Disclaimer

This document is an exam summary that follows the slides of the *Introduction to Machine Learning* lecture at ETH Zurich. The contribution to this is a short summary that includes the most important concepts, formulas and algorithms. This summary was created during the spring semester 2018 by Yannik Merkli and adapted in 2024 by Nicolas Wehrli. Due to updates to the syllabus content, some material may no longer be relevant for future versions of the lecture.

I do not guarantee correctness or completeness, nor is this document endorsed by the lecturers. Feel free to point out any erratas. For the full LaTeX source code, consider `https://github.com/nwehrli/ymerkli-eth-summaries`.

Basics

Orth: A: $det(A) \in \{+1,-1\}, AA^T = A^TA = I$
$trace(ABC) = trace(BCA) = trace(CAB)$
$trace(A) = \sum \lambda_i(A)$; $\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$
$A = \sum_{k=1}^{rk(A)} \sigma_{k,k} u_k (v_k)^T, A^\dagger = US'V^T; \sigma'_{k,k} = \frac{1}{\sigma_{k,k}}$

Deriv: $\frac{\partial}{\partial x} b^T x = \frac{\partial}{\partial x} x^T b = b^T, \frac{\partial}{\partial x}\|x\|_2^2 = 2x^T, \frac{\partial}{\partial x}\|x - a\|_2 = \frac{(x-a)^T}{\|x-a\|_2}, \frac{\partial}{\partial x}(x^TAx) = x^T(A^T + A), \frac{\partial}{\partial x}(b^TAx) = A^Tb, \nabla_X(c^TXb) = cb^T, \nabla_X(c^TX^Tb) = bc^T; \|A\|_{op} = sup_{\|x\|_2=1}\|Ax\|_2$

(Strong) Convexity:
0: $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) - \frac{m}{2}\lambda(1-\lambda)\|x-y\|_2^2$
1: $f(y) \geq f(x) + \nabla f(x)^\top(y-x) - \frac{m}{2}\|x-y\|_2^2$
2: Hessian $\nabla^2 f(x) \succcurlyeq 0 + mI$ (psd)
i: $\alpha f + \beta g$, $\alpha, \beta \geq 0$, convex if f, g convex · **ii**: $f \circ g$, convex if f convex and g affine or f convex non-decreasing and g convex · **iii**: $\max(f,g)$, convex if f,g convex

PSD: $M \in \mathbb{R}^{n\times n}$ PSD $\Leftrightarrow \forall x \in \mathbb{R}^n : x^\top Mx \geq 0$ $\Leftrightarrow$ all principal minors of M have non-negative determinant $\Leftrightarrow \lambda \geq 0\ \forall \lambda \in \sigma(M)$

$p_{\mu,\Sigma}(x) = \frac{1}{\sqrt{(2\pi)^p det(\Sigma)}} \exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$
$\mathbf{X} \sim \mathcal{N}(\mu, \Sigma) \implies A\mathbf{X} + b \sim \mathcal{N}(A\mu + b, A\Sigma A^T)$
$\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X \mid Y]] = \sum_y \mathbb{E}[X \mid Y]\mathbb{P}[Y = y]$
If $X \geq 0:\ \mathbb{E}[X] = \int_0^\infty (1 - F(x))dx$
$\mathrm{Var}(AX) = A\mathrm{Var}(X)A^\top$
Jensen ineq: $g(E[X]) \leq E[g(X)]$, g convex

Regression

Linear Regression $f(x) = w^Tx; X \in \mathbb{R}^{n\times d}$
$L(w) = \|Xw - y\|_2^2; X^TX\hat{w} = X^Ty$
$d \leq n : \hat{w} = (X^TX)^{-1}X^Ty$ if $rk(X) = d$
$n < d : \hat{w} = (X^TX)^\dagger X^Ty$; $rk(X) = n$ $\|\hat{w}\|_2$ min.
$\nabla_w L(w) = 2X^T(Xw - y)$

Gradient Descent
1. Start arbitrary $w_o \in \mathbb{R}$
2. Do $w_{t+1} = w_t - \eta\nabla L(w_t)$ until $\|w^t - w^{t-1}\|_2 \leq \epsilon$

