Probabilities

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

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

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

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

Important

$ln(x) \leq x-1, x>0$; $\|x\|_2 = \sqrt{x^T x}$; $\nabla_x \|x\|_2^2 = 2x$

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

$D_{KL} = \mathbb{E}_p[log(\frac{p(x)}{q(x)})]$; $D_{KL}(P\|Q) = \sum_{x\in X} P(x) \cdot log\frac{P(x)}{Q(x)} = \int_{-\infty}^{+\infty} p(x) log\frac{p(x)}{q(x)} dx$ always nonneg

Standard Gaussian: CDF: $\Phi(x) = \int_{-\infty}^{x} \phi(t)\partial t$;

PDF: $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-(1/2)x^2}$; $\int \phi(x)\partial x = \Phi(x) + c$;

$\int x\phi(x) = -\phi(x) + c$; $\int x^2 \phi(x) \partial x = \Phi(x) - x\phi(x) + c$

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 i do $w_{t+1} = w_t - \eta_t \nabla \hat{R}(w_t)$

Generalization error

Goal: minimize true risk (iid data) $R(w) = \int P(x,y)(y - w^T x)^2 \partial x \partial y = \mathbb{E}_{x,y}[(y - w^T x)^2]$

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

Generalization error = |Estim. R - true R|

Evaluate model on validation set, because $\mathbb{E}_D[\hat{R}_D(w_D)] \ll \mathbb{E}[R(w_D)]$

Crossvalidation (=pick multiple test sets)

For each model m repeat for $i = 1:k$

Split dataset $D = D_t^{(i)} \uplus D_v^{(i)}$, train model $\hat{\mathbf{w}}_i$ and estimate error $\hat{R}_m^{(i)}$ on validation set

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

Ridge regression

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

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: $\mu = 0$, $\sigma^2 = 1$: $\tilde{x}_{i,j} = \frac{(x_{i,j} - \hat{\mu}_j)}{\hat{\sigma}_j}$

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

Classification

0/1 loss

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.

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

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

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

Theorem: If the data is linearly separable, the Perceptron will obtain a linear separator.

Stochastic Gradient Descent (SGD)

1. Start at an arbitrary $w_0 \in \mathbb{R}^d$
2. For $t = 1, 2, ...$ 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 alg. uses SGD with Perceptron loss

Support Vector Machine

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

Theorem: SVM finds solution with max margin to decision boundary.

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

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

Feature selection: Greedy and Lasso

Lasso uses L1-norm $\|\cdot\|_1$ to encourage sparse solutions. Fast, but for linear models only. Greedy forw adds best elm, greedy backw remove worse elm. General, but comp. expensive and might lead to bad solutions.

Kernels

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

Kernel optimization problems are convex

Kernel trick: Perceptron and SVM

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

or in short $\underset{\alpha}{min} \sum_{i=1}^n max\{0, -y_i \alpha^T k_i\}$

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

with $k_i = [y_1 k(x_i, x_1), ..., y_n k(x_i, x_n)]$, $D_y = diagy$

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

Kernelized Perceptron

Initialize $\alpha_{1:n} = 0$ and repeat for $t = 1, 2, ...$

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 kernel

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

Kernel matrix must be positive semi-definite.

Kernel matrix $K_{i,j} = k(x_i, xx_j)$

Semi-definite matrices $\Leftrightarrow$ kernels

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

Kernel engineering

$k_1(x,y) + k_2(x,y)$; $k_1(x,y)\cdot k_2(x,y)$; $c\cdot k_1(x,y), c > 0$; $f(k_1(x,y))$, f polynomial with coeff ≥ 0 or exp

