$\frac{\partial x^TA}{\partial x} = A^T$, $\frac{\partial x^TAx}{\partial x} = A^Tb$, Hessian $\frac{\partial^2}{\partial w_i \partial w_j}$ pos. sem. def ⇒ loss convex
zero mean, unit variance: $\tilde{x}_j = \frac{(x_j - \mu_j)}{\sigma_j}$, μ_j, σ_j for each dim.

Imbalanced Data: (classify all data to larger class ⇒ small error but bad classifier! - up/downsampling: sample from smaller class or throw away from larger. - normal trained classifier with threshold: $y = \text{sign}(w^Tx - \tau)$ - Use cost-sensitive loss: $\ell(y, h(x)) = c_y \ell(y, h(x))$, c_y is tuned and depends on class.
Confusion matrix: + is smaller class, true pos. rate
Precision: $\frac{TP}{TP+FP}$, Recall = $\frac{TP}{TP+FN}$, FPR = $\frac{FP}{TN+FP}$
Accuracy = $\frac{TP+TN}{n}$, F_1 score = $\frac{2TP}{2TP+FP+FN} = \frac{2}{\frac{1}{prec} + \frac{1}{recall}}$ harm. mean
Precision/Recall curve: takes class ratios into account: best, random: prec = $\frac{n_+}{n}$ = fix, recall = p = prob(y = +), can vary p
ROC: (Recall/FPR): class ratios not taken into account: best, rand.
Alg 1 dominates Alg 2 (always higher) in ROC ⇔ 1 dom. 2 in Prec/Recall
Use area und curve (ROC or other) to compare algorithms. Every point on curve corresponds to a threshold τ.

Gradient Descent: $w_{t+1} = w_t - \eta_t \nabla_w R(w_t)$
step size sufficiently small ⇒ converge to local opt.
loss convex: $g(\lambda x + (1-\lambda)x') \leq \lambda g(x) + (1-\lambda)g(x')$ ⇒ local opt = global
η_t: - line search at every step $\eta_t = \arg\min_\eta R(w_t - \eta \nabla_w R(w_t))$ same for descent
- bold driver heuristic: $R(w_{t+1}) < R(w_t) \Rightarrow \eta_{t+1} = \eta_t \cdot c_{inc}$, $c_{inc} > 1$

lin. regr. analysis: Gradient descent: $G(w) = w - \frac{\eta}{n}[X^TXw - X^Ty] = [I - \frac{\eta}{n}X^TX]w + \frac{\eta}{n}X^Ty$
w^* opt: $G(w) - w^* = [I - \frac{\eta}{n}X^TX](w - w^*) + \frac{\eta}{n}[X^Ty - X^TXw^*]$, contraction factor $g = \sup_{\|A v\|}$ ...
we want $g < 1$
Use $\Sigma = \frac{1}{n}X^TX$ and $\Sigma v = \lambda v$ (v evec of Σ) ⇒ $[I - \eta\Sigma]v = (1 - \eta\lambda)v$ ⇒ $\|Av\| = 1 - \eta\lambda$
Σ symm: $\lambda \geq 0$, Σ regular: $\lambda \neq 0$ ⇒ $\lambda > 0$: $|1 - \eta\lambda| \leq g < 1$ ⇒ $\eta \geq \frac{1-g}{\lambda}$ and $\eta \leq \frac{1+g}{\lambda}$ $\forall \lambda$
Use strongest bound: $\eta \geq \frac{1-g}{\lambda_{min}}$, $\eta \leq \frac{1+g}{\lambda_{max}}$, Use highest possible: $\eta_* = \frac{1+g}{\lambda_{max}}$
g: $\eta_0 = \frac{1-g}{\lambda_{min}} \Rightarrow g = \frac{1-\frac{\lambda}{\lambda+g}}{...} \leq 1 - \frac{\lambda}{...} < 1 - \frac{\lambda_{min}}{\lambda_{max}}$ small a preferred! a: condition number of Σ
Regularization $\frac{\mu}{2}\|w\|_2^2 \Rightarrow a = \frac{\lambda_{max} + \mu}{\lambda_{min} + \mu}$: choose $\mu = \sqrt{\lambda_{min}\lambda_{max}}$: $a \approx \sqrt{a}$
⇒ faster convergence, $\frac{1}{k}$ = desired precision = $g^k \Rightarrow k = \frac{\ln k}{-\ln(1 - \frac{1}{a})} \leq a \ln k$ #iter
Stochastic (SGD): choose i randomly (uniform)
Use only $\hat{R}(w_t, x_i, y_i) \Rightarrow \mathbb{E}[\nabla\hat{R}] = \nabla R_{(w_t, X, Y)}$, Guaranteed to converge if: $\sum\eta_t = \infty$ and $\sum\eta_t^2 < \infty$, i.e. $\eta_t = \frac{1}{t}$

Kernels: efficient, implicit representation of inner products, act in very high dimension ⇒ regularize to avoid overfitting (already done in kern. lin. regr. and SVM, but not in kern. Perceptron)
Kernels transform parametric models into nonparametric models ⇒ complexity grows with size of data, all kernelized versions and kNN
Kernels cannot explicitly embedd high-dim data (unless have appropriate kernel). Good kernel: - valid (sym. psd) - fast to compute - makes sense in problem (similarity) h, b, k etc via CV
linear kernel x^Tx', polynomial kernel up to degree m: $(1 + x^Tx')^m$, only degree m: $(x^Tx')^m$, Gaussian (RBF) kern: $\exp\left(-\frac{\|x - x'\|_2^2}{h^2}\right)$
RBF: $[0, 1]$, 1 if x near x', plots points on infinite dim unit sphere
Sigmoid kernel (tanh kernel): $\tanh(Kx^Tx' - b)$, Laplacian kernel: $\exp(-\gamma\|x - x'\|_1)$, Spectrum kernel: Bag of words = occurrence vectors with 1 words, 2 words etc until substring of size k, for sequences
Gram-matrix $K = k(x_i, x_j) = \Phi^T\Phi$, K must be symm ($k(x, y) = k(y, x)$) and pos. sem. def.: $w^TKw \geq 0 \; \forall w$. All w can be written as lin. combination of eigenvectors u of K: $Ku_i = \lambda_i u_i$
: $u^TKu = \lambda u^Tu = \lambda \geq 0$, $K = USU^T = \Phi^T\Phi = \sqrt{S}U^T$, $K = \Phi^T\Phi \Rightarrow w^TKw = \|\Phi w\|_2^2 \geq 0$
For kernel trick: requirement $w^* = \sum \alpha_i x_i$
Representer theorem: loss $= f(w^Tx) + c\|w\|_2^2 \Rightarrow w^* = \sum \alpha_i x_i$
Mercers theorem: X compact set and k = kernel ⇒ $k(x, x') = \sum_{i=1}^\infty \lambda_i \phi_i(x) \cdot \phi_i(x')$ Kernel engineering:
k_1, k_2 kernels, then k also kernel if: $k = k_1 + k_2$, $k = c \cdot k_1$, $k = k_1 \cdot k_2$, $c > 0$
$k = f(k_1)$ iff f = exponential fct. or polynomial with positive coefficients, $k = x^TMx$ if M = symm. and pos. sem. def.
$k = 2^{k_1} = \exp(k_1 \ln 2)$ = kernel, $|X \cap Y| = \langle x, y \rangle$ where $x_i = 1$ if element i is in X, 0 else. $|X \cup Y| = |X| + |Y| - |X \cap Y|$

