

Machine Learning Cheat Sheet

General

- Multivariate normal: $\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) := \frac{1}{\sqrt{(2\pi)^D \det(\boldsymbol{\Sigma})}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right]$
- Bayes' Rule: $\text{pst} = \frac{\text{likh} \cdot \text{pri}}{\text{evi}} \iff P(\mathbf{w} \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \mathbf{w})P(\mathbf{w})}{P(\mathcal{D})}$
 - $P(\mathcal{D}) = \int P(\mathcal{D} \mid \mathbf{w})P(\mathbf{w})d\mathbf{w}$
- Logistic sigmoid: $\sigma(a) := \frac{1}{1 + \exp(-a)}$
- Softmax: $f_{k(a)} := \frac{\exp(a_k)}{\sum_j \exp(a_j)}$ with $a_k = \ln p(\mathbf{x} \mid C_k)p(C_k)$
- One-hot coding: $\mathbf{C}_k = ((k-1) * [\mathbf{0}] + [\mathbf{1}] + (n-k) * [\mathbf{0}])^T$

Linear basis function regression

- Model $y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x})$
- Target $t = y(\mathbf{x}, \mathbf{w}) + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \beta^{-1})$
 - $p(t \mid \mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t \mid y(\mathbf{x}, \mathbf{w}), \beta^{-1})$
- Maximum likelihood estimation (constant prior)
 - optimize likelihood $p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \beta) = \prod_n \mathcal{N}(t_n \mid \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_n), \beta^{-1})$
 - $\ln p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \beta) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \frac{\beta}{2} \sum_n [t_n - \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_n)]^2$
 - $\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} \ln p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \beta) = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \mathbf{t}$ with $\boldsymbol{\Phi}_{nj} = \boldsymbol{\varphi}_j(\mathbf{x}_n)$
 - $\hat{\beta}^{-1} = \arg \max_{\beta^{-1}} \ln p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \beta) = \frac{1}{N} \sum_n [t_n - \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_n)]^2$ (residual variance)
- Maximum A-Posteriori estimation (Gaussian prior)
 - optimize posterior $p(\mathbf{w} \mid \mathcal{D}) \propto p(\mathcal{D} \mid \mathbf{w})p(\mathbf{w})$
 - $\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} (\ln p(\mathcal{D} \mid \mathbf{w}) + \ln p(\mathbf{w})) = (\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \alpha \mathbf{I})^{-1} \boldsymbol{\Phi}^T \mathbf{t}$
- Expected loss: $E_{\mathcal{D}}[L] = \text{bias}^2 + \text{variance}$
 - $E_{\mathcal{D}}[(y(\mathbf{x}, \mathcal{D}) - h(\mathbf{x}))^2] = (E_{\mathcal{D}}[y(\mathbf{x}, \mathcal{D})] - h(\mathbf{x}))^2 + E_{\mathcal{D}}[(y(\mathbf{x}, \mathcal{D}) - E_{\mathcal{D}}[y(\mathbf{x}, \mathcal{D})])^2]$ with $\mathcal{D}$ = dataset, $h(\mathbf{x})$ = true data-generating function, and $y(\mathbf{x}, \mathcal{D})$ = linear function fitted using $\mathcal{D}$
- k -fold cross-validation: split data into k partitions, use $k - 1$ for training and 1 for testing and repeat k times, always using a different partition for testing

Fisher's linear discriminant

- project $\mathbf{x}$ to 1D: $y = \mathbf{w}^T \mathbf{x}$
- class mean: $\mathbf{m}_k := \frac{1}{N_k} \sum_{n \in C_k} \mathbf{x}_n$
- projected class mean: $m_k := \mathbf{w}^T \mathbf{m}_k$
- transformed within-class variance: $s_k^2 := \sum_{n \in C_k} (y_n - m_k)^2$
- Fisher criterion: $J(\mathbf{w}) := \frac{\text{between-class variance}}{\text{within-class variance}} = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$
 - $\mathbf{S}_B := (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$
 - $\mathbf{S}_W := \sum_{n \in C_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in C_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$
- Fisher's linear discriminant: $\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} J(\mathbf{w}) \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$

