supervised learning:

given: training dataset: $\{\vec{x}_1, \dots, \vec{x}_N\}$

target dataset: $\{\vec{t}_1, \dots, \vec{t}_N\}$

learn: a function y s.t. $y(\vec{x}_i) \approx \vec{t}_i \ \forall i$

test: check y on new datapoints

ex: classification, regression

unsupervised learning

given: training data

ex: clustering, density estimation, visualization

Probability

sum rule - $p(x) = \sum_y p(x,y)$ if discrete; $p(x) = \int_y p(x,y)dy$ if continuous

product rule - $p(x,y) = p(x|y)p(y) = p(y|x)p(x) = p(y,x)$

Bayes' rule - $p(y|x) = \frac{p(x|y)\,p(y)}{p(x)} = \frac{p(x|y)\,p(y)}{\sum_y p(x|y)p(y)}$

$p(y|x)$ = posterior probability

$p(y)$ = prior probability

given a probability distribution function $p(x)$, the expectation or average of a function $f(x)$ in $p(x)$ is defined as

$\mathbb{E}[f(x)] = \sum_x p(x) f(x)$ if discrete; $\mathbb{E}[f(x)] = \int p(x) f(x)\,dx$ if continuous

note: $\mathbb{E}[f(x)] = \frac{1}{N} \lim_{N\to\infty} \sum_{n=1}^{N} f(x_n)$

$\mathbb{E}_x[f(x,y)]$ = expectation of $f(x,y)$ wrt marginal dist of x

$\mathbb{E}[f(x)|y] = \sum_x p(x|y) f(x)$ = conditional expectation on $f(x)$

the variance, a measure of how "variable" a random variable is, is defined as

$\text{var}[f] = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2] = \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2$

the covariance of two rv's is

$\text{cov}[x,y] = \mathbb{E}_{x,y}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])] = \mathbb{E}_{x,y}[x,y] - \mathbb{E}[x]\mathbb{E}[y]$

$\text{cov}[x,y] = 0$ if x and y are independent

Gaussian distributions

a Gaussian (or normal) r.v. has a probability density function

$$\boxed{\mathcal{N}(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}}$$

random variables $\vec{x} = (x_1, \dots, x_N)$ are iid (independent and identically distributed)

in this case, the joint density is $p(\vec{x}) = \prod_{n=1}^{N} p(x)$

maximum likelihood of normally distributed iid samples

$\mu_{ML} = \underset{\mu}{\text{argmax}}\ p(\vec{x}|\mu,\sigma^2) = \underset{\mu}{\text{argmax}} \ln p(\vec{x}|\mu,\sigma^2) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)$

set the derivative to 0 to find the max

$\frac{1}{2\sigma^2}\sum_{n=1}^{N} 2(x_n-\mu) = 0 \Rightarrow \sum_{n=1}^{N}(x_n-\mu) = 0 \Rightarrow \sum_{n=1}^{N} x_n = N\mu \Rightarrow \mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n$ = sample mean

$\sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{ML})^2$ = sample variance

bias - sample variance underesimates

$\mathbb{E}[\mu_{ML}] = \mu$

$\mathbb{E}[\sigma^2_{ML}] = \left(\frac{N-1}{N}\right)\sigma^2 \Rightarrow$ unbiased estimator for variance is $\tilde{\sigma}^2 = \frac{N}{N-1}\sigma^2_{ML}$

ex: input $\vec{x} = (x_1, \dots, x_n)^T$, target $\vec{t} = (t_1, \dots, t_N)^T$

assume $t_n \sim \mathcal{N}(y(x_n, \vec{w}), \beta^{-1})$, i.e. the targets approximate $y(\vec{x}, \vec{w})$ with a little bit of noise

MLE approach

maximize $p(\vec{t} \mid \vec{x}, \vec{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(\vec{t} \mid y(x_n, \vec{w}), \beta^{-1})$

log likelihood: $\boxed{-\frac{\beta}{2}\sum_{n=1}^{N}[y(x_n, \vec{w}) - t_n]^2 + \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi)}$

MLE = $\arg\min_{\vec{w}} E(\vec{w}) = \arg\min_{\vec{w}} \frac{1}{2}\sum_{n=1}^{N}[y(x_n, \vec{w}) - t_n]^2$

precision (average error) = $\frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}[y(x_n, \vec{w}_{ML}) - t_n]^2$

