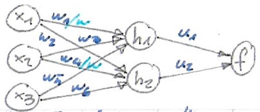


Perceptron

$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^n \overbrace{\max\{0, -y_i w^T x_i\}}^{=:\ell_P(w;y_i,x_i)} + \lambda\|w\|_2^2$

Lasso → $\lambda\|w\|_1$ L1, $\lambda\|w\|_2^2$ L2 } regularisation

Note: Perceptron algo. doesn't do max. margin hyperplanes, the algo. has no way of choosing between them.

- $w = \sum_{j=1}^n \alpha_j y_j x_j$ $\quad \max\{0, -y_i(\sum_{j=1}^n \alpha_j y_j x_j)^T x_i\}$
- kernelised ℓ_P: $= \max\{0, -y_i \sum_{j=1}^n \alpha_j y_j k(x_j, x_i)\}$
- kernelised perceptron: $\alpha^* = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^n \max\{0, -y_i \sum_{j=1}^n \alpha_j y_j k(x_j, x_i)\}$

Linear SVM

hinge loss → $w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^n \overbrace{\max\{0, 1 - y_i w^T x_i\}}^{\ell_{SVM}(w;y_i,x_i)\ \text{hard margin}} + \lambda\|w\|_2^2$ ← soft margin

$\lambda\|w\|_1$ ← automatic feature selection!!!

Note: we seek a max. margin hyperplane

- kernelised SVM: $\alpha^* = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^n \max\{0, 1 - y_i \sum_{j=1}^n \alpha_j y_j k(x_j, x_i)\}$
- Multiclass: • One-vs-All: we solve c binary classifiers one for each class. We then classify using the classifier w/ largest confidence.
- One-vs-One: Train $c(c-1)/2$ classifiers, one for each pair of classes and let them "vote". $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n\times d}$, $w \in \mathbb{R}^d$

Linear Regression

- Squared loss fct: $\underset{w}{\operatorname{argmin}} \overbrace{\sum_{i=1}^n (y_i - w^T x_i)^2}^{\|y - Xw\|} + \lambda\|w\|_2^2$ ← ridge regres. ($\lambda\|w\|_1$ ← real Lasso)
- Closed form sol.: $w^* = (X^T X + \lambda \mathbb{1})^{-1} X^T y$

Logistic Regression (linear)

- $w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^n \overbrace{\log(1 + \exp(-y_i w^T x_i))}^{=:\ell_{logistic}(w;y_i,x_i)} + \lambda\|w\|_2^2$ ($\lambda\|w\|_1$) } regularisation

Note: For logistic regression models unbalanced training data affects only the estimate of the model intercept (although this skews all the predicted prob., which compromises prediction)

- kernelised logistic regression: $\alpha^* = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^n \log(1 + \exp(-y_i \sum_{j=1}^n \alpha_j y_j k(x_j, x_i)))$
- logistic function: $\sigma(w^T x) = 1/(1 + \exp(-y w^T x))$
- We assume Bernoulli noise: $P(y|x,w) = \mathrm{Ber}(y; \sigma(w^T x)) = \frac{1}{1 + \exp(-y w^T x)}$

MLE: Get Logistic loss $L(w) = \sum_{i=1}^n \log(1 + \exp(-y_i w^T x_i))$

Gradient: $\nabla_w L(w) = 1/(1 + \exp(y w^T x)) \cdot (-yx) = P(-y|x,w) \cdot (-yx)$

MAP: add L1 or L2 regulariser to get the MAP estimate.

Stochastic Gradient Descent (SGD)

(A) Start at an arbitrary (random) $w_0 \in \mathbb{R}^d$ [e.g. $w_0 = 0$]

(B) For $t = 1, 2, \dots$ do

(i) Pick data point $(x', y') \in D$ from training set uniformly at random. (with replacement)

(ii) $w_{t+1} = w_t - \eta_t \nabla \ell(w_t; x', y')$, where $\eta_t \hat{=}$ learning rate

Note: mild conditions needed: (1) $\sum_t \eta_t = \infty$ (2) $\sum_t \eta_t^2 < \infty$, e.g. $\eta_t = 1/t$ (i.e. harmonic progression)

K-means

$L(\mu) = L(\mu_1, \dots, \mu_k)$

- $\mu^* = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^n \overbrace{\min_{j\in\{1,\dots,k\}} \|x_i - \mu_j\|_2^2}^{=:\ell(\mu, x_i)} + \lambda k$ (regularisation)
- K-means algorithm (Lloyd's heuristic):

(0) Initialise cluster centers $\mu^{(0)} = [\mu_1^{(0)}, \dots, \mu_k^{(0)}]$

(1) "E-step": Assign point to closest center $z_i \leftarrow \underset{j\in\{1,\dots,k\}}{\operatorname{argmin}} \|x_i - \mu_j^{(t-1)}\|_2^2$

(2) "M-step": Update center as mean of assigned data points $\mu_j^{(t)} \leftarrow 1/n_j \sum_{i:z_i=j} x_i$, $n_j := |\{x_i | z_i = j\}| \hat{=}$ #points of cluster j

- k-means++ initialisation:

(i) start with random point as center

(ii) pick $\mu_j = x_i$ randomly from the distribution $P(\mu_j = x_i) = 1/z \min_{\ell\in\{1,\dots,j-1\}} \|x_i - \mu_\ell\|_2^2$, $z \hat{=}$ normalisation const

