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

convex $\iff f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda)f(y); f(y) \ge f(x) + \langle \nabla f(x), y-x \rangle; D^2 f(x) \ge 0$
$\alpha f + \beta g$ **c.**; $max(f,g)$ **c.** if f,g **c.**, $\alpha, \beta \ge 0$
$f \circ g = f(g(x))$ **c.** if f **c.**, g **a.** $\vee$ f **c., non-dec.**, g **c.**
$p_{\mu,\Sigma}(x) = \frac{1}{\sqrt{(2\pi)^p det(\Sigma)}} \exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu))$
$\mathbf{X} \sim \mathcal{N}(\mu, \Sigma) \implies A\mathbf{X} + b \sim \mathcal{N}(A\mu + b, A\Sigma A^T)$
Jensen ineq: $g(E[X]) \le E[g(X)]$, g convex

Regression

Linear Regression $f(x) = w^Tx; X \in \mathbb{R}^{n \times d}$
$L(w) = \|Xw - y\|_2^2; X^TX\hat{w} = X^Ty$
$d \le n: \hat{w} = (X^TX)^{-1}X^Ty$ 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 $\|w^t - w^{t-1}\|_2 \le \epsilon$

GD conv. to $\hat{w}$ if $rk(X^TX) = d, \eta < \frac{2}{\lambda_{max}(X^TX)}$
$\|w^{t+1} - \hat{w}\| \le \|I - \eta X^TX\|_{op}\|w^t - \hat{w}\|_2 \le \rho^{t+1}\|w^0 - \hat{w}\|_2; \eta_{opt} = \frac{2}{\lambda_{max}+\lambda_{min}}; \rho_{min} = 1 - \eta_{opt}\lambda_{min} = \frac{\kappa-1}{\kappa+1}$
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
$\text{Bias}^2_{\mathcal{D}}(\hat{f}_{\mathcal{D}}, x) := (E_{\mathcal{D}}(\hat{f}_{\mathcal{D}}(x)) - f^*(x))^2$
$\text{Bias}^2_{\mathcal{D}}(\hat{f}_{\mathcal{D}}) := E_X(\text{Bias}^2_{\mathcal{D}}(\hat{f}_{\mathcal{D}}, X)); \text{Var}_{\mathcal{D}}(\hat{f}_{\mathcal{D}}) := E_X(\text{Var}_{\mathcal{D}}(\hat{f}_{\mathcal{D}}(X))); E_{\mathcal{D}}(L(\hat{f}_{\mathcal{D}}; \mathbb{P}_{X,Y})) = \text{Var}_{\mathcal{D}}(\hat{f}_{\mathcal{D}}) + \text{Bias}^2_{\mathcal{D}}(\hat{f}_{\mathcal{D}}) + \sigma^2$

Ridge closed form: $\hat{w} = (X^TX + \lambda I)^{-1}X^Ty$

Classification

Perceptron $y = sign(f(x)) = sign(w^Tx)$
Perceptron loss is convex and not differentiable, but gradient is informative.
$l_P(w; y_i, x_i) = max\{0, -y_i w^T x_i\}$
$w^* = \text{argmin}_w \sum_{i=1}^n l_p(w; y_i, x_i)$
$\nabla_w l_p(w; y_i, x_i) = (-y_i x_i) 1[y_i w^T x_i < 0]$

Stochastic Gradient Descent (SGD)
1. Start at an arbitrary $w_0 \in \mathbb{R}^d$
2. For $t = 1, 2, ...$ do:

Pick data point $(x', y') \in_{u.a.r.} D$
$w_{t+1} = w_t - \eta_t \nabla_w l(w_t; x', y')$
Perceptron Alg: SGD with Perceptron loss

Support Vector Machine
Hinge loss: $l_H(w; x, y) = max\{0, 1 - yw^Tx\}$
Goal: Max. a "band" around the separator.
$w^* = \underset{w}{\text{argmin}} \frac{1}{n}\sum_{i=1}^n (max\{0, 1 - y_i w^T x_i\} + \lambda\|w\|_2^2)$
$g_i(w) = max\{0, 1 - y_i w^T x_i\} + \lambda\|w\|_2^2$
$\nabla_w g_i(w) = \begin{cases} -y_i x_i + 2\lambda w & \text{, if } y_i w^T x_i < 1 \\ 2\lambda w & \text{, if } y_i w^T x_i \ge 1 \end{cases}$

