases} x \text{ if } x \geq 0 \\ \alpha \text{ else} \end{cases}$
Exponantial linear: $f(x) = \begin{cases} x \text{ if } x \geq 0 \\ \alpha(e^x - 1) \text{ else} \end{cases}$

Expectation Maximization

Expectation: $\mathcal{Q}(\Phi|\Phi^t) = \mathbb{E}[\mathcal{L}_C(\Phi|\mathcal{X},\mathcal{Z})|\mathcal{X},\Phi^t]$
Maximization: $\Phi^{t+1} = \arg\max_\Phi \mathcal{Q}(\Phi|\Phi^t)$

Deep-learning Concepts

Xavier Initialization: (for sigmoid activation):
- Zero-mean normal with variance: $\frac{2}{n_{\text{in}}+n_{\text{out}}}$
- Uniform in $[-r, r]$ where $r = \sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}$

He Initialization: (for ReLUs activation):
- Zero-mean normal with variance: $\frac{4}{n_{\text{in}}+n_{\text{out}}}$
- Uniform in $[-r, r]$ where $r = \sqrt{\frac{12}{n_{\text{in}}+n_{\text{out}}}}$

Batch normalization: zero-center and normalize every layer.
Dropout: probabilistically set some hidden unit to 0.
Data augmentation: Translate, scale, shift, light/dim, flip to generate new data.
AdaGrad:

$$s \leftarrow s + \bigtriangledown_w L \circ \bigtriangledown_w L$$
$$w \leftarrow w - \eta \bigtriangledown_w L \oslash \sqrt{s+\epsilon}$$

RMSProp:

$$s \leftarrow \beta s + (1-\beta) \bigtriangledown_w L \circ \bigtriangledown_w L$$
$$w \leftarrow w - \eta \bigtriangledown_w L \oslash \sqrt{s+\epsilon}$$

Adam:

$$\Delta w \leftarrow \frac{\beta_1 \Delta w - (1-\beta_1) \bigtriangledown_w L}{1-\beta_1^t}$$
$$s \leftarrow \frac{\beta_2 s + (1-\beta_2) \bigtriangledown_w L \circ \bigtriangledown_w L}{1-\beta_2^t}$$
$$w \leftarrow w - \eta \Delta w \oslash \sqrt{s+\epsilon}$$

Derivative Rules

c	0	e^x	e^x
x	1	a^x	$\ln(a)a^x$
cx	c	$\ln(x)$	$\frac{1}{x}$
x^n	nx^{n-1}	fg	$f'g \cdot fg'$

Integration Rules

$\int uvdx = u\int vdx + \int u'(\int vdx)dx$
$\int f(g(x))g'(x)dx = \int f(u)du$

Nonparametric Methods

Histogram Estimator:
$\hat{p}(x) = \frac{\#\{x^{(l)} \text{ in the same bin as } x\}}{Nh}$
Naive Estimator: $\hat{p}(x) = \frac{\sum_l w(\frac{x-x^{(l)}}{h})}{Nh}$
where $w(u) = \begin{cases} 1 \text{ if } |u| < 1/2 \\ 0 \text{ else} \end{cases}$
Kernel Estimator: $\hat{p}(x) = \frac{\sum_l K(\frac{x-x^{(l)}}{h})}{Nh}$
K-nearest neighbour: $\hat{p}(x) = \frac{k}{2Nd_k(x)}$
where $d_k(x)$ is the distance of x and k-th nearest neighbour of x
KNN-kernel: $\hat{p}(x) = \frac{\sum_l K(\frac{x-x^{(l)}}{d_k(x)})}{Nd_k(x)}$
Regressogram: Mean of the same bin
Running-mean smoother: Mean of the bin around x
Kernel smoother: $\hat{g}(x) = \frac{\sum_l K(\frac{x-x^{(l)}}{h})y^{(l)}}{\sum_l K(\frac{x-x^{(l)}}{h})}$
Running line smoother: With piecewise linear fit

