Regression

Linear Regression

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

$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^n (y_i - w^T x_i)^2$

Closed form: $w^* = (X^T X)^{-1} X^T y$

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

Convex / Jensen's inequality

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

$g(\lambda x_1 + (1-\lambda)x_2) \leq \lambda g(x_1) + (1-\lambda) g(x_2)$

Gradient Descent

1. Start arbitrary $w_o \in \mathbb{R}$
2. For $t = 1, 2, \ldots$ do $w_{t+1} = w_t - \eta_t \nabla \hat{R}(w_t)$

Expected Error (True Risk)

Assumption: data set generated iid: $R(w) = \int P(x,y)(y - w^T x)^2 \partial x \partial y = \mathbb{E}_{x,y}[(y - w^T x)^2]$

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

Gaussian/Normal Distribution

σ = standard deviation, σ^2 = var., μ = mean:

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

L2-reg: Ridge Regression

Regularization: $\min_w \sum_{i=1}^n (y_i - w^T x_i)^2 + \lambda \|w\|_2^2$

Closed form solution: $w^* = (X^T X + \lambda I)^{-1} X^T y$

$(X^T X + \lambda I)$ always invertible.

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

Standardization

Goal: each feature: $\mu = 0$, unit σ^2: $\tilde{x}_{i,j} = \frac{(x_{i,j} - \hat{\mu}_j)}{\hat{\sigma}_j}$

$\hat{\mu}_j = \frac{1}{n} \sum_{i=1}^n x_{i,j}$, $\hat{\sigma}_j^2 = \frac{1}{n} \sum_{i=1}^n (x_{i,j} - \hat{\mu}_j)^2$

Classification

0/1 loss - "NP-Hard"

0/1 loss is not convex and not differentiable.

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

Perceptron loss

Perceptron loss is convex and not differentiable, but gradient is informative.

$l_P(w; y_i, x_i) = max\{0, -y_i w^T x_i\}$

$\nabla_w l_p(w; y_i, x_i) = \begin{cases} 0 & \text{, if } -y_i w^T x_i \leq 0 \\ -y_i x_i & \text{, if } -y_i w^T x_i > 0 \end{cases}$

$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^n l_p(w; y_i, x_i)$

Stochastic Gradient Descent (SGD)

1. Start at an arbitrary $w_0 \in \mathbb{R}^d$
2. For $t = 1, 2, \ldots$ do:

Pick data point $(x', y') \in_{u.a.r.} D$

$w_{t+1} = w_t - \eta_t \nabla_w l(w_t; x', y')$

Perceptron Algo: SGD with Perceptron loss

Support Vector Machine - "Max Margin"

Hinge loss: $l_H(w; x, y) = max\{0, 1 - y w^T x\}$

Goal: max. the margin around the separator.

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

$g_i(w) = max\{0, 1 - y_i w^T x_i\} + \lambda \|w\|_2^2$

$\nabla_w g_i(w) = \begin{cases} -y_i x_i + 2\lambda w & \text{, if } y_i w^T x_i < 1 \\ 2\lambda w & \text{, if } y_i w^T x_i \geq 1 \end{cases}$

L1-SVM

$\min_w \lambda \|w\|_1 + \sum_{i=1}^n max(0, 1 - y_i w^T x_i) \rightarrow$ enourages coefficients to be zero (only linear models).

Kernels = Non-Parametric

Reformulating the perceptron

Ansatz: $w = \sum_{j=1}^n \alpha_j y_j x_j$

$\min_{w \in \mathbb{R}^d} \sum_{i=1}^n \max[0, -y_i w^T x_i]$

$= \min_{\alpha_{1:n}} \sum_{i=1}^n \max[0, -\sum_{j=1}^n \alpha_j y_i y_j x_i^T x_j]$

Kernelized Perceptron

1. Initialize $\alpha_1 = \ldots = \alpha_n = 0$
2. For $t = 1, 2, \ldots$ do

Pick data $(x_i, y_i) \in_{u.a.r} D$

Predict $\hat{y} = sign(\sum_{j=1}^n \alpha_j y_j k(x_j, x_i))$

If $\hat{y} \neq y_i$ set $\alpha_i = \alpha_i + \eta_t$

Predict new point x: $\hat{y} = sign(\sum_{j=1}^n \alpha_j y_j k(x_j, x))$

Properties of a Kernel

k must be a function: $f : X \times X \rightarrow R$

k must be symmetric: $k(x,y) = k(y,x)$

Matrix K must be positive semi-definite (psd).