GD conv. to $\hat{w}$ if $rk(X^TX) = d, \eta < \frac{2}{\lambda_{max}(X^TX)}$
$\|w^{t+1} - \hat{w}\| \leq \|I - \eta X^TX\|_{op}\|w^t - \hat{w}\|_2 \leq \rho^{t+1}\|w^0 - \hat{w}\|_2$; $\eta_{opt} = \frac{2}{\lambda_{max}+\lambda_{min}}$; $\rho_{min} = 1 - \eta_{opt}\lambda_{min} = \frac{\kappa-1}{\kappa+1}$
minibatch SGD: $\nabla L_S(w)$ on random $S \subset D$ every iter.; $|S| = 1$ SGD; $E_S(\nabla L_S(w)) = \nabla L(w)$
strictly c. $\implies$ stationary point is unique g. min.; **strongly c.** $\implies$ unique g. min. exists

Errors
exp. estim. err.: $E_X(\ell(f(X), f^*(X)))$; $y = f^*(x) + \varepsilon$
generaliz. err.: $L(f;\mathbb{P}_{X,Y}) = E_{X,Y}(\ell(f(X), Y))$
$L(\hat{f};\mathbb{P}_{X,Y}) = E_X((\hat{f}(X) - f^*(X))^2) + \sigma^2$(sq. loss)
$L(\hat{f}_\mathcal{D};\mathcal{D}_{\text{test}}) = \frac{1}{|\mathcal{D}_{\text{test}}|}\sum_{(x,y)\in\mathcal{D}_{\text{test}}} \ell(\hat{f}_\mathcal{D}(x), y)$ estim. generaliz. err.; g. err. + const = exp. estim err.
k-fold CV: $\uparrow$k $\implies \hat{f}_{M_i,\mathcal{D}'} \approx \hat{f}_{M_i,\mathcal{D}_{\text{use}}}$, $CV_k(M_i) \not\approx L(\hat{f}_{M_i,\mathcal{D}_{\text{use}}};\mathbb{P}_{X,Y})$; extreme: LOOCV

Bias-Variance Tradeoff
$\text{Bias}^2_\mathcal{D}(\hat{f}_\mathcal{D}, x) := (E_\mathcal{D}(\hat{f}_\mathcal{D}(x)) - f^*(x))^2$
$\text{Bias}^2_\mathcal{D}(\hat{f}_\mathcal{D}) := E_X(\text{Bias}^2_\mathcal{D}(\hat{f}_\mathcal{D}, X))$; $\text{Var}_\mathcal{D}(\hat{f}_\mathcal{D}) := E_X(\text{Var}_\mathcal{D}(\hat{f}_\mathcal{D}(X)))$; Pred. err. $E_\mathcal{D}(L(\hat{f}_\mathcal{D};\mathbb{P}_{X,Y})) = \text{Var}_\mathcal{D}(\hat{f}_\mathcal{D}) + \text{Bias}^2_\mathcal{D}(\hat{f}_\mathcal{D}) + \sigma^2$
Ridge closed form: $\hat{w} = (X^TX + \lambda I)^{-1}X^Ty$
Kernelized: $\min_w \frac{1}{n}\|y - \Phi w\|_2^2 + \lambda\|w\|_2^2 = \min_\alpha \frac{1}{n}\|y - K\alpha\|_2^2 + \lambda\alpha^TK\alpha$

Classification

$\hat{y} = sign(f(x)) = sign(w^Tx), z = yf(x)$ Surrogate Losses for 0-1: $\ell_{\exp(z)} = e^{-z}, \ell_{\log}(z) = \log(1 + e^{-z})$
$\nabla_z\ell_{\exp}$ explodes for $z \to -\infty$, sens. to outliers.
Logistic Reg. $L(w) = \frac{1}{n}\sum_{i=1}^n \log(1 + e^{-y_iw^Tx_i})$ (linear boundary!)

MM and SVM
$w_{\text{MM}} = \arg\max_{\|w\|_2=1} \min_{1\leq i\leq n} y_i\langle w, x_i\rangle$
$w_{\text{SVM}} = \arg\min \|w\|_2$ s.t. $y_i\langle w, x_i\rangle \geq 1, \forall i$
If data **linearly sep.**, 1. $w_{\text{SVM}} = w_{\text{MM}}\|w_{\text{SVM}}\|_2$
2. GD on logistic reg. ($\eta = 1$): $\frac{w^t}{\|w^t\|_2} \to w_{\text{MM}}$
If **not** and $ker(X) = \emptyset$, GD on logistic reg. ($\eta = \frac{4}{\lambda_{max}(X)}$) $w^t \to \hat{w}$, $\hat{w}$ global min.
Hinge loss: $l_H(w;x,y) = max\{0, 1 - yw^Tx\}$
soft-mar. $w^* = \underset{w}{\text{argmin}}\|w\|_2^2 + \lambda\sum_{i=1}^n l_H(w;x_i,y_i)$
$g_i(w) = max\{0, 1 - y_iw^Tx_i\} + \lambda\|w\|_2^2$
$\nabla_w g_i(w) = \begin{cases} -y_ix_i + 2\lambda w & \text{, if } y_iw^Tx_i < 1 \\ 2\lambda w & \text{, if } y_iw^Tx_i \geq 1 \end{cases}$

