Introduction to Machine Learning SS20

True risk and estimated error

True risk: $R(w) = \int P(x,y)(y - w^T x)^2 \partial x \partial y = \mathbb{E}_{x,y}[(y - w^T x)^2]$, Est. error: $\hat{R}_D(w) = \frac{1}{|D|} \sum_{(x,y)\in D}(y - w^T x)^2$, **Training error (empirical risk) systematically underestimates true risk**, thus we need a separate test set.

Standardization

Centered data, unit variance: $\tilde{x}_i = \frac{x_i - \hat{\mu}}{\hat{\sigma}}$, $\hat{\mu} = \frac{1}{n}\sum_{i=1}^n x_i$, $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^n (x_i - \hat{\mu})^2$, **use it before regularization**

Cross-Validation

Data should be iid. **Select** k: Small: overfitting to test set, little data for training, underfitting to train set, Large: Better performance (LOOCV: $k = n$), higher computational complexity.

Gradient Descent (GD)

Pick arbitrary $w_0 \in \mathbb{R}^d$, 2. $w_{t+1} = w_t - \eta_t \nabla \hat{R}(w_t)$, mild assumptions, step size sufficiently small: stationary point ($\nabla = 0$); converges linearly for constant stepsize and squared loss. *Convex* problems $\rightarrow$ finds optimum! **Compare GD vs Closed Form**: Comp. Complex., Problem may not offer closed form sol.

Stochastic Gradient Descent (SGD)

1. Pick arbitrary $w_0 \in \mathbb{R}^d$
2. $w_{t+1} = w_t - \eta_t \nabla_w l(w_t; x', y')$, with u.a.r. data point $(x', y') \in D$, if data lin. seperable, finds lin. seperator! SGD $O(1)$ vs. $O(n)$ for GD. *Mini-batches* exploit parallelism, reduce variance.

Feature Selection

Greedy: **+**: Any pred. method, convex **-**: Slower (Train many models). // **Greedy Forward**: Add best elements according to loss and stop once error increases. (faster) **Greedy Backward**: Find best element to remove according to loss until error doesn't decrease anymore. (handles dependant features) **L1**: **+** Faster (training and feature selection joint), **-** Only works with linear models.

Regression

Solve $w^* = \text{argmin}_w \hat{R}(w) + \lambda C(w)$

Linear Regression

$\hat{R}(w) = \sum_{i=1}^n (y_i - w^T x_i)^2 = ||Xw - y||_2^2$
$\nabla_w \hat{R}(w) = -2\sum_{i=1}^n (y_i - w^T x_i) \cdot x_i$
$w^* = (X^T X)^{-1} X^T y$,
$\mathbf{E}[w^*] = w$, $\mathbf{V}[w^*] = (X^T X)^{-1}\sigma^2$

Ridge regression

$\hat{R}(w) = \sum_{i=1}^n (y_i - w^T x_i)^2 + \lambda ||w||_2^2$
$\nabla_w \hat{R}(w) = -2\sum_{i=1}^n (y_i - w^T x_i) \cdot x_i + 2\lambda w$
$w^* = (X^T X + \lambda I)^{-1} X^T y$
$\mathbf{E}[w^*] = (X^T X + \lambda I)^{-1}(X^T X) w$
$\mathbf{V}[w^*] = \sigma^2 (X^T X + \lambda I)^{-1}(X^T X)[(X^T X + \lambda I)^{-1}]^\top$

L1-regularized regression (Lasso)

$\hat{R}(w) = \sum_{i=1}^n (y_i - w^T x_i)^2 + \lambda ||w||_1$, Lasso performs variable selection as coefficients go to 0 if $\lambda \rightarrow \infty$

Classification

Solve $w^* = \text{argmin}_w \, l(w; x_i, y_i)$; loss function l

0/1 loss

$l_{0/1}(w; y_i, x_i) = 1$ if $y_i \neq \text{sign}(w^T x_i)$ else 0

Perceptron algorithm

Use $l_P(w; y_i, x_i) = \max(0, -y_i w^T x_i)$ and SGD

$\nabla_w l_P(w; y_i, x_i) = \begin{cases} 0 & \text{if } y_i w^T x_i \geq 0 \\ -y_i x_i & \text{otherwise} \end{cases}$