Kernel Matrix K

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

positive semi-definite matrices $\Leftrightarrow$ kernels

Definition of PSD

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

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

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

Nearest Neighbor k-NN

$y = sign(\sum_{i=1}^n y_i [x_i \text{ among k nn of } x])$

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

Linear kernel: $k(x,y) = x^T y$

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

Gaussian kernel: $k(x,y) = exp(-\|x-y\|_2^2 / h^2)$

Laplacian kernel: $k(x,y) = exp(-\|x-y\|_1 / h)$

h = bandwidth $\approx 1\sigma$

Kernel Properties / Rules

$k_1(x,y) + k_2(x,y)$; $k_1(x,y) \cdot k_2(x,y)$; $c \cdot k_1(x,y), c > 0$; $f(k_1(x,y))$, where f is a polynomial with pos. coeffs. or the exponential function

Perceptron and SVM

Perceptron: $\min_\alpha \sum_{i=1}^n max\{0, -y_i \alpha^T k_i\}$

SVM: $k_i = [y_1 k(x_i, x_1), \ldots, y_n k(x_i, x_n)]$:

$\min_\alpha \sum_{i=1}^n max\{0, 1 - y_i \alpha^T k_i\} + \lambda \alpha^T D_y K D_y \alpha$

Prediction: $y = sign(\sum_{j=1}^n \alpha_j y_j k(x_j, x))$

Kernelized linear regression

Ansatz: $w^* = \sum_i \alpha_i x$

Parametric: $w^* = \underset{w}{\operatorname{argmin}} \sum_i (w^T x_i - y_i)^2 + \lambda \|w\|_2^2$

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

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

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

Imbalance

Cost Sensitive Classification

Replace loss by: $l_{CS}(w; x, y) = c_y l(w; x, y)$

Metrics

Accuracy: $\frac{TP+TN}{TP+TN+FP+FN}$, Precision: $\frac{TP}{TP+FP}$

TPR = Recall: $\frac{TP}{TP+FN}$, F1 score: $\frac{2TP}{2TP+FP+FN}$

FPR = $\frac{FP}{TN+FP}$

Multi-class Hinge Loss

One vs. One | One vs. All | Maintain $w^{(1)}, .., w^{(c)}$

Neural Networks

Learning features

Parameterize the feature maps and optimize over the parameters:

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

One possibility: $\phi(x, \theta) = \varphi(\theta^T x) = \varphi(z)$

Activation functions

Sigmoid: $\varphi(z) = \frac{1}{1 + exp(-z)}$

Tanh: $\varphi(z) = tanh(z) = \frac{exp(z) - exp(-z)}{exp(z) + exp(-z)}$

ReLu: $\varphi(z) = max(z, 0)$

Forward propagation

For each unit j on input layer, set value $v_j = x_j$

For each layer $l = 1 : L-1$: For each unit j on layer l set its value $v_j = \varphi(\sum_{i \in Layer_{l-1}} w_{j,i} v_i)$

For each unit j on output layer, set its value $f_j = \sum_{i \in Layer_{L-1}} w_{j,i} v_i$

Predict $y_j = f_j$ for reg. / $y_j = sign(f_j)$ for class.