Multi-Class Classification
$\hat{y}(x) = \text{argmax}_{k\in\{1,..,K\}} f_k(x)$; $f = (f_1, ..., f_K)$
OvR: For each class $k \in [K]$:
1. Relabel $\tilde{y}_i = 1$ if $y_i = k$, else $\tilde{y}_i = -1$ as $\mathcal{D}_k$
2. Train f_k as binary classifier on $\mathcal{D}_k$

Cross-E. Loss: $\ell_{\text{ce}}(f(x), y) = -\log\left(\frac{e^{f_y(x)}}{\sum_{k=1}^K e^{f_k(x)}}\right)$
OvO: $L(w) = \sum_{i=1}^n \ell_{\text{ce}}(f_w(x_i), y_i)$; GD on $L(w)$
Multiplicative noise model: $y = y^*(x)\varepsilon$

Cost Sensitive Classification
Replace loss by: $l_{CS}(w;x,y) = c_y l(w;x,y)$

Metrics (convention: positive = rare)
Accuracy$= \frac{\#correct\ predictions}{\#all\ predictions} = \frac{TP+TN}{TP+TN+FP+FN}$,
Precision$= \frac{\#correct\ '+'\ predictions}{\#all\ '+'\ predictions} = \frac{TP}{TP+FP}$ =1-FDR
Rec.=TPR$= \frac{TP}{TP+FN} = \frac{TP}{n_+}$, FPR$= \frac{FP}{TN+FP} = \frac{FP}{n_-}$=T1
F1 score $= \frac{2TP}{2TP+FP+FN} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$
$\hat{y}_\tau(x) = \text{sign}(\hat{f}(x) - \tau)$ (varying threshold)

Kernels $k : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$

Reparam. $w = \phi^T\alpha$; $f(x) = \sum_{i=1}^n \alpha_i\langle\phi(x_i), \phi(x)\rangle$
A kernel is **valid** if K is sym.: $k(x,z) = k(z,x)$ and psd: $z^\top Kz \geq 0$
mono.: $k(x,y) = (x^\top y)^m$, **poly**: $k(x,y) = (1 + x^\top y)^m$, **RBF**: $k(x,z) = \exp(-\frac{\|x-z\|_\alpha}{\tau})$, $\alpha = 1 \Rightarrow$ Laplacian, $\alpha = 2 \Rightarrow$ Gaussian
Mercers Theorem: Valid kernels can be decomposed into a lin. comb. of inner products.
Kernel composition $k = k_1 + k_2, k = k_1 \cdot k_2, \forall c > 0. k = c\cdot k_1, k = f(k_1), f$ pwr. series w/ non-neg. coeffs., $k((\begin{smallmatrix}x\\y\end{smallmatrix}), (\begin{smallmatrix}x'\\y'\end{smallmatrix})) = k(x,x')k(y,y')$, $k((\begin{smallmatrix}x\\y\end{smallmatrix}), (\begin{smallmatrix}x'\\y'\end{smallmatrix})) = k(x,x') + k(y,y')$, $k(x,x') = g(\langle x,x'\rangle)$, g all Taylor coefficients non-negative, $\forall f.\ k(x,y) = f(x)k_1(x,y)f(y)$
Kern. Ridge Reg. $\min_w \frac{1}{n}\|y - \Phi w\|^2 + \lambda\|w\|_2^2 = \min_\alpha \frac{1}{n}\|y - K\alpha\|_2^2 + \lambda\alpha^\top K\alpha$

k Nearest Neighbor classifier
i: Pick k and distance metric d · **ii**: For given x, find among $x_1, ..., x_n \in D$ the k closest to $x \to x_{i_1}, ..., x_{i_k}$ · **iii**: Output the majority vote of labels