Probabilistic Modelling: Assumption: $(x_i, y_i) \sim P(x, y)$ iid
Don't know $P(x, y)$ ⇒ use estimate $\hat{P}(x, y)$ from data, Expected value $\mathbb{E}_x[x] = \int P(x) x \, dx = \mu$ (mean), Prediction error: $R(h) = \int P(x, y) \ell(y, h(x)) \, dx \, dy = \mathbb{E}_{x,y}[\ell(y, h(x))]$
$\min \mathbb{E}[x] \geq \mathbb{E}[\min x]$

Bayes rule: $P(w|x_i, y_i) = \frac{P(w|x_i) \cdot P(y_i|x_i, w)}{P(y_i|x_i)}$
Bayes optimal predictor = $\min_h R(h)$ ⇒ (squared loss): $h^*(x) = \mathbb{E}[Y|X = x] = \int y P(y|x) \, dy$
Maximum likelihood: $\arg\max_w \sum_{i=1}^n -\log P(y_i|x_i, w)$ = parametric estimation: choose param. form for $\hat{P}(\cdot)$ and optimize: Assume Gaussian noise: $P(y|x) = \mathcal{N}(w^Tx, \sigma^2)$
$-\log\hat{P}(y|X, w) = \frac{n}{2}\ln(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum_{i=1}^n (y_i - w^Tx_i)^2$ ⇒ least squares
Bias-variance tradeoff: $y = f(x) + \varepsilon$, $f(x)$ = true, $\mathbb{E}[\varepsilon]$, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, $\hat{f}(x)$ = estimated:
Max a posteriori: $\arg\min_w \sum_{i=1}^n -\log \hat{P}(w|x_i, y_i)$
Prediction error $= \mathbb{E}[(y - \hat{f}(x))^2] = \mathbb{E}[y^2] + \mathbb{E}[\hat{f}^2] - 2\mathbb{E}[y\hat{f}] = \text{Var}(y) + \mathbb{E}[y]^2 + \text{Var}(\hat{f}(x)) + \mathbb{E}[\hat{f}(x)]^2 - 2f(x)\mathbb{E}[\hat{f}(x)] = \sigma^2 + \text{Var}(\hat{f}(x)) + \mathbb{E}[f(x) - \mathbb{E}[\hat{f}(x)]]^2$
$= \text{noise} + \text{var} + \text{bias}^2$
MLE = unbiased with min variance among all unbiased, but prone to overfit ⇒ trade bias for less variance via regularization.

Regularization vs. prior: Conj. prior: post ∝ prior: Dirichlet prior & categorical/multinomial likelihood, Beta prior & Bernoulli (Binomial likelihood), noise $\mathcal{N}(0, \sigma^2)$ & prior $w \sim \mathcal{N}(0, \beta^2 I)$
⇒ ridge regression $\lambda = \sigma^2/\beta^2$, Laplacian prior: $\frac{1}{2b}\exp\left(-\frac{|x - \mu|}{b}\right)$ ⇒ ℓ_1 regularizer $\hat{=}$ LASSO
Student-t likelihood is more robust against outliers than Gaussian
$\propto \left(1 + \frac{(y - w^Tx)^2}{\nu\sigma^2}\right)^{-\frac{\nu+1}{2}}$, Beta prior: $\frac{1}{B(\alpha_+, \alpha_-)} p_+^{\alpha_+ - 1} \cdot (1 - p_+)^{\alpha_- - 1}$ with conj. likelihood
⇒ MAP: $\hat{\theta} = \frac{\alpha_+ + n_+ - 1}{\alpha_+ + n_+ + \alpha_- + n_- - 2}$: α_+, α_- = pseudo-counts
MAP: $\arg\min \sum$ loss + reg. loss $= -\ln P(y_i|x_i, w)$, reg $= -\ln P(w)$

Bayesian decision theory: $a^* = \arg\min_{a \in A} \mathbb{E}[C(y, a)|X] = \sum_y \ell(y, a) P(y|x)$
log. regression: $P(y|x) = \sigma(y w^Tx)$, $A = \{-1, +1\}$, $C(y, a) = \begin{cases} 1 & y \neq a \\ 0 & y = a \end{cases}$ ⇒ $a^* = \arg\max_y P(y|x) = \text{sign}(w^Tx)$
$A = \{-1, +1, D\}$, $C(y, a) = \begin{cases} 1 & a \neq y \\ 0 & a = y \\ c & a = D \end{cases}$, $c < 0.5$ ⇒ $a^* = \begin{cases} y & \text{if } P(y|x) \geq 1 - c \\ D & \text{else} \end{cases}$ classify only when confident enough
lin. regression: $P(y|x) = \mathcal{N}(w^Tx, \sigma^2)$, $A = \mathbb{R}$, $C(y, a) = (y - a)^2 \Rightarrow \frac{\partial}{\partial a}\mathbb{E}_y[\cdot] = \mathbb{E}_y[\frac{\partial}{\partial a}\cdot] = 0 \Rightarrow a^* = \mathbb{E}[y|x] = w^Tx$
Asymmetric cost: $C(y, a) = c_1\max(y - a, 0) + c_2\max(a - y, 0)$, $\mathbb{E}[C(y, a)|X] = \int C(y, a)\mathcal{N}(\cdot) \, dy$ shifts line/dec. boundary
$\frac{\partial}{\partial a}\mathbb{E}[\cdot] = c_1\int \mathcal{N}(w^Tx, \sigma^2) \, dy + c_2\int \mathcal{N}(w^Tx, \sigma^2) \, dy = c_1(1 - \Phi(a; w^Tx, \sigma^2)) + c_2\Phi(a; w^Tx, \sigma^2) = 0$
$\Phi(a; w^Tx, \sigma^2) = \Phi\left(\frac{a - w^Tx}{\sigma}\right)$, $\Phi(b)$ = CDF of $\mathcal{N}(0, 1)$ = error fct. $a^* = w^Tx + \sigma\Phi^{-1}\left(\frac{c_1}{c_1 + c_2}\right)$

Confidence Bounds: Assumption: $(x_i, y_i) \overset{iid}{\sim} P$
test-error (F) $= \mathbb{E}[\text{error}(F; x, y)] = TE(F) = \varepsilon$, Use validation set X with k samples:
$\frac{\overbrace{\#\text{mistakes}(F; X)}^{m(F, X)}}{|X|} \xrightarrow{k \to \infty} \varepsilon$
$\Pr(m(F, X) = \ell) = \binom{k}{\ell}(1 - \varepsilon)^{k-\ell}\varepsilon^\ell$, $\ell \leq k$
$\Pr(m(F, X) \leq \ell) = \sum_{r=0}^{\ell}\binom{k}{r}(1 - \varepsilon)^{k-r}\varepsilon^r = \delta(\varepsilon, \ell, k)$
Hoeffding's ineq. $\delta(\varepsilon, \ell, k) \leq \exp\left(-2\frac{(k\varepsilon - \ell)^2}{k}\right) \Rightarrow \varepsilon \leq \frac{\ell}{k} + \sqrt{\frac{-\ln\delta}{2k}}$ and by logic $\varepsilon \geq \frac{\ell}{k}$, $k \to \infty \Rightarrow \varepsilon = \frac{\ell}{k}$
Define confidence bound $1 - \delta_0$
$\Pr(m(F, X) \leq \ell \wedge \varepsilon \geq \varepsilon(\delta_0, \ell, k)) = \Pr(m(F, X) \leq \ell \,|\, \varepsilon \in \varepsilon(\delta_0, \ell, k)) = \delta \leq \delta_0$
Check often on same validation set ⇒ overfit to valid. set ⇒ confidence bounds not realistic anymore ⇒ use multiple val. sets ($\binom{n}{k}$ possible) or k-fold CV and average
If we have s classifiers F_i, each with confidence bound $1 - \delta_0$ and choose best ⇒ all confidence bounds need to be fulfilled simultaneously. ⇒ Union bound:
$\Pr(\bigcup A_i) \leq \sum \Pr(A_i) \Rightarrow \delta \leq s\delta_0 \Rightarrow \frac{\delta}{s} \leq \delta_0$ ⇒ either keep above ε but confidence bound drops from $1 - \delta_0$ to $1 - s\delta_0$, or use $\varepsilon \leq \frac{\ell}{k} + \sqrt{\frac{-\ln\delta_0 + \ln s}{2k}}$ with bound $1 - \delta_0$
If $\ell = 0$: $\sum_{r=0}^{0}\binom{k}{r}(1 - \varepsilon)^{k-r}\varepsilon^r = (1 - \varepsilon)^k \leq \frac{\delta}{s}$