Multi-Class Classification
Confidence $\to$ Distance from Decision Bound.
$y = \text{amin}_{i\in\{1,..,c\}} f_i(x), f_i(x) = \widetilde{w_i}^T x, \widetilde{w_i} = \frac{w_i}{\|w_i\|_2}$
OvA: $\hat{y}_i = \text{argmax}_{j\in\{1,..,c\}} w_j^T x_i$; C bin. classif
OvO: Train $\frac{c(c-1)}{2}$ bin. classif., one for each pair (i,j). Voting $\to$ class with most positive preditions wins (slower, but no confidence needed)

Kernels $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}; x_i^T x_j \to k(x_i, x_j)$

Reformulating the perceptron
Ansatz: $w = \sum_{j=1}^n \alpha_j y_j x_j$
$w^* = \min_{w\in\mathbb{R}^d} \sum_{i=1}^n \max[0, -y_i w^T x_i]$
$\Leftrightarrow \alpha^* = \min_{\alpha_{1:n}} \sum_{i=1}^n \max[0, -\sum_{j=1}^n \alpha_j y_i y_j x_i^T x_j]$

Kernelized Perceptron
1. Initialize $\alpha_1 = ... = \alpha_n = 0$
2. For $t = 1, 2, ...$ do

Pick data $(x_i, y_i) \in_{u.a.r} D$
Predict $\hat{y} = sign(\sum_{j=1}^n \alpha_j y_j k(x_j, x_i))$
If $\hat{y} \ne y_i$ set $\alpha_i^{(t)} = \alpha_i^{(t-1)} + \eta_t$ | else: $\alpha_i^{(t)} = \alpha_i^{(t-1)}$
Predict new point x: $\hat{y} = sign(\sum_{j=1}^n \alpha_j y_j k(x_j, x))$

Perceptron and SVM
Perceptron: $\underset{\alpha}{min} \sum_{i=1}^n max\{0, -y_i \alpha^T k_i\}$
SVM: $k_i = [y_1 k(x_i, x_1), ..., y_n k(x_i, x_n)]$:
$\underset{\alpha}{min} \sum_{i=1}^n max\{0, 1 - y_i \alpha^T k_i\} + \lambda \alpha^T D_y K D_y \alpha$
Prediction: $y = sign(\sum_{j=1}^n \alpha_j y_j k(x_j, x))$

Properties of kernel $k(x,y) = \phi(x)^T \phi(y)$
k must be symmetric: $k(x,y) = k(y,x)$
Kernel matrix must be positive semi-definite.

positive semi-definite matrices
$M \in \mathbb{R}^{n\times n}$ is psd $\Leftrightarrow \forall x \in \mathbb{R}^n : x^TMx \ge 0 \Leftrightarrow$ all eigenvalues of M are positive: $\lambda_i \ge 0$

k Nearest Neighbor classifier
$y = sign(\sum_{i=1}^n y_i [x_i \text{ among k nn of } x])$

Examples of kernels on $\mathbb{R}^d$
Linear kernel: $k(x,y) = x^Ty$
Polynomial kernel: $k(x,y) = (x^Ty + 1)^d$
Gaussian kernel: $k(x,y) = exp(-\|x-y\|_2^2/h^2)$
Laplacian kernel: $k(x,y) = exp(-\|x-y\|_1/h)$

Kernel engineering
$k_1(x,y) + k_2(x,y); k_1(x,y)\cdot k_2(x,y); c\cdot k_1(x,y), c > 0; f(k_1(x,y))$, where f is a polynomial with positive coefficients or the exponential function