Neural Networks

$F(x) = W^L\phi^{L-1}(W^{L-1}...(\phi^1(W^1x)...))$
ReLU: $\max(0,z)$, **Tanh:** $\frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$
Sigmoid: $\varphi(z) = \frac{1}{1+\exp(-z)}$, $\varphi' = (1-\varphi)\varphi$
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: $z^{(l)} = W^{(l)}v^{(l-1)}, v^{(l)} = [\varphi(z^{(l)});1]$

Backpropagation

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

Only compute **the gradient**. Rand. init. weights by distr. assumption for φ. ($2/n_{in}$ for ReLu and $1/n_{in}$ or $2/(n_{in} + n_{out})$ for Tanh)

Overfitting
Regularization; **Early Stopping**; **Dropout**: ignore hidden units with prob. $1 - p$, after training use all units and scale weights by p; **Batch Normalization**: normalize the input data for each mini-batch, rescale and shift;

CNN $\varphi(W * v^{(l)})$
The out. dim. of applying k $f\times f\times d$ filters (d = # channels) to $n\times m$ image with padding p and stride s is: $\left(\frac{n+2p-f}{s} + 1\right)\times\left(\frac{m+2p-f}{s} + 1\right)$ with k channels. If $t\times t$ pooling is applied, both dimensions are divided by t, nr. channels stays.
Don't forget bias: 1 per filter + #outputs for any fully connected layer.

Learning with momentum
$a \leftarrow m\cdot a + \eta_t\nabla_W l(W;y,x)$; $W \leftarrow W - a$

Clustering

k-mean
$\hat{R}(\mu) = \sum_{i=1}^n \min_{j\in\{1,...k\}}\|x_i - \mu_j\|_2^2$, non-convex, NP-hard, kernelizable, local opt., spherical bias
Algorithm (Lloyd's heuristic):
Initialize cluster centers $\mu^{(0)} = [\mu_1^{(0)}, ..., \mu_k^{(0)}]$
While still changes in assignments:
$z_i^{(t)} = \underset{j\in\{1,...,k\}}{\text{argmin}}\|x_i - \mu_j^{(t-1)}\|_2^2$; $\mu_j^{(t)} = \frac{1}{n_j^{(t)}}\sum_{i:z_i^{(t)}=j} x_i$
$\mathcal{O}(nkd)$ per it., worst-case exponential it.
Conv. proof: $\hat{R}(\mu, z) := \sum_{i=1}^n \|x_i - \mu_{z_i}\|_2^2$. $\hat{R}(\mu^{(t)}, z^{(t)}) \geq \hat{R}(\mu^{(t)}, z^{(t+1)}) \geq \hat{R}(\mu^{(t+1)}, z^{(t+1)})$
k-mean++
- Start with random data point as center
- for $j = 2$ to k: i_j sampled with prob.

$P(i_j = i) = \frac{1}{z}\min_{1\leq l<j}\|x_i - \mu_l\|_2^2$; $\mu_j \leftarrow x_{i_j}$

in exp. conv. to $\mathcal{O}(\log k)$·OPT
Selecting k: elbow method, regularization

Dimension Reduction

Principal component analysis (PCA)
Given: $D = \{x_1, ..., x_n\} \subset \mathbb{R}^d$, $1 \leq k \leq d$
$\Sigma_{d\times d} = \frac{1}{n}\sum_{i=1}^n x_ix_i^T$, $\mu = \frac{1}{n}\sum_{i=1}^n x_i = 0$!!
Sol.: $(W, z_1, ..., z_n) = \text{argmin}\sum_{i=1}^n \|Wz_i - x_i\|_2^2$,
where $W \in \mathbb{R}^{d\times k} : W^TW = I_k$, $W^* = (v_1|...|v_k)$ w/ v_i evec. of Σ and evals $\lambda_1 \geq ... \geq \lambda_d \geq 0$.
Projections $z_1, ..., z_n \in \mathbb{R}^k$ are given by $z_i = W^Tx_i$ where $\Sigma = \sum_{i=1}^d \lambda_iv_iv_i^T$,

Kernel PCA
For general $k \geq 1$, the Kernel PC are given by $\alpha^{(1)}, ..., \alpha^{(k)} \in \mathbb{R}^n$, where $\alpha^{(i)} = \frac{1}{\sqrt{\lambda_i}}v_i$ is obtained from: $K = \sum_{i=1}^n \lambda_iv_iv_i^T$, $\lambda_1 \geq ... \geq \lambda_d \geq 0$
Point x projected as $z \in \mathbb{R}^k$: $z_i = \sum_{j=1}^n \alpha_j^{(i)}k(x, x_j)$