Feature selection: greedy forward selection: Start with 0 features $S = \{\}$, Find best feature to add: $s_i = \arg\min L(S \cup \{j\})$, $L(\cdot)$ is CV error, add s_j to S and repeat until $L(\cdot)$ increases with additional feature or desired #features reached., Greedy backward selection: start with all features and remove at each step one: better, bcs can capture effects of two joint-features, but comp. more expensive ⇒ slow. Greedy methods can be used for any method, but high comp. cost bcs need to retrain model always, also: found set can be suboptimal. Other way: make W sparse ⇒ use only few features ⇒ $\lambda\|w\|_0$ regularizer or set constraint $\|w\|_0 \leq k$, but $\|w\|_0$ is np-hard ⇒ use $\|w\|_1$ ⇒ convex: Pro: faster, Con: only usable for linear models, and cannot use kernels (representer th.)
Why feature selection?: - interpretability, better generalization, needs less ressources (faster)

Supervised Learning: Linear Regression: $f(x) = w^Tx + b$, $R(w) = \frac{1}{n}\sum \ell(y_i, h(x_i)) = \frac{1}{n}\sum\frac{1}{2}(y_i - w^Tx_i)^2$
$\frac{\partial R}{\partial b} = 0 \Rightarrow b^* = \langle w, \bar{x} \rangle \Rightarrow f(x) = w^Tx$ where x is centered. Nonlinear basis fct: $f(x) = \sum_j w_j \phi_j(x)$
1D polynomial degree m: $f(x) = \sum_{i=0}^m w_i x^i$, d dimensional polynom degree m: dimension grows from d to $O = \binom{d+m}{m} = O(d^m)$
Least square regression: convex, X: n×d = design matrix, $X^TX = \sum x_i x_i^T$
$w^* = \arg\min_w \frac{1}{n}\sum_{i=1}^n \frac{1}{2}(y_i - w^Tx_i)^2 = \frac{1}{2}\|Xw - y\|_2^2 = R(w)$ = squared loss, closed form sol: $w^* = (X^TX)^{-1}X^Ty$
$O(nd^2 + d^3)$, normal equation = $X^TXw = X^Ty$, gradient descent: $O(\text{iter} \cdot nd)$, other errors usable: absolute error: $|y - w^Tx|$: more robust against outliers, or p^{th} power: $|y - w^Tx|^p$: convex for $p \geq 1$
$p = \infty$: penalize violation of tolerance
Ridge regression: convex, $w^* = \arg\min_w R(w) + \frac{\lambda}{2}\|w\|_2^2$
scale of x matters ⇒ normalize data, closed form: $w^* = (X^TX + n\lambda I)^{-1}X^Ty$, for SGD: $\eta_t n\lambda < 1$!
Kern. ridge regression: $\alpha^* = \arg\min_\alpha \|\alpha^TK - y\|_2^2 + \lambda\alpha^TK\alpha \Rightarrow \alpha^* = (K + \lambda I)^{-1}y$, $w^* = \sum_i \alpha_i x_i$
Prediction: $y_j = \sum_{i=1}^n \alpha_i k(x_i, x_j)$, $K = XX^T$: n×n, Parametric models often too rigid and non-param fail to extrapolate ⇒ semi-param: $k = c_1 k(\text{RBF}) + c_2 k(\text{linear})$
Lasso: convex: $w^* = \arg\min_w R(w) + \lambda\|w\|_1$

Linear classifier: $y = \text{sign}(w^Tx)$, if $x^Tw = 0$: $y = 1$, nonlinear: $y = \text{sign}(w^T\Phi(x))$ very efficient! $w^* = \arg\min_w \frac{1}{n}\sum_{i=1}^n [y_i \neq \text{sign}(w^Tx_i)]$: 0/1 loss $\to y w^T x$
0/1 loss not convex, nor differentiable: np-hard. Uncertainty sampling: pick most uncertain data, estimate its label ⇒ gets classifier fast.

Perceptron: $w^* = \arg\min_w \frac{1}{n}\sum_{i=1}^n \underbrace{\max(0, -y_i w^Tx_i)}_{\text{Perceptron loss}}$: convex, Algorithm: $y \in \{-1, 1\}$: Init $w = 0$, loop over all (x_i, y_i): $w = \begin{cases} w & \text{if } y_i w^T x_i \geq 0 \\ w + y_i x_i & \text{else} \end{cases}$ until convergence, $\hat{=}$ SGD with $\eta = 1$
Convergence: necessary cond. data linearly seperable, sufficient cond: seperation margin $\gamma = \max_{\|w\|_2 = 1} (\min_i (y_i w^Tx_i)) > 0$, γ is a property of the data!
Novikoff's theorem: $R = \max_i \|x_i\|$ and data assumed lin. separable ($\gamma > 0$) ⇒ perception performs no more than $\left(\frac{R}{\gamma}\right)^2$ updates/mistakes until conv. If not lin. seperable: converges still if $\eta_t = \frac{1}{t}$ (maybe SGD conditions on η) $w \in \text{span}(X) = $ lin. comb. of all x_i, Perception finds any hyperplane, don't know which.

Kern. Perceptron: $k(x_i, x_j)$ init $\alpha_{1:n} = 0$
$\alpha^* = \arg\min_\alpha \left(\sum_{i=1}^n \max\left(0, -y_i \left(\sum_{j=1}^n \alpha_j y_j x_j\right)^T x_i\right)\right)$, SGD: $(x_i, y_i) \sim \text{Uni}(1, n)$
$\alpha_j^{t+1} = \alpha_j^t \; \forall j \neq i$, $\alpha_i^{t+1} = \alpha_i^t + \eta_t$ if $y_i \left[\sum_j \alpha_j^t y_j x_j\right]^T x_i < 0$, else stays
$w^* = \sum_{j=1}^n \alpha_j y_j x_j$, prediction for new: $y_i = \text{sign}\left(\sum_j \alpha_j^* y_j k(x_i, x_j)\right)$
If $k = $ RBF: $k(x, x') = [0, 1]$: kern. Perceptron and kern. SVM $\hat{=}$ kNN with optimized weights = soft-kNN, h param of RBF defines # neighbors.