Kernelized linear regression
Ansatz: $w^* = \sum_i \alpha_i x$
Parametric: $w^* = \underset{w}{\text{argmin}} \sum_i (w^T x_i - y_i)^2 + \lambda\|w\|_2^2$
$= \text{amin}_\alpha \|\alpha^T K - y\|_2^2 + \lambda \alpha^T K \alpha, \alpha^* = (K + \lambda I)^{-1} y$
Prediction: $y = w^{*T} x = \sum_{i=1}^n \alpha_i^* k(x_i, x)$

Imbalance

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=$\frac{\#correct'+'predictions}{\#all'+'predictions} = \frac{TP}{TP+FP}$
Recall=TPR=$\frac{TP}{TP+FN} = \frac{TP}{n_+}$, FPR=$\frac{FP}{TN+FP} = \frac{FP}{n_-}$
F1 score $= \frac{2TP}{2TP+FP+FN} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$

Multi-class Hinge Loss

$l_{MC-H}(w^{(1)}, ..., w^{(c)}; x, y) = \max_{j\in\{[1,...,c]\setminus y\}}(0, 1 + \max w^{(j)T}x - w^{(y)T}x)$

Neural Networks

$F(x) = \Sigma_{i=1}^k w_i^{(2)} \phi(\Sigma_{j=1}^m w_{ij}^{(1)} x_j) = W^{(2)}\phi(W^{(1)}x)$
$F(x) = \phi^{(L)}(W^{(L)}\phi^{(L-1)}(W^{(L-1)}...(\phi^{(1)}(W^{(1)}x)...)))$

Learning features
Parametr. feat. maps & optimize over params:
$w^* = \text{argmin}_{w,\theta} \sum_{i=1}^n l(y_i; \sum_{j=1}^m w_j \phi(x_i, \theta_j))$
One possibility: $\phi(x, \theta) = \varphi(\theta^T x) = \varphi(z)$

Activation functions
Sigmoid: $\varphi(z) = \frac{1}{1+exp(-z)}; \varphi'(z) = (1 - \varphi(z))\cdot\varphi(z)$
Tanh$_{[-1,1]}$: $\varphi(z) = tanh(z) = \frac{exp(z) - exp(-z)}{exp(z) + exp(-z)}$
ReLu: $\varphi(z) = max(z, 0)$

Forward propagation
For each unit j on input layer, set value $v_j = x_j$
For each layer $l = 1 : L-1$: For each unit j on layer l set its value $v_j = \varphi(\sum_{i\in Layer_{l-1}} w_{j,i} v_i)$
For each unit j on output layer, set its value $f_j = \sum_{i\in Layer_{L-1}} w_{j,i} v_i$ resp. $\vec{f} = W^{(L)} v^{(L-1)}$
Predict $y_j = f_j$ for reg. / $y_j = sign(f_j)$ for class.

Backpropagation
For each unit j on the output layer L:
- Compute error signal: $\delta_j^{(L)} = \ell'_j(f_j)$
- For each unit i on layer $L-1$: $\frac{\partial l}{\partial w_{j,i}^{(L)}} = \delta_j^{(L)} v_i^{(L-1)}$

For each unit j on hidden layer $l = \{L-1, ..., 1\}$:
- Error sig: $\delta_j^{(l)} = \varphi'(z_j^{(l)}) \sum_{i\in Layer_{l+1}} w_{i,j}^{(l+1)} \delta_i^{(l+1)}$
- For each unit i on layer $l-1$: $\frac{\partial l}{\partial w_{j,i}^{l}} = \delta_j^{(l)} v_i^{(l-1)}$

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

Clustering

k-mean
Loss: $\hat{R}(\mu) = \hat{R}(\mu_1, ..., \mu_k) = \sum_{i=1}^n \min_{j\in\{1,...k\}} \|x_i - \mu_j\|_2^2$
$\hat{\mu} = \underset{\mu}{argmin} \hat{R}(\mu)$; non-convex, $\mathcal{O}(NP)$

Algorithm (Lloyd's heuristic):
Initialize cluster centers $\mu^{(0)} = [\mu_1^{(0)}, ..., \mu_k^{(0)}]$
While still changes in assignments:
$z_i = \underset{j\in\{1,...,k\}}{\text{argmin}} \|x_i - \mu_j^{(t-1)}\|_2^2; \mu_j^{(t)} = \frac{1}{n_j} \sum_{i:z_i=j} x_i$

k-mean++:
- Start with random data point as center
- Add centers 2 to k randomly, proportionally to squared distance to closest selected center for $j = 2$ to k: i_j sampled with prob.