Convex surrogate $\rightarrow$ if data lin. separable $\Leftrightarrow$ obtains a lin. separator (not necessarily optimal)

Support Vector Machine (SVM)

Hinge loss: $l_H(w; x_i, y_i) = \max(0, 1 - y_i w^T x_i)$

$\nabla_w l_H(w; y, x) = \begin{cases} 0 & \text{if } y_i w^T x_i \geq 1 \\ -y_i x_i & \text{otherwise} \end{cases}$

$w^* = \text{argmin}_w \, l_H(w; x_i, y_i) + \lambda ||w||_2^2$

Kernels

Choosing Kernels Domain knowledge, Brute force or heuristic search, CV **Choosing Param.** CV **Trick** Reformulate such that inner product appear, replace **+** Explicit control, Incorporation of prior knowledge by kernel eng. **-** Avoid large d but solution is in $\mathbb{R}$, Hard to design **Proof Validity** Show Symmetry and p.d., Find an explicit feature map, Derive the kernel from others **Proof Inval.** Disp. Symmetry or p.d. by Ex.

Properties of kernel

$k: X \times X \rightarrow \mathbb{R}$, k must be some inner product (efficientm implicit, symmetric, positive-definite, linear) for some space V. i.e. $k(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_V \overset{Eucl.}{=} \varphi(\mathbf{x})^T \varphi(\mathbf{x}')$ and $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$
Kernel matrix: Positive semi-definite!

Important kernels

Linear: $k(x,y) = x^T y$, Poly: $k(x,y) = (x^T y + 1)^d$
Gaussian: $k(x,y) = exp(-||x-y||_2^2/(2h^2))$
Laplacian: $k(x,y) = exp(-||x-y||_1/h)$

Composition rules

Valid kernels k_1, k_2, also valid kernels: $k_1 + k_2$; $k_1 \cdot k_2$; $c \cdot k_1$, $c > 0$; $f(k_1)$ if f polynomial with pos. coeffs. or exponential

Reformulating the perceptron

Ansatz: $w^* \in \text{span}(X) \Rightarrow w = \sum_{j=1}^n \alpha_j y_j x_j$
$\alpha^* = \underset{\alpha \in \mathbb{R}^n}{\text{argmin}} \sum_{i=1}^n \max(0, -\sum_{j=1}^n \alpha_j y_i y_j x_i^T x_j)$

Kernelized perceptron and SVM

Use $\alpha^T k_i$ instead of $w^T x_i$,
use $\alpha^T D_y K D_y \alpha$ instead of $||w||_2^2$
$k_i = [y_1 k(x_i, x_1), \ldots, y_n k(x_i, x_n)]$, $D_y = \text{diag}(y)$
Prediction: $\hat{y} = \text{sign}(\sum_{i=1}^n \alpha_i y_i k(x_i, \hat{x}))$ SGD update: $\alpha_{t+1} = \alpha_t$, if mispredicted: $\alpha_{t+1,i} = \alpha_{t,i} + \eta_t$ (c.f. updating weights towards mispredicted point)

Kernelized linear regression (KLR)

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

k Nearest Neighbours (*k*NN)

$y = \text{sign}\left(\sum_{i=1}^n y_i [x_i \text{ among } k\text{NN of } x]\right)$ – No weights $\rightarrow$ no training! Depends on all data. Can be kernelized!