Support Vector Machines (SVM): convex
$w^* = \arg\min_w \frac{1}{n}\sum_{i=1}^n \max(0, 1 - y_i w^T x_i) + \frac{\lambda}{2}\|w\|_2^2$: Hinge loss and soft
Derivation: SVM: $\max \gamma$ s.t. $y_i w^T x_i \geq \gamma \; \forall i$ and $\|w\| = 1$, else we can rescale w, γ. Define $w' = \frac{w}{\gamma} \Rightarrow \gamma = \frac{\|w\|}{\|w'\|} = \frac{1}{\|w'\|}$ ⇒ $\max \gamma \hat{=} \min \|w'\| \Rightarrow \min \|w'\|^2$ s.t. $y_i w'^T x_i \geq 1$
convex, hard SVM, bcs constraint! of 0/1 loss ⇒ $\gamma^* = \min(y_i w^T x_i)$, Hinge loss = upper bound

Kernelized SVM, same as kern. Perceptron but with $1 - y_i \cdots$ in cost instead of $-y_i$ and regularized: $\min_\alpha \left(\sum \max(0, 1 - y_i (\sum \alpha_j y_j k(x_i, x_j))\right) + \lambda \alpha^T K \alpha$

$\ell 1$-SVM: normal $+ \lambda \|w\|_1$ convex

k-Nearest Neighbor: $y_i = \text{sign} \sum_{j=1}^n z_j y_j$ where $z_j = 1$ if x_j among k nearest of x_i, 0 else: no training, but inefficient bcs depends on all data ⇒ need to store all data, non-parametric, k via CV

Logistic Regression: probabilistic model for classification: get class prob (SVM and Perceptron sparse bcs don't need prob ⇒ SVM doesn't know anything about points outside margin) Bernoulli noise assumed: $P(Y = y|x) = \frac{1}{1 + \exp(-y w^T x)}$
$y \in \{-1, +1\}$, convex: MLE: $w^* = \arg\min_w \frac{1}{n}\sum_{i=1}^n \underbrace{\log(1 + \exp(-y_i w^T x_i))}_{\text{logistic loss}}$
prediction: $y = \arg\max P(y|x, w^*) = \text{sign}(w^{*T}x)$, SGD: $\nabla_w = -y_i x_i \cdot P(Y = -y_i | x_i, w)$
Regularization: Gaussian ($\ell 2$) or Laplace prior ($\ell 1$) ($= +\lambda\|w\|_2^2$ or $+\lambda\|w\|_1$)
multiclass: soft-max $P(Y = i|X, w_{1:c}) = \frac{\exp(w_i^T x)}{\sum_{j=1}^c \exp(w_j^T x)} = p_i$: $p_i \geq 0 \; \forall i$, $\sum_i p_i = 1$
one weight vector per class
Replace all $w_i = w_i + \Delta \Rightarrow P(\cdot)$ stays same ⇒ w_i not unique ⇒ set $w_c = 0$: unique

Generative vs Discriminative: Discr: model $P(y|x)$ directly: more robust bcs no assumptions on features. Gen: model $P(x, y) \Rightarrow P(y|x)$: usually model 1. $P(y)$ and 2. $P(x|y)$ = needs to specify $P(x|y)$ distr. ⇒ assumptions, but can detect outliers: $P(x) = \sum_y P(x, y) \leq \tau \Rightarrow$ outlier, τ chosen to max F_1 score, if outlier examples available, otherwise hard to choose. all data
For both: $y = \arg\max_{y'} P(y'|x) \overset{\text{Bayes}}{=} \arg\max_{y'} P(y') P(X|y')$

y categorical: 2 classes: $P(Y|p) = \prod_{i=1}^n p^{\overbrace{\#[y_i = 1]}^{n_+}} (1-p)^{\overbrace{\#[y_i \neq 1]}^{n_-}}$, $p = P(Y = 1)$
MLE: $p = \frac{n_+}{n}$, in general $P(Y = j) = \frac{\#y = j}{n}$

Naive Bayes Classifier: $p(X|Y) = \prod_{j=1}^d p(x_j|y)$ one data point
$p(x_j|y)$ can be different for every j: less parameter ⇒ less overfitting, but assumes features cond. ind. given class ⇒ very strong limitation and often wrong

Gaussian Naive Bayes Classifier (GNBC): $P(x_i|y) = \mathcal{N}(x_i; \mu_{y,i}, \sigma_{y,i}^2)$
x_i = feature of x, $\mu_{y,i}$ and $\sigma_{y,i}$: class and feature specific. Naive: $\Sigma_y = $ diag
MLE: $\hat{\mu}_{y,i} = \frac{1}{n_y}\sum_{j: y_j = y}^n x_{ji}$, $\hat{\sigma}_{y,i}^2 = \frac{1}{n_y}\sum_{j: y_j = y}^n (x_{ji} - \hat{\mu}_{y,i})^2$, 2 classes:
$y = \arg\max_{y'} P(y'|x) = \text{sign}\left(\ln\left(\frac{P(Y = 1|x)}{P(Y = -1|x)}\right)\right) = \text{sign}(f(x))$, $f(x)$ = discriminant fct.
GNBC with $P(Y = 1) = 0.5$, 2 classes and class ind. var σ_i^2: $f(x) = w^Tx + w_0$ ⇒ linear discriminant, $P(Y = 1|x) = \sigma(f(x))$ ⇒ if assumptions correct, GNBC = log. regr.
But copying one feature d times: $f_d(x) = d \cdot f(x) \Rightarrow \sigma(d \cdot f(x)) \Rightarrow$ over confident bcs assumption violated!

General Gaussian Bayes classifier (GBC): $P(x|y) = \overset{\text{multivariate}}{\mathcal{N}}(x; \mu_y, \Sigma_y)$: captures correlation between features ⇒ more parameters and overfitting! MLE:
$\hat{\mu}_y = \frac{1}{n_y}\sum_{j: y_j = y}^n x_j$, $\hat{\Sigma}_y = \frac{1}{n_y}\sum_{j: y_j = y} (x_j - \hat{\mu}_y)(x_j - \mu_y)^T$, 2 classes, general ⇒ Quadratic Discriminant Analysis
QDA: nonlinear decision boundaries $y = \text{sign}(f(x))$, 2 classes, $p(Y = 1) = 0.5$, $\Sigma_+ = \Sigma_-$ ⇒ $f(x) = w^Tx + x_0$ ⇒ Fishers LDA classifier $y = \text{sign}(f(x))$, if assumptions correct ⇒ LDA = log. regr., LDA: 1D projection with max ration of between class to within class variances, PCA: max variance of 1D projection

Categorical Naive Bayes classifier: x discrete too: $P(x_i = c|Y = y) = \theta_{c|y}^i$, MLE: $\hat{\theta}_{c|y}^i = \frac{\#[x_i = c, Y = y]}{n_y}$, Categorical general bayes cl.: #param ~ $O(\exp(d))$: intractable and overfitting: no

Multiclass classifier: generative probabilistic models are naturally multiclass, otherwise:
One vs all: train c binary classifiers $f_i(x) = \frac{w_i^T x}{\|w_i\|_2}$ = confidence, $y = \arg\max_c f_i(x)$ ⇒
Pro: faster than one vs. one, Con: always imbalanced, need confidence score, (one class may be not linear seperable from rest) soft SVM etc. Normalized, bcs $w_i' = 100 w_i$ ⇒ same hyperplane but higher confidence $f_i(x)$. | One vs One: $\frac{c(c-1)}{2}$ binary classifier, apply voting scheme: one with highest votes wins: Pro: no confidence needed, slower, bcs more classifiers
One vs all, One vs One often hard to beat, Use coding theory: error correcting code, Hamming distance, to do only $O(\log c)$ classifiers., Multiclass SVM:
$\min_{w_1, \ldots, w_c} \sum_{i=1}^n \max(1 - \max_{j \neq y_i} [w_{y_i}^T x - w_j^T x], 0) + \lambda \sum_{j=1}^c \|w_j\|_2^2$, where $w_j^T x = f_j(x)$