Probabilistic generative models

- Two classes
 - $p(C_1 \mid \mathbf{x}) = \sigma(a) = \sigma(\mathbf{w}^T \mathbf{x} + w_0)$ with $a = \ln \frac{p(\mathbf{x} \mid C_1)p(C_1)}{p(\mathbf{x} \mid C_2)p(C_2)}$, $\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$, and $w_0 = \frac{1}{2}(\boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_2) + \ln \frac{p(C_1)}{p(C_2)}$
 - Maximum likelihood estimation
 - $\hat{\pi} = \frac{N_1}{N}$ with $\pi := p(C_1)$, $p(C_2) = 1 - \pi$
 - $\hat{\boldsymbol{\mu}}_1 = \frac{1}{N_1} \sum_n t_n \mathbf{x}_n$, $\hat{\boldsymbol{\mu}}_2 = \frac{1}{N_2} \sum_n (1 - t_n) \mathbf{x}_n$
 - $\hat{\boldsymbol{\Sigma}} = \frac{N_1}{N} \mathbf{S}_1 + \frac{N_2}{N} \mathbf{S}_2$ with $\mathbf{S}_k = \frac{1}{N_k} \sum_{n \in C_k} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T$
- k classes
 - $p(C_k \mid \mathbf{x}) = \text{Softmax}(\mathbf{a}) = \text{Softmax}(\mathbf{W}^T \mathbf{x} + \mathbf{w}_0)$ (assuming shared $\boldsymbol{\Sigma}$)
 - $a_k = \ln p(\mathbf{x} \mid C_k)p(C_k)$

- $\mathbf{w}_k = \Sigma^{-1} \boldsymbol{\mu}_k$
- $\mathbf{w}_{k0} = -\frac{1}{2} \boldsymbol{\mu}_k^T \Sigma^{-1} \boldsymbol{\mu}_k + \ln p(C_k)$
- Decision boundary between class i and j : $0 = (\mathbf{w}_j - \mathbf{w}_i)^T \mathbf{x} + (w_{j0} - w_{i0})$
 - if Σ is not shared, discriminant is quadratic, otherwise linear

Probabilistic discriminative models

- $y(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$, $p(C_1 | \mathbf{x}) = y$, $p(C_2 | \mathbf{x}) = 1 - y$
- Maximum likelihood estimate (generalizes to multi-class case)
 - $p(\mathbf{t} | \mathbf{w}) = \prod_n y_n^{t_n} (1 - y_n)^{1-t_n}$ with $\mathbf{t} = \{0, 1\}$
 - $E(\mathbf{w}) = -\ln p(\mathbf{t} | \mathbf{w}) = -\sum_n [t_n \ln y_n + (1 - t_n) \ln(1 - y_n)]$
 - $\nabla_{\mathbf{w}} E(\mathbf{w}) = \sum_n [\sigma(\mathbf{w}^T \mathbf{x}_n) \mathbf{x}_n^{T_n} - t_n \mathbf{x}_n^T] = 0$ (no direct solution)
 - Newton's method (IRLS): $\mathbf{w}_{k+1} = (\mathbf{X}^T \mathbf{R} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{R} \mathbf{z}$ with $R_{nn} = y_n(1 - y_n)$ and $\mathbf{z} = \mathbf{X} \mathbf{w}_k - \mathbf{R}^{-1}(\mathbf{y} - \mathbf{t})$

Neural networks

- Forward pass (apply input $\mathbf{x}_n$): $z_j = h(a_j)$ with h = activation function, $a_j = \sum_i w_{ji} z_i$
- Evaluate $\delta_k = y_k - t_k$ at the output neurons, backpropagate $\delta_j = h'(a_j) \sum_k w_{kj} \delta_k$
- Gradient $\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$

