Introduction to Machine Learning SS20
Let's learn some CBB

Basics

Fundamental Assumption

Data is iid for unknown P: $(x_i, y_i) \sim P(X,Y)$

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: $\hat{R}_D(w) = \frac{1}{|D|}\sum_{(x,y)\in D}(y - w^T x)^2$

Standardization

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

Cross-Validation

For all models m, for all $i \in \{1,\dots,k\}$ do:
1. Split data: $D = D_{train}^{(i)} \uplus D_{test}^{(i)}$ (Monte-Carlo or k-Fold)
2. Train model: $\hat{w}_{i,m} = \operatorname*{argmin}_w \hat{R}_{train}^{(i)}(w)$
3. Estimate error: $\hat{R}_m^{(i)} = \hat{R}_{test}^{(i)}(\hat{w}_{i,m})$

Select best model: $\hat{m} = \operatorname*{argmin}_m \frac{1}{k}\sum_{i=1}^k \hat{R}_m^{(i)}$

Gradient Descent

1. Pick arbitrary $w_0 \in \mathbb{R}^d$
2. $w_{t+1} = w_t - \eta_t \nabla \hat{R}(w_t)$

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 $(x', y') \in D$

Regression

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

Linear Regression

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

Ridge regression

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

L1-regularized regression (Lasso)

$\hat{R}(w) = \sum_{i=1}^n (y_i - w^T x_i)^2 + \lambda ||w||_1$

Classification

Solve $w^* = \operatorname*{argmin}_w l(w; x_i, y_i)$; 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
$\nabla_w l_P(w; y_i, x_i) = \begin{cases} 0 & \text{if } y_i w^T x_i \geq 0 \\ -y_i x_i & \text{otherwise} \end{cases}$
Data lin. separable ⇔ obtains a lin. separator (not necessarily optimal)

Support Vector Machine (SVM)

Hinge loss: $l_H(w; x_i, y_i) = \max(0, 1 - y_i w^T x_i)$
$\nabla_w l_H(w; y, x) = \begin{cases} 0 & \text{if } y_i w^T x_i \geq 1 \\ -y_i x_i & \text{otherwise} \end{cases}$
$w^* = \operatorname*{argmin}_w l_H(w; x_i, y_i) + \lambda ||w||_2^2$

Kernels

efficient, implicit inner products

Properties of kernel

$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, k must be some inner product (symmetric, positive-definite, linear) for some space $\mathcal{V}$. i.e. $k(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{V}} \overset{Eucl.}{=} \varphi(\mathbf{x})^T \varphi(\mathbf{x}')$ and $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$

Kernel matrix

$$K = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \dots & k(x_n, x_n) \end{bmatrix}$$

Positive semi-definite matrices ⇔ kernels k

Important kernels

Linear: $k(x, y) = x^T y$
Polynomial: $k(x, y) = (x^T y + 1)^d$
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: $w^* \in \text{span}(X) \Rightarrow w = \sum_{j=1}^n \alpha_j y_j x_j$
$\alpha^* = \operatorname*{argmin}_{\alpha \in \mathbb{R}^n} \sum_{i=1}^n \max(0, -\sum_{j=1}^n \alpha_j y_i y_j x_i^T x_j)$

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 $||w||_2^2$
$k_i = [y_1 k(x_i, x_1), \dots, y_n k(x_i, x_n)]$, $D_y = \text{diag}(y)$
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: $w^* = \sum_{i=1}^n \alpha_i x$
$\alpha^* = \operatorname*{argmin}_\alpha ||\alpha^T K - y||_2^2 + \lambda \alpha^T K \alpha$
$= (K + \lambda I)^{-1} y$
Prediction: $\hat{y} = \sum_{i=1}^n \alpha_i k(x_i, \hat{x})$

k-NN

$y = \text{sign}(\sum_{i=1}^n y_i [x_i$ among k nearest neighbours of $x])$ – No weights ⇒ no training!
But depends on all data :(

Imbalance

up-/downsampling

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 score: $\frac{2TP}{2TP + FP + FN} = \frac{2}{\frac{1}{prec} + \frac{1}{rec}}$
ROC Curve: $y = $TPR, $x = $FPR