Autoencoders $f_1 : \mathbb{R}^d \to \mathbb{R}^k, f_2 : \mathbb{R}^k \to \mathbb{R}^d$
Try to learn identity function: $x \approx f(x;\theta)$
$f(x;\theta) = f_2(f_1(x_1;\theta_1);\theta_2)$; f_1 : en-, f_2 : decoder
Lin. activation func. & square loss => PCA

Probability Modeling

Assumption: Data set is generated iid
Find $h: X \to Y$ that minimizes pred. error
$\hat{y} = h^*(x) = \mathbb{E}[Y|X=x]$ for sq. loss

Maximum Likelihood Estimation (MLE)
Choose a particular parametric $\hat{p}(Y|X,\theta)$
$\theta^* = \underset{\theta}{\text{amax}}\,\hat{p}(y_{1:n}|x_{1:n},\theta)$
$\overset{\text{iid}}{=} \text{amin}_\theta - \sum_{i=1}^n log\hat{p}(y_i|x_i,\theta)$

Ex. Conditional Linear Gaussian
Assume Gaussian noise $y = f(x) + \epsilon$ with $\epsilon \sim \mathcal{N}(0,\sigma^2)$ and $f(x) = w^\top x$:
$$\hat{p}(y \mid x, \theta) = \mathcal{N}(y; w^\top x, \sigma^2)$$
The optimal $\hat{w}$ can be found using MLE:
$\hat{w} = \underset{w}{\text{argmax}}\; p(y|x,\theta) = \underset{w}{\text{argmin}} \sum (y_i - w^\top x_i)^2$

Maximum a Posteriori Estimate
Assume $y = f(x;\theta^*) + \epsilon$, $\epsilon \sim \mathcal{N}(0,\sigma^2)$, but $\theta^* \sim \mathcal{N}(0, \sigma_\theta^2 I_d)$ The posterior distribution of θ is given by: $p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)}{p(\mathcal{D})} \cdot p(\theta)$
Now we want to find the MAP for θ:
$\hat{\theta} = \text{argmax}_\theta\; p(\theta \mid \mathcal{D})$
$= \text{argmax}_\theta\; p(\mathcal{D} \mid \theta) \cdot p(\theta)$
$= \text{argmin}_\theta - \sum_{i=1}^n \log p(y_i \mid x_i, \theta) - \log p(\theta)$
$= \text{argmin}_\theta\; \frac{\sigma^2}{\sigma_\theta^2}\|\theta\|_2^2 + \sum_{i=1}^n (y_i - f(x_i;\theta))^2$
Regularization can be understood as MAP inference, with different priors (= regularizers) and likelihoods (= loss functions).

Statistical Models for Classification
f minimizing the population risk: $f^*(x) = \text{argmax}_{\hat{y}}\; p(\hat{y} \mid x)$
This is called the Bayes' optimal predictor for the 0-1 loss. Assuming iid. Bernoulli noise, the conditional probability is:
$$p(y \mid x, w) \sim \text{Ber}(y; \sigma(w^\top x))$$
Where $\sigma(z) = \frac{1}{1+\exp(-z)}$ is the sigmoid function.
Using MLE we get:
$$\hat{w} = \underset{w}{\text{argmin}} \sum_{i=1}^n \log(1 + \exp(-y_i w^\top x_i))$$
Which is the logistic loss. Instead of MLE we can estimate MAP, e.g. with a Gaussian prior:
$$\hat{w} = \underset{w}{\text{argmin}}\; \lambda\|w\|_2^2 + \sum_{i=1}^n \log(1 + e^{-y_i w^\top x_i})$$

Bayesian Decision Theory

Given $p(y \mid x)$, a set of actions A and a cost $C: Y \times A \mapsto \mathbb{R}$, pick the action with the maximum expected utility.
$$a^* = \text{argmin}_{a \in A}\; \mathbb{E}_y[C(y,a) \mid x]$$
Can be used for asymmetric costs or abstention.