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

Dimension Reduction

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

Kernel PCA
For general $k \ge 1$, 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: $K = \sum_{i=1}^n \lambda_i v_i v_i^T, \lambda_1 \ge ... \ge \lambda_d \ge 0$
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

d input, d output units, 1 layer w/ $k<d$ units

$W^*=\text{argmin}_W \sum_{i=1}^n \|x_i - W^{(2)}\varphi(W^{(1)}x^{(i)})\|_2^2$

$\varphi(z)=z: NNA=PCA, w^{(1)}=PCA(x)=w^{(2)^T}$

Probability Modeling

Assumption: Data set is generated iid

Find $h: X\to Y$ that minimizes pred. error

$R(h)=\int P(x,y)l(y;h(x))\partial x\partial y=\mathbb{E}_{x,y}[l(y;h(x))]$

$h^*(x)=\mathbb{E}[Y|X=x]$ for $R(h)=\mathbb{E}_{x,y}[(y-h(x))^2]$

Prediction: $\hat y=\hat{\mathbb{E}}[Y|X=x]=\int \hat P(y|X=x)y\partial y$

Maximum Likelihood Estimation (MLE)

Choose a particular parametric form $\hat P(Y|X,\theta)$

$\theta^*=\underset{\theta}{\text{amax}}\hat P(y_{1:n}|x_{1:n},\theta)\overset{\text{iid}}{=}\underset{\theta}{\text{amax}}\prod_{i=1}^n\hat P(y_i|x_i,\theta)$

$=\text{amin}_\theta -\sum_{i=1}^n log\hat P(y_i|x_i,\theta)$

Ex: $y_i\sim\mathcal N(w^Tx_i,\sigma^2)$: $w^*=\text{amin}_w\sum_i^n(y_i-w^Tx_i)$

Bias/Variance/Noise

Prediction error $= Bias^2+Variance+Noise$

Maximum a posteriori estimate (MAP)

Introduce bias by expressing assumption through a Bayesian prior $w_i\sim\mathcal N(0,\beta^2)$

Bayes: $P(w|x,y)=\frac{P(w|x)P(y|x,w)}{P(y|x)}=\frac{P(w)P(y|x,w)}{P(y|x)}$

assume w indep. of x: $\text{argmax}_w P(w|x,y)=$

$=\text{argmin}_w -logP(w)-logP(y|x,w)+const.$

$=\text{argmin}_w \lambda\|w\|_2^2+\sum_{i=1}^n(y_i-w^Tx_i)^2$, $\lambda=\frac{\sigma^2}{\beta^2}$

($=\underset{w}{\text{argmax}}P(w)\prod_i P(y_i|x_i,w)$, assuming noise $P(y|x,w)$ iid Gaussian, prior $P(w)$ Gaussian)

Logistic regression

Assume iid Bernoulli noise instead of Gauss.

$P(y|x,w)=Ber(y;\sigma(w^Tx))=\frac{1}{1+exp(-yw^Tx)}$

$l_{logistic}(w;x_i,y_i)=log(1+exp(-y_iw^Tx_i))$

$\nabla_w l(w)=\frac{(-y_ix_i)}{1+exp(+y_iw^Tx_i)}=P(Y=-y|x,w)(-y_ix_i)$

Example: MLE for logistic regression

$\text{argmax}_w P(y_{1:n}|w,x_{1:n})$

$=\text{argmin}_w -\sum_{i=1}^n logP(y_i|w,x_i)$

$=\text{argmin}_w\sum_{i=1}^n log(1+exp(-y_iw^Tx_i))$

$\hat R(w)=\sum_{i=1}^n log(1+exp(-y_iw^Tx_i))$ (neg log l. f.)