Backpropagation

For each unit j on the output layer:

- Compute error signal: $\delta_j = \ell_j'(f_j)$
- For each unit i on layer L: $\frac{\partial}{\partial w_{j,i}} = \delta_j v_i$

For each unit j on hidden layer $l = \{L-1, \ldots, 1\}$:

- Error signal: $\delta_j = \varphi'(z_j) \sum_{i \in Layer_{l+1}} w_{i,j} \delta_i$
- For each unit i on layer $l-1$: $\frac{\partial}{\partial w_{j,i}} = \delta_j v_i$

Learning with momentum

$a \leftarrow m \cdot a + \eta_t \nabla_W l(W; y, x)$; $W \leftarrow W - a$

Clustering

k-mean

$\hat{R}(\mu) = \hat{R}(\mu_1, \ldots, \mu_k) = \sum_{i=1}^n \min_{j \in \{1, \ldots k\}} \|x_i - \mu_j\|_2^2$

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

not convex! $\rightarrow$ only local optimum!

choosing k is difficult (plot)!

Algorithm (Lloyd's heuristic):

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

While not converged

$z_i \leftarrow arg \min_{j \in \{1, \ldots, k\}} \|x_i - \mu_j^{(t-1)}\|_2^2$; $\mu_j^{(t)} \leftarrow \frac{1}{n_j} \sum_{i: z_i = j} x_i$

k-mean++

- Start with random data point as center
- Add centers 2 to k randomly, proportionally to squared distance to closest selected center for $j = 2$ to k: i_j sampled with prob.

$P(i_j = i) = \frac{1}{z} \min_{1 \leq l < j} \|x_i - \mu_l\|_2^2$; $\mu_j \leftarrow x_{i_j}$

Dimension Reduction

Principal component analysis (PCA)

Given: $D = x_1, \ldots, x_n \subset \mathbb{R}^d$, $1 \leq k \leq d$

$\Sigma_{d \times d} = \frac{1}{n} \sum_{i=1}^n x_i x_i^T$, $\mu = \frac{1}{n} \sum_{i=1}^n x_i = 0$

Sol.: $(W, z_1, \ldots, z_n) = argmin \sum_{i=1}^n \|W z_i - x_i\|_2^2$,

where $W \in \mathbb{R}^{d \times k}$ is orthogonal, $z_1, \ldots, z_n \in \mathbb{R}^k$ is given by $W = (v_1 | \ldots | v_k)$ and $z_i = W^T x_i$ where $\Sigma = \sum_{i=1}^d \lambda_i v_i v_i^T$, $\lambda_1 \geq \ldots \geq \lambda_d \geq 0$

Kernel PCA

For general $k \geq 1$, the Kernel PC are given by $\alpha^{(1)}, \ldots, \alpha^{(k)} \in \mathbb{R}^n$, where $\alpha^{(i)} = \frac{1}{\sqrt{\lambda_i}} v_i$ is obtained from: $K = \sum_{i=1}^n \lambda_i v_i v_i^T$, $\lambda_1 \geq \ldots \geq \lambda_d \geq 0$

Given this, a new point x is projected as $z \in \mathbb{R}^k$:

$z_i = \sum_{j=1}^n \alpha_j^{(i)} k(x, x_j)$

Autoencoders

Try to learn identity function: $x \approx f(x; \theta)$

$f(x; \theta) = f_2(f_1(x; \theta_1); \theta_2)$; f_1 : en-, f_2 : decoder

Training: $\min_w \sum_{i=1}^n \|x_i - f(x_i; W)\|_2^2$

Probability Modeling

Assumption: Data set is generated iid
Find $h: X \to Y$ that minimizes pred. error $R(h) = \mathbb{E}_{x,y}[l(y; h(x))]$ $h^*(x) = \mathbb{E}[Y|X = x]$ for $R(h) = \mathbb{E}_{x,y}[(y - h(x))^2]$ Pred: $\hat{y} = \hat{\mathbb{E}}[Y|X = x]$

Maximum Likelihood Estimation (MLE)

Choose a particular parametric form $\hat{P}(Y|X, \theta)$, then optimize the parameters using MLE.