Optimization algorithms

- Newton's method: $\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{f'(\mathbf{x}_k)}{f''(\mathbf{x}_k)}$, multidimensional: $\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}^{-1} \nabla f(\mathbf{x}_k)$
- Stochastic gradient descent: $\mathbf{w}_{k+1} = \mathbf{w}_k - \lambda \nabla_{\mathbf{w}} E(\mathbf{x}, \mathbf{w}_k)$
- SGD with momentum: velocity $v_{k+1} = \beta v_k + \lambda \nabla_{\mathbf{w}} E(\mathbf{x}, \mathbf{w}_k)$ with $\beta \in [0, 1)$
 - $\mathbf{w}_{k+1} = \mathbf{w}_k - v_{k+1}$
- AdaGrad (adaptive gradient per parameter) $w_i - \frac{\eta}{\sqrt{G_i}} g_i$; $G_i = \sum_{\tau=1}^t (g_i^{(\tau)})^2$
- RMSProp (root mean square propagation) $v^{(t)} = \gamma v^{t-1} + (1 - \gamma) (g_i^{(t)})^2$; $w_i - \frac{\eta}{\sqrt{v^{(t)}}} g_i^{(t)}$ forgets earlier gradient magnitudes over time
- Adam (adaptive moment estimation, Momentum + RMSProp) $w^{(t)} \leftarrow w^{(t-1)} - \eta \frac{\tilde{m}_w}{\sqrt{\tilde{v}_w + \epsilon}}$; $m_w^{(t)} \leftarrow \beta_1 m^{(t-1)} + (1 - \beta_1) (g^{(t)})^2$; $v_w^{(t)} \leftarrow \beta_2 m^{(t-1)} + (1 - \beta_2) (g^{(t)})^2$

Convnets

- Slide a filter $\mathbf{w}$ over the image, computing $\mathbf{w}^T \mathbf{x} + b$ for a local chunk $\mathbf{x}$ of the image
- valid padding = no padding: $n \times n \Rightarrow (n - k + 1) \times (n - k + 1)$ with $k \times k$ kernel
- same padding = padding with $\frac{k}{2} - 1$ zero-pixels: $n \times n \Rightarrow n \times n$
- Receptive field size $R = 1 + \sum_l (k_l - 1)$
- s -strided convolution: output size $M = \frac{N-k}{s} + 1$
- last layer: Flatten last convolutional layer and fully-connect to output

Decision trees

- for every observation $\mathbf{x} \in R_i$ predict the mean of all training observations in region R_m
- Construction via greedy recursive binary splitting
 - regression trees: use exhaustive search to find predictor x_j and cut point θ that minimize MSE $\min_{j, \theta} \left[\min_{c_-} \sum_{n \in R_-} (t_n - c_-)^2 + \min_{c_+} \sum_{n \in R_+} (t_n - c_+)^2 \right]$ with optimal prediction $c_{\pm} := \frac{1}{N_{\pm}} \sum_{n \in R_{\pm}} t_n$ and halfplanes $R_{\pm(j, \theta)} := \{\mathbf{x} \mid x_j \gtrless \theta\}$
 - classification trees: use cross-entropy $H := -\sum_k p_{mk} \ln p_{mk}$ or Gini index $G := \sum_k p_{mk} (1 - p_{mk})$ as loss function instead, $p_{mk} := \frac{1}{N_m} \sum_{n \in R_m} I(t_n = k)$
 - repeat for each half plane until stopping criterion is reached
- Bagging to reduce variance (samples are correlated, variance barely decreases)

