Basics

Fundamental Assumption

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

True risk and estimated error

True risk: $R(w) = \int P(x,y)(y-w^Tx)^2 \partial x \partial y = \mathbb{E}_{x,y}[(y-w^Tx)^2]$

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

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 $i \in \{1,...,k\}$ do:

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

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

Parametric vs. Nonparametric models

Parametric: have finite set of parameters. e.g. linear regression, linear perceptron

Nonparametric: grow in complexity with the size of the data, more expressive. e.g. k-NN

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

Linear Regression

$\hat{R}(w) = \sum_{i=1}^n (y_i - w^Tx_i)^2 = ||Xw-y||_2^2$

$\nabla_w \hat{R}(w) = -2\sum_{i=1}^n (y_i - w^Tx_i)\cdot x_i$

$w^* = (X^TX)^{-1}X^Ty$

Ridge regression

$\hat{R}(w) = \sum_{i=1}^n (y_i - w^Tx_i)^2 + \lambda||w||_2^2$

$\nabla_w \hat{R}(w) = -2\sum_{i=1}^n (y_i - w^Tx_i)\cdot x_i + 2\lambda w$

$w^* = (X^TX + \lambda I)^{-1}X^Ty$

L1-regularized regression (Lasso)

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

Classification

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

Loss functions

- $l_{0/1}(y,x) = 1$ if $y \neq \operatorname{sign}(w^Tx)$ else 0
- $l_{\text{hinge}}(z) = \max(0, 1-z)$
- $l_{\text{squared}}(z) = (1-z)^2$
- $l_{\text{logistic}}(z) = \log(1+e^{-z})$
- $l_{\exp}(z) = e^{-z}$

Perceptron algorithm

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

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

Data lin. separable $\Leftrightarrow$ obtains a lin. separator (not necessarily optimal)

Support Vector Machine (SVM)

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

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

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

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 $\Leftrightarrow$ kernels k

Important kernels

Linear: $k(x,y) = x^Ty$

Polynomial: $k(x,y) = (x^Ty+1)^d$

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: $w^* \in \operatorname{span}(X) \Rightarrow w = \sum_{j=1}^n \alpha_j y_j x_j$

$\alpha^* = \underset{\alpha \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^n \max(0, -\sum_{j=1}^n \alpha_j y_i y_j x_i^T x_j)$

Kernelized perceptron and SVM

Use α^Tk_i instead of w^Tx_i,

use $\alpha^T D_y K D_y \alpha$ instead of $||w||_2^2$

$k_i = [y_1k(x_i,x_1), ..., y_nk(x_i,x_n)]$, $D_y = \operatorname{diag}(y)$

Prediction: $\hat{y} = \operatorname{sign}(\sum_{i=1}^n \alpha_i y_i k(x_i, \hat{x}))$

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: $w^* = \sum_{i=1}^n \alpha_i x$

$\alpha^* = \underset{\alpha}{\operatorname{argmin}} ||\alpha^TK - y||_2^2 + \lambda\alpha^TK\alpha$

$= (K+\lambda I)^{-1}y$

Prediction: $\hat{y} = \sum_{i=1}^n \alpha_i k(x_i, \hat{x})$

k-NN

$y = \operatorname{sign}\left(\sum_{i=1}^n y_i[x_i \text{ among } k \text{ nearest neighbours of } x]\right)$ – No weights $\Rightarrow$ 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

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

Neural networks

Parameterize feature map with θ: $\phi(x,\theta) = \varphi(\theta^Tx) = \varphi(z)$ (activation function φ)

$\Rightarrow w^* = \underset{w,\theta}{\operatorname{argmin}} \sum_{i=1}^n l(y_i; \sum_{j=1}^m w_j \phi(x_i, \theta_j))$

$f(x; w, \theta_{1:d}) = \sum_{j=1}^m w_j \varphi(\theta_j^T x) = w^T\varphi(\Theta x)$

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 $l = 1, ..., L-1$:

$v^{(l)} = \varphi(z^{(l)})$, $z^{(l)} = W^{(l)}v^{(l-1)}$

$f = W^{(L)}v^{(L-1)}$

Predict f for regression, $\operatorname{sign}(f)$ 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 $l = L-1, ..., 1$:

$\delta_j = \varphi'(z_j)\cdot\sum_{i \in Layer_{l+1}} w_{i,j}\delta_i$, $\frac{\partial}{\partial w_{j,i}} = \delta_j v_i$

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

$\hat{R}(\mu) = \sum_{i=1}^n \min_{j \in \{1,...k\}} ||x_i - \mu_j||_2^2$

$\hat{\mu} = \underset{\mu}{\operatorname{argmin}}\ \hat{R}(\mu)$ ...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

$D = x_1, ..., x_n \subset \mathbb{R}^d$, $\Sigma = \frac{1}{n}\sum_{i=1}^n x_i x_i^T$, $\mu = 0$