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

$= \underset{\theta}{\text{argmax}} \prod_{i=1}^{n} \hat{P}(y_i|x_i, \theta)$ (iid)

$= \underset{\theta}{\text{argmin}} - \sum_{i=1}^{n} log \hat{P}(y_i|x_i, \theta)$

Example: MLE for Linear Gaussian

$y_i \sim \mathcal{N}(w^T x_i, \sigma^2)$:
$y_i = w^T x_i + \epsilon_i, \epsilon_i \sim \mathcal{N}(0, \sigma^2)$
Maximizing the log likelihood:
$\underset{w}{\text{argmax}} P(y_1, ..., y_n|x_1, ..., x_n, w)$

$= \underset{w}{\text{argmin}} \sum_i^n (y_i - w^T x_i)^2$

Bias/Variance/Noise

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

Maximum a posteriori estimate (MAP)

Introduce bias by expressing assumption through a Bayesian prior $w_i \in \mathcal{N}(0, \beta^2)$

Bayes rule: $P(w|x, y) = \frac{P(w|x)P(y|x,w)}{P(y|x)}$

$= \frac{P(w)P(y|x,w)}{P(y|x)}$, we assume w is indep. of x.
$\underset{w}{\text{argmax}} P(w|x, y)$

$= \underset{w}{\text{argmin}} - logP(w) - logP(y|x, w) + const.$

$= \underset{w}{\text{argmin}} \frac{1}{2\beta^2}||w||_2^2 + \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - w^T x_i)^2$

$= \underset{w}{\text{argmin}} \lambda ||w||_2^2 + \sum_{i=1}^{n} (y_i - w^T x_i)^2$, $\lambda = \frac{\sigma^2}{\beta^2}$

($= \underset{w}{\text{argmax}} P(w) \prod_i P(y_i|x_i, w)$, assuming noise $P(y|x, w)$ iid Gaussian, prior $P(w)$ Gaussian)

Logistic Regression

Link function: $\sigma(w^T x) = \frac{1}{1+exp(-w^T x)}$ (Sigmoid)

Logistic regression replaces the assumption of Gaussian noise by iid Bernoulli noise. Can naturally output probabilities.
$P(y|x, w) = Ber(y; \sigma(w^T x)) = \frac{1}{1+exp(-yw^T x)}$

Example: MLE for logistic regression

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

$= \underset{w}{\text{argmin}} - \sum_{i=1}^{n} logP(y_i|w, x_i)$

$= \underset{w}{\text{argmin}} \sum_{i=1}^{n} log(1 + exp(-y_i w^T x_i))$

$\hat{R}(w) = \sum_{i=1}^{n} log(1 + exp(-y_i w^T x_i))$ (neg log l. f.)

SGD for logistic regression

1. Initialize w; 2. For t=1,2,...
Pick data point $(x, y) \in_{u.a.r} D$
Compute probability of misclassification
$\hat{P}(Y = -y|w, x) = \frac{1}{1+exp(yw^T x)}$

Update $w \leftarrow w + \eta_t y x \hat{P}(Y = -y|w, x)$

Logistic regression and regularization

$s = ||w||_2^2$ L2 (Gaussian prior)/$||w||_1$ L1 (Laplace)
$\min_w \sum_{i=1}^{n} log(1 + exp(-y_i w^T x_i)) + \lambda s$

SGD for L2-regularized logistic regression

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

Bayesian decision theory

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

Pick action that minimizes the expected cost:
$a^* = \underset{a \in \mathcal{A}}{\text{argmin}} \mathbb{E}_y[C(y, a)|x] = \sum_y P(y|x) \cdot C(y, a)$

Optimal decision for logistic regression

$a^* = \underset{y}{argmax} \hat{P}(y|x) = sign(w^T x)$

Doubtful logistic regression

Est. cond. distr.: $\hat{P}(y|x) = Ber(y; \sigma(\hat{w}^T x))$
Action set: $\mathcal{A} = \{+1, -1, D\}$; Cost function:

$C(y, a) = \begin{cases} [y \neq a] & \text{if } a \in \{+1, -1\} \\ c & \text{if } a = D \end{cases}$

The action that minimizes the expected cost
$a^* = y$ if $\hat{P}(y|x) \geq 1 - c$, D otherwise