Generative Modeling

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

Gaussian Bayes Classifier
No independence assumption, model the features with a multivariate Gaussian $\mathcal{N}(x; \mu_y, \Sigma_y)$:
$$\mu_y = \frac{1}{\text{Count}(Y=y)} \sum_{j \mid y_j = y} x_j$$
$$\Sigma_y = \frac{1}{\text{Count}(Y=y)} \sum_{j \mid y_j = y} (x_j - \hat{\mu}_y)(x_j - \hat{\mu}_y)^\top$$
This is also called the **quadratic discriminant analysis** (QDA). LDA: $\Sigma_+ = \Sigma_-$, Fisher LDA: $p(y) = \frac{1}{2}$, classify x as outlier if: $p(x) \leq \tau$.

Gaussian Naive Bayes Classifier
GBC with diagonal Σs (assume features independent). Estimate the parameters via MLE:
MLE for class prior: $p(y) = \hat{p}_y = \frac{\text{Count}(Y=y)}{n}$ MLE for feature distribution:
$$p(x_i \mid y) = \mathcal{N}(x_i; \hat{\mu}_{y,i}, \sigma_{y,i}^2)$$
Where:
$$\mu_{y,i} = \frac{1}{\text{Count}(Y=y)} \sum_{j \mid y_j = y} x_{j,i}$$
$$\sigma_{y,i}^2 = \frac{1}{\text{Count}(Y=y)} \sum_{j \mid y_j = y} (x_{j,i} - \hat{\mu}_{y,i})^2$$
Predictions are made by:
$$y = \underset{\hat{y}}{\text{argmax}}\; p(\hat{y} \mid x) = \underset{\hat{y}}{\text{argmax}}\; p(\hat{y}) \cdot \prod_{i=1}^d p(x_i \mid \hat{y})$$
Equivalent to decision rule for bin. class.:
$$y = \text{sgn}\left(\log \frac{p(Y=+1 \mid x)}{p(Y=-1 \mid x)}\right)$$
Where $f(x)$is called the discriminant function. If the conditional independence assumption is violated, the classifier can be overconfident.

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

Fisher's linear discriminant analysis (LDA; c=2)
Assume: $p = 0.5$; $\hat{\Sigma}_- = \hat{\Sigma}_+ = \hat{\Sigma}$
discriminant f.: $f(x) = log\frac{p}{1-p} + \frac{1}{2}[log\frac{|\hat{\Sigma}_-|}{|\hat{\Sigma}_+|}$
$+ ((x - \hat{\mu}_-)^T \hat{\Sigma}_-^{-1}(x - \hat{\mu}_-)) - ((x - \hat{\mu}_+)^T \hat{\Sigma}_+^{-1}(x - \hat{\mu}_+))]$
Predict: $y = sign(f(x)) = sign(w^T x + w_0)$
$w = \hat{\Sigma}^{-1}(\hat{\mu}_+ - \hat{\mu}_-)$; $w_0 = \frac{1}{2}(\hat{\mu}_-^T \hat{\Sigma}^{-1}\hat{\mu}_- - \hat{\mu}_+^T \hat{\Sigma}^{-1}\hat{\mu}_+)$

Outlier Detection
$P(x) = \sum_{y=1}^c P(y)P(x|y) = \sum_y \hat{p}_y \mathcal{N}(x|\hat{\mu}_y, \hat{\Sigma}_y) \leq \tau$

Gaussian Mixture Model

Mixture modeling $P(x|\theta) = P(x|\mu; \Sigma, w)$
1)Model each cluster j as prob. distr. $P(x|\theta_j)$
2)data iid, lklh.: $P(D|\theta) = \prod_{i=1}^n \sum_{j=1}^k w_j P(x_i|\theta_j)$
3)θ should minimize neg log-likelihood:
$\theta^* = \underset{\theta}{\text{amin}} L(D;\theta) = \underset{\theta}{\text{amin}} - \sum_i log \sum_j w_j P(x_i|\theta_j)$

Ex: $P(x|\theta) = \Sigma_i w_i \mathcal{N}(x; \mu_i, \Sigma_i)$, $P(z_i = j) = w_j$
$\Sigma w_i = 1$, $P(z,x) = w_z \mathcal{N}(x|\mu_z, \Sigma_z)$

Gaussian-Mixture Bayes classifiers
Estimate class prior $P(y)$; Est. cond. distr. for each class: $P(x|y) = \sum_{j=1}^{k_y} w_j^{(y)} \mathcal{N}(x; \mu_j^{(y)}, \Sigma_j^{(y)})$
$P(y|x) = \frac{1}{P(x)} p(y) \sum_{j=1}^{k_y} w_j^{(y)} \mathcal{N}(x; \mu_j^{(y)}, \Sigma_j^{(y)})$