Kernels

- $w^* = \sum_{i=1}^n (\alpha_i y_i) x_i$

Kernel Perceptron: $\alpha^* = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^n \max(0, -y_i \sum_{j=1}^n \alpha_j y_j \overbrace{(x_j^T x_i)}^{k(x_j,x_i)})$

Kernel SVM: $\alpha^* = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^n \max(0, 1 - y_i \alpha^T k_i) + \lambda \alpha^T K \alpha$

where $k_i = [y_1 k(x_i, x_1), \dots, y_n k(x_i, x_n)]$

Predict by: $y = \mathrm{sign}(\sum_{i=1}^n \alpha_i y_i k(x_i, x))$

Kernel linear Regression: $\alpha^* = \underset{\alpha}{\operatorname{argmin}} \|\alpha^T K - y\|_2^2 + \lambda \alpha^T K \alpha$

Closed form solution: $\alpha^* = (K + \lambda \mathbb{1})^{-1} y$

Predict by: $y = \sum_{i=1}^n \alpha_i^* k(x_i, x)$

Kernel trick: (1) Initialise $\alpha_1 = \dots = \alpha_n = 0$

(2) For $t = 1, 2, \dots$ do

- Pick data point (x_i, y_i) uniformly at random
- Predict
- if $\hat{y} \neq y_i$ set $\alpha_i = \alpha_i + \eta_t$

Definition: → Data space X; → A kernel is a fct $k: X \times X \to \mathbb{R}$

→ Properties: (i) Symmetry: $\forall x, x' \in X: k(x,x') = k(x',x)$

(ii) Pos. semi-def.: For any n, any set $S = \{x_1, \dots, x_n\} \subset X$, the kernel (Gram) matrix $K = \begin{pmatrix} k(x_1,x_1) & \cdots & k(x_1,x_n) \\ \vdots & & \vdots \\ k(x_n,x_1) & \cdots & k(x_n,x_n) \end{pmatrix}$ must be pos. semi-def., i.e. it holds:

(A) EW of K are ≥ 0 (EW are real by symmetry)

(B) $\forall w \in \mathbb{R}^d$: $w^T K w \geq 0$

(C) Hauptminoren-Kriterium für Definitheit: Für eine symm. Matrix $A \in \mathbb{R}^{n\times n}$ gilt: A pos. def. $\Leftrightarrow \det(A_k) > 0$, für $k = 1, \dots, n$

Linalg: Satz: Sei A symm./herm., dann gilt: (i) A hat nur reelle EW; (ii) EV zu versch. EW sind paarw. orthogonal

Kor.: (Fischer) Eine symm. Matrix $A \in \mathbb{R}^{n\times n}$ ist genau dann pos. def., wenn ihre EW $\lambda_1, \dots, \lambda_n > 0$ sind.

Kernel engineering

Suppose we have two kernels k_1, k_2 def. on X.

(1) Summation: $k(x,x') = k_1(x,x') + k_2(x,x')$

(2) Multiplication: $k(x,x') = k_1(x,x') k_2(x,x')$

(3) Scaling: $k(x,x') = c k_1(x,x')$, $c > 0$

(4) Mapping: $k(x,x') = f(k_1(x,x'))$ where f is a polynomial with positive coeffs or exponential function!

Examples of kernels on $\mathbb{R}^d$:

- Lin. kernel: $k(x,x') = x^T x'$
- Polynomial kernel: $k(x,x') = (x^T x' + 1)^d$
- Gaussian (RBF, squared exp.) kernel: $k(x,x') = \exp(-\|x - x'\|_2^2 / h^2)$
- Sigmoid (tanh) kernel: $k(x,x') = \tanh(k x^T x' - b)$, $k \hat{=}$ slope

Artificial Neural Networks

- $f = \sum_{j=1}^m u_j \varphi(\underbrace{w_j^T x}_{z})$ ← v_j

predict by: $y_j = f_j$, regression; $y_j = \mathrm{sign}(f_j)$ classification (sometimes with softmax for multiclass)

Back propagation: • For each unit j on the output layer L:

(i) $\delta_j = \ell_j'(f_j)$ where $\ell_i(f_i) = (y_i - f_i)^2$

(ii) For each unit i on layer $L-1$: $\partial_{w_{j,i}} = \delta_j v_i$

- For each unit j on the hidden layer $\ell = L-1 : -1 : 1$:

(i) $\delta_j = \varphi'(z_j) \sum_{j \in \mathrm{Layer}_{\ell+1}} w_{i,j} \delta_i$ where $z_j = \sum_k w_{j,k} v_{j-1,k}$

(ii) For each unit i on layer $L-1$: $\partial_{w_{j,i}} = \delta_j v_i$

Example S3A1: NN with two logistic hidden units h_1, h_2 and three inputs x_1, x_2, x_3. The output neuron f is a linear unit and we are using the squared error cost fct. $E = (y - f)^2$. Logistic fct: $\Phi(x) = 1/(1 + \exp(-x))$.