Nearest Neighbor k-NN

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

Imbalance

Upsampling, downsampling of dataset

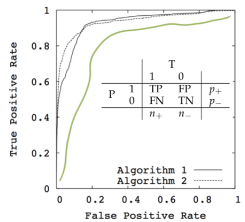
Cost Sensitive Classification

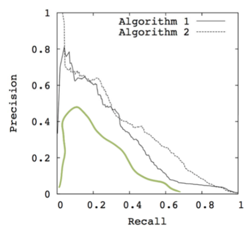
weight losses per class $l_{CS}(w; x, y) = c_y l(w; x, y)$

Metrics

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

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





Multi-class Hinge Loss

1) One-vs-all: train c bin classifiers (conf.)
2) One-vs-one: train $c\frac{c-1}{2}$ binary classifiers
3) Maintain c different $w^{(i)}$ vectors

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

Neural Networks

Learning features

Parameterize feat. maps (eg $\phi(x,\theta) = \varphi(\theta^T x)$) optimize params (non-convex problem):

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

SGD step: $W \leftarrow W - \eta_t \nabla_W l(W; y, x)$

Activation functions

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

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

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

Forward propagation

For input layer $v^0 = x$

For each hidden layer $l = 1 : L-1$

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

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

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

Backpropagation to compute $\nabla_W l(W; y, x)$

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 $l = \{L-1, ..., 1\}$:

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

e.g. $\frac{\partial L}{\partial w_{ij}^{(2)}} = \frac{\partial L}{\partial f_i}\frac{\partial f_i}{\partial w_{ij}^{(2)}}$, $\frac{\partial L}{\partial w_{ij}^{(1)}} = \sum_i \frac{\partial L}{\partial f_i}\frac{\partial f_i}{\partial v_j}\frac{\partial v_j}{\partial w_{jk}^{(1)}}$

Learning with momentum

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

Regularization

Dropout, early stopping (train and val set)

CNNs: input -> convolution -> pooling -> NN

$L = \frac{n+2p-f}{s} + 1$, stride s, padding p, $n \times n$ image

Clustering

k-means (non-convex)

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

Algorithm (Lloyd's heuristic), O(nkd) per it:

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

While not converged

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

problems: k =?, number of iterations exponential, circles, non-convex, can't regularize

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

Dimension Reduction
Principal component analysis (PCA)
Assume $\Sigma=\frac{1}{n}\sum_{i=1}^n x_i x_i^T$, $\mu=\frac{1}{n}\sum_{i=1}^n x_i=0$
General $(W,z_1,...,z_n)=\operatorname{argmin}\sum_{i=1}^n\|Wz_i-x_i\|_2^2$ with $W\in\mathbb{R}^{d\times k}$ orthogonal, $z_1,...,z_n\in\mathbb{R}^k$
Solution $W=(v_1|...|v_k)$ and $z_i=W^Tx_i$, where v_k eigenv of $\Sigma=\sum_{i=1}^d\lambda_i v_i v_i^T$, $\lambda_1\ge...\ge\lambda_d\ge 0$
$k=1$: $w^*=\operatorname{argmin}_{\|w\|=1}w^T\Sigma w=v_1$

Kernel PCA
Optim ($k=1$): $\alpha^{(*)}=\operatorname{argmax}_{\alpha^TK\alpha=1}\alpha^TK^TK\alpha$
Sol ($k\ge1$): $\alpha^{(i)}=\frac{1}{\sqrt{\lambda_i}}v_i$ from eigenv decomposition of $K=\sum_{i=1}^n\lambda_i v_i v_i^T$, $\lambda_1\ge...\ge\lambda_d\ge0$
x is projected as $z\in\mathbb{R}^k$ by $z_i=\sum_{j=1}^n\alpha_j^{(i)}k(x,x_j)$

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 $R(h)=\int P(x,y)l(y;h(x))\partial x\partial y=\mathbb{E}_{x,y}[l(y;h(x))]$
$h^*(x)=\mathbb{E}[Y|X=x]$ for $R(h)=\mathbb{E}_{x,y}[(y-h(x))^2]$
Prediction: $\hat{y}=\hat{\mathbb{E}}[Y|X=x]=\int\hat{P}(y|X=x)y\partial y$
Choose a particular parametric form $\hat{P}(Y|X,\theta)$