Hard-EM algorithm
Initialize parameters $\theta^{(0)}$
For $t = 1, 2..$: Predict class z_i for each x_i:
E: $z_i^{(t)} = \text{argmax}_z\, P(z|x_i, \theta^{(t-1)}) =$
$= \text{argmax}_z\, P(z|\theta^{(t-1)})P(x_i|z, \theta^{(t-1)}) =$
$= \text{argmax}_z\, w_z^{(t-1)} \mathcal{N}(x_i|\mu_z^{(t-1)}, \Sigma_z^{(t-1)})$
M: Compute the MLE as for the Gaussian B. class.: $\theta^{(t)} = \text{argmax}_\theta\, P(D^{(t)}|\theta)$
$\forall i.(x_i, z_i^{(t)}) \in D^{(t)}$; works poorly if clust. overlap
Special case: fix $w_z = \frac{1}{k}$, spher. cov. $\Sigma_z = \sigma^2 \mathbb{I}$
$\to$ k-means: **E:** $z_i^{(t)} = \text{argmin}_z \|x_i - \mu_z^{(t-1)}\|_2^2$
M: $\mu_j^{(t)} = \frac{1}{n_j} \Sigma_{i: z_i^{(t)} = j}\, x_i$

Soft-EM algorithm: While not converged
E-step: For each i and j calculate $\gamma_j^{(t)}(x_i)$
$\gamma_j^t(x_i) = P(Z_i = j|x_i, \theta_t) = \frac{P(x_i|Z_i=j,\theta_t)P(Z_i=j|\theta_t)}{P(x_i;\theta_t)} =$
$= \frac{w_j P(x|\Sigma_j, \mu_j)}{\Sigma_l w_l P(x|\Sigma_l, \mu_l)} = \frac{w_j \mathcal{N}(x; \Sigma_j, \mu_j)}{\Sigma_l w_l \mathcal{N}(x; \Sigma_l, \mu_l)}$
$Q(\theta; \theta^{(t-1)}) = \mathbb{E}_{z_{1:n}}[\log P(x_{1:n}, z_{1:n}|\theta)|x_{1:n}, \theta^{(t-1)}]$
$= \mathbb{E}_{z_{1:n}}[\log \Pi_{i=1}^n P(x_i, z_i|\theta)|x_{1:n}, \theta^{(t-1)}] =$
$= \sum_{i=1}^n \mathbb{E}_{z_i}[\log P(x_{1:n}, z_i; \theta)|x_i, \theta^{(t-1)}] =$
$\sum_{i=1}^n \sum_{j=1}^k P(z_i = j|x_i, \theta^{(t-1)}) \log(P(x_i, z_i = j|\theta))$
$= \sum_{i=1}^n \sum_{j=1}^k \gamma_j^t(x_i) \log(P(z_i = j)P(x_i|z_i = j, \theta))$
If constraint $\Sigma_{j=1}^m P(z_i = j; \theta) = 1$ (m: #labels):
$\to \mathcal{L}(\theta, \lambda) = Q(\theta|\theta^{(t-1)}) + \lambda(\Sigma_j^m P(z_i = j) - 1)$
M-step: Fit clusters to weighted data points:
Genrl: $\theta^{(t)} = \text{argmax}_\theta\, Q(\theta; \theta^{(t-1)}), \gamma_j^t(x_i) fixed!$
$$w_j^{(t)} \leftarrow \frac{1}{n} \sum_{i=1}^n \gamma_j^{(t)}(x_i);\; \mu_j^{(t)} \leftarrow \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i) x_i}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)}$$
$$\Sigma_j^{(t)} \leftarrow \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i)(x_i - \mu_j^{(t)})(x_i - \mu_j^{(t)})^T}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)} \{+\nu^2 \mathbb{I}\}$$
SSL w/ GMMs: labeled p.: z_i: $\gamma_j^{(t)}(x_i) = 1[j = z_i]$
unl. p.: $\gamma_j^{(t)}(x_i) = P(Z = j|x_i, \mu^{(t-1)}, \Sigma^{(t-1)}, w^{(t-1)})$

Large Language Models

Sequence-to-sequence: Use RNN with hidden state, keep hidden state, encoder: current input + last hidden state $\to$ new hidden state, decoder: current hidden state $\to$ output token + new hidden state