Neural Networks: linear unit: $\varphi(z) = z$, sigmoid/logistic: $\varphi(z) = \sigma(z) = \frac{1}{1 + \exp(-z)}$, tanh: $\varphi(z) = 2\sigma(2z) - 1 = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} \in [-1, 1]$, $\sigma'(z) = \sigma(z)(1 - \sigma(z))$, ReLU: $\varphi(z) = \max(0, z) = \frac{z + |z|}{2}$, Absolut value: $\varphi(z) = |z|$, units act elementwise, Multilayer Perceptron (MLP) with 1 hidden layer, output act. fct: linear (often for regression): $F(x) = \sum_{i=1}^k w_i^{(2)} \varphi\left(\sum_{j=1}^m w_{ij}^{(1)} x_j\right)$, $W^{(1)} = (w_{ij}^{(1)})$, $W^{(2)}$ here only vector, if multiple outputs ⇒ $W^{(2)}$ matrix. General: $F(x) = \varphi^{(L)}(W^{(L)} \varphi^{(L-1)}(\ldots \varphi^{(1)}(W^{(1)} x) \ldots))$, $x^{(0)} = x$, $x^{(\ell)} = \varphi^{(\ell)}(W^{(\ell)} x^{(\ell-1)})$
$W^{(\ell)}$: $k \times m$, m = inputs of ℓ, k = units of ℓ, $x^{(L-1)}$ = learned representation

Universal theorem: - lin. units: only linear, $\varphi^{(\ell)}$ = polynomial degree $\leq d^{(\ell)} \Rightarrow F$ = polynomial of at most degree $\prod_\ell d^{(\ell)}$ - non-polynomial, nonlinear such as σ, ReLU etc: Any bounded $g(x)$: ∃ MLP with one hidden layer, s.t. $|g(x) - F(x)| < \varepsilon \; \forall x, \forall \varepsilon > 0$

Regression: $\varphi^{(L)}(z) = z$, $\ell(y^*, y) = \frac{1}{2}(y^* - y)^2$ or $\frac{1}{2}\|y^* - y\|_2^2$ if y = vector, y^* = true!
Binary classification: $\varphi^{(L)}(z) = z \Rightarrow y = \text{sign}(z)$ or $\varphi^{(L)}(z) = \sigma(z)$, $y^* \in \{-1, 1\}$, $y \in [0, 1]$, cross-entropy loss: $\ell(y^*, y) = -\frac{1 + y^*}{2}\ln y - \frac{1 - y^*}{2}\ln(1 - y)$, Multiclass: $y^* \in \{1, \ldots, s\}$, s outputs, each with act. fct: $F_r(x) = \frac{\exp(\langle w_r^{(L)}, x^{(L-1)} \rangle)}{\sum_s \exp(\langle w_s^{(L)}, x^{(L-1)} \rangle)}$ for class r! = soft-max, $\sum_s F_s(x) = 1$, $F_r(\beta x) \xrightarrow{\beta \to \infty} \begin{cases} 1 & \text{if } r = \arg\max_s \langle w_s^{(L)}, x^{(L-1)} \rangle \\ 0 & \text{else} \end{cases}$
loss $\ell(y^*, F(x)) = -\ln F_{y^*}(x)$

Backpropagation: risk $R(\Theta) = \frac{1}{n}\sum_{i=1}^n \ell(y_i^*, y_i)$ where $y_i = F(x_i; \Theta)$, $\frac{\partial \ell(y_i^*, y_i)}{\partial \Theta} = \frac{\partial \ell(y_i^*, y_i)}{\partial y_i} \cdot \frac{\partial y_i}{\partial \Theta}$, $f(g(x)) = (f \circ g)(x)$, $(f \circ g)'(x) = (f' \circ g)(x) \cdot g'(x)$
Jacobians: $J_F = \begin{pmatrix} \nabla^T F_1 \\ \nabla^T F_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots \\ \vdots & & \end{pmatrix}$: $k \times m$, $\nabla_{x^{(\ell)}} R = \delta^{(\ell)} = J_F^{T(\ell+1)} \cdots \underbrace{J_F^{T(L)} \nabla_y R}_{\delta^{(L)}}$
$\delta^{(\ell)} = J_F^{T(\ell+1)} \delta^{(\ell+1)}$ ~~$D\varphi = \text{diag}(\varphi'(\langle w_1, x \rangle) \ldots$~~
$z = \varphi(Wx)$: $\frac{\partial z_i}{\partial x_j} = \frac{\partial \varphi(\sum_r w_{ir} \cdot x_r)}{\partial x_j} = \varphi'(\langle w_i, x \rangle) \cdot w_{ij} \Rightarrow J = D_\varphi W$ where $D\varphi = \text{diag}(\varphi'(\langle w_1, x \rangle), \ldots, \varphi'(\langle w_k, x \rangle))$, if $\varphi(z) = z \Rightarrow J = W$
$\frac{\partial R}{\partial w_{ij}^{(\ell)}} = \frac{\partial R}{\partial x_i^{(\ell)}} \cdot \frac{\partial x_i^{(\ell)}}{\partial w_{ij}^{(\ell)}} = \frac{\partial R}{\partial x_i^{(\ell)}} \cdot \varphi'(\langle w_i^{(\ell)}, x^{(\ell-1)} \rangle) \cdot x_j^{(\ell-1)}$

Instead of SGD, also possible Mini-batch SGD: $R(\Theta; I) = \frac{1}{|I|}\sum_{i=1}^{|I|} \ell(y_i^*, F(x_i; \Theta))$
Initialization: weights = random small numbers. Regularization: $-\frac{\lambda}{2}\|\Theta\|^2$ – weight norm constraint: $\|w_i^{(\ell)}\|^2 < \varrho \; \forall i, \ell$ ⇒ projected gradient descent – Early stopping: use validation error and stop when it increases – Dropout: randomly drop out unit with some prob p at each step (only for forward pass, backward pass still with all units)

Sequential (time series) models: iid assumption wrong, 1st order Markov chain: $P(Y_1 = y) = p_y$
$P(Y_t = y|Y_{t-1} = y') = \theta_{y|y'} = P(Y_{t+1} = y|Y_t = y') \hat{=}$ stationary. 1st order $\hat{=}$ memoryless: state today defines state tomorrow, k^{th} order MC: previous k states: can transform to 1st order, $O(c^{k+1}) \Rightarrow k$ small, $P(Y_{t+2} = y|Y_t = y_t) = \sum_{y'} \theta_{y|y'} \cdot \theta_{y'|y_t}$ or sample, Prob. vector $p^{(t)} = [p_1^{(t)}, \ldots, p_c^{(t)}]$
Transition matrix T: $T_{ij} = \theta_{j|i} \Rightarrow p^{(t+m)} = p^{(t)} \cdot T^m$, transition diagram/finit stochastic automaton
Stationary distr. (in general does not exist, sufficient cond: $T_{ij} > 0$): $\lim_{t \to \infty} p^{(t)}$: $p = pT$