Logistic regression and regularization

$\underset{w}{\min}\sum_{i=1}^n log(1+exp(-y_iw^Tx_i))+\lambda(\|w\|_1 or \|w\|_2^2)$

SGD for logistic regression

Update $w\leftarrow w+\eta_t yx\hat P(Y=-y|w,x)$

L2 regularized logistic regression:

Update $w\leftarrow w(1-2\lambda\eta_t)+\eta_t yx\hat P(Y=-y|w,x)$

Multiclass Logistic Regression

$P(Y=i|x,w_1,..,w_c)=exp(w_i^Tx)/\sum_j^c exp(w_j^Tx)$

Bayesian decision theory

- Conditional distribution over labels $P(y|x)$
- Set of actions $\mathcal A$ - Cost function $C: Y\times\mathcal A\to\mathbb R$

Pick action that minimizes the expected cost:

$a^*=\text{argmin}_{a\in\mathcal A}\mathbb E_y[C(y,a)|x]=\sum_y P(y|x)C(y,a)$

$\mathbb E_y[C(y,+)|x]=P(-|x)C(-,+)$;

$\mathbb E_y[C(y,-)|x]=P(+|x)C(+,-)$;

$\mathbb E_y[C(y,D)|x]=P(+|x)C_{d+}+P(-|x)C_{d-}$

Optimal decision for logistic regression

$a^*=\text{argmin}_y \hat P(y|x)=sign(w^Tx)$

Doubtful logistic regression

Est. cond. distr.: $\hat P(y|x)=Ber(y;\sigma(\hat w^Tx))$

Action set: $\mathcal A=\{+1,-1,D\}$; Cost function:

$C(y,a)=\begin{cases}[y\neq a] & \text{if } a\in\{+1,-1\}\\ c & \text{if } a=D\end{cases}$

$\to a^*=y$ if $\hat P(y|x)\ge 1-c$, D otherwise

Linear regression

Est. cond. distr.: $\hat P(y|x,w)=\mathcal N(y;w^Tx,\sigma^2)$

$\mathcal A=\mathbb R$; $C(y,a)=(y-a)^2$

$\to a^*=\mathbb E_y[y|x]=\int\hat P(y|x)\partial y=\hat w^Tx$

Asymmetric cost for regression

Est. cond. distr.: $\hat P(y|x)=\mathcal N(\hat y;\hat w^Tx,\sigma^2)$

$\mathcal A=\mathbb R$; $C(y,a)=c_1\max(y-a,0)+c_2\max(a-y,0)$

$\to a^*=\hat w^Tx+\sigma\Phi^{-1}(\frac{c_1}{c_1+c_2})$, Φ: Gaussian CDF

Discriminative vs. Generative Modeling

Discriminative models: aim to estimate $P(y|x)$

G. m.: aim to estimate joint distribution $P(y,x)$

Typical approach to generative modeling:

- Estimate prior on labels $P(y)$
- Estimate cond. distr. $P(x|y)$ for each class y
- Obtain predictive distr. using Bayes' rule:

$P(y|x)=\frac{P(y)P(x|y)}{P(x)}=\frac{P(x,y)}{P(x)}$, $P(x)=\sum_y P(x,y)$

Example MLE for P(y)

Want: $P(Y=1)=p, P(y=-1)=1-p$

Given: $D=\{(x_1,y_1),...,(x_n,y_n)\}$

$P(D|p)=\prod_{i=1}^n p^{1[y_i=+1]}(1-p)^{1[y_i=-1]}$

$=p^{n_+}(1-p)^{n_-}$, where $n_+=\#$ of $y=+1$

$\frac{\partial}{\partial p}logP(D|p)=n_+\frac1p-n_-\frac{1}{1-p}\overset{!}{=}0\Rightarrow p=\frac{n_+}{n_++n_-}$

Example MLE for P=(x|y)