(i) Forward propagation: $h_1 = \Phi(v_1)$, $h_2 = \Phi(v_2)$

$v_1 = w_1 x_1 + w_3 x_2 + w_5 x_3$, $v_2 = w_2 x_1 + w_4 x_2 + w_6 x_3$

$\Rightarrow f = u_1 h_1 + u_2 h_2$

(ii) $\nabla_w E$? $\quad \nabla_w E = \frac{\partial E}{\partial f}\frac{\partial f}{\partial h_1}\frac{\partial h_1}{\partial v_1}\frac{\partial v_1}{\partial w} + \frac{\partial E}{\partial f}\frac{\partial f}{\partial h_2}\frac{\partial h_2}{\partial v_2}\frac{\partial v_2}{\partial w}$

$= -2(y - f)[u_1 \Phi(v_1)(1 - \Phi(v_1)) x_1 + u_2 \Phi(v_2)(1 - \Phi(v_2)) x_2]$

Probability/Statistics

Bayes: $P(y|x) = 1/z\, P(y) P(x|y)$, where $z = P(x) = \sum_y P(y) P(x|y)$

Wiki: • simple case: $P(A|B) = P(B|A) P(A) / P(B)$

- extended form: $P(A_i|B) = P(B|A_i) P(A_i) / P(B)$ where $P(B) = \sum_j P(B|A_j) P(A_j)$

Chain rule: $P(y,x) = P(x|y)P(y) = P(y|x)P(x) \Rightarrow P(y|x) = P(y,x)/P(x) = \frac{1}{z} P(y,x)$

Naive Bayes Model

Assumption: $P(y|x) = P(y,x)/P(x)$

- Estimate prior: $\hat{P}(Y=y) = \mathrm{Cnt}(Y=y)/n$ — MLE $P(y)$
- Estimate feature distribution: $\hat{P}(x_i|y) = \mathcal{N}(x_i; \hat{\mu}_{y,i}, \hat{\sigma}_{y,i}^2)$ — MLE $P(x_i|y)$

where $\hat{\mu}_{y,i} = 1/\mathrm{cnt}(Y=y) \sum_{j:y_j=y} x_{j,i}$

$\hat{\sigma}_{y,i}^2 = 1/\mathrm{cnt}(Y=y) \sum_{j:y_j=y} (x_{j,i} - \hat{\mu}_{y,i})^2$

- Predict: $y = \underset{y}{\operatorname{argmax}}\, P(y|x) = \underset{y}{\operatorname{argmax}}\, P(y) \prod_{i=1}^D P(x_i|y)$
- Derivation MLE $P(y)$: Want: $p := P(Y=1)$, $(1-p) = P(Y=0)$

Data: D & $D_y = \{y^{(1)}, \dots, y^{(n)}\}$

$P(D_y|p) = \prod_{i=1}^n p^{\mathbb{1}[y_i=1]} (1-p)^{\mathbb{1}[y_i=0]} = p^{n_1}(1-p)^{n_0}$

$\log(P(D_y|p)) = \log(p^{n_1}(1-p)^{n_0}) = n_1 \log(p) + n_0 \log(1-p)$

$\partial_p \log(P(D_y|p)) = n_1/p - n_0/(1-p) \overset{!}{=} 0 \Leftrightarrow \dots \Leftrightarrow \hat{P}(Y=y) = \frac{\mathrm{cnt}(Y=y)}{n}$

Rem. NB: NB model features are conditionally independent given Y: $P(X_1, \dots, X_d | Y) = \prod_{i=1} P(X_i|Y)$

Example 2016 A4(iv): NB classifier with d binary feature $X_1, \dots, X_d \in \{0,1\}$. We estimate $P(Y=1) = P(Y=0) = 0.5$, $P(X_i = 1|Y=0) = \theta$ and $P(X_i=1|Y=1) = 1-\theta$, $\theta \in [0, 0.5)$ $\forall i \in \{1, \dots, d\}$. Given a new example with $X_1 = \dots = X_d = 1$, determine the smallest #features d s.t. the human expert is not consulted (the smallest number should be a function of θ, the cost c_h, c_m.)

$P(Y=1) = 0.5$; $P(X_i = 1|Y=0) = \theta$
$P(Y=0) = 0.5$; $P(X_i = 1|Y=1) = 1-\theta$ } $\theta \in [0, 0.5)$ (*)

$\Rightarrow y = \underset{y'}{\operatorname{argmax}}\, P(y'|x) = \underset{y'}{\operatorname{argmax}}\, P(y') \prod_{i=1}^d P(x_i|y') = \underset{y'}{\operatorname{argmax}} \frac{1}{2}\prod_{i=1}^d P(x_i|y')$

$\Rightarrow P(y=1|x) = \frac{1}{2}\prod_{i=1}^d P(x_i=1|y=1) = \frac{1}{2}\prod_{i=1}^d (1-\theta) = \frac{1}{2}(1-\theta)^d$; $P(y=0|x) = \frac{1}{2}\theta^d$

(*) $\Rightarrow \theta < 1-\theta \Rightarrow \theta^d < (1-\theta)^d \Rightarrow \frac{1}{2}\theta^d < \frac{1}{2}(1-\theta)^d$

$\Rightarrow y = \underset{y'}{\operatorname{argmax}} \frac{1}{2}\prod_{i=1}^d P(x_i|y) = \underset{y'}{\operatorname{argmax}} \{\frac{1}{2}(1-\theta)^d, \frac{1}{2}\theta^d\} = 1$

$\Rightarrow p_1 := P(y=1|x) = \frac{1}{2}(1-\theta)^d$

(*2) $\Rightarrow (1-p_1)c_m < c_h \Leftrightarrow \dots \Leftrightarrow \frac{\log(2(c_m - c_h)/c_m)}{\log(1-\theta)} < d$

(iii) Suppose for some ex. the posterior is $P(Y=1|X_1=x_1, X_2=x_2, X_3=x_3) = p$ and assume that there is a cost c_m for misclassification. For each prediction you have the choice between trusting the NBC or asking a human expert with cost $c_h < c_m$, who is 100% accurate. Which of these two choices minimises the expected cost as a fct of p?