Linear regression

Est. cond. distr.: $\hat{P}(y|x, w) = \mathcal{N}(y; w^T x, \sigma^2)$
$\mathcal{A} = \mathbb{R}$; $C(y, a) = (y - a)^2$
The action that minimizes the expected cost
$a^* = \mathbb{E}_y[y|x] = \int \hat{P}(y|x)\partial y = \hat{w}^T x$

Asymmetric cost for regression

Est. cond. distr.: $\hat{P}(y|x) = \mathcal{N}(\hat{y}; \hat{w}^T x, \sigma^2)$
$\mathcal{A} = \mathbb{R}$; $C(y, a) = c_1 \max(y - a, 0) + c_2 \max(a - y, 0)$
Action that minimizes the expected cost:
$a^* = \hat{w}^T x + \sigma\Phi^{-1}(\frac{c_1}{c_1+c_2})$, Φ: Gaussian CDF

Discriminative vs. Generative Modeling

Discriminative estimates $P(y|x)$
Generative estimates joint distribution $P(y, x)$
Typical approach to generative modeling:
- Estimate prior on labels $P(y)$
- Estimate cond. distr. $P(x|y)$ for each class y
- Obtain predictive distr. using Bayes' rule:

$P(y|x) = \frac{P(y)P(x|y)}{P(x)} = \frac{P(x,y)}{P(x)}$, $P(x) = \sum_y P(x, y)$

Example MLE for P(y)

Want: $P(Y = 1) = p, P(y = -1) = 1 - p$
Given: $D = \{(x_1, y_1), ..., (x_n, y_n)\}$
$P(D|p) = \prod_{i=1}^{n} p^{1[y_i=+1]}(1 - p)^{1[y_i=-1]}$
$= p^{n_+}(1 - p)^{n_-}$, where n_+ = # of $y = +1$
$\frac{\partial}{\partial p} logP(D|p) = n_+ \frac{1}{p} - n_- \frac{1}{1-p} \stackrel{!}{=} 0 \Rightarrow p = \frac{n_+}{n_+ + n_-}$

Example MLE for P=(x|y)

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

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

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

Deriving decision / classification rule

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

$= \underset{y'}{\text{argmax}} logP(y') + \sum_{i=1}^{d} logP(x_i|y')$

Gaussian Naive Bayes classifier

MLE for class prior: $\hat{P}(Y = y) = \hat{p}_y = \frac{\text{Count}(Y=y)}{n}$

MLE for feature distr.: $\hat{P}(x_i|y) = \mathcal{N}(x_i; \hat{\mu}_{y,i}, \sigma_{y,i}^2)$

$\hat{\mu}_{y,i} = \frac{1}{\text{Count}(Y=y)} \sum_{j:y_j=y} x_{j,i}$

$\sigma_{y,i}^2 = \frac{1}{\text{Count}(Y=y)} \sum_{j:y_j=y} (x_{j,i} - \hat{\mu}_{y,i})^2$

Prediction given new point x:
$y = \underset{y'}{\text{argmax}} \hat{P}(y'|x) = \underset{y'}{\text{argmax}} \hat{P}(y') \prod_{i=1}^{d} \hat{P}(x_i|y')$

Gaussian Bayes Classifier

MLE for class prior: $\hat{P}(Y = y) = \hat{p}_y = \frac{\text{Count}(Y=y)}{n}$

MLE for feature distr.: $\hat{P}(x|y) = \mathcal{N}(x; \hat{\mu}_y, \hat{\Sigma}_y)$

$\hat{\mu}_y = \frac{1}{\text{Count}(Y=y)} \sum_{i:y_i=y} x_i \in \mathbb{R}^d$

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

Fisher's linear discriminant analysis (LDA; c=2)

Assume: $p = 0.5$; $\hat{\Sigma}_- = \hat{\Sigma}_+ = \hat{\Sigma}$

discriminant f.: $f(x) = log\frac{p}{1-p} + \frac{1}{2}[log\frac{|\hat{\Sigma}_-|}{|\hat{\Sigma}_+|}$