Imbalance

Cost-Sensitive Classification

Scale loss by cost: $l_{CS}(w; x, y) = c_{\pm} l(w; x, y)$

Metrics

$n = n_+ + n_-$, $n_+ = TP + FN$, $n_- = TN + FP$
Accuracy: $\frac{TP+TN}{n}$, Precision: $\frac{TP}{TP+FP}$
Recall/TPR: $\frac{TP}{n_+}$, FPR: $\frac{FP}{n_-}$, F1: $\frac{2TP}{2TP+FP+FN} = \frac{2}{\frac{1}{prec} + \frac{1}{rec}}$, ROC Curve: y=TPR, x=FPR

Multi-class

One-vs-all (Classif. c) but requires confidence in prediction, One-vs-one (Classif. $\frac{c \times (c-1)}{2}$), encoding

Multi-class Hinge loss

$l_{MC-H}(w^{(1)}, \ldots, w^{(c)}; x, y) = \max(0, 1 + \underset{j \in \{1, \cdots, y-1, y+1, \cdots, c\}}{\max} w^{(j)T} x - w^{(y)T} x)$

Neural networks

Parameterize feature map with θ: $\phi(x, \theta) = \varphi(\theta^T x) = \varphi(z)$ (activation function φ)
$\Rightarrow w^* = \text{argmin}_{w,\theta} \sum_{i=1}^n l(y_i; \sum_{j=1}^m w_j \phi(x_i, \theta_j))$
$f(x; w, \theta_{1:d}) = \sum_{i=1}^m w_j \varphi(\theta_j^T x) = w^T \varphi(\Theta x)$ **Overfitting**: Early Stopping, Regularization $\lambda ||W||_F^2$, Dropout: Randomly ignore hidden units during each SGD iter. with probability p, Change weights after training to compensate (All units present), **Batch Norm.**: Reduces cov. shift, larger LR possible, regularizing effect, **Difference Kernels**: Kernels optimize α only $\Rightarrow$ convex, ANNs optimize w and θ.

Activation functions

Sigmoid: $\frac{1}{1+exp(-z)}$, $\varphi'(z) = (1 - \varphi(z)) \cdot \varphi(z)$
tanh: $tanh(z) = \frac{exp(z) - exp(-z)}{exp(z) + exp(-z)}$, ReLU: $\max(z, 0)$

Forward Propagation

Input layer: $\mathbf{v}^{(0)} = \mathbf{x}$; **Hidden layers:** $\mathbf{z}^{(\ell)} = \mathbf{W}^{(\ell)} \mathbf{v}^{(\ell-1)}$, $\mathbf{v}^{(\ell)} = \phi(\mathbf{z}^{(\ell)})$; **Output layer:** $f = \mathbf{W}^{(L)} \mathbf{v}^{(L-1)}$ **E.g.:** $E = (y - u_1 \rho(w_1 x_1 + w_2 x2) + \ldots)^2$

SGD for ANNs

$\hat{\mathbf{W}} = \text{argmin}_{\mathbf{W}} \sum_{i=1}^n \ell(\mathbf{W}; \mathbf{x}_i, y_i)$; $\ell(\mathbf{W}; \mathbf{x}, \mathbf{y}) = \ell(\mathbf{y} - f(\mathbf{x}, \mathbf{W}))$; For random $(\mathbf{x}, \mathbf{y})$, $\mathbf{W}_{t+1} = \mathbf{W}_t - \eta_t \nabla_{\mathbf{W}} \ell(\mathbf{W}; \mathbf{x}, \mathbf{y})$

Backpropagation

Output layer: Err: $\delta^{(L)} = \mathbf{l}'(\mathbf{f}) = [l'(f_1), \ldots, l'(f_p)]$
Grad: $\nabla_{\mathbf{W}^{(L)}} \ell(\mathbf{W}; \mathbf{y}, \mathbf{x}) = \delta^{(L)} \mathbf{v}^{(L-1)T}$
Hidden layers: Err: $\delta^{(\ell)} = \phi'(\mathbf{z}^{(\ell)}) \odot \mathbf{W}^{(\ell+1)T} \delta^{(\ell+1)}$ Grad: $\nabla_{\mathbf{W}^{(\ell)}} \ell(\mathbf{W}; \mathbf{y}, \mathbf{x}) = \delta^{(\ell)} \mathbf{v}^{(\ell-1)T}$

