Machine Learning Cheatsheet (IN2064)

Linear Algebra shits and giggles

$(A^T)^T = A$
$(AB)^T = B^T A^T$
$(A+B)^T = A^T + B^T$
$A = A^T \iff A$ symmetric
$A = -A^T \iff A$ anti-symmetric
trace of square matrix A is sum of diagonal elements
$tr(A) = \sum_{i=1}^n A_{ii}$
$||x||_2 = \sqrt{\sum_{i=1}^n x_i^2}$
$||x||_1 = \sum_{i=1}^n |x_i|$
$||x||_\infty = max_i |x_i|$
$rank(A) \hat{=}$ number of linearly independent columns/rows
Invertable matrix $\hat{=}$ Non-singular matrix $\hat{=} A^{-1}$ exists
Orthonomal matrix: columns are orthogonal to eachother and are normalized ($||x||_2 = 1$): $U^T U = I = UU^T$ (second equality holds only when U is square).
Span of a set of vectors is the set of all vectors that can be expressed as linear combination of them.
range of a matrix is span of its column vectors $R(A)$.
Projection of y on $span(\{x_1, \dots, x_n\})$ is $v \in span(\{x_1, \dots, x_n\})$, while $||v - y||_2$ is minimal.
$Proj(y; A) = \text{argmin}_{v \in R(A)} ||v - y||_2 = A(A^T A)^{-1} A^T y$
Nullspace $N(A) = \{x \in \mathbb{R}^n : Ax = 0\}$
$A \in \mathbb{S}^n, x \in \mathbb{R}^n$ is a non-zero vector:

- $x^T Ax > 0 \implies A > 0$ A is PD
- $x^T Ax \geq 0 \implies A \geq 0$ A is PSD
- $x^T Ax < 0 \implies A < 0$ A is ND
- $x^T Ax \leq 0 \implies A \leq 0$ A is NSD

else indefinite

$\lambda \in \mathbb{C}$ is eigenvalue, $x \in \mathbb{C}^n$ is eigenvector if:

$$Ax = \lambda x, x \neq 0$$
$$(\lambda I - A)x = 0$$

$tr(A) = \sum_{i=1}^n \lambda_i$, $det(A) = \prod_{i=1}^n \lambda_i$
$rank(A)$ is number of nonzero eigenvalues
if eigenvectors are linearly independant: $A = X \Lambda X^{-1}$
Definiteness depends on sign of eigenvalues.
Gradient: $(\nabla_A f(A))_{ij} = \frac{\partial f(A)}{\partial A_{ij}}$
Hessian: $(\nabla_x^2 f(x))_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$

Probability Theory

Bayes theorem $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$
Normal distribution $\frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2)$
Beta distribution $\text{Beta}(a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{\beta-1}$
Multivariant Gaussian
$\frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(x-\mu_c)^T \Sigma^{-1}(x-\mu_c)\}$

K Nearest Neighbors

Classification: Look at k nearest neighbors and pick majority vote.
$p(y=c|x,k) = \frac{1}{k}\sum_{i \in N_k(x)} \mathbb{I}(y_i = c)$
$\hat{y} = \text{argmax}_c p(y=c|x,k)$
Weighted k-NN:
$p(y=c|x,k) = \frac{1}{Z}\sum_{i \in N_k(x)} \frac{1}{d(x,x_i)} \mathbb{I}(y_i = c)$
$Z = \sum_{i \in N_k(x)} \frac{1}{d(x,x_i)}$
$d(x,x_i)$ distance between x, x_i
Regression:
$\hat{y} = \frac{1}{Z}\sum_{i \in N_k(x)} \frac{1}{d(x,x_i)} y_i$
$\hat{y}$ will be the weighted mean of neighbors
Distance measures:
L_1 norm: $\sum_i |u_i - v_i|$
L_2 norm: $\sqrt{\sum_i (u_i - v_i)^2}$
L_∞ norm: $\max_i |u_i - v_i|$
Mahalanobis distance: $\sqrt{(u-v)^T \Sigma^{-1}(u-v)}$, (Σ) is P(S)D and symmetric
Circument scaling issues:
Data standardization: $x_{i,std} = \frac{x_i - \mu_i}{\sigma_i}$
use mahalanobis distance with $\Sigma = diag(\sigma_1^2, \dots, \sigma_n^n)$

Decision Trees

Classification: $p(y=c|R) = \frac{n_{c,R}}{\sum_{c_i \in C} n_{c_i,R}}$
$\hat{y} = \text{argmax}_c p(y=c|x) = \text{argmax}_c n_{c,R}$
Improvement of split:
$\delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R)$
Impurity measures ($\pi_c = p(y=c|t)$)
Misclassification rate: $i_E = 1 - \max_c \pi_c$
Entropy: $i_H = -\sum_{c_i} \pi_{c_i} \log_2(\pi_{c_i})$
Gini-index: $i_G = -\sum_{c_i} \pi_{c_i}(1-\pi_{c_i}) = 1 - \sum_{c_i \in C} \pi_{c_i}^2$
Regression:
At leaves use mean instead over outputs
Use mean squared error as splitting heuristic