Est. cond. distr.: $\hat{P}(y'|x) = \hat{P}(y') \prod_{i=1}^3 \hat{P}(x_i|y')$

Action set: $A := \{1, 0, D\}$

Cost function: $\mathrm{Cost}(y,a) = \begin{cases} c_m \mathbb{1}[y \neq a], & a \in \{1,0\} \\ c_h < c_m, & a = D \end{cases}$

$a^* = \underset{a\in A}{\operatorname{argmin}}\, \mathbb{E}[C(y,a)]$; $p_1 := \hat{P}(y=1|x)$

$\Rightarrow \mathbb{E}[C(y,a)] = C_1 \mathbb{1}[a=1] + C_0 \mathbb{1}[a=0] + C_D \mathbb{1}[a=D] = $ (*3)

$C_1 = \mathbb{E}[C(y,1)] = p_1 \cdot 0 + (1-p_1) \cdot c_m = (1-p_1)c_m$

$C_0 = \mathbb{E}[C(y,0)] = p_1 \cdot c_m + (1-p_1) \cdot 0 = p_1 c_m$

$C_D = \mathbb{E}[C(y,D)] = p_1 \cdot c_h + (1-p_1) \cdot c_h = c_h$

(*3) $= \underbrace{(1-p_1)c_m \mathbb{1}[a=1]} + p_1 c_m \mathbb{1}[a=0] + \underbrace{c_h \mathbb{1}[a=D]}$

$a^* = \begin{cases} y, & \hat{P}(-y|x) c_m < c_h \Leftrightarrow p_1 \in I \quad (*2) \\ D, & \text{otherwise} \end{cases}$ ← sufficient!

- $(1-p_1)c_m < c_h \Leftrightarrow \frac{c_m - c_h}{c_m} < p_1$ • $p_1 c_m < c_h \Leftrightarrow p_1 < \frac{c_h}{c_m}$

$\Rightarrow p_1 \in [(c_m - c_h)/c_m, c_h/c_m] =: I$ s.t. the expected cost is minimised.

Example 2015 A2(iii) Consider the class posterior distr. $P(y|x)$, assume that the cost c_{10} for classifying a relevant message as irrelevant is larger than the cost c_{01} of classifying an irrelevant message as relevant. The cost of classifying correctly is assumed to be zero. How does the classification rule change?

Est. cond. distr.: $\hat{P}(y'|x) = \hat{P}(y') \prod_{i=1}^d \hat{P}(x_i|y')$

Action set: $A := \{0, 1\}$

Cost function: $c(y,a) = \begin{cases} c_{10} \mathbb{1}[y \neq a], & a = 0 \\ c_{01} \mathbb{1}[y \neq a], & a = 1 \end{cases}$ & $c_{01} < c_{10}$

$a^* = \underset{a\in A}{\operatorname{argmin}}\, \mathbb{E}[C(y,a)]$, $p_1 := \hat{P}(y=1|x)$

$\mathbb{E}[C(y,a)] = C_1 \mathbb{1}[a=1] + C_0 \mathbb{1}[a=0]$

$C_1 = \mathbb{E}[C(y,1)] = p_1 \cdot 0 + (1-p_1) c_{01} = (1-p_1) c_{01}$

$C_2 = \mathbb{E}[C(y,0)] = p_1 c_{10} + (1-p_1) \cdot 0 = p_1 c_{10}$

$= \underset{a\in A}{\operatorname{argmin}}\, (1-p_1) c_{01} \mathbb{1}[a=1] + p_1 c_{10} \mathbb{1}[a=0] = \begin{cases} 1, & p_1 > c_{01}/(c_{01} + c_{10}) \\ 0, & \text{otherwise} \end{cases}$

Gaussian Bayes Classifiers

No longer assume that feature are independent

Use multivariant gaussian: $P(x|y) = \mathcal{N}(x; \mu_y, \Sigma_y)$

- Estimate prior: $\hat{P}(Y=y) = \mathrm{cnt}(Y=y)/n$
- Estimate feature distr.: $\hat{P}(x_i|y) = \mathcal{N}(x_i | \hat{\mu}_y, \Sigma_y)$

where $\hat{\mu}_y = 1/\mathrm{cnt}(Y=y) \sum_{j:y_j=y} x_j$

$\hat{\Sigma}_y = 1/\mathrm{cnt}(Y=y) \sum_{j:y_j=y} (x_j - \mu_y)(x_j - \mu_y)^T$

- Predict: $y = \underset{y}{\operatorname{argmax}}\, p(y|x) = \underset{y}{\operatorname{argmax}}\, P(y) \prod_{j=1}^D P(x_j|y)$