Multi-class

one-vs-all (c), one-vs-one ($\frac{c(c-1)}{2}$), encoding

Multi-class Hinge loss

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

Neural networks

Parameterize feature map with θ: $\phi(x, \theta) = \varphi(\theta^T x) = \varphi(z)$ (activation function φ)
$\Rightarrow w^* = \operatorname*{argmin}_{w, \theta} \sum_{i=1}^n l(y_i; \sum_{j=1}^m w_j \phi(x_i, \theta_j))$
$f(x; w, \theta_{1:d}) = \sum_{j=1}^m w_j \varphi(\theta_j^T x) = w^T \varphi(\Theta x)$

Activation functions

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

Predict: forward propagation

$v^{(0)} = x$; for $l = 1, \dots, L-1$:
$v^{(l)} = \varphi(z^{(l)})$, $z^{(l)} = W^{(l)} v^{(l-1)}$
$f = W^{(L)} v^{(L-1)}$
Predict f for regression, $\text{sign}(f)$ for class.

Compute gradient: backpropagation

Output layer: $\delta_j = l_j'(f_j)$, $\frac{\partial}{\partial w_{j,i}} = \delta_j v_i$
Hidden layer $l = L-1, \dots, 1$:
$\delta_j = \varphi'(z_j) \cdot \sum_{i \in Layer_{l+1}} w_{i,j} \delta_i$, $\frac{\partial}{\partial w_{j,i}} = \delta_j v_i$

Learning with momentum

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

Clustering

k-mean

$\hat{R}(\mu) = \sum_{i=1}^n \min_{j \in \{1, \dots k\}} ||x_i - \mu_j||_2^2$
$\hat{\mu} = \operatorname*{argmin}_\mu \hat{R}(\mu)$ ...non-convex, NP-hard
Algorithm (Lloyd's heuristic): Choose starting centers, assign points to closest center, update centers to mean of each cluster, repeat

Dimension reduction

PCA

$D = x_1,\dots,x_n \subset \mathbb{R}^d$, $\Sigma = \frac{1}{n}\sum_{i=1}^n x_i x_i^T$, $\mu = 0$
$(W, z_1,\dots,z_n) = \text{argmin} \sum_{i=1}^n ||Wz_i - x_i||_2^2$,
$W = (v_1|\dots|v_k) \in \mathbb{R}^{d\times k}$, orthogonal; $z_i = W^T x_i$
v_i are the eigen vectors of Σ

Kernel PCA

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

Autoencoders

Find identity function: $x \approx f(x;\theta)$
$f(x;\theta) = f_{decode}(f_{encode}(x;\theta_{encode});\theta_{decode})$

Probability modeling

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

For least squares regression

Best 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)

$\theta^* = \underset{\theta}{\text{argmax}}\, \hat{P}(y_1,\dots,y_n|x_1,\dots,x_n,\theta)$
E.g. lin. + Gauss: $y_i = w^T x_i + \varepsilon_i, \varepsilon_i \sim \mathcal{N}(0,\sigma^2)$
i.e. $y_i \sim \mathcal{N}(w^T x_i, \sigma^2)$, With MLE (use argmin $-\log$): $w^* = \underset{w}{\text{argmin}} \sum (y_i - w^T x_i)^2$

Bias/Variance/Noise

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

Maximum a posteriori estimate (MAP)

Assume bias on parameters, e.g. $w_i \in \mathcal{N}(0,\beta^2)$
Bay.: $P(w|x,y) = \frac{P(w|x)P(y|x,w)}{P(y|x)} = \frac{P(w)P(y|x,w)}{P(y|x)}$

Logistic regression

Link func.: $\sigma(w^T x) = \frac{1}{1+exp(-w^T x)}$ (Sigmoid)
$P(y|x,w) = Ber(y;\sigma(w^T x)) = \frac{1}{1+exp(-yw^T x)}$