Maximum Likelihood Estimation (MLE)
$\theta^*=\operatorname{argmax}_\theta\hat{P}(y_1,...,y_n|x_1,...,x_n,\theta)$

Prediction error = $Bias^2+Variance+Noise$
Bias $\mathbb{E}_\mathbf{X}[\mathbb{E}_D\hat{h}_D(\mathbf{X})-h^*(\mathbf{X})]^2$
V $\mathbb{E}_\mathbf{X}\mathbb{V}_D[\hat{h}_D(\mathbf{X})]^2=\mathbb{E}_\mathbf{X}\mathbb{E}_D[\hat{h}_D(\mathbf{X})-\mathbb{E}_{D'}\hat{h}_{D'}(\mathbf{X})]^2$
Noise $\mathbb{E}_{\mathbf{X},Y}[Y-h^*(\mathbf{X})]^2$

Maximum a posteriori estimate (MAP)
Introduce bias through prior $w_i\in\mathcal{N}(0,\beta^2)$
Bayes: $P(w|x,y)=\frac{P(w|x)P(y|x,w)}{P(y|x)}=\frac{P(w)P(y|x,w)}{P(y|x)}$
$\operatorname{argmax}_w P(w|x,y)=\operatorname{argmax}_w\ln P(w)+\ln P(y|x,w)$

Example: MLE and MAP for linear Gaussian
$y_i\sim\mathcal{N}(w^Tx_i,\sigma^2): y_i=w^Tx_i+\epsilon_i,\epsilon_i\sim\mathcal{N}(0,\sigma^2)$
$\operatorname{argmax}_w P(y_{1:n}|x_{1:n},w)=\operatorname{argmin}_w\sum_i^n(y_i-w^Tx_i)^2$
MAP $\operatorname{argmin}_w\frac{1}{2\beta^2}\|w\|_2^2+\frac{1}{2\sigma^2}\sum_{i=1}^n(y_i-w^Tx_i)^2$

Logistic regression
replaces Gaussian noise assumption by iid Bernoulli noise $P(y|x,w)=Ber(y;\sigma(w^Tx))$
Sigmoid link function $\sigma(w^Tx)=\frac{1}{1+exp(-w^Tx)}$
$\nabla_w l(\mathbf{w})=-y\mathbf{x}P(Y=-y|\mathbf{w},\mathbf{x})$
Predict e.g. most likely $P(y|x,w)$ class label.

Example: MLE for logistic regression
$\operatorname{argmax}_w P(y_{1:n}|w,x_{1:n})=\operatorname{argmax}_w\sum\log P(y_i|w,x_i)$
$=\operatorname{argmin}_w\sum_{i=1}^n\log(1+exp(-y_iw^Tx_i))$
$\hat{R}(w)=\sum_{i=1}^n\log(1+exp(-y_iw^Tx_i))$ (neg log l. f.)

Logistic regression and regularization
$s=\|w\|_2^2$ L2 (Gaussian prior)/$\|w\|_1$ L1 (Laplace)
$\min_w\sum_{i=1}^n log(1+exp(-y_iw^Tx_i))+\lambda s$

SGD for logistic regression
Initialize w and repeat for t=1,2,...
Pick data $(x,y)\in_{u.a.r}D$
Compute probability of misclassification $\hat{P}(Y=-y|w,x)=\frac{1}{1+exp(yw^Tx)}$
Update $w\leftarrow w+\eta_t yx\hat{P}(Y=-y|w,x)$ or regularized $w\leftarrow w(1-2\lambda\eta_t)+\eta_t yx\hat{P}(Y=-y|w,x)$

Bayesian decision theory (BDT)
- 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:
$a^*=\operatorname{argmin}_{a\in\mathcal{A}}\mathbb{E}_y[C(y,a)|x]=\sum_y P(y|x)\cdot C(y,a)$