Mixture Models

- Model each cluster as a probability distr. (e.g. Gaussian, Poisson): $P(x|\theta_j)$
- Assuming data is iid, likelihood of data is: $P(D|\theta) = \sum_{j=1}^k P(D, \theta_j) = \sum_j P(\theta_j) P(D|\theta_j) = \sum_j w_j P(D|\theta_j) \overset{D\ \text{iid}}{=} \prod_{i=1}^n \sum_j w_j P(x_i|\theta_j) =: L(D; \theta)$
- Choose parameters to minimise neg. log-likelihood: $-\ell(D;\theta) = -\log(L(D;\theta)) = -\log(\prod_i \sum_j w_j P(x_i|\theta_j)) = -\sum_i \log(\sum_j w_j P(x_i|\theta_j))$

Gaussian Mixture Model

$P(x|\theta) = P(x|\mu, \Sigma, w) = \sum_{z=1}^k P(x, z_i = z | \mu, \Sigma, w) = \sum_z P(z_i = z; \theta) P(x_i | z_i = z, \mu, \Sigma, w)$
$= \sum_z w_z \mathcal{N}(x; \mu_z, \Sigma_z)$

Expectation Maximation

Assume $P(X,Y)$ is a Gaussian Bayes classifier (GBC). Assume labels are missing.

Hard-EM: Initialise parameters. Predict most likely class for each data point. Complete data with these labels. Compute MLE for the GBC. Repeat!

Soft-EM: Initialise parameters.

E-step: Calculate cluster membership weights for each point.

$\gamma_{z_i}(x_i) := P(z_i|x_i; \theta^{t-1}) = \frac{P(x_i|z_i; \theta^{t-1}) P(z_i; \theta^{t-1})}{P(x_i; \theta^{t-1})} = \frac{P(x_i|z_i; \theta^{t-1}) P(z_i; \theta^{t-1})}{\sum_{z_i=1}^k P(x_i|z_i; \theta^{t-1}) P(z_i; \theta^{t-1})}$

$= \frac{\mathcal{N}(x_i; \mu_{z_i}^{t-1}, \Sigma_{z_i}^{t-1}) w_{z_i}}{\sum_{z_i=1}^k \mathcal{N}(x_i; \mu_{z_i}, \Sigma_{z_i}) w_{z_i}}$; $\gamma_{z_i}(x_i) = \mathbb{1}[z_i = y_i]$

M-step: Fit cluster to weighted data points

$Q(\theta, \theta^{t-1}) = \mathbb{E}_{p(z_{1:n}|x_{1:n}, \theta^{t-1})}[\log(P(x_{1:n}, z_{1:n}|\theta))] = \mathbb{E}_{z_{1:n}}[\log(P(x_{1:n}, z_{1:n}|\theta)) | x_{1:n}, \theta^{t-1}]$
$\overset{iid}{=} \mathbb{E}[\log(\prod_{i=1}^n P(x_i, z_i|\theta)) | x_{1:n}, \theta^{t-1}] = \sum_{i=1}^n \mathbb{E}_{z_{1:n}}[\log(P(x_i, z_i|\theta)) | x_{1:n}, \theta^{t-1}]$
$\overset{iid}{=} \sum_{i=1}^n \sum_{z_i=1}^k \mathbb{E}_{z_i}[\log(P(x_i, z_i|\theta)) | x_i, \theta^{t-1}] = \sum_i \sum_{z_i} P(z_i|x_i, \theta^{t-1}) \log(P(x_i, z_i|\theta))$
$= \sum_i \sum_{z_i} \gamma_{z_i}(x_i) \log(P(x_i, z_i|\theta))$

$\Rightarrow \theta^t = \operatorname{argmax} Q(\theta; \theta^{t-1})$

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

$\Sigma_j^t \leftarrow \frac{\sum_{i=1}^n \gamma_j^t(x_i)(x_i - \mu_j^t)(x_i - \mu_j^t)^T}{\sum_{i=1}^n \gamma_j^t(x_i)}$ ← transposed

Repeat!

Markov Chain

Note: Data not iid! Next state only depends on current state.

We have a starting state Y_1 with $P(Y_1 = y) = p_y$

And a probability for the next state: $T_{y',y} = P(Y_t = y | Y_{t-1} = y') = \theta_{y|y'}$

Sequence Prediction: Timestep depends on k last steps.

$(Y_{t+1}|y_1, \dots, y_t) = P(Y_{t+1}|y_{t-k+1}, \dots, y_t)$. Can be reduced to iid supervised learning.