Assume: $P(X=x_i|y)=\mathcal N(x_i;\mu_{i,y},\sigma_{i,y}^2)$

Given: $D, D_{x_i|y}=\{x, \text{s.t. } x_{j,i}=x, y_j=y\}$

Thus MLE yields: $\hat\mu_{i,y}=\frac{1}{n_y}\sum_{x\in D_{x_i|y}}x$;

$\hat\sigma_{i,y}^2=\frac{1}{n_y}\sum_{x\in D_{x_i|y}}(x-\hat\mu_{i,y})^2$

Deriving decision rule

$P(y|x)=\frac1Z P(y)P(x|y)$, $Z=\sum_y P(y)P(x|y)$

$y=\text{argmax}_{y'}P(y'|x)=\text{argmax}_{y'}P(y')\prod_{i=1}^d P(x_i|y')$

$=\text{argmax}_{y'}logP(y')+\sum_{i=1}^d logP(x_i|y')$

Gaussian Naive Bayes classifier

Indep. feat. giv. Y: $P(X_1,..,X_n|Y)=\prod_{i=1}^d P(X_i|Y)$

MLE for class prior: $\hat P(Y=y)=\hat p_y=\frac{\text{Count}(Y=y)}{n}$

MLE for feature distr.: $\hat P(x_i|y_i)=\mathcal N(x_i;\hat\mu_{y,i},\sigma_{y,i}^2)$

$\hat\mu_{y,i}=\frac{1}{\text{Count}(Y=y)}\sum_{j:y_j=y}x_{j,i}$

$\sigma_{y,i}^2=\frac{1}{\text{Count}(Y=y)}\sum_{j:y_j=y}(x_{j,i}-\hat\mu_{y,i})^2$

Prediction given new point x:

$y=\text{argmax}_{y'}\hat P(y'|x)=\text{argmax}_{y'}\hat P(y')\prod_{i=1}^d\hat P(x_i|y')$

Categorical Naive Bayes Classifier

MLE class prior: $\hat P(Y=y)=p_y=\frac{Count(Y=y)}{n}$

MLE for feature distr.: $\hat P(X_i=c|Y=y)=\theta_{c|y}^{(i)}$

$\theta_{c|y}^{(i)}=\frac{Count(X_i=c,Y=y)}{Count(Y=y)}$, Pred: $y=\text{amax}_{y'}\hat P(y'|x)$

Discr fnc: $f(x)=log\frac{P(y=1|x)}{P(y=-1|x)}$; $p(x)=\frac{1}{1+exp(-f(x))}$

Gaussian Bayes Classifier

MLE for class prior: $\hat P(Y=y)=\hat p_y=\frac{\text{Count}(Y=y)}{n}$

MLE for feature distr.: $\hat P(\mathbf x|y)=\mathcal N(\mathbf x;\hat\mu_y,\hat\Sigma_y)$

$\hat{\boldsymbol\mu}_y=\frac{1}{\text{Count}(Y=y)}\sum_{i:y_i=y}\mathbf x_i\in\mathbb R^d$

$\hat\Sigma_y=\frac{1}{\text{Count}(Y=y)}\sum_{i:y_i=y}(x_i-\hat\mu_y)(x_i-\hat\mu_y)^T\in\mathbb R^{d\times d}$

Fisher's linear discriminant analysis (LDA; c=2)

Assume: $p=0.5$; $\hat\Sigma_-=\hat\Sigma_+=\hat\Sigma$

discriminant f.: $f(x)=log\frac{p}{1-p}+\frac12[log\frac{|\hat\Sigma_-|}{|\hat\Sigma_+|}$

$+((x-\hat\mu_-)^T\hat\Sigma_-^{-1}(x-\hat\mu_-))-((x-\hat\mu_+)^T\hat\Sigma_+^{-1}(x-\hat\mu_+))]$

Predict: $y=sign(f(x))=sign(w^Tx+w_0)$

$w=\hat\Sigma^{-1}(\hat\mu_+-\hat\mu_-)$; $w_0=\frac12(\hat\mu_-^T\hat\Sigma^{-1}\hat\mu_- - \hat\mu_+^T\hat\Sigma^{-1}\hat\mu_+)$

Outlier Detection

$P(x)=\sum_{y=1}^c P(y)P(x|y)=\sum_y\hat p_y\mathcal N(x|\hat\mu_y,\hat\Sigma_y)\le\tau$

Latent: Missing Data (Gaussian distr.)