Classification: Use $P(y|x,w)$, predict most likely class label.
MLE: $\underset{w}{\text{argmax}}\, P(y_{1:n}|w,x_{1:n})$
$\Rightarrow w^* = \underset{w}{\text{argmin}} \sum_{i=1}^n log(1 + exp(-y_i w^T x_i))$
SGD update: $w = w + \eta_t y x \hat{P}(Y=-y|w,x)$
$\hat{P}(Y=-y|w,x) = \frac{1}{1+exp(yw^T x)}$
MAP: Gauss. prior $\Rightarrow ||w||_2^2$, Lap. p. $\Rightarrow ||w||_1$
SGD: $w = w(1-2\lambda\eta_t) + \eta_t y x \hat{P}(Y=-y|w,x)$

Bayesian decision theory

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

$a^* = \underset{a\in\mathcal{A}}{\text{argmin}}\, \mathbb{E}[C(y,a)|x]$
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 / generative modeling

Discr. estimate $P(y|x)$, generative $P(y,x)$
Approach (generative): $P(x,y) = P(x|y)\cdot P(y)$ - Estimate prior on labels $P(y)$
- Estimate cond. distr. $P(x|y)$ for each class y
- Pred. using Bayes: $P(y|x) = \frac{P(y)P(x|y)}{P(x)}$

$P(x) = \sum_y P(x,y)$

Examples

MLE for $P(y) = p = \frac{n_+}{n}$
MLE for $P(x_i|y) = \mathcal{N}(x_i; \mu_{i,y}, \sigma_{i,y}^2)$:
$\hat{\mu}_{i,y} = \frac{1}{n_y}\sum_{x\in D_{x_i|y}} x$
$\hat{\sigma}_{i,y}^2 = \frac{1}{n_y}\sum_{x\in D_{x_i|y}} (x - \hat{\mu}_{i,y})^2$
MLE for Poi.: $\lambda = \text{avg}(x_i)$
$\mathbb{R}^d$: $P(X=x|Y=y) = \prod_{i=1}^d Pois(\lambda_y^{(i)}, x^{(i)})$

Deriving decision rule

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

Gaussian Bayes Classifier

$\hat{P}(x|y) = \mathcal{N}(x; \hat{\mu}_y, \hat{\Sigma}_y)$
$\hat{P}(Y=y) = \hat{p}_y = \frac{n_y}{n}$
$\hat{\mu}_y = \frac{1}{n_y}\sum_{i:y_i=y} x_i \in \mathbb{R}^d$
$\hat{\Sigma}_y = \frac{1}{n_y}\sum_{i:y_i=y} (x_i - \hat{\mu}_y)(x_i - \hat{\mu}_y)^T \in \mathbb{R}^{d\times d}$

Fisher's lin. discrim. analysis (LDA, c=2)

Assume: $p = 0.5$; $\hat{\Sigma}_- = \hat{\Sigma}_+ = \hat{\Sigma}$
discriminant function: $f(x) = \log\frac{p}{1-p} + \frac{1}{2}[\log\frac{|\hat{\Sigma}_-|}{|\hat{\Sigma}_+|} + \left((x-\hat{\mu}_-)^T \hat{\Sigma}_-^{-1}(x-\hat{\mu}_-)\right) - \left((x-\hat{\mu}_+)^T \hat{\Sigma}_+^{-1}(x-\hat{\mu}_+)\right)]$
Predict: $y = \text{sign}(f(x)) = \text{sign}(w^T x + w_0)$
$w = \hat{\Sigma}^{-1}(\hat{\mu}_+ - \hat{\mu}_-)$;
$w_0 = \frac{1}{2}(\hat{\mu}_-^T \hat{\Sigma}^{-1}\hat{\mu}_- - \hat{\mu}_+^T \hat{\Sigma}^{-1}\hat{\mu}_+)$

Outlier Detection

$P(x) \leq \tau$

Categorical Naive Bayes Classifier

MLE for feature distr.: $\hat{P}(X_i = c|Y=y) = \theta_{c|y}^{(i)}$
$\theta_{c|y}^{(i)} = \frac{Count(X_i=c, Y=y)}{Count(Y=y)}$
Prediction: $y^* = \underset{y}{argmax}\, \hat{P}(y|x)$

Missing data

Mixture modeling

Model each c. as probability distr. $P(x|\theta_j)$
$P(D|\theta) = \prod_{i=1}^n \sum_{j=1}^k w_j P(x_i|\theta_j)$
$L(w,\theta) = -\sum_{i=1}^n \log \sum_{j=1}^k w_j P(x_i|\theta_j)$