Maximum a posteriori estimate (MAP) & MLE

MLE: $\theta^* = \underset{\theta}{\operatorname{argmax}}\, P(y_1, \dots, y_n | x_1, \dots, x_n) = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^n P(y_i|x_i) = \underset{\theta}{\operatorname{argmin}} - \sum_{i=1}^n \log(P(y_i|x_i, \theta))$

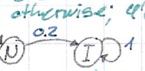
MAP: We add a prior $P(w; b)$: $P(w|x_1, \dots, x_n, y_1, \dots, y_n) = \frac{P(w) P(y_1, \dots, y_n | x_1, \dots, x_n, w)}{P(y_1, \dots, y_n | x_1, \dots, x_n)} \propto P(w) P(y_1, \dots, y_n | x_1, \dots, x_n, w)$

ANN (II) Derivatives of activation functions

- Sigmoid: $\varphi(z) := 1/(1 + \exp(-z))$; $\varphi'(z) = \varphi(z)(1 - \varphi(z))$
- Rectified Linear Unit (ReLU): $\varphi(z) = \begin{cases} z, & z > 0 \\ 0, & \text{otherwise} \end{cases}$; $\varphi'(z) = \begin{cases} 1, & z > 0 \\ 0, & z < 0 \end{cases}$

Norms

- L1-norm: $\|x\|_1 = \sum_{i=1}^n |x_i|$
- L2-norm: $\|x\|_2^2 = \sum_{i=1}^n |x_i|^2$
- Loo-norm: $\|x\|_\infty = \max_i |x_i|$



$\Rightarrow T = \begin{pmatrix} 0.8 & 0.2 \\ 0 & 1 \end{pmatrix}$

Markov Chains (II)

$P(Y_1) = [1, 0]$, $P(Y_t|Y_{t-1}) = T_{t-1,t}$

What is the probability $P(Y_4 = N)$ that Harry isn't infected in the fourth week?

$P(Y_4) = P(Y_1) T^3 = \dots = [0.512\ \ 0.488] \Rightarrow P(Y_4 = N) = 0.512$

Want to show: $P(Y_6, Y_4 | Y_5) = P(Y_6|Y_5) P(Y_4|Y_5)$

Proof: $P(Y_6, Y_4|Y_5) = \frac{P(Y_4, Y_5, Y_6)}{P(Y_5)} = \frac{P(Y_6|Y_4, Y_5) P(Y_4, Y_5)}{P(Y_5)} \overset{\text{Markov property}}{=} \frac{P(Y_6|Y_5) P(Y_4, Y_5)}{P(Y_5)}$
$= \frac{P(Y_6|Y_5) P(Y_4|Y_5) P(Y_5)}{P(Y_5)} = P(Y_6|Y_5) P(Y_4|Y_5)$ □

Categorial Naive Bayes Classifiers

MLE $P(y)$: $\hat{P}(Y=y) = \mathrm{cnt}(Y=y)/n$

MLE $P(X_i = c|y)$: $\hat{P}(X_i = c|y) = \frac{\mathrm{cnt}(X_i = c, Y = y)}{\mathrm{cnt}(Y=y)}$

Predict: $y = \underset{y'}{\operatorname{argmax}}\, \hat{P}(y'|x) = \underset{y'}{\operatorname{argmax}}\, \hat{P}(y') \prod_{i=1}^d \hat{P}(x_i|y') = \underset{y'}{\operatorname{argmax}} \log(\hat{P}(y')) + \sum_{i=1}^d \log(\hat{P}(x_i|y'))$

Probabilities/Statistics (II)

- $f_{X,Y}(x,y) = f_{Y|X}(y|X=x) f_X(x)$
- Bayes (continuous): $f_{X|Y}(x|Y=y) = \frac{f_{X,Y}(x,y)}{f_Y(y)} = \frac{f_{X|Y}(y|X=x) f_X(x)}{f_Y(y)}$
- Multiplikationsregel:
  $P[A \cap B] = P[A,B] = P[B|A]P[A]$
  if iid then: $P[A,B] = P[A]P[B]$
- Poisson distribution: $P(X=x) = \begin{cases} \frac{\lambda^x}{x!} e^{-\lambda}, & \text{if } x \in \{0,1,\dots\} \\ 0, & \text{otherwise} \end{cases}$
- Normal distr.: $X \sim N(\mu, \sigma^2)$
  $\rightsquigarrow f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-1}{2} \frac{(x-\mu)^2}{\sigma^2}\right)$
  $\rightsquigarrow$ Multivariate normal distr.: $X \sim N(\mu, \Sigma)$ $\Sigma \in \mathbb{R}^{k \times k}$
  $f_X(x) = 1/\sqrt{(2\pi)^k \det(\Sigma)} \exp(\frac{-1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu))$
- Cumulative distr. fct. (CDF):
  $\Phi(x) := 1/\sqrt{2\pi} \int_{-\infty}^{x} e^{-t^2/2} dt$
  $\Phi(u; v, w) = \Phi(\frac{u-v}{\sqrt{w}}; 0, 1)$.

K-medians Loss function: $L(\mu) = \sum_{i=1}^{n} \min_{j \in \{1,\dots,k\}} \|x_i - \mu_j\|_1$