Bayesian approach (MAP)

assume prior $p(\vec{w}|\alpha) = N(\vec{w} \mid \vec{0}, \alpha^{-1}I) = \left(\frac{\alpha}{2\pi}\right)^{(m+1)/2} \exp\left(-\frac{\alpha}{2}\vec{w}^T\vec{w}\right)$ where α is a hyperparameter that acts like β

by Baye's theorem, $p(\vec{w} \mid \vec{x}, \vec{t}, \alpha, \beta) \propto p(\vec{t} \mid \vec{x}, \vec{w}, \beta) \cdot p(\vec{w} \mid \alpha)$

log of posterior $p(\vec{w} \mid \vec{x}, \vec{t}, \alpha, \beta) \Rightarrow \boxed{\frac{\beta}{2}\sum_{n=1}^{N}[y(x_n, \vec{w}) - t_n]^2 + \frac{\alpha}{2}\vec{w}^T\vec{w}}$ = regularized sum of squares

multivariate Gaussian: $\boxed{\mathcal{N}(\vec{x} \mid \vec{\mu}, \Sigma) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma|^{1/2}}\exp\left\{\underbrace{-\frac{1}{2}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})}_{\text{Mahalanobis distance}}\right\}}$

$\vec{\mu}$ = mean vector, Σ = D×D covariance matrix, $|\Sigma| = \det(\Sigma)$

$$\mathbb{E}[\vec{x}] = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma|^{1/2}}\int \exp\left\{-\frac{1}{2}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})\right\}\vec{x}\,d\vec{x}$$

$$= \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma|^{1/2}}\int \exp\left\{-\frac{1}{2}\vec{z}^T\Sigma^{-1}\vec{z}\right\}(\vec{z}+\vec{\mu})\,d\vec{z}$$

$$= \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma|^{1/2}}\left[\vec{\mu}\int \exp\left\{-\frac{1}{2}\vec{z}^T\Sigma^{-1}\vec{z}\right\}d\vec{z} + \underbrace{\int \vec{z}\cdot\exp\left\{-\frac{1}{2}\vec{z}^T\Sigma^{-1}\vec{z}\right\}d\vec{z}}_{=0}\right]$$

$= \vec{\mu}$ = sample mean

$$\mathbb{E}[\vec{x}\vec{x}^T] = \vec{\mu}\vec{\mu}^T + \Sigma \Rightarrow \text{cov}[\vec{x}] = \mathbb{E}[(\vec{x}-\vec{\mu})(\vec{x}-\vec{\mu})^T] = \mathbb{E}[\vec{x}\vec{x}^T] - 2\vec{\mu}\vec{\mu}^T + \vec{\mu}\vec{\mu}^T = \vec{\mu}\vec{\mu}^T + \Sigma - \vec{\mu}\vec{\mu}^T = \Sigma = \text{sample covariance}$$

suppose $\vec{x} \sim \mathcal{N}(\vec{x} \mid \vec{\mu}, \Sigma)$, $\vec{x} = \begin{bmatrix}\vec{x}_a \\ \vec{x}_b\end{bmatrix}$, $\vec{\mu} = \begin{bmatrix}\vec{\mu}_a \\ \vec{\mu}_b\end{bmatrix}$, $\Sigma = \begin{bmatrix}\Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb}\end{bmatrix}$, $\Lambda = \Sigma^{-1} = \begin{bmatrix}\Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb}\end{bmatrix}$

conditional Gaussian: $p(\vec{x}_a \mid \vec{x}_b) = \mathcal{N}(\vec{x} \mid \vec{\mu}_{a|b}, \Lambda_{aa}^{-1})$

$$\Lambda_{aa}^{-1} = \Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}$$

$$\vec{\mu}_{a|b} = \vec{\mu}_a - \Lambda_{aa}^{-1}\Lambda_{ab}(\vec{x}_b - \vec{\mu}_b)$$

marginal Gaussian: $p(\vec{x}_a) = \mathcal{N}(\vec{x}_a \mid \vec{\mu}_a, \Sigma_{aa})$

Dual Representation and Kernels

kernel function: $k(\vec{x}, \vec{x}') = \vec{\Phi}(\vec{x})^T\vec{\Phi}(\vec{x}')$; $\vec{\Phi}(\vec{x})$ is a feature space mapping

Mercer's Theorem - a symmetric function $k(\vec{x}, \vec{y})$ is a kernel function iff the Gram matrix K is positive semidefinite for any collection of data $\{\vec{x}_n\}$

Linear Basis Function Models

linear regression - $y(\vec{x}, \vec{w}) = w_0 + w_1x_1 + \dots + w_Dx_D = \vec{w}\vec{\Phi}(\vec{x}) \approx t$; where $\vec{\Phi}(\vec{x}) = \begin{bmatrix}1 \\ \vec{x}\end{bmatrix}$

polynomial fitting - $y(x, \vec{w}) = w_0 + w_1x + \dots + w_Mx^M = \vec{w}\vec{\Phi}(x) \approx t$; where $\vec{\Phi}(x) = \begin{bmatrix}1 \\ x \\ x^2 \\ \vdots \\ x^M\end{bmatrix}$