Dimensionality Reduction

Feature selection: Choose k from d features
Forward search v.s. Backward search
Feature extraction: Project $\mathbf{x}$ to $\mathcal{R}^k$
Principal Component Analysis:
Map $\mathbf{x}$ to k orthogonal dimensions
$\mathbf{z}_n = \mathbf{w}_n^T \mathbf{x}, \text{Var}(\mathbf{z}_1) = \mathbf{w}_1^T \mathbf{\Sigma} \mathbf{w}_1, \mathbf{\Sigma} = \text{Cov}(x)$
Maximize $\text{Var}(\mathbf{z}_1)$ s.t. $||\mathbf{w}_1|| = 1$ is eigenvalue of $\mathbf{\Sigma}$
Factor Analysis: Sample $\mathcal{X}$, $\mathbb{E}[\mathbf{x}] = \mu$, $\text{Cov}(\mathbf{x}) = \mathbf{\Sigma}$
Factors z_j, $\mathbb{E}[z_j] = 0$, $\text{Var}(z_j) = 1$, $\text{Cov}(\mathbf{z}) = \mathbf{I}$
Noise ϵ_i, $\mathbb{E}[\epsilon_i] = 0$, $\text{Var}(\epsilon_i) = \Psi_i$, $\text{Cov}(\epsilon) = \mathbf{\Psi I}$
$\mathbf{x} - \mu = \mathbf{Vz} + \epsilon$, $\mathbf{\Sigma} = \mathbf{VV}^T + \mathbf{\Psi}$
Multidimensional Scaling:
$\mathbf{B} = \mathbf{XX}^T = \mathbf{CDC}^T = (\mathbf{CD}^{1/2})(\mathbf{CD}^{1/2})^T$
where $\mathbf{C}$ is eigenvectors as columns and
$\mathbf{D}^{1/2}$ is diagonal matrix of square root of eigenvalues
Drop eigenvectors with low eigenvalues in $\mathbf{C}$ and $\mathbf{D}$
Linear Discriminant Analysis:
Between-class scatter: $\mathbf{w}^T \mathbf{S}_B \mathbf{w}$,
$\mathbf{S}_b = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T, \sum_i N_i(\mathbf{m}_i - \mathbf{m})(\mathbf{m}i - \mathbf{m})^T$
Within-class scatter: $\mathbf{w}^T \mathbf{S}_W \mathbf{w}$,
$\mathbf{S}_W = \sum_i \mathbf{S}_i, \mathbf{S}_i = \sum_l y_i^{(l)}(x^{(l)} - \mathbf{m}_i)(x^{(l)} - \mathbf{m}_i)^T$

Ensemble Learning

No-free-lunch theory: No single model is the best
Combine several simple models into group
Voting: Convex combination of base learners
$y = f(d_1, \cdots, d_L|\mathbf{\Phi}) = \sum_{j=1}^{L} w_j d_j$
Other voting rules: Weighted sum, median, min, max, product
Mixture of experts, Gating: $y = \sum_{j=1}^{L} w_j(x) d_j$
Bayesian model combination:
$P(C_i|x) = \sum_{\mathcal{M}_j} P(C_i|x, \mathcal{M}_j) P(\mathcal{M}_j)$,
w_j estimates prior model probability $P(\mathcal{M}_j)$
Bagging, Bootstrap aggregating: Base learners trained on slightly different training sets.
Draw N from $\mathcal{X}$ with replacement
Boosting: Combine weak learner into strong learner
AdaBoost: Make wrongly-labeled data have higher weight in next learner's training set

Regularization

Lasso Regression (L1 Regularization):
- Cost: $\lambda \sum_i |w_i|$

Ridge Regression (L2 Regularization):
- Cost: $\lambda \sum_i w_i^2$

Problems with Machine Learning

Overfitting
Underfitting
Explanability
Hardware limitation
Time limitation
Space/Time complexity of learning algorithm

Matrix Factorization

Given a non-negative matrix $\mathbf{V}$, find $\mathbf{V} \approx \mathbf{WH}$ Cost functions (Lower bound = 0 iff $\mathbf{A} = \mathbf{B}$):
- Euclidean distance (Frobenius norm):
$||\mathbf{A} - \mathbf{B}||_F^2 = \sum_{ij}(A_{ij} - B_{ij})^2$
- Kullback-Leibler divergence:
$D(\mathbf{A}||\mathbf{B}) = \sum_{ij}(A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij})$

Optimization: convex w.r.t. to $\mathbf{W}$ or $\mathbf{H}$ separately, not both
Multiplicative update rules:
For Euclidean distance:
$H_{a\mu} \leftarrow H_{a\mu} \frac{(\mathbf{W}^T\mathbf{V})_{a\mu}}{(\mathbf{W}^T\mathbf{WH})_{a\mu}}$, $W_{ia} \leftarrow W_{ia} \frac{(\mathbf{VH}^T)_{ia}}{(\mathbf{WHH}^T)_{ia}}$,
For KL-divergence:
$H_{a\mu} \leftarrow H_{a\mu} \frac{\sum_i W_{ia} V_{i\mu}/(\mathbf{WH})_{i\mu}}{\sum_k W_{ka}}$,
$W_{ia} \leftarrow W_{ia} \frac{\sum_\mu H_{a\mu} V_{i\mu}/(\mathbf{WH})_{i\mu}}{\sum_v H_{av}}$,
Probabiistic Matrix Factorization:
$p(R_{ij}|\mathbf{U}_i, \mathbf{V}_j, \sigma^2) = \mathcal{N}(R_{ij}|\mathbf{U}_i^T\mathbf{V}_j, \sigma^2)$
$p(\mathbf{U}_i|\sigma_U^2) = \mathcal{N}(\mathbf{U}_i|0, \sigma_U^2\mathbf{I})$
$p(\mathbf{V}_j|\sigma_V^2) = \mathcal{N}(\mathbf{V}_j|0, \sigma_V^2\mathbf{I})$
MAP estimation with quadratic regularization terms:

$$
\begin{aligned}
E =& \frac{1}{2} \sum_i \sum_j I_{ij}(R_{ij} - \mathbf{U}_i^T\mathbf{V}_j)^2 \\
&+ \frac{\lambda_U}{2} \sum_i ||\mathbf{U}_i||^2 + \frac{\lambda_V}{2} \sum_j ||\mathbf{V}_i||^2
\end{aligned}
$$

where $\lambda_X = \sigma^2/\sigma_X^2$
Variation: $p(R_{ij}|\mathbf{U}_i, \mathbf{V}_j, \sigma^2) = \mathcal{N}(R_{ij}|\varsigma(\mathbf{U}_i^T\mathbf{V}_j), \sigma^2)$

Hidden Markov Model Definition

States: $S = \{S_1, S_2, \cdots, S_N\}$
Observation: $V = \{v_1, v_2, \cdots, v_M\}$
State transition probabilities:
$\mathbf{A} = [a_{ij}]$ where $a_{ij} \equiv P(q_{t+1} = S_j|q_t = S_i)$
Observation probabilities:
$\mathbf{B} = [b_j(m)]$ where $b_j(m) \equiv P(O_t = v_m|q_t = S_j)$
Initial state probabilities:
$\pi = [\pi_i]$ where $\pi_i \equiv P(q_1 = S_i)$

Hidden Markov Model Algorithms

Model parameter: $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$
Forward variables:

$$
\begin{aligned}
\alpha_t(i) &\equiv P(O_1, \cdots, O_t, q_t = S_i|\lambda) \\
\alpha_i(i) &\equiv P(O_1, q_1 = S_i|\lambda) = \pi_i b_i(O_1) \\
\alpha_{t+1}(i) &= [\sum_i \alpha_t(i) a_{ij}] b_j(O_t + 1) \\
P(O|\lambda) &= \sum_i \alpha_T(i) \text{ time } O(N^2T)
\end{aligned}
$$

Backward variables:

$$
\begin{aligned}
\beta_t(i) &\equiv P(O_t + 1, \cdots, O_T|q_t = S_i, \lambda) \\
\beta_T(i) &= 1 \\
\beta_t(i) &= \sum_j a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \\
P(O|\lambda) &= \sum_i \beta_1(i) \pi_i b_i(O_1)
\end{aligned}
$$

Finding states: Viterbi Algorithm:

$$
\begin{aligned}
\delta_1(i) &= \pi_i b_i(O_1) \\
\psi_i(i) &= 0 \\
\delta_t(j) &= (\max_i \delta_{t-1}(i) a_{ij}) \cdot b_j(O_t) \\
\psi_t(j) &= \arg\max_i \delta_{t-1}(i) a_{ij} \\
p^* &= \max_i \delta_T(i) \\
q_T^* &= \arg\max_i \delta_T(i) \\
q_t^* &= \psi_{t+1}(q_{t+1}^*), t = T-1, \cdots, 1
\end{aligned}
$$

Learn model parameter: Baum-Welch algorithm:

$$
\begin{aligned}
\zeta_t(i,j) &\equiv P(q_t = S_i, q_{t+1} = S_j|O, \lambda) \\
&= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_k \sum_l \alpha_t(k) a_{kl} b_l(O_{t+1}) \beta_{t+1}(l)} \\
\mathbb{E}[z_i^t] &= \gamma_t(i), \quad \mathbb{E}[z_{ij}^t] = \zeta_t(i,j) \\
\hat{a}_{ij} &= \frac{\sum_{k=1}^K \sum_{t=1}^{T_k-1} \zeta_t^k(i,j)}{\sum_{k=1}^K \sum_{t=1}^{T_k-1} \gamma_t^k(i)} \\
\hat{b}_j(m) &= \frac{\sum_{k=1}^K \sum_{t=1}^{T_k} \gamma_t^k(j) \mathbf{1}(O_t^{(k)} = v_m)}{\sum_{k=1}^K \sum_{t=1}^{T_k} \gamma_t^k(j)} \\
\hat{\pi}_i &= \frac{\sum_{k=1}^K \gamma_1^k(i)}{K}
\end{aligned}
$$