$+ ((x - \hat{\mu}_-)^T \hat{\Sigma}_-^{-1}(x - \hat{\mu}_-)) - ((x - \hat{\mu}_+)^T \hat{\Sigma}_+^{-1}(x - \hat{\mu}_+))]$

Predict: $y = sign(f(x)) = sign(w^T x + w_0)$

$w = \hat{\Sigma}^{-1}(\hat{\mu}_+ - \hat{\mu}_-)$; $w_0 = \frac{1}{2}(\hat{\mu}_-^T \hat{\Sigma}^{-1}\hat{\mu}_- - \hat{\mu}_+^T \hat{\Sigma}^{-1}\hat{\mu}_+)$

Outlier Detection

$P(x) = \sum_{y=1}^{c} P(y)P(x|y) = \sum_y \hat{p}_y \mathcal{N}(x|\hat{\mu}_y, \hat{\Sigma}_y) \leq \tau$

Categorical Naive Bayes Classifier

MLE class prior: $\hat{P}(Y = y) = \frac{Count(Y=y)}{n}$

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

$\theta_{c|y}^{(i)} = \frac{Count(X_i=c, Y=y)}{Count(Y=y)}$, Pred.: $y = \underset{y'}{argmax} \hat{P}(y'|x)$

Latent: Missing Data

Mixture modeling

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

data iid, likelih.: $P(D|\theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} w_j P(x_i|\theta_j)$

$\underset{min}{argmin} L(D; \theta) = \underset{min}{argmin} - \sum_i log \sum_j w_j P(x_i|\theta_j)$

Gaussian-Mixture Bayes classifiers

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

$P(y|x) = \frac{1}{P(x)} p(y) \sum_{j=1}^{k_y} w_j^{(y)} \mathcal{N}(x; \mu_j^{(y)}, \Sigma_j^{(y)})$

Hard-EM

Initialize parameters $\theta^{(0)}$; For $t = 1, 2, ..$ (predict most likely class for each data point):
$z_i^{(t)} = \underset{z}{\text{argmax}} P(z|x_i, \theta^{(t-1)})$

$= \underset{z}{\text{argmax}} P(z|\theta^{(t-1)}) P(x_i|z, \theta^{(t-1)})$;

Compute the MLE (see Gauss Bayes Classifier):
$\theta^{(t)} = \underset{\theta}{\text{argmax}} P(D^{(t)}|\theta)$

Soft-EM: "While not converged repeat"

E-step: For each i and j calculate $\gamma_j^{(t)}(x_i)$
M-step: Fit clusters to weighted data points:

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

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

EM for semi-supervised learning with GMMs:
unl. p.: $\gamma_j^{(t)}(x_i) = P(Z = j|x_i, \mu^{(t-1)}, \Sigma^{(t-1)}, w^{(t-1)})$

labeled points y_i: $\gamma_j^{(t)}(x_i) = [j = y_i]$

Things To Remember

$ln(x) \leq x - 1, x > 0$; $||x||_2 = \sqrt{x^T x}$; $\nabla_x ||x||_2^2 = 2x$
$f(x) = x^T A x$; $\nabla_x f(x) = (A + A^T)x$
Standard Gaussian: CDF: $\Phi(x) = \int_{-\infty}^{x} \phi(t)\partial t$;
PDF: $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-(1/2)x^2}$; $\int \phi(x)\partial x = \Phi(x) + c$;
$\int x\phi(x) = -\phi(x) + c$; $\int x^2 \phi(x)\partial x = \Phi(x) - x\phi(x) + c$

Probabilities

$\mathbb{E}_x[X] = \begin{cases} \int x \cdot p(x)\partial x & |\mathbb{E}_x[f(x)] = \\ \sum_x x \cdot p(x) & |\int f(x) \cdot p(x)\partial x \end{cases}$

$Var[X] = \mathbb{E}[(X - \mu_X)^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$
$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$; $p(Z|X, \theta) = \frac{p(X,Z|\theta)}{p(X|\theta)}$
$P(x, y) = P(x \cap y) = P(y|x) \cdot P(x) = P(x|y) \cdot P(y)$