Optimal decision for logistic regression
$a^*=\operatorname{argmax}_y\hat{P}(y|x)=sign(w^Tx)$

Doubtful logistic regression
Est. cond. distr.: $\hat{P}(y|x)=Ber(y;\sigma(\hat{w}^Tx))$
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 $\hat{P}(y|x)\ge1-c$, D otherwise

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

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

Discriminative vs. Generative Modeling
DM estimate $P(y|x)$, GM estimate $P(y,x)$ using prior $P(y)$ and $P(x|y)$ and predict using
$P(y|x)=\frac{P(y)P(x|y)}{P(x)}=\frac{P(x,y)}{P(x)}$, $P(x)=\sum_y P(x,y)$

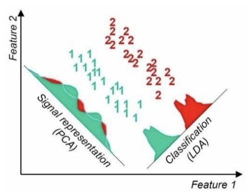
Example MLE for P(y)
Want: $P(Y=1)=p, P(y=-1)=1-p$
Given: $D=\{(x_1,y_1),...,(x_n,y_n)\}$
$P(D|p)=\prod_{i=1}^n p^{1[y_i=+1]}(1-p)^{1[y_i=-1]}$
$=p^{n_+}(1-p)^{n_-}$, where $n_+=$ # of $y=+1$
$\frac{\partial}{\partial p}logP(D|p)=n_+\frac{1}{p}-n_-\frac{1}{1-p}\overset{!}{=}0\Rightarrow p=\frac{n_+}{n_++n_-}$

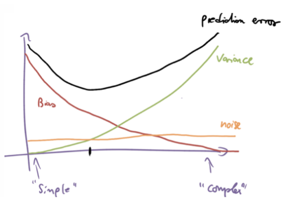
Deriving decision rule
$P(y|x)=\frac{1}{Z}P(y)P(x|y)$, $Z=\sum_y P(y)P(x|y)$
$y=\operatorname{argmax}_{y'}P(y'|x)=\operatorname{argmax}_{y'}P(y')\prod_{i=1}^d P(x_i|y')$
$=\operatorname{argmax}_{y'}logP(y')+\sum_{i=1}^d logP(x_i|y')$

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.: $\hat{P}(x_i|y)=\mathcal{N}(x_i;\hat{\mu}_{y,i},\sigma_{y,i}^2)$
$\hat{\mu}_{y,i}=\frac{1}{\text{Count}(Y=y)}\sum_{j:y_j=y}x_{j,i}$
$\sigma_{y,i}^2=\frac{1}{\text{Count}(Y=y)}\sum_{j:y_j=y}(x_{j,i}-\hat{\mu}_{y,i})^2$
Predict using Bayesian Decision Theory, e.g.
$y=\operatorname{argmax}_{y'}\hat{P}(y'|x)=\operatorname{argmax}_{y'}\hat{P}(y')\prod_{i=1}^d\hat{P}(x_i|y')$
NB might be overconfident (close to 1 or 1)

Gaussian Bayes Classifier
MLE for class prior: $\hat{P}(Y=y)=\hat{p}_y=\frac{\text{Count}(Y=y)}{n}$
MLE for feature distr.: $\hat{P}(x|y)=\mathcal{N}(x;\hat{\mu}_y,\hat{\Sigma}_y)$
$\hat{\mu}_y=\frac{1}{\text{Count}(Y=y)}\sum_{i:y_i=y}x_i\in\mathbb{R}^d$
$\hat{\Sigma}_y=\frac{1}{\text{Count}(Y=y)}\sum_{i:y_i=y}(x_i-\hat{\mu}_y)(x_i-\hat{\mu}_y)^T\in\mathbb{R}^{d\times d}$
$c=2$: predict $y=sign(f(x))$ with discriminant func: $f(x)=\log(\frac{P(Y=1|x)}{P(Y=0|x)})=\log\frac{p}{1-p}+\frac{1}{2}[log\frac{|\hat{\Sigma}_-|}{|\hat{\Sigma}_+|}$
$+((x-\hat{\mu}_-)^T\hat{\Sigma}_-^{-1}(x-\hat{\mu}_-))-((x-\hat{\mu}_+)^T\hat{\Sigma}_+^{-1}(x-\hat{\mu}_+))]$