Learning with momentum

$a \leftarrow m \cdot a + \eta_t \nabla_W l(W; y, x)$; $W_{t+1} \leftarrow W_t - a$

CNN

Output dim: $L \times L \times M$, M: filters, F: size of filters, S: stride (how many positions the filters are moved), P: padding (pad inputs with 0). $L = \left\lfloor \frac{N-F+2P}{S} - 1 \right\rfloor$
Pooling: Aggregate Units to decrease width of the network (avg or max)

Clustering

k-mean

$\hat{R}(\mu) = \sum_{i=1}^n \min_{j \in \{1, \ldots k\}} ||x_i - \mu_j||_2^2$, $\hat{\mu} = \text{argmin}_\mu \hat{R}(\mu)$
non-convex, NP-hard, only conv. to local opt., iter can be exponential $\mathbf{O}(nkd)$ **Selecting k**: Elbow method, increasing k leads to negligible decrease in loss, CV can't be used as test loss keeps decreasing **Lloyd's Heuristic:** 0. Initialize cluster centers While not converged: 1.Assign points, 2.Update centers. (converges to local optimum) **k-Means++:** Start with rand. data pts. as center, add centers rand. $\propto$ squared dist. to closest center. $O(\log k) \cdot$ cost of opt. k-Means sol. **Spectral clustering** = kernelized k-means.

Dimensionality reduction

PCA (linear dim. reduction)

Linear mapping $W^T x$ that projects vectors x into a k-dim. subspace such that the reconstruction error (euclidian) is minimal. $D = x_1,...,x_n \subset \mathbb{R}^d$, $\Sigma = \frac{1}{n}\sum_{i=1}^n x_i x_i^T$, $\mu = 0$
$(W, z_1, ..., z_n) = \text{argmin} \sum_{i=1}^n ||W z_i - x_i||_2^2$,
$W = (v_1|...|v_k) \in \mathbb{R}^{d \times k}$, orthogonal; $z_i = W^T x_i$
v_i are the eigenvectors of $\Sigma = \sum_{i=1}^d \lambda_i v_i v_i^\top$. Only if $k = d$ x_i can be reconstructed from k principal components. Via SVD $\rightarrow k$ first columns of V.

Kernel PCA

Kernel PC: $\alpha^{(1)}, ..., \alpha^{(k)} \in \mathbb{R}^n$, $\alpha^{(i)} = \frac{1}{\sqrt{\lambda_i}} v_i$, $K = \sum_{i=1}^n \lambda_i v_i v_i^T$, $\lambda_1 \geq ... \geq \lambda_d \geq 0$
New point: $\hat{z} = f(\hat{x}) = \sum_{j=1}^n \alpha_j^{(i)} k(\hat{x}, x_j)$

Autoencoders

Find identity function: $x \approx f(x; \theta)$, $f(x;\theta) = f_{decode}(f_{encode}(x;\theta_{encode}); \theta_{decode})$, if $\phi(z) = z \rightarrow$ equal to PCA.

Probabilistic modeling

Find $h : X \rightarrow Y$ that minimize prediction error: $R(h) = \int P(x,y) l(y; h(x)) \partial x \partial y = \mathbb{E}_{x,y}[l(y; h(x))]$

For least squares regression

Bayes optimal predictor h: $h^*(x) = \mathbb{E}[Y|X = x]$
Pred.: $\hat{y} = \hat{\mathbb{E}}[Y|X = \hat{x}] = \int \hat{P}(y|X = \hat{x}) y \partial y$

Maximum Likelihood Estimation (MLE)

$\theta^* = \text{argmax}_\theta \hat{P}(y_1, ..., y_n | x_1, ..., x_n, \theta)$
E.g. lin. + Gauss: $y_i = w^T x_i + \varepsilon_i, \varepsilon_i \sim N(0, \sigma^2)$
i.e. $y_i \sim N(w^T x_i, \sigma^2)$, With MLE (use argmin $-\log$): $w^* = \text{argmin}_w \sum (y_i - w^T x_i)^2$. For Gaussian MLE equal to least squares solution.