- $\mu^* = \operatorname{argmin}_{\mu} \sum_{i=1}^{n} \min_{j \in \{1,\dots,k\}} \|x_i - \mu_j\|_1 + \lambda k$ ← regularisation
  $\mu^{(0)} = [\mu_1^{(0)}, \dots, \mu_k^{(0)}]$
- K-medians Algo: (0) Initialise cluster centers
  (1) "E-step": $z_i = \operatorname{argmin}_{j \in \{1,\dots,k\}} \|x_i - \mu_j\|_1$
  (2) "M-step": $\mu_{j,l} = \text{median}(x_{i,l}, i : z_i = j)$ $\forall j = 1,\dots,k$ $\forall l = 1,\dots,d$
- Derivation of the M-step:
  $\mu = \operatorname{argmin}_{\mu} \sum_{i=1}^{n} \min_{j \in \{1,\dots,k\}} \|x_i - \mu_j\|_1 \overset{(1)}{=} \operatorname{argmin}_{\mu} \sum_{i=1}^{n} \|x_i - \mu_{z_i}\|_1$
  Fix $z_i = j$: $\Rightarrow \mu_j = \operatorname{argmin}_{\mu_j} \sum_{i: z_i = j} \|x_i - \mu_j\|_1$
  $= \operatorname{argmin}_{\mu_j} \sum_{i: z_i = j} \sum_{l=1}^{d} |x_{i,l} - \mu_{j,l}|$

  separate component-wise:
  $\mu_{j,l} = \operatorname{argmin}_{\mu_{j,l}} \sum_{i: z_i = j} |x_{i,l} - \mu_{j,l}|$ (*)
  derive (*) & set to zero:
  $\partial_\mu \left[\sum_{i: z_i = j \wedge x_{il} \le \mu_{jl}} (\mu_{jl} - x_{il}) + \sum_{i: z_i = j \wedge x_{il} > \mu_{jl}} (x_{il} - \mu_{jl}) \right] \overset{!}{=} 0$
  $\Rightarrow |\{i : z_i = j \wedge x_{il} \le \mu_{jl}\}| - |\{i : z_i = j \wedge x_{il} > \mu_{jl}\}| = 0$
  $\Leftrightarrow \mu_{jl} = \text{median}(x_{il}, i : z_i = j)$ //

S2A1(b) Find corresponding feature mappings Φ.

$k(x,y) = a_1 k_1(x,y) + a_2 k_2(x,y) = \langle \sqrt{a_1}\phi_1(x), \sqrt{a_1}\phi_1(y) \rangle + \langle \sqrt{a_2}\phi_2(x), \sqrt{a_2}\phi_2(y) \rangle$
$= \langle (\sqrt{a_1}\phi_1(x), \sqrt{a_2}\phi_2(x))^T, (\sqrt{a_1}\phi_1(y), \sqrt{a_2}\phi_2(y))^T \rangle = \langle \phi(x), \phi(y) \rangle$
where $\phi(x) := (\sqrt{a_1}\phi_1(x), \sqrt{a_2}\phi_2(x))^T$, $\phi: X \to \mathbb{R}^{d_1+d_2}$

c) $k(x,y) = k_i(x,y) k_j(x,y) = \langle \phi_i(x), \phi_i(y) \rangle \langle \phi_j(x), \phi_j(y) \rangle$
$= \sum_{k=1}^{d_i} \sum_{l=1}^{d_j} \phi_i(x)_k \phi_i(y)_k \phi_j(x)_l \phi_j(y)_l = \sum_k \sum_l \phi_i(x)_k \phi_j(x)_l \phi_i(y)_k \phi_j(y)_l$
$= \sum_k \sum_l \Phi(x)_{kl} \Phi(y)_{kl} = \langle \Phi(x), \Phi(y) \rangle$.

Propositions SVD: $X = U \Sigma V^T$ $\Sigma =$ "$\text{diag}(\sigma_1, \dots)$"; $\sigma_i = \sqrt{\lambda_i}$
$X, \Sigma \in \mathbb{R}^{n \times d}$
$U \in \mathbb{R}^{n \times n}$
$V \in \mathbb{R}^{d \times d}$

Rem: The optimisation problem $\operatorname{argmin}_w \|Xw - y\|_2^2$ admits a unique sol. only if $\sigma_i \neq 0$ $\forall i \in \{1,\dots,d\}$. $\Leftrightarrow$ rank$(X) \hat{=}$ full.

Prop: Convex fcts are closed under addition, thus show that each term is convex.

Satz: Sei $f \in C^2(\Omega)$. Dann ist f genau dann konvex (konkav), wenn $f''(x) \ge 0$ (≤ 0) für $\forall x \in \Omega$ gilt.

Satz: (Jensen) Sei $f: (a,b) \to \mathbb{R}$ konvex. Dann gilt für beliebige Pkte $x_1, \dots, x_n \in (a,b)$ & Zahlen $0 \le t_1, \dots, t_n \le 1$ mit $\sum_{i=1}^{n} t_i = 1$ die Ungl.: $f(\sum_{i=1}^{n} t_i x_i) \le \sum_{i=1}^{n} t_i f(x_i)$

Rem.: (Jensen & $\mathbb{E}$)
- φ is convex: $\varphi(\mathbb{E}[X]) \le \mathbb{E}[\varphi(X)]$
- φ is concave: $\varphi(\mathbb{E}[X]) \ge \mathbb{E}[\varphi(X)]$.

Prop.: $X^T X + \lambda \mathbb{1}_d$ has EW bounded from below by $\lambda > 0$
$\Rightarrow X^T X + \lambda \mathbb{1}_d$ is invertible.

Prop.: M pos. semi-def. $\Rightarrow$ $\exists$ EW decompisition $M = V \Lambda V^T$ where V is orthogonal.

Rem.: (about kernel) $y = x^T X^T z = x^T \sum_{i=1}^{n} x_i z_i = \sum_{i=1}^{n} x^T x_i z_i = \sum_{i=1}^{n} k(x, x_i) z_i$.