Probabilistic Inference

MLE
$\theta_{MLE} = \text{argmax}_\theta p(D|\theta)$
Here in MLE we are trying to guess/estimate our random variable θ. We assume our data D is distributed depending on θ. So θ_{MLE} is our *best guess*. Looking at D, θ_{MLE} seems to be the most likely θ.
Bayesian Inference:
$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$
Maximum a posteriori:
Estimating the MAP takes prior beliefs into account and thus performs well if less data if available.
$\theta_{MAP} = \text{argmax}_\theta p(\theta|D)$
$= \text{argmax}_\theta \frac{p(D|\theta)p(\theta)}{p(D)}$
$= \text{argmax}_\theta p(D|\theta)p(\theta)$
Estimating the posterior distribution:
Finding $p(\theta|D)$ boils down to finding $p(D)$. (Use pattern matching)
Full Bayesian approach/analysis:
$p(y|D) = \int_0^1 p(y,\theta|D)d\theta$
$= \int_0^1 p(y|\theta,D)p(\theta|D)d\theta = \int_0^1 p(y|\theta)p(\theta|D)d\theta$

Linear regression

Linear regression: $f(x) = w^T x$
Least squared loss function:
$E_{LS}(w) = \frac{1}{2}\sum_{i=1}^N (w^T x_i - y_i)^2$
Closed form:
$w^* = \text{argmin}_w \frac{1}{2}(Xw-y)^T(Xw-y)$
$= (X^T X)^{-1} X^T y (= w_{ML})$
Polynomial model: $\phi_j = \mathbb{R}^d \to \mathbb{R}$:
$f_w(x) = w^T \phi(x)$
$E_{LS}(w) = \frac{1}{2}(\Phi w - y)^T(\Phi w - y)$
$w^* = (\Phi^T \Phi)^{-1} \Phi^T y$
Ridge regression: (controls overfitting)
$E_{ridge}(w) = \frac{1}{2}\sum_{i=1}^N [w^T \phi(x_i) - y_i]^2 + \frac{\lambda}{2}||w||_2^2$
equivalent to MAP estimation with Gaussian prior.
$w^*_{ridge} = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T y$

Linear Classification

Hyperplane as decision boundary defined by normal vector w

$$w^T x \begin{cases} = 0, \text{if x on plane} \\ > 0, \text{if x normals side (class 1)} \\ < 0, \text{else} \end{cases}$$

Generative model:
to obtain posterior $p(y=c|x) \propto p(x|y=c)p(y=c)$
class conditionals $p(x|y=c)$ we choose multivariant normal

LDA:
posterior: $p(y=1|x) = \ldots = \frac{1}{1+\exp(-a)} = \sigma(a)$
$a = \log(\frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)}) = \ldots = w^T x + w_0$
$w = \Sigma^{-1}(\mu_1 - \mu_0)$
$w_0 = -\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 + \log p(y=1)p(y=0)$
$\implies p(y=1|x) = \sigma(w^T x + w_0)$
For more classes we use softmax function:
$\sigma_i = \frac{\exp(x_i)}{\sum_{k=1}^{K}\exp(x_k)}$

Naive Bayes:
continous data with likelihood as gaussian:
$p(x|y=c) = \mathcal{N}(x|\mu_c, \Sigma_c) \rightarrow$ different Σ_c for each class unlike LDA!
$a = x^T W_2 x + w_1^T x + w_0 \rightarrow$ quadratic decision boundary

Discriminative model:
Logistic regression: $y|x \sim \text{Bernoulli}(\sigma(w^T x + w_0))$, w, w_0 free parameters
$E(w)$
$= \sum_{i=1}^{N}(y_i \log(\sigma(w^T x_i)) + (1 - y_i \log(1 - \sigma(w^T x_i)))$
binary cross entropy, $w^* = \text{argmin} E(w)$, also exists with regularization
Multiclass logistic regression:
$E(w) = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\frac{\exp(w_c x^T)}{\sum_{c'}\exp(w_c^T x)})$, with y_{ic} one hot vector

Optimization

Convexity: X is a convex set
$\iff \forall x, y \in X. \lambda x + (1-\lambda)y \in X, \lambda \in [0,1]$
$f(x)$ is convex on $X \iff$
$\forall x, y \in X. \lambda f(x) + (1-\lambda)f(y) \geq f(\lambda x + (1-\lambda)y)$
$\lambda \in [0,1]$
each local minimum is a global minimum
$f(y) \geq f(x) + (y-x)^T \nabla f(x)$
$\nabla^2 f(x) \geq 0$ (PSD)