Fisher's linear discriminant analysis (LDA; c=2)
$f(x)$ from GBC simplified with $p=0.5$; $\hat{\Sigma}_-=\hat{\Sigma}_+=\hat{\Sigma}$ predict: $y=sign(f(x))=sign(w^Tx+w_0)$
$w=\hat{\Sigma}^{-1}(\hat{\mu}_+-\hat{\mu}_-)$; $w_0=\frac{1}{2}(\hat{\mu}_-^T\hat{\Sigma}^{-1}\hat{\mu}_--\hat{\mu}_+^T\hat{\Sigma}^{-1}\hat{\mu}_+)$





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

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.: $y=\operatorname{argmax}_{y'}\hat{P}(y'|x)$

Latent: Missing Data
Mixture modeling
Model each cluster as probability distr. $P(x|\theta_j)$
data iid, likelih.: $P(D|\theta)=\prod_{i=1}^n\sum_{j=1}^k w_jP(x_i|\theta_j)$
$argminL(D;\theta)=argmin-\sum_i log\sum_j w_jP(x_i|\theta_j)$

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

EM
E-step: calculate $\gamma_z(\mathbf{x})=P(Z=z|\mathbf{X}=\mathbf{x},\theta^{(t-1)})$
M-step: $\theta^{(t+1)}=\operatorname{argmax}_\theta Q(\theta;\theta^{(t-1)})$, where
$Q(\theta,\theta^{(t)})=\mathbb{E}_{\mathbf{Z}|\mathbf{X}=\mathbf{x},\theta^{(t)}}[\log P(\mathbf{X},Z|\theta)]=$
$\sum_{i=1}^n\mathbb{E}_{\mathbf{Z}|\mathbf{X}=\mathbf{x}_i,\theta^{(t)}}[\log P(\mathbf{x}_i,Z|\theta)]=$
$\sum_{i=1}^n\sum_{z=1}^k P(Z=z|\mathbf{x}_i,\theta^{(t)})\log P(\mathbf{X}=\mathbf{x}_i,Z=z|\theta)$

Hard-EM algorithm
Initialize parameters $\theta^{(0)}$ and repeat $t=1,2,...$:
Predict most likely class for each data point:
$z_i^{(t)}=\operatorname{argmax}_z P(z|x_i,\theta^{(t-1)})$
$=\operatorname{argmax}_z P(z|\theta^{(t-1)})P(x_i|z,\theta^{(t-1)})$;
Compute the MLE as for the Gaussian B. class.:
$\theta^{(t)}=\operatorname{argmax}_\theta P(D^{(t)}|\theta)$

Soft-EM algorithm: While not converged
E-step: For each i and j calculate $\gamma_j^{(t)}(x_i)$
M-step: Fit clusters to weighted data points:
$w_j^{(t)}\leftarrow\frac{1}{n}\sum_{i=1}^n\gamma_j^{(t)}(x_i)$; $\mu_j^{(t)}\leftarrow\frac{\sum_{i=1}^n\gamma_j^{(t)}(x_i)x_i}{\sum_{i=1}^n\gamma_j^{(t)}(x_i)}$
$\Sigma_j^{(t)}\leftarrow\frac{\sum_{i=1}^n\gamma_j^{(t)}(x_i)(x_i-\mu_j^{(t)})(x_i-\mu_j^{(t)})^T}{\sum_{i=1}^n\gamma_j^{(t)}(x_i)}(+\nu^2I)$
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 with label y_i: $\gamma_j^{(t)}(x_i)=[j=y_i]$