Data: m sequences of different length n_i: $P(D|\Theta) = \prod_{i=1}^m \left(P(y_1^i) \prod_{t=2}^{n_i} \theta_{y_t^i | y_{t-1}^i}\right)$, c = # states: $y_t \in \{1, \ldots, c\}$
⇒ $\log P(D|\Theta) = \sum_{j=1}^c [Y_1 = j] \ln P(Y_1 = j) + \sum_{j=1}^c \sum_{j'=1}^c [Y_{t+1} = j, Y_t = j'] \ln \theta_{j|j'}$ ⇒ MLE: $\hat{P}(Y_1 = y) = \frac{\#[Y_1 = y]}{m}$
$\hat{\theta}_{y|y'} = \frac{\#[Y_t = y, Y_{t-1} = y']}{\#[Y_{t-1} = y']}$ use pseudo counts (Beta prior) for regularization/smoothing.
Supervised learning!: Y_t known = label for $P(Y_t|Y_{t-1:t-1})$ etc = autoregressiv

y continuous, k^{th} order Gaussian linear time series: $P(Y_{t+1} = y|Y_{t-k+1:t}) = \mathcal{N}(y; \underbrace{\sum_{i=1}^k w_i \cdot Y_{t-k+i}}_{f(\cdot)}, \sigma^2)$
= same as assuming Gaussian noise: $Y_t = f(Y_{t-k+1:t}; \Theta) + \varepsilon_t$ where $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$
Use SGD. If $f(\cdot)$ nonlinear: nonlinear Gaussian..., $\hat{P}(Y_{k+1:T}|Y_{1:k}, \Theta) = \prod_{t=k}^{T-1} \hat{P}(Y_{t+1}|Y_t, \Theta)$?
Bernoulli-time-series: $P(Y_{t+1} = y|Y_{t-k+1:t}) = \frac{1}{1 + \exp(-f(Y_{t-k+1:t}; \Theta))}$

Predict multiple timesteps: $P(Y_{t+2}|Y_t) = \int P(Y_{t+2}|Y_{t+1}) P(Y_{t+1}|Y_t) \, dY_{t+1}$ when y cont. ⇒ intractable ⇒ approximate via Forward sampling algorithm: Input: model $P(Y_{t+1}|Y_{t-k+1:t})$ and data $y_{1:T}$. Predict horizon τ; N = # samples: Algo: For $i = 1:N$: [Set $y_{1:T}^i = y_{1:T}$, For $t = T$: $T + \tau - 1$ do: Sample $y_{t+1}^i \sim P(Y_{t+1}|y_{t-k+1}^i, \ldots, y_t^i)$] Then predict mean (y cont.) $\mathbb{E}[Y_{T+\tau}] \approx \frac{1}{N}\sum_{i=1}^N y_{T+\tau}^i$, How to choose N? Hoeffding's inequality: f bounded $[0, C]$
$P(|\mathbb{E}_p[f(X)] - \frac{1}{N}\sum_i f(x_i)| > \varepsilon) \leq 2\exp\left(\frac{2N\varepsilon^2}{C^2}\right)$: choose ε ⇒ gives N.
Report weighted aver.
Temporal CV: k+1 fold: D_1 = first $\frac{T}{k+1}$ etc: For $i = 2 : k+1$ do: train on $D_1, \ldots, D_{i-1}$, test on D_i: k tests

Sequence to sequence: HMM (Hidden...): generative: $x_{1:T}$ observations, $Y_{1:T}$ hidden states

X, Y categorical. Same p_Y and $\Theta_{y|y'}$, additional $P(X_t = x | Y_t = y) = n_{x|y}$, $P(X_{1:T}, Y_{1:T}) = p_{Y_1} \cdot \prod_{t=1}^{T} n_{x_t|y_t} \prod_{t=2}^{T} \Theta_{y_t|y_{t-1}}$ HMM assumption

MLE: $\hat{n}_{x|y} = \frac{\#[X_t = x, Y_t = y]}{\#[Y_t = y]}$ | Most probable state seq. $y^*_{1:T} = \arg\max_{y_{1:T}} P(Y_{1:T} | X_{1:T})$

Inference tasks: Filtering: $P(Y_t | x_{1:t})$, Prediction: $P(Y_{t+\tau} | x_{1:t})$, Smoothing: $P(Y_t | x_{1:T})$ for $1 \le t \le T$

Bayesian filtering: at time t, we have $P(Y_t | x_{1:t-1})$: 1. Conditioning: $P(Y_t = y | x_{1:t}) \overset{\text{Bayes}}{=} \frac{P(Y_t = y | x_{1:t-1}) n_{x_t|y}}{\sum_{y'} P(Y_t = y' | x_{1:t-1}) n_{x_t|y'}}$

2. Prediction: $P(Y_{t+1} = y | x_{1:t}) = \sum_{y'} \overbrace{P(Y_t = y' | x_{1:t})}^{\text{Filtering}} \Theta_{y|y'}$

Matrix/vector notation of filtering: $P(Y_t | x_{1:t})$ as vector $p^{(t)} = [p_1^{(t)}, \dots, p_c^{(t)}]$, Transition matrix T as before, $O^{(t)} = [n_{x_t|1}, \dots, n_{x_t|c}]$ = vector repr. of $P(X_t = x_t | Y_t = y) \Rightarrow p^{(t+1)} \propto (p^{(t)} T) \cdot O^{(t+1)}$ (need to renormalize)

Prediction: Given $P(Y_t | X_{1:t})$ as vector $p^{(t)}$: $P(Y_{t+\tau} | x_{1:t}) = p^{(t+\tau)} = p^t T^\tau$ Markov property

Smoothing: $P(Y_t = y | x_{1:T}) = \frac{1}{Z} P(Y_t = y, x_{1:T}) = \frac{1}{Z} \underbrace{P(Y_t = y, x_{1:t})}_{a_t(y)} \cdot \underbrace{P(x_{t+1:T} | Y_t = y, \cancel{x_{1:t}})}_{b_t(y)}$

Forward-Backward-Algorithm:

Forward pass: Init: $a_1(y) = p_y n_{x_1|y}$

For $t = 2:T$: $a_t(y) = \sum_{y'=1}^{c} a_{t-1}(y') \cdot \Theta_{y|y'} \cdot n_{x_t|y'}$

Backward pass: Init $b_T(y) = 1 \ \forall y$

For $t = T-1:1$: $b_t(y) = \sum_{y'=1}^{c} b_{t+1}(y') \cdot \Theta_{y'|y} \cdot n_{x_t|y'}$

$\Big\} \ O(T \cdot c^2)$

For $t = 1:T$: $P(Y_t = y | x_{1:T}) = \frac{1}{Z} a_t(y) b_t(y)$, where $Z = \sum_{y'} a_t(y') b_t(y')$

Active Learning: expensive labels (need to ask expert): Uncertainty sampling: 1. estimate $\hat{p}(y_i | x_i)$ given current labeled data. 2. pick most uncertain ($p \sim 0.5$) unlabeled data $i = \arg\min |0.5 - \hat{p}(y_i | x_i)|$ 3. expert labels $y_i \Rightarrow$ iid assumpt. violated!

Unsupervised Learning: (Anomaly = doesn't fit to any cluster): dimension reduction = unsupervised regression, clustering = classification.

Dim Reduction:

Principal Component Analysis (PCA): linear $f(x) = Ax$, dimension reduction $d \to k$

$k = 1$: $\max z^T \Sigma z + \lambda(z^T z - 1)$. $k \ge 1$: $z_i \in \mathbb{R}^k$, $x_i \in \mathbb{R}^d$, $W: d \times k$, $X: n \times d$, $\Sigma: d \times d$, W orthog.