- draw bootstrap samples $\left\{ \left(x_1^{(b)}, t_1^{(b)} \right), \dots, \left(x_N^{(b)}, t_N^{(b)} \right) \right\}_{b=1}^B$ from the training set (with replacement) and train y_b^* on b th sample, average models: $y_{\text{bag}}(x) = \frac{1}{B} \sum_b y_b^*(x)$
- Random forests to reduce variance: construct B trees on bootstrap samples using a random subset of $J \leq P$ (ex. $J = \text{sqrt } P$) features for each split, then average the trees
- Gradient boosted regression trees: initialize $Y_0(x) = 0$, for all k fit regression tree y_k to residuals $r_{kn} = t_n - Y_{k-1}(x_n)$ and update $Y_k(x) = Y_{k-1}(x) + y_k(x)$

Principal Component Analysis

- minimize squared reconstruction error, equivalent to maximizing variance of projection
- principal components are main axes of covariance ellipse
- data covariance matrix $S = \frac{1}{N} X^T X = U \Lambda U^T$ (eigenvalue decomposition)
- Principal components = eigenvectors U

Probabilistic PCA

- define multivariate Gaussian density model with reduced number of parameters: $x \sim \mathcal{N}(x \mid \mu, \mathbb{W}^T + \sigma^2 I)$
- maximum likelihood: $\widehat{W} = U_M \frac{(\Lambda_M - \sigma^2 I)^{-1}}{2} R$ with U_M, Λ_M = first M eigenvectors/eigenvalues and R = arbitrary orthogonal matrix

Stochastic neighbor embedding

- preserve local neighborhood relationships
- minimize $L := \sum_{i \neq j} p_{ij} \ln \frac{p_{ij}}{q_{ij}} \propto - \sum_{i \neq j} p_{ij} \ln a_{ij} + \ln \sum_{k \neq l} a_{kl}$ via gradient descent
 - $q_{ij} = \frac{a_{ij}}{\sum_{k \neq l} a_{kl}}$ and $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$ with $p_{j|i} = \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)}$ with $d_{ij}^2 = \frac{|x_i - x_j|^2}{2\sigma_i^2}$
 - SNE: $a_{ij} = \exp(-|y_i - y_j|^2)$, t-SNE: $a_{ij} = \frac{1}{1 + |y_i - y_j|^2}$

K-means

- minimize $J := \sum_n \sum_k r_{nk} |x_n - \mu_k|^2$ with r_{nk} one-hot encoded
- find optimal r_{nk} given current prototypes μ_k : $r_{nk} = 1$ if $k = \arg \min_j |x_n - \mu_j|^2$
- find optimal μ_k given current cluster assignments r_{nk} : $\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$, iterate

Gaussian mixture models

- $p(x) = \sum_k \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k)$ with priors $\pi_k := p(z_k = 1)$ with z one-hot encoded
- responsibility $\gamma(z_k) := p(z_k = 1 \mid x) = \frac{\pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(x \mid \mu_j, \Sigma_j)}$
- Maximum likelihood: $\ln p(x \mid \pi, \mu, \Sigma) = \sum_n \ln \left[\sum_k \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k) \right]$ (no direct solution)
- compute γ_{nk} given current parameters π_k, μ_k, Σ_k
- find optimal parameters using current γ_{nk} : $\mu_k = \frac{1}{N_k} \sum_n \gamma_{nk} x_n$, $\Sigma_k = \frac{1}{N_k} \sum_n \gamma_{nk} (x_n - \mu_k)(x_n - \mu_k)^T + \lambda I_D$, $\pi_k = \frac{N_k}{N}$ with $N_k := \sum_n \gamma_{nk}$

Transformers

- self Attention $y_i = \sum_{j=1}^N w_{i,j} x_j; \tilde{w}_{i,j} = x_i^T x_j; w_{i,j} = \text{Softmax}(\tilde{w}_{i,j})$
- Each input is query, key, value .
- Transformers are permutation invariant -> positional encodings

No Free Lunch Theorem Averaged over all possible functions f , all algorithms are equivalent in terms of $\mathbb{E}[E|d]$.