Gaussian-Mixture Bayes classifiers

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

Hard-EM algorithm

Initialize parameters $\theta^{(0)}$
E-step: Predict most likely class for each point: $z_i^{(t)} = \underset{z}{\text{argmax}}\, P(z|x_i,\theta^{(t-1)})$
$= \underset{z}{\text{argmax}}\, P(z|\theta^{(t-1)})P(x_i|z,\theta^{(t-1)})$;
M-step: Compute the MLE: $\theta^{(t)} = \underset{\theta}{\text{argmax}} P(D^{(t)}|\theta)$, i.e. $\mu_j^{(t)} = \frac{1}{n_j}\sum_{i:z_i=j} x_j$

Soft-EM algorithm

E-step: Calc p for each point and cls.: $\gamma_j^{(t)}(x_i)$
M-step: Fit clusters to weighted data points:
$w_j^{(t)} = \frac{1}{n}\sum_{i=1}^n \gamma_j^{(t)}(x_i)$; $\mu_j^{(t)} = \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i) x_i}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)}$
$\sigma_j^{(t)} = \frac{\sum_{i=1}^n \gamma_j^{(t)}(x_i)(x_i - \mu_j^{(t)})^T(x_i - \mu_j^{(t)})}{\sum_{i=1}^n \gamma_j^{(t)}(x_i)}$

Soft-EM for semi-supervised learning

labeled 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

Probabilities

$$\mathbb{E}_x[X] = \begin{cases} \int x\cdot p(x)\partial x & \text{if continuous} \\ \sum_x x\cdot p(x) & \text{otherwise} \end{cases}$$
$\text{Var}[X] = \mathbb{E}[(X-\mu_X)^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$
$P(A|B) = \frac{P(B|A)\cdot P(A)}{P(B)}$; $p(Z|X,\theta) = \frac{p(X,Z|\theta)}{p(X|\theta)}$
$P(x,y) = P(y|x)\cdot P(x) = P(x|y)\cdot P(y)$
$\mathbb{E}_x[b + cX] = b + c\cdot\mathbb{E}_x[X]$
$\mathbb{E}_x[b + CX] = b + C\cdot\mathbb{E}_x[X], C \in \mathbb{R}^{n\times n}$
$\mathbb{V}_x[b + cX] = c^2\mathbb{V}_x[X]$
$\mathbb{V}_x[b + CX] = C\cdot\mathbb{V}_x[X]C^\top, C \in \mathbb{R}^{n\times n}$
$\text{Cov}[X,Y] = \mathbb{E}[(X - \mathbb{E}(X)(Y - \mathbb{E}(Y))] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$
$\mathbb{V}[X+Y] = \mathbb{V}[X] + \mathbb{V}[Y] + 2\text{Cov}[X,Y]$

Bayes Rule

$P(A|B) = \frac{P(B|A)\cdot P(A)}{P(B)}$

P-Norm

$||x||_p = (\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}}$, $1 \leq p < \infty$

Some gradients

$\nabla_x ||x||_2^2 = 2x$

$f(x) = x^T Ax$; $\nabla_x f(x) = (A + A^T)x$

E.g. $\nabla_w \log(1+\exp(\text{-yw}^\text{T}\text{x})) =$

$\frac{1}{1+\exp(-yw^T x)} \cdot \exp(-yw^T x) \cdot (-yx) =$

$\frac{1}{1+\exp(yw^T x)} \cdot (-yx)$

Convex / Jensen's inequality

g(x) convex $\Leftrightarrow g''(x) > 0 \Leftrightarrow x_1, x_2 \in \mathbb{R}, \lambda \in [0, 1] : g(\lambda x_1 + (1-\lambda)x_2) \leq \lambda g(x_1) + (1-\lambda)g(x_2)$

Positive semi-definite matrices

$M \in \mathbb{R}^{n \times n}$ is *p.s.d* $\Leftrightarrow$

$\forall x \backslash \{0\} \in \mathbb{R}^n : x^T Mx \geq 0 \Leftrightarrow$

all eigenvalues of M are positive: $\lambda_i \geq 0$