$\Sigma = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T = X^T X \frac{1}{n}$, assume centered: $\mu = 0$, reconstruction loss minimization:

$(W^*, z^*) = \arg\min_{W, z} \sum_{i=1}^{n} \|W z_i - x_i\|_2^2$, $\tilde{x}_i = W z_i$ = reconstruction = low dim approx.

variance maximization: $W^* = \arg\max_{W^T W = I} \sum_{i=1}^{n} (W^T x_i)^2 = \arg\max_{W^T W = I} W^T \Sigma W \cdot \underbrace{n}_{\text{irrelevant}}$

Solve via variance max: Use SVD: $\Sigma = \frac{1}{n} V S^T U^T U S V^T = \frac{1}{n} V D V^T$ ($X = U S V^T$)

$\Rightarrow \Sigma = \sum_{i=1}^{d} \lambda_i v_i v_i^T$, $\lambda_1 \ge \lambda_2 \ge \dots \ge \lambda_d$. Then $W^* = (v_1, \dots, v_k)$ = first k columns of V corresponding to top k eigenvalues of Σ = principal components and then $z_i = W x_i$. k chosen visually or as in k-means heuristics

Nonlinear methods: kernel PCA: for kernels we require $w = \sum_{i=1}^{n} \alpha_i x_i$, From normal PCA: λ = eigenvalue of $X^T X \Rightarrow X^T X w = \lambda w$, define $X w = \beta \Rightarrow w = \frac{X^T \beta}{\lambda} = \sum_{i=1}^{n} \frac{\beta_i}{\lambda} x_i$

$\Rightarrow \alpha_i = \frac{\beta_i}{\lambda} \Rightarrow$ requirement fulfilled. Normal PCA $\max_{\|w\|_2 = 1} \sum_i (w^T x_i)$, now let $x \to \phi(x) \Rightarrow w = \sum_{j=1}^{n} \alpha_j \phi(x_j)$ then normal PCA gets $\max_{1 = \alpha^T K \alpha} \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \alpha_j \phi(x_j)^T \phi(x_i)\right)^2$

$= \max_{\alpha^T K \alpha} \sum_{i=1}^{n} (\alpha^T K_i)^2 \Rightarrow \alpha^* = \arg\max_{\alpha^T K \alpha = 1} \alpha^T K^T K \alpha$, $x: \mathbb{R}^d \to \phi(x) \in \mathbb{R}^D \to z \in \mathbb{R}^k$, $D \gg d \ge k$

Solution: $K = \sum_{i=1}^{n} \lambda_i v_i v_i^T$, $\lambda_1 \ge \dots \ge \lambda_n \Rightarrow \alpha^{(i)} = \frac{1}{\sqrt{\lambda_i}} v_i$,

Projection $z_i = \sum_{j=1}^{n} \alpha_j^{(i)} k(x_j, x) = i^{th}$ component of $z \in \mathbb{R}^k$, $\alpha^{(1)}, \dots, \alpha^{(k)} \hat{=}$ princ. comp.

Center the kernel: $E = \frac{1}{n} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} (1 \dots 1)$: $K' = K - KE - EK + EKE$

Neural Network Autoencoders: try to learn the identity fct. d inputs, d output units, one layer with $k < d$ units: $(W^{(2)*}, W^{(1)*}) = \arg\min \sum_{i=1}^{n} \|x_i - W^{(2)} \varphi(W^{(1)} x_i)\|_2^2$

$\Rightarrow$ SGD, backpropagation, Initialization matters and is challenging, $\varphi(z) = z \Rightarrow$ PCA

Locally Linear Embedding (LLE): 1. identify neighbors $N(i)$ of x_i 2. weighted comb. of neighbors: $x_i \approx \sum_{j \in N(i)} w_{ij} x_j$, where $W = \arg\min \sum_{i=1}^{n} \|x_i - \sum_j w_{ij} x_j\|_2^2$

s.t. $\sum_j w_{ij} = 1$ and $w_{ij} = 0$ if $x_j \notin N(i)$ 3. Project to low-dim space:

$z_{1:n} = \arg\min \sum_{i=1}^{n} \|z_i - \sum_{j \in N(i)} w_{ij} z_j\|_2^2$ s.t. $\sum_i z_i = 0$, $\frac{1}{n} \sum_i z_i z_i^T = I_k$. 2. & 3. convex

Multi-dimensional scaling (MDS): Given: dissimilarity matrix D (i.e. city distances): $n \times n$ Find $z_{1:n}$ s.t. $\|z_i - z_j\| \approx D_{ij}$ in 2D (= map)

$z_{1:n} = \arg\min \sum_{ij} w_{ij} (\|z_i - z_j\|_2 - D_{ij})^2$ where $w_{ij} = 1 \hat{=}$ unnormalized or $w_{ij} = \frac{1}{D_{ij}^2}$ non-convex! (rotate map $\hat{=}$ equivalent $\Rightarrow$ local opt)

Clustering: k-means clustering: $\mu = (\mu_1, \dots, \mu_k)$

$L(\mu) = \sum_{i=1}^{n} \min_{j = 1:k} \|x_i - \mu_j\|_2^2$, $\mu^* = \arg\min_\mu L(\mu)$: non convex np-hard! Lloyd's heuristic: Init $\mu^{(0)} = (\mu_1^{(0)}, \dots, \mu_k^{(0)})$

1. $z_i = \arg\min_j \|x_i - \mu_j\|_2^2$ 2. Define $c_{ij} = \begin{cases} 1 & \text{if } z_i = j \\ 0 & \text{else} \end{cases}$

$\mu_j = \arg\min_\mu \sum_{i=1}^{n} \sum_{j=1}^{k} c_{ij} \|x_i - \mu_j\|_2^2$

$\Rightarrow \mu_j = \frac{1}{n_j} \sum_{i: z_i = j} x_i$ Complexity $O(\text{iter} \cdot nkd)$

guaranteed to local opt:

Assignment step: $z_i^* = \text{new}$, $z_i = \text{old}$

$L(\mu, z^*) - L(\mu, z) \le 0$ where $L(\mu, z) = \sum_i \|x_i - \mu_{z_i}\|_2^2$ bcs z_i^* assigns to nearest cluster

Refitting step: $L(\mu, z) = \sum_{j=1}^{k} \left(\sum_{i: z_i = j} \|x_i - \mu_j\|_2^2\right)$: $L(\overset{\text{new}}{\mu^*}, z) - L(\overset{\text{old}}{\mu}, z) \le 0$ bcs μ_j^* minimizes this quantity.

k-means vs PCA: k-means: $W = (\mu_1, \dots, \mu_k)$: $(W, z_{1:n}) = \arg\min_{W, z} \sum_{i=1}^{n} \|W z_i - x_i\|_2^2$

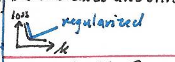
W arbitrary, z_i = unit vector $(0, 0 \dots 0, 1, 0 \dots 0)^T \Rightarrow$ PCA with diff. constraints

kern. k-means: $\|x_i - \mu_j\|_2^2 = x_i^T x_i - 2 x_i^T \mu_j + \mu_j^T \mu_j \hat{=} k(x_i, x_i) + k(\mu_j, \mu_j) - 2k(x_i, \mu_j)$

$\Rightarrow$ non-spherical shapes.

Initialization: multiple random restarts, furthest point heuristic (prone to outliers), k-means++: 1^{st} point random, then sample

$P(\mu_j = x_i) = \frac{1}{Z} \min_{m \in \{1, \dots, j-1\}} \|x_i - \mu_m\|_2^2$, Choosing k: Cannot use validation error bcs decreases also strict with k. $\Rightarrow$ regularization: $L(\mu) + \lambda k$ or heuristic at knee



Mixture Models: Bayes classifier with unknown labels, generative $\Rightarrow$ can detect outliers, generate new point: sample cluster $y_i \sim$ categorical with $P(Y_i = j) = w_j$, then sample $x_i \sim f(x; \Theta_j)$, non convex (exchange component indizes = equivalent solution $\Rightarrow$ multiple local optima) also for GMM: hard to maintain constraint of symm. psd variances, parameters k, ν via CV (works here) or as in k-means. GMM degeneracy: m = # components, if $m = n$, best is: place each μ_j on a data point x_i $\Rightarrow$ overfitting: neg. log-likelihood $= -\infty$ when variances $\to 0 \Rightarrow$ either reset any component that falls on a data point or use Wishart prior to add always smth to the covariance matrices.