Optimization

Convexity preserving operations:
let $f_1 : \mathbb{R}^d \rightarrow \mathbb{R}, f_2 : \mathbb{R}^d \rightarrow \mathbb{R}$ be convex, and $g : \mathbb{R}^d \rightarrow \mathbb{R}$ be concave.
- $h(x) = f_1(x) + f_2(x)$ is convex
- $h(x) = \max\{f_1(x), f_2(x)\}$ is convex
- $h(x) = c * f_1(x)$ is convex if $c \geq 0$
- $h(x) = c * g_1(x)$ is convex if $c \leq 0$
- $h(x) = f_1(Ax + b)$ is convex
- $h(x) = m(f_1(x))$ is convex if $m : \mathbb{R} \rightarrow \mathbb{R}$ is convex and non decreasing

Gradient descent:
repeat:
1. $\Delta\theta := -\nabla(\theta)$
2. Linesearch: $t^* = \text{argmin}_{t>0} f(\theta + t\Delta\theta)$
3. Update $\theta := \theta + t^*\Delta\theta$

until stopping criterion satisfied.
Introduce learning rate

$$\theta_{t+1} \leftarrow \theta_t - \tau\, \nabla\Delta\theta \begin{cases} \tau \text{too small} \rightarrow \text{slow convergence} \\ \tau \text{too big} \rightarrow \text{overshooting, oscillation} \end{cases}$$

Use learning rate adaption!
Newton method ?!
Stochastic gradient descent:
1. randomly pick a small subset (S) of the points (minibatch)
2. compute gradient descent on minibatch
3. update $\theta_{t+1} \leftarrow \theta_t - \tau * \frac{n}{|S|}\sum_{j \in S}\nabla L_j(\theta_t)$
4. pick new subset and repeat with 2

A full iteration through the data $D = S_1 \cup \ldots \cup S_n$ is called epoch. (S_i disjoint)

Deep Learning

Instead of adding linear transformations (like in logistic regression) we can learn these linear transformations too.
(learn w_{ijk} where i=layer, j=input node, k=output node)
Use cross entropy for loss function
By adding more hidden layers we get a *deep* neural network

Deep Learning

Activation functions:
Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$
$ReLU(x) = \max\{0, x\}$

$$ELU(x) = \begin{cases} x & x \geq 0 \\ \alpha(e^{-x} - 1) & else \end{cases}$$

$tanh(x)$
Leaky ReLU $max(0.1x, x)$
Swish $x\sigma(x)$
Functions that can be compactly represented with k layers may require exponentially many hidden units when using $k-1$ layers.
Different tasks

Target	$p(y\|x)$	Final layer	Loss function
Binary	Bernoulli	Sigmoid	Bin-cross entro
Discrete	Categorical	Softmax	cross entropy
Continous	Gaussian	Identity	Squared error

For optimization often use gradient descent.

Support Vector Machines

Linear classifier, find hyperplane with biggest margin between two classes. $m = \frac{2}{||w||}$
$w^T x_i + b \geq 1$ for $y_i = 1$
$w^T x_i + b \leq 1$ for $y_i = -1$
$\implies y_i(w^T x_i + b) \geq 1$
Minimize $f_0(w, b) = \frac{1}{2}w^T w$
subject to $f_i(w, b) = y_i(w^T x_i + b) - 1 \geq 0$
Use Lagrangian:
$L(w, b, \alpha) = \frac{1}{2}w^T w - \sum_{i=1}^{N}\alpha_i[y_i(w^T x_i + b) - 1]$
$w = \sum_{i=1}^{N}\alpha_i y_i x_i$
if $\alpha_i \neq 0$, point lies on margin (support vector)

Soft margin SVM:
introduce slack variables: $\eta_i \geq 0$
$g(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_i\sum_j y_i y_j \alpha_i \alpha_j x_i^T x_j$
Subject to $\sum_i \alpha_i y_i = 0$
$0 \leq \alpha_i \leq C$
Rewrite as unconstrained:
$min_{w,b}\frac{1}{2}w^T w + C\sum_{i=1}^{N}\max\{0, 1 - y_i(w^T x_i + b)\}$ (hinge)

Kernels

$\phi : \mathbb{R}^D \to \mathbb{R}^M$ to make data linearly separable.
Kernel is valid if
$k(x_1, x_2) = \phi(x_1)^T \phi(x_2)$ can be represented as product
Kernel matrix

$$K = \begin{pmatrix} k(x_1, x_1) & \dots & k(x_1, x_N) \\ \dots & \dots & \dots \\ k(x_N, x_1) & \dots & k(x_N, x_N) \end{pmatrix} \text{ is PSD}$$

Kernel preserving operations:
$k_1 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $k_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \mathcal{X} \subseteq \mathbb{R}^M$

- $k(x_1, x_2) = k_1(x_1, x_2) + k_2(x_1, x_2)$
- $k(x_1, x_2) = c * k_1(x_1, x_2)$ with $c > 0$
- $k(x_1, x_2) = k_1(x_1, x_2) * k_2(x_1, x_2)$
- $k(x_1, x_2) = k_1(\phi(x_1), \phi(x_2))$
- $k(x_1, x_2) = x_1 A x_2$ A square, symmetric, PSD