Bias Variance trade-off

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

Maximum a posteriori estimate (MAP)

likelihood = loss function, regularizer = prior.
$\hat{\theta} = \text{argmax}_\theta f(\theta|x) = \text{argmax}_\theta\, g(\theta) \prod_{i=1}^n f(x|\theta)$
Gauss. prior $\equiv ||w||_2^2$, Laplace prior $\equiv ||w||_1$
SGD: $w = w(1 - 2\lambda\eta_t) + \eta_t y x \hat{P}(Y = -y|w, x)$

Logistic regression

Link func.: $\sigma(w^T x) = \frac{1}{1 + exp(-w^T x)}$ (Sigmoid), iid Bernoulli noise instead: $P(y|x, w) = Ber(y; \sigma(w^T x))$
Classification: Use $P(y|x, w)$, predict most likely class label. (Boundary shifted towards less training data points) MLE: $\text{argmax}_w P(y_{1:n}|w, x_{1:n}) \Rightarrow w^* = \text{argmin}_w \sum_{i=1}^n log(1 + exp(-y_i w^T x_i))$,
Convex $\rightarrow$ SGD: $w = w + \eta_t y x \cdot \hat{P}(Y = -y|w, x)$
$\hat{P}(Y = -y|w, x) = \frac{1}{1 + exp(y w^T x)}$, can be regularized (L1, L2) + kernelized, apply to NN for class probabilities, multi-class setting: maintain 1 w per class, model the others.

Bayesian decision theory

Conditional distribution over labels $P(y|x)$, Set of actions A, Cost function $C : Y \times A \rightarrow \mathbb{R}$
$a^* = \text{argmin}_{a \in A}\, \mathbb{E}[C(y, a)|x]$
Calculate $\mathbb{E}$ via sum/integral.
Classification: $C(y, a) = [y \neq a]$; asymmetric:
$$C(y,a) = \begin{cases} c_{FP}\text{ , if } y = -1, a = +1 \\ c_{FN}\text{ , if } y = +1, a = -1 \\ 0\text{ , otherwise} \end{cases}$$
Regression: $C(y, a) = (y - a)^2$; asymmetric: $C(y, a) = c_1 \max(y - a, 0) + c_2 \max(a - y, 0)$
E.g. $y \in \{-1, +1\}$, predict $+$ if $c_+ < c_-$, $c_+ = \mathbb{E}(C(y, +1)|x) = P(y = 1|x) \cdot 0 + P(y = -1|x) \cdot c_{FP}$, c_- likewise

Discriminative vs. generative modeling

Discriminative estimate $P(y|x)$, generative $P(y, x)$. Generative approach uses chain rule: $P(x, y) = P(x|y) \cdot P(y) = P(y|x) \cdot P(x)$. Discriminative: generally more robust, but cannot detect outliers. Generative: Can be more powerful (e.g., detect outliers) if model assumptions are met.

Naive Bayes

Naive: features conditionally **independent** given Y, prior on labels $P(y)$, Estimate conditional distribution $P(x|y)$ for each class y, Classes can be mixture of categorical and continuous features. Prediction using **Bayes rule**: $P(y|x) = \frac{P(y)P(x|y)}{P(x)} = \frac{P(y)P(x|y)}{\sum_y P(x,y)} = \frac{P(y)P(x|y)}{\sum_y P(y)P(x|y)}$.
Use conjugate priors (posterior dist. same as prior) to avoid overfitting.