linear basis function model - $y(\vec{x}, \vec{w}) = w_0 + \sum_{j=1}^{M-1} w_j\phi_j(\vec{x}) = \vec{w}^T\vec{\Phi}(\vec{x}) \approx t$; where $\vec{\Phi}(\vec{x}) = \begin{bmatrix}\phi_0(\vec{x}) \\ \phi_1(\vec{x}) \\ \vdots \\ \phi_{M-1}(\vec{x})\end{bmatrix}$, $\phi_0(\vec{x}) = 1$

ex: let $t = y(\vec{x}, \vec{w}) + \varepsilon$, $\varepsilon \sim \mathcal{N}(\varepsilon | 0, \beta^{-1})$

input: $X = \{\vec{x}_1, \ldots, \vec{x}_N\}$, target: $\{t_1, \ldots, t_N\}$, samples are iid

$p(\vec{t} | X, \vec{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n | y(\vec{x}_n, \vec{w}), \beta^{-1}) = \prod_{n=1}^{N} \mathcal{N}(t_n | \vec{w}^T \vec{\phi}(\vec{x}), \beta^{-1})$

maximum likelihood - $\ln p(\vec{t} | X, \vec{w}, \beta) = -\frac{N}{2} \ln(2\pi) + \frac{N}{2} \ln \beta - \frac{\beta}{2} \sum_{n=1}^{N} (t_n - \vec{w}^T \vec{\phi}(\vec{x}_n))^2$

$\vec{w}_{ML} = (\Phi^T \Phi)^{-1} \Phi^T \vec{t} = \Phi^+ \vec{t}$

<u>design matrix</u> $\Phi = \begin{bmatrix} \phi_0(\vec{x}_1) & \phi_1(\vec{x}_1) & \cdots & \phi_{M-1}(\vec{x}_1) \\ \phi_0(\vec{x}_2) & \phi_1(\vec{x}_2) & \cdots & \phi_{M-1}(\vec{x}_2) \\ \vdots & \vdots & & \vdots \\ \phi_0(\vec{x}_N) & \phi_1(\vec{x}_N) & \cdots & \phi_{M-1}(\vec{x}_N) \end{bmatrix} = \begin{bmatrix} — \vec{\phi}(\vec{x}_1) — \\ — \vec{\phi}(\vec{x}_2) — \\ \vdots \\ — \vec{\phi}(\vec{x}_N) — \end{bmatrix} \in \mathbb{R}^{N \times M}$

Φ^+ is the <u>Moore-Penrose pseudoinverse</u> of Φ

calculating the pseudoinverse is at least $O(\min\{M, N\}^w)$

$\frac{1}{\beta_{ML}} = \frac{1}{N} \sum_{n=1}^{N} (t_n - \vec{w}_{ML}^T \vec{\phi}(\vec{x}_n))^2$ = <u>sample variance</u>

$w_0 = -\frac{\beta}{2} \sum_{n=1}^{N} (t_n - \vec{w}^T \vec{\phi}(\vec{x}_n))^2 = -\frac{\beta}{2} \sum_{n=1}^{N} (t_n - \sum_{j=1}^{M-1} w_j \phi_j(\vec{x}_n))^2$ = <u>bias</u>

maximized $w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j \bar{\phi}_j$ where $\bar{t} = \frac{1}{N} \sum_{n=1}^{N} t_n$ and $\bar{\phi}_j = \frac{1}{N} \sum_{n=1}^{N} \phi_j(\vec{x}_n)$

sequential learning - <u>stochastic gradient descent</u>

general setup - minimize $E(\vec{w}) = \sum_{n=1}^{N} E_n(\vec{w})$

$\Rightarrow \vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla E_n(\vec{w}^{(k)})$ for a random n, learning rate η

<u>least-mean-squares algorithm</u>

minimize $\sum_{n=1}^{N} \{t_n - \vec{w}^T \vec{\phi}(\vec{x}_n)\}^2 = \|\vec{t} - \Phi \vec{w}\|^2$

$\Rightarrow \vec{w}^{(k+1)} = \vec{w}^{(k)} + \eta [t_n - \vec{w}^{(k)T} \vec{\phi}(\vec{x}_n)] \vec{\phi}(\vec{x}_n)$

Bayesian Linear Regression

likelihood - $p(\vec{t} | \vec{w}) = \prod_{n=1}^{N} \mathcal{N}(t_n | \vec{w}^T \vec{\phi}(\vec{x}_n), \beta^{-1})$

prior - $p(\vec{w}) = \mathcal{N}(\vec{w} | \vec{m}_0, S_0)$

posterior - $p(\vec{w} | \vec{t}) = \mathcal{N}(\vec{w} | (S_0^{-1} + \Phi^T \beta I \Phi)^{-1} (\Phi^T \beta I \vec{t} + S_0^{-1} \vec{m}_0), (S_0^{-1} + \Phi^T \beta I \Phi)^{-1})$