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.: $P(D|\theta)=\prod_{i=1}^n\sum_{j=1}^k w_jP(x_i|\theta_j)$

3)θ should minimize neg log-likelihood:

$\theta^*=\underset{\theta}{\text{amin}}L(D;\theta)=\underset{\theta}{\text{amin}}-\sum_i log\sum_j w_jP(x_i|\theta_j)$

Ex: $P(x|\theta)=\sum_i w_i\mathcal N(x;\mu_i,\Sigma_i)$, $P(z_i=j)=w_j$

$\sum w_i=1$, $P(z,x)=w_z\mathcal N(x|\mu_z,\Sigma_z)$

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.: $\theta^{(t)}=\text{argmax}_\theta P(D^{(t)}|\theta)$

Special case: fix $w_z=\frac1k$, 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: $\mu_j^{(t)}=\frac{1}{n_j}\sum_{i:z_i^{(t)}=j}x_i$

Soft-EM algorithm: While not converged

E-step: For each i and j calculate $\gamma_j^{(t)}(x_i)$

$\gamma_j^t(x_i)=P(Z_i=j|x_i,\theta_t)=\frac{P(x_i|Z_i=j,\theta_t)P(Z_i=j|\theta_t)}{P(x_i;\theta_t)}=$

$=\frac{w_jP(x|\Sigma_j,\mu_j)}{\sum_l w_lP(x|\Sigma_l,\mu_l)}=\frac{w_j\mathcal N(x;\Sigma_j,\mu_j)}{\sum_l w_l\mathcal N(x;\Sigma_l,\mu_l)}$

$Q(\theta;\theta^{(t-1)})=\mathbb E_{y_{1:n}}[\log P(x_{1:n},y_{1:n}|\theta)|x_{1:n},\theta^{(t-1)}]$

$=\mathbb E_{y_{1:n}}[\log\prod_{i=1}^n P(x_i,y_i|\theta)|x_{1:n},\theta^{(t-1)}]=$

$=\sum_{i=1}^n\mathbb E_{y_i}[\log P(x_{1:n},y_i;\theta)|x_i,\theta^{(t-1)}]=$

$\sum_{i=1}^n\sum_{j=1}^k P(y_i=j|x_i,\theta^{(t-1)})\log(P(x_i,y_i=j;\theta))$

$=\sum_{i=1}^n\sum_{j=1}^k\gamma_j^t(x_i)\log(P(y_i=j)P(x_i|y_i=j;\theta))$

If constraint $\sum_{j=1}^m P(y_i=j;\theta)=1$ (m: #labels):

$\to\mathcal L(\theta,\lambda)=Q(\theta;\theta^{(t-1)})+\lambda(\sum_j^m P(y_i=j)-1)$

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!*

$w_j^{(t)}\leftarrow\frac1n\sum_{i=1}^n\gamma_j^{(t)}(x_i)$; $\mu_j^{(t)}\leftarrow\frac{\sum_{i=1}^n\gamma_j^{(t)}(x_i)x_i}{\sum_{i=1}^n\gamma_j^{(t)}(x_i)}$

$\Sigma_j^{(t)}\leftarrow\frac{\sum_{i=1}^n\gamma_j^{(t)}(x_i)(x_i-\mu_j^{(t)})(x_i-\mu_j^{(t)})^T}{\sum_{i=1}^n\gamma_j^{(t)}(x_i)}\{+\nu^2\mathbb I\}$

SSL w/ GMMs: labeled p.: y_i: $\gamma_j^{(t)}(x_i)=1[j=y_i]$

unl. p.: $\gamma_j^{(t)}(x_i)=P(Z=j|x_i,\mu^{(t-1)},\Sigma^{(t-1)},w^{(t-1)})$

Additions: small variance & high bias $\to$ too simple model

CNN filter output size: $L=\frac{n+2p-f}{s}+1$