Hard EM: Compute hard assignments $y_i^{(t)} = \arg\max_y P(y | x_i, \Theta^{(t-1)}) \overset{\text{Bayes}}{=} \arg\max_y P(y | \Theta^{(t-1)}) P(x_i | y, \Theta^{(t-1)})$

M-Step: now got labels $\Rightarrow$ Bayes classifier: $\Theta^{(t)} = \arg\max_\Theta P(x_{1:n}, y_{1:n} | \Theta)$

Hard-EM bad, bcs: if components overlap, and also: model is uncertain, but output assignments are fixed!

GMM with hard assignments $\hat{=} \gamma_j(x_i) \in \{0, 1\} \hat{=}$ variance $\sigma^2 \to 0$, uniform weights $w_j = \frac{1}{m}$ and identical spherical covariances $\Sigma_j = \sigma^2 I \Rightarrow k$-means.

Soft EM: Introduce latent variable $y_i \in \{1, \dots, m\}$: m components. $P(Y_i = j) = w_j$, $w_j \ge 0$, $\sum_j w_j = 1$

MLE: likelihood for one data point: $P(x | \Theta) = \sum_{j=1}^{m} w_j \cdot f(x; \Theta_j)$, Gaussian: $f(x; \Theta_j) = \mathcal{N}(\mu_j, \Sigma_j)$

Complete likelihood $P(X | \Theta) = \prod_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_i = j | \Theta) = \prod_i \sum_j P(y_i = j | \Theta) \cdot P(x_i | y_i = j, \Theta) = \prod_i \sum_j w_j f(x_i; \Theta_j)$

log-likelihood: $\ell(\Theta) = \sum_{i=1}^{n} \log \sum_{j=1}^{m} w_j f(x_i; \Theta_j)$

E-Step: Compute posterior $\hat{=}$ expected labels $\hat{=}$ responsibilities:

$\gamma_j^{(t)}(x_i) = P(y_i = j | x_i, \Theta^{(t-1)}) \overset{\text{Bayes}}{=} \frac{P(x_i | y_i = j, \Theta) P(y_i = j | \Theta)}{P(x_i | \Theta)} = \frac{w_j f(x_i; \Theta_j)}{\sum_l w_l f(x_i; \Theta_l)}$, $\sum_j \gamma_j(x_i) = 1$, $\sum_i \sum_j \gamma_j(x_i) = n$

Other method: Define $Q(y_{1:n}) = P(y_{1:n} | x_{1:n}, \Theta) \Rightarrow$ Expected log-likelihood $\overbrace{\mathbb{E}_Q[\ell(\Theta)]}^{Q(\Theta, \Theta^{(t-1)})}$ = normal $\ell(\Theta)$ but y_i exchanged to $\mathbb{E}_Q[y_i]$

M-Step: $\Theta^{(t)} = \Theta^* = \arg\max_\Theta \sum_{i=1}^{n} \sum_{j=1}^{m} \gamma_j^{(t)}(x_i) \ln \underbrace{P(x_i, y_i = j | \Theta)}_{w_j f(x_i; \Theta_j)} + \lambda\left(\sum_{j=1}^{m} w_j - 1\right) = \arg\max_\Theta \mathbb{E}_Q[\ell(\Theta)] + \lambda\left(\sum_j w_j - 1\right)$

MLE: $\forall$ Mixture Models: $w_j^{(t)} = \frac{1}{n} \sum_{i=1}^{n} \gamma_j^{(t)}(x_i)$, GMM: $\mu_j^{(t)} = \frac{\sum_{i=1}^{n} \gamma_j^{(t)}(x_i) \cdot x_i}{\sum_{i=1}^{n} \gamma_j^{(t)}(x_i)}$, $\Sigma_j^{(t)} = \frac{\sum_{i=1}^{n} \gamma_j^{(t)}(x_i) (x_i - \mu_j^{(t)})(x_i - \mu_j^{(t)})^T}{\sum_{i=1}^{n} \gamma_j^{(t)}(x_i)} + \nu^2 I$ MAP with (Wishart prior) ν via CV

Naive Bayes classifier with k binary features: $\Theta_{l|j} = P(X_l = 1 | Y_i = j)$

$f(x_i; \Theta_j) = \prod_{l=1}^{k} \Theta_{l|j}^{x_{il}} (1 - \Theta_{l|j})^{1 - x_{il}}$, E-Step normal, M-Step: $\Theta_{l|j} = \frac{\sum_{i=1}^{n} [x_{il} = 1] \gamma_j(x_i)}{\sum_{i=1}^{n} \gamma_j(x_i)}$, w_j normal

Labelled and unlabelled data: $y_i = 0$ = unlabelled: log-likelihood:

$\ell(\Theta) = \sum_{i: y_i = 0} \log \sum_j w_j f(x_i; \Theta_j) + \sum_{i: y_i \neq 0} \log w_{y_i} f(x_i; \Theta_{y_i})$, E-Step: for unlabelled normal, for labelled: $\gamma_j(x_i) = [y_i = j] \in \{0, 1\}$, M-Step normal!

As k-means, guaranteed to conv. to local opt, Initialization: $\gamma_j^{(0)}(x_i) = \frac{1}{m}$, $\mu_j^{(0)}$ = random or as in k-means++, $\Sigma_j^{(0)}$: do E-Step with $\gamma_j^{(0)}(x_i)$ and $\mu_j^{(0)}$!

Gaussian math: $f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$, $f(y|x) = \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{(y-x)^2}{2\sigma^2}\right)$

$\Rightarrow f(x, y) \propto \exp\overbrace{\left(-\frac{1}{2} \frac{(x-a)^2}{b^2}\right)}^{A} \cdot \exp\overbrace{\left(-\frac{1}{2} \frac{(y-c)^2}{d^2}\right)}^{B}$ $a = a(y)$, $b, c, d = \text{const}$

$f(y) = \int f(x, y) dx \propto \underbrace{\int \exp(A) dx}_{\sqrt{2\pi} b = \text{const}} \exp(B) \propto \exp(B) \overset{\Rightarrow \text{rearrange}}{\Big\}} f(y) = c \exp(B)$ to Gaussian

Multivariate: $f(x) = \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{\det \Sigma}} \exp\left(-\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu)\right)$, $Y = X_1 + X_2$ given $Z = X_1 - X_2 = 0$

$\Rightarrow Y = 2X_1 = 2X_2$

$\Rightarrow X_1 = X_2 = \frac{Y}{2}$ $f_{Y|Z}(y|0) = \frac{f_{Y,Z}(y, 0)}{f_Z(0)} \propto f_{Y,Z}(y, 0) \propto f_X\begin{pmatrix} y/2 \\ y/2 \end{pmatrix}$ same Σ $\propto$ vary

$\propto \exp\left(-\frac{1}{2} \left(\begin{pmatrix} y/2 \\ y/2 \end{pmatrix} - \mu\right)^T \Sigma^{-1} \left(\begin{pmatrix} y/2 \\ y/2 \end{pmatrix} - \mu\right)\right)$: compute and read off mean and

Or: Define $R = \begin{pmatrix} Y \\ Z \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} X = AX$: $\mathbb{E}[R] = A\mathbb{E}[X] = A\mu$, $\Sigma_R = \mathbb{E}[(R - \mathbb{E}[R])(R - \mathbb{E}[R])^T] \hat{=} A \Sigma A^T$

$f_{Y,Z}(y, 0) \propto \exp(\cdot)$ part of $f_R\begin{pmatrix} y \\ 0 \end{pmatrix}$