$= \mathcal{N}(\vec{w} | \vec{m}_N, S_N)$

where $\vec{m}_N = S_N (S_0^{-1} \vec{m}_0 + \beta \Phi^T \vec{t})$

$S_N = (S_0^{-1} + \beta \Phi^T \Phi)^{-1}$

<u>maximum posterior</u> $\vec{w}$ is the mode of $p(\vec{w} | \vec{t})$

$\Rightarrow \vec{w}_{MAP} = \vec{m}_N$

ex: $S_0 = \alpha^{-1} I$ is "infinitely broad"/uniform over the parameter space and offers no information

sequential method

- begin with prior $p(\vec{w})$
- update the posterior $p(\vec{w} | \vec{t})$ w/ data points
- use posterior $p(\vec{w} | \vec{t})$ as prior $p(\vec{w})$ for the next iteration, and repeat

special case: mean 0, <u>isotropic</u> Gaussian prior $p(\vec{w} | \alpha) = \mathcal{N}(\vec{w} | \vec{0}, \alpha^{-1} I)$

then $p(\vec{w} | \vec{t}) = \mathcal{N}(\vec{w} | \vec{m}_N, S_N)$

where $\vec{m}_N = S_N (\alpha I \vec{0} + \beta \Phi^T \vec{t}) = \beta S_N \Phi^T \vec{t}$

$S_N = (\alpha I + \beta \Phi^T \Phi)^{-1}$

$\ln p(\vec{w} | \vec{t}) = -\frac{\beta}{2} \sum_{n=1}^{N} (t_n - \vec{w}^T \vec{\phi}(\vec{x}_n))^2 - \frac{\alpha}{2} \vec{w}^T \vec{w} + \text{const}$ = <u>regularized least-squares objective</u>

note: isotropic means cov is in the form $\alpha^{-1} I$

predictive distribution: want to determine t given a new input $\vec{x}$

$$p(t|\vec{t},\alpha,\beta) = \int p(t|\vec{w},\beta)\, p(\vec{w}|\vec{t},\alpha,\beta)\, d\vec{w}$$

which is given by

$$p(t|\vec{x},\vec{t},\alpha,\beta) = \mathcal{N}(t|\vec{m}_N^T\vec{\phi}(\vec{x}), \sigma_N^2(\vec{x}))$$

$$\sigma_N^2(\vec{x}) = \frac{1}{\beta} + \vec{\phi}(\vec{x})^T S_N \vec{\phi}(\vec{x})$$

Evidence Approximation

maximize <u>evidence function</u> $p(\vec{t}|\alpha,\beta) = \int p(\vec{t}|\vec{w},\beta)\, p(\vec{w}|\alpha)\, d\vec{w}$

where $p(\vec{t}|\vec{w},\beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\vec{w}^T\vec{\phi}(\vec{x}_n), \beta^{-1})$

$$p(\vec{w}|\alpha) = \mathcal{N}(\vec{w}|\vec{0}, \alpha^{-1}I)$$

$$p(\vec{t}|\alpha,\beta) = \int \left(\frac{\beta}{2\pi}\right)^{N/2}\left(\frac{1}{2\pi}\right)^{M/2} |\alpha^{-1}I|^{-1/2} \exp\left\{-\frac{\beta}{2}\sum_{n=1}^{N}((t_n - \vec{w}^T\vec{\phi}(\vec{x}_n))^2\right\} \exp\left\{-\frac{\alpha}{2}\vec{w}^T\vec{w}\right\} d\vec{w}$$

$$= \left(\frac{\beta}{2\pi}\right)^{N/2}\left(\frac{\alpha}{2\pi}\right)^{M/2} \int \exp\left\{-\frac{\beta}{2}\|\vec{t} - \Phi\vec{w}\|^2 - \frac{\alpha}{2}\vec{w}^T\vec{w}\right\} d\vec{w}$$

$$= \left(\frac{\beta}{2\pi}\right)^{N/2}\left(\frac{\alpha}{2\pi}\right)^{M/2} \int \exp\left\{-\frac{1}{2}(\vec{w}-\vec{m}_N)^T A(\vec{w}-\vec{m}_N) - \frac{\beta}{2}\|\vec{t} - \Phi\vec{m}_N\|^2 - \frac{\alpha}{2}\vec{m}_N^T\vec{m}_N\right\} d\vec{w}$$

$$= \left(\frac{\beta}{2\pi}\right)^{N/2}\left(\frac{\alpha}{2\pi}\right)^{M/2} \exp\left\{-\frac{\beta}{2}\|\vec{t} - \Phi\vec{m}_N\|^2 - \frac{\alpha}{2}\vec{m}_N^T\vec{m}_N\right\} \int \exp\left\{-\frac{1}{2}(\vec{w}-\vec{m}_N)^T A(\