Deriving decision rule

$P(y|x) = \frac{1}{Z} P(y) P(x|y)$, $Z = P(x)^{-1} = \sum_y P(y) P(x|y)$
$y^* = \text{amax}_y\, P(y|x) = \text{amax}_y\, P(y) \prod_{i=1}^d P(x_i|y)$

Gaussian (Naive) Bayes Classifier

Gaussian Naive Bayes: $\hat{\mu}_y, \hat{\sigma}_y$, indep. assumption. # param. $= O(c \cdot d)$ **Gaussian Bayes**: corr. among features, but $O(c \cdot d^2)$, MLE for GB: $\hat{P}(x|y) = N(x; \hat{\mu}_y, \hat{\Sigma}_y)$, $\hat{P}(Y = y) = \hat{p}_y = \frac{n_y}{n}$, $\hat{\mu}_y = \frac{1}{n_y} \sum_{i: y_i = y} x_i \in \mathbb{R}^d$, $\hat{\Sigma}_y = \frac{1}{n_y} \sum_{i: y_i = y} (x_i - \hat{\mu}_y)(x_i - \hat{\mu}_y)^T \in \mathbb{R}^{d \times d}$
Discriminant function:
$f(\mathbf{x}) = \log \frac{p}{1-p} + \frac{1}{2}\Big[\log \frac{|\hat{\Sigma}_-|}{|\hat{\Sigma}_+|} + (\mathbf{x} - \hat{\mu}_-)^T \hat{\Sigma}_-^{-1} (\mathbf{x} - \hat{\mu}_-) - (\mathbf{x} - \hat{\mu}_+)^T \hat{\Sigma}_+^{-1} (\mathbf{x} - \hat{\mu}_+)\Big]$
Fisher's linear discriminant analysis: Assuming equal class probabilities and covariances:
$f(\mathbf{x}) = \mathbf{w}^T\ \mathbf{x} + w_0$, $w_0 = \frac{1}{2}(\hat{\mu}_-^T \hat{\Sigma}^{-1} \hat{\mu}_- - \hat{\mu}_+^T \hat{\Sigma}^{-1} \hat{\mu}_+)$, $\mathbf{w} = \hat{\Sigma}^{-1}(\hat{\mu}_+ - \hat{\mu}_-)$
if assumptions met, Gaussian NB and LDA same pred. as *logistic* regression, LDA maximizes ratio of between-class and within-class variances.
Quadratic discriminant analysis: we predict $y = \text{sign}(f(\mathbf{x}))$ ($f(\mathbf{x})$ = discriminant function)

Categorical Naive Bayes Classifier

MLE for $P(y) = p = \frac{n_+}{n}$, MLE for feature distr.: $\hat{P}(X_i = c|Y = y) = \theta_{c|y}^{(i)} = \frac{Count(X_i = c, Y = y)}{Count(Y = y)}$,
Prediction: $y^* = \text{argmax}_y \hat{P}(y|x)$
Could lift Naive assumption by modeling joint cond. dist., but exponential in d and prone to overfitting.

Mixture models

Model each c. as probability distr. $P(x|\theta_j)$
$P(D|\theta) = \prod_{i=1}^n \sum_{j=1}^k w_j P(x_i|\theta_j)$
$L(w, \theta) = -\sum_{i=1}^n \log \sum_{j=1}^k w_j P(x_i|\theta_j)$. ($\rightarrow$ Fitting a GMM = Training a GBC without labels; Clustering = latent variable modeling)

Gaussian-Mixture Bayes classifiers

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

Hard-EM algorithm

Initialize parameters $\theta^{(0)}$, for $t = 1, 2, ...$
E-step: Predict most likely class for each point:
$z_i^{(t)} = \text{argmax}_z\, P(z|x_i, \theta^{(t-1)})$
$= \text{argmax}_z\, P(z|\theta^{(t-1)}) P(x_i|z, \theta^{(t-1)})$; now we have comlete data: $D^{(t)} = \{(x_1, z_1^{(t)} ..., x_n, z_n^{(t)}\}$
M-step: Compute the MLE:
$\theta^{(t)} = \text{argmax}_\theta P(D^{(t)}|\theta)$, i.e. $\mu_j^{(t)} = \frac{1}{n_j} \sum_{i: z_i = j} x_j$
with spherical covariances same as k-means. CV can be used to determine # cluster centers. Alternating optimization on the complete data likelihood.