Transformers
Use encoder/decoder architecture, don't need recurrence because of (self-) attention (multi-head), encoder: self-attention + feed-forward (FCNN), decoder: self-attention, encoder-decoder attention, feed-forward
Self-attention: For the tokens X, generate **query** $Q = X \times W_Q$, **key** $K = X \times W_K$, **value** $V = X \times W_V$. For each token, perform dot product of query and key, use soft-maxed version to scale value, i.e. $Z = \text{softmax}(\frac{Q \times K^\top}{\sqrt{d_k}})V$
Positional Encodings: Add to word embeddings, e.g. sine functions w/ different freq.
Enc.-Dec. Att.: dec.: Q so far. enc.: K, V

Task First, plug in the complete data log-likelihood into the equation for Q:
$Q(\lambda; \lambda^j) := \mathbb{E}_{z_{1:m}}[(n+m)\log(\lambda) - \lambda \sum_{i=1}^m t_i - \lambda \sum_{i=1}^m z_i \;|t_{1:n}; \lambda^{(j)}]$
The 1. and 2. summand don't depend on $z_{1:m}$ nor $t_{1:n}$ nor $\lambda^{(j)}$:
$= (n+m)\log(\lambda) + \lambda \sum_{i=1}^m t_i + \mathbb{E}_{z_{1:m}}[\lambda \sum_{i=1}^m z_i|t_{1:n}; \lambda^{(j)}]$
The third summand does not depend on $t_{1:n}$ butz_i ~Exp($\lambda^{(j)}$). $\mathbb{E}_{z_{1:m}}[\lambda \sum_{i=1}^m z_i|t_{1:n}; \lambda^{(j)}] = \lambda \sum_{i=1}^m \mathbb{E}_{z_{1:m}}[z_i|\lambda^{(j)}]$
We know that $z_i \geq \tau$. We use a new Exp($\lambda^{(j)}$) distributed variable z_i' to model this:
$\mathbb{E}_{z_{1:m}}[z_i|\lambda^{(j)}] = \mathbb{E}[z_i'|z_i' \geq \tau; \lambda^{(j)}]$
$= \frac{1}{\mathbb{P}[z_i' \geq \tau]} \int_\tau^{\inf} z_i' \lambda^{(j)} e^{-\lambda^{(j)} z_i'} dz_i'$ (below τ the value is 0)
$= \frac{1}{\mathbb{P}[z_i' \geq \tau]} (\tau\lambda^{(j)} + 1) e^{-\tau\lambda^{(j)}} \frac{1}{\lambda^{(j)}}$ (hint)
Furthermore:
$\mathbb{P}[z_i' \geq \tau] = \int_\tau^{\inf} \lambda^{(j)} e^{-\lambda^{(j)} z_i'} dz_i' = e^{-\lambda^{(j)}\tau}$
Thus we get:
$\frac{1}{e^{-\lambda^{(j)}\tau}} (\tau\lambda^{(j)} + 1) e^{-\tau\lambda^{(j)}} \frac{1}{\lambda^{(j)}} = (\tau\lambda^{(j)} + 1) \frac{1}{\lambda^{(j)}}$
Assembling this back together, we get for the third summand:
$\lambda \sum_{i=1}^m \mathbb{E}_{z_{1:m}}[z_i|\lambda^{(j)}] = \lambda^{(j)} \sum_{i=1}^m ((\tau\lambda^{(j)} + 1)\frac{1}{\lambda^{(j)}}) = \lambda^{(j)} m(\tau\lambda^{(j)} + 1)\frac{1}{\lambda^{(j)}} = m(\tau\lambda^{(j)} + 1) = m\lambda^{(j)}(\tau + \frac{1}{\lambda^{(j)}})$
Finally, summing the three summands again:
$= \mathbb{E}_{z_{1:m}}[(n+m)\log(\lambda)|t_{1:n}; \lambda^{(j)}] - \mathbb{E}_{z_{1:m}}[\lambda \sum_{i=1}^m t_i|t_{1:n}; \lambda^{(j)}] - \mathbb{E}_{z_{1:m}}[\lambda \sum_{i=1}^m z_i|t_{1:n}; \lambda^{(j)}]$
$= (n+m)\log(\lambda) - \lambda \sum_{i=1}^m t_i - m\lambda^{(j)}(\tau + \frac{1}{\lambda^{(j)}})$