$(W, z_1, ..., z_n) = \operatorname{argmin} \sum_{i=1}^n ||Wz_i - x_i||_2^2$,

$W = (v_1|...|v_k) \in \mathbb{R}^{d\times k}$, orthogonal; $z_i = W^Tx_i$

v_i are the eigen vectors of Σ

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: $x \approx f(x;\theta)$

$f(x;\theta)=f_{decode}(f_{encode}(x;\theta_{encode});\theta_{decode})$

Probability modeling

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

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)

$\theta^*=\underset{\theta}{\text{argmax}}\ \hat{P}(y_1,...,y_n|x_1,...,x_n,\theta)$

E.g. lin. + Gauss: $y_i=w^Tx_i+\varepsilon_i,\varepsilon_i\sim\mathcal{N}(0,\sigma^2)$ i.e. $y_i\sim\mathcal{N}(w^Tx_i,\sigma^2)$, With MLE (use argmin $-\log$): $w^*=\underset{w}{\text{argmin}}\sum(y_i-w^Tx_i)^2$

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.: $\sigma(w^Tx)=\frac{1}{1+exp(-w^Tx)}$ (Sigmoid)

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

Classification: Use $P(y|x,w)$, predict most likely class label.

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

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

SGD update: $w=w+\eta_t yx\hat{P}(Y=-y|w,x)$

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

MAP: Gauss. prior $\Rightarrow ||w||_2^2$, Lap. p. $\Rightarrow ||w||_1$

SGD: $w=w(1-2\lambda\eta_t)+\eta_t yx\hat{P}(Y=-y|w,x)$

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

$\hat{\mu}_{i,y}=\frac{1}{n_y}\sum_{x\in D_{x_i|y}}x$

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

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

$\hat{P}(x|y)=\mathcal{N}(x;\hat{\mu}_y,\hat{\Sigma}_y)$

$\hat{P}(Y=y)=\hat{p}_y=\frac{n_y}{n}$

$\hat{\mu}_y=\frac{1}{n_y}\sum_{i:y_i=y}x_i\in\mathbb{R}^d$

$\hat{\Sigma}_y=\frac{1}{n_y}\sum_{i:y_i=y}(x_i-\hat{\mu}_y)(x_i-\hat{\mu}_y)^T\in\mathbb{R}^{d\times d}$

Outlier Detection

$P(x)\le\tau$

Categorical Naive Bayes Classifier

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

$\theta^{(i)}_{c|y}=\frac{Count(X_i=c,Y=y)}{Count(Y=y)}$

Prediction: $y^*=\underset{y}{argmax}\hat{P}(y|x)$

Missing data

Mixture modeling

Model each c. as probability distr. $P(x|\theta_j)$

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

$L(w,\theta)=-\sum_{i=1}^n\log\sum_{j=1}^k w_jP(x_i|\theta_j)$

Gaussian-Mixture Bayes classifiers

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

EM Algorithm - Theory

E-Step (compute where belongs): $Q(\theta;\theta^{(t-1)})=\mathbb{E}_{z_{1:n}}\left[\log P(x_{1:n},z_{1:n}\,|\,\theta)\,|\,x_{1:n},\theta^{(t-1)}\right]$ M-Step (find best model params): $\theta^{(t)}=\text{argmax}_\theta Q(\theta;\theta^{(t-1)})$

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. $\mu_j^{(t)}=\frac{1}{n_j}\sum_{i:z_i=j}x_i$

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

$$\mathbb{E}_x[X]=\begin{cases}\int x\cdot p(x)\partial x & \text{if continuous} \\ \sum_x x\cdot p(x) & \text{otherwise}\end{cases}$$

$\text{Var}[X]=\mathbb{E}[(X-\mu_X)^2]=\mathbb{E}[X^2]-\mathbb{E}[X]^2$

$P(A|B)=\frac{P(B|A)\cdot P(A)}{P(B)}$; $p(Z|X,\theta)=\frac{p(X,Z|\theta)}{p(X|\theta)}$

$P(x,y)=P(y|x)\cdot P(x)=P(x|y)\cdot P(y)$

Bayes Rule

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

P-Norm

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

Some gradients

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

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

E.g. $\nabla_w \log(1+\exp(-yw^Tx))=\frac{1}{1+\exp(-yw^Tx)}\cdot\exp(-yw^Tx)\cdot(-yx)=\frac{1}{1+\exp(yw^Tx)}\cdot(-yx)$

Convex / Jensen's inequality

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

Gaussian / Normal Distribution

$f(x)=\frac{1}{\sqrt{2\pi\sigma^2}}exp(-\frac{(x-\mu)^2}{2\sigma^2})$

Multivariate Gaussian

$\Sigma=$ covariance matrix, μ = mean

$f(x)=\frac{1}{2\pi\sqrt{|\Sigma|}}e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$

Empirical: $\hat{\Sigma}=\frac{1}{n}\sum_{i=1}^n x_ix_i^T$ (needs centered data points)

Positive semi-definite matrices

$M\in\mathbb{R}^{n\times n}$ is psd $\Leftrightarrow$

$\forall x\in\mathbb{R}^n: x^TMx\ge 0\Leftrightarrow$

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