Rem.: (trace trick) $\|A\|^2 = tr(AA^T) = tr(A^T A)$

Rem.: (trace properties)
- $tr(X^T Y) = tr(XY^T) = tr(Y^T X) = tr(YX^T) = \sum_{ij} X_{ij} Y_{ij}$
- $tr(A) = tr(A^T)$

Example 2015 A7 i) Write down the objective that the M-step is optimising wrt the parameters $\pi, \lambda_1, \lambda_2$.

$L(\pi, \lambda_1, \lambda_2) = \mathbb{E}_{Q(z|x)}[\log(P(x_1, z_1, x_2, z_2, x_3, z_3))] = \mathbb{E}_{Q(z|x)}[\log(\prod_{i=1}^{3} P(x_i, z_i))]$
$= \sum_{i=1}^{3} \mathbb{E}_{Q(z|x)}[\log(P(z_i) P(x_i | z_i))]$
$= \sum_{i=1}^{3} \sum_{z_i=1}^{2} Q(Z = z_i | X = x_i) \log(P(Z = z_i) P(X = x_i | Z = z_i))$
$= \sum_i Q(Z=1|X=x_i) \log(P(Z=1) P(X=x_i|Z=1))$
$+ Q(Z=2|X=x_i) \log(P(Z=2) P(X=x_i|Z=2))$
$= \sum_i Q(Z=1|X=x_i) \log(\pi \lambda_1^{x_i}/x_i! \exp(-\lambda_1)) + Q(Z=2|X=x_i) \log((1-\pi) \lambda_2^{x_i}/x_i! \exp(-\lambda_2))$
$= \sum_i Q(Z=1|X=x_i)(\log(\pi) + x_i \log(\lambda_1) - \log(x_i!) - \lambda_1 \log(e))$
$+ Q(Z=2|X=x_i)(\log(1-\pi) + x_i \log(\lambda_2) - \log(x_i!) - \lambda_2$ //

ii) Perform the M-step update and compute the new parameters $\pi, \lambda_1, \lambda_2$.
$\rightsquigarrow$ derive along parameters, set to zero, plug in measured data...

iii) Kate would also like to experiment with different mixture components, but she doesn't know how to compare the resulting models. What strategy would you suggest? How should we choose a model?
Compare log-likelihood of the models on the test set.

Example 2016 A7 We want to maximise $\log(P(x|\theta))$ but this is hard, we instead maximise $L^{(k)}(\theta) = \mathbb{E}_{p^{(k)}}[\log(P(x, z|\theta))]$ where $P^{(k)}(z) = P(z|x, \theta^{(k)})$.

$\Rightarrow P(\alpha, \beta, z_r, z_g, z_b | \theta) = \underbrace{P(\alpha, \beta | z_r, z_g, z_b, \theta)}_{\text{doesn't depend on } \theta} P(z_r, z_g, z_b | \theta)$

Therefore we have only to maximise $P(z_r, z_g, z_b | \theta)$:

i) $L^{(k)}(\theta) = \mathbb{E}_{p(k)}[\log(P(z_r, z_g, z_b|\theta))] = \mathbb{E}_{p(k)}[\log(\frac{1}{c} 0.5^{z_r} \theta^{z_g} (\frac{1}{2} - \theta)^{z_b})]$
$= \mathbb{E}_{p(k)}[\log(c) - z_r \log(2) + z_g \log(\theta) + z_b \log(\frac{1}{2} - \theta)]$, $z_b = n - \alpha = n - z_r - z_g$
$= \mathbb{E}_{p(k)}[-\log(c) - z_r \log(2) + z_g \log(\theta) + (n - z_r - z_g) \log(\frac{1}{2} - \theta)]$
$= -\log(c) + \underbrace{\mathbb{E}_{p(k)}[z_r]}_{=\xi_r^{(k)}}(-\log(2) - \log(\frac{1}{2} - \theta)) + \underbrace{\mathbb{E}_{p(k)}[z_g]}_{\xi_g^{(k)}}(\log(\theta) - \log(\frac{1}{2} - \theta)) + n\log(\frac{1}{2} - \theta)$

ii) Compute $\theta^{(k+1)}$ by maximising L^k wrt θ. (M-step)
$\partial_\theta L^k(\theta) = \xi_r^k / (\frac{1}{2} - \theta) + \xi_g^k / \theta + \xi_g^k / (\frac{1}{2} - \theta) - n / (\frac{1}{2} - \theta) \overset{!}{=} 0 \Leftrightarrow \dots \Leftrightarrow \hat{\theta} = \frac{\xi_g^k}{2(n - \xi_r^k)}$

iii) Compute ξ_g^k by noting that x_g follows a binomial distribution. (E-step)
[Hint: The mean of a bin. distr. with m trials & success probability p is pm.]

$z_g | \alpha, \beta, \theta = z_g | \alpha, \theta$ independent from $x_b = \beta$ | $z_g | \alpha, \theta$ follows a bin. distr. with $\alpha \hat{=} n_{bin} = $ # trials, and $p_{bin} = p_g / (p_g + p_r)$

$\Rightarrow \xi_g^k = n_{bin} p_{bin} = \frac{\alpha \theta^k}{\theta^k + 0.5}$ //