Soft-EM algorithm

E-step: Calc. cluster membership weights for each point: $\gamma_j^{(t)}(x_i)$ given estimates of previous iterations.
M-step: Fit clusters to weighted data points:
$w_j^{(t)} = \frac{1}{n} \sum_{i=1}^n \gamma_j^{(t)}(x_i)$; $\mu_j^{(t)} = \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i) x_i}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)}$
$\sigma_j^{(t)} = \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i)(x_i - \mu_j^{(t)})^T (x_i - \mu_j^{(t)})}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)}$

Avoid degeneravy: small term to the diagonal of the MLE. Constrained GMM: different covariance matrices $\rightarrow$ different sizes of clusters. (Diagonal = GNB). Each iteration in M-step is equiv. to training a GBC with weighted data. $\rightarrow$ Closed form solution

Soft-EM for semi-supervised learning

learning from unlabeled and labeled data.
E-step: labeled points: y_i: $\gamma_j^{(t)}(x_i) = [j = y_i]$, unlabeled: $\gamma_j^{(t)}(x_i) = P(Z = j|x_i, \mu^{(t-1)}, \Sigma^{(t-1)}, w^{(t-1)})$

Useful Math

Calculus

$F'(x) = f'(g(x)) g'(x)$
$\frac{\delta x^T A x}{\delta x} = (A + A^T)x$, $\frac{\delta x^T a}{\delta x} = \frac{\delta a^T x}{\delta x} = a$, $\frac{\delta a^T X b}{\delta X} = ab^T$, $\frac{\delta a^T X^T b}{\delta X} = ba^T$, $\frac{\delta a^T X a}{\delta X} = \frac{\delta a^T X^T a}{\delta X} = aa^T$

Probabilities

$\mathbb{E}_x[X] = \int x \cdot p(x) \partial x$ (cont.), $\mathbb{E}_x[X] = \sum_x x \cdot p(x)$
$\text{Var}[X] = \mathbb{E}[(X - \mu_X)^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$
$\mathbb{E}_x[b + cX] = b + c \cdot \mathbb{E}_x[X]$ $\mathbb{E}_x[b + CX] = b + C \cdot \mathbb{E}_x[X], C \in \mathbb{R}^{n \times n}$
$\mathbb{V}_x[b + cX] = c^2 \mathbb{V}_x[X]$, $\mathbb{V}_x[b + CX] = C \mathbb{V}_x[X] C^\top, C \in \mathbb{R}^{n \times n}$
$\text{Cov}[X, Y] = \mathbb{E}[(X - \mathbb{E}(X)(Y - \mathbb{E}(Y))] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$, $\mathbb{V}[X + Y] = \mathbb{V}[X] + \mathbb{V}[Y] + 2\text{Cov}[X, Y]$

Invertible/nonsingular Matrices

$A^{m \times m}$: $A^{-1} A = I_d = A A^{-1}$ only if $\det(A) \neq 0$; $Ax = 0$ has only trivial solution $x = 0$.

Orthogonal Matrices

$A^{m \times m} : A^\top A = I_d = A A^\top \Leftrightarrow A^\top = A^{-1}$

Symmetric Positive Definite Matrices

Symmetric: $A^{n \times n}$: $A^\top = A$, symmetric positive definite if: $\forall x \backslash \{0\} \in \mathbb{R}^n : x^\top A x > 0$ (semi-definite if: ≥ 0) $\Leftrightarrow$ all eigenvalues of A are positive.

Eigendecomposition

$AP = PD \Leftrightarrow A = PDP^{-1}$ iff eigenvectors of A form a basis in $\mathbb{R}^n$. D diagonal matrix of eigenvalues, Eigenvectors in P. $Ap = \lambda p$

Cholesky decomposition

$A^{n \times n} : A = LL^\top$, symmetric and positive definite.

Singular value decomposition

$A = U \Sigma V^\top$; $A^{m \times n}; U^{m \times m}, V^{n \times n} : U, V$ orthogonal and $\Sigma^{m \times n}$ diagonal with singular values $\sigma = \sqrt{\lambda(A^\top A)}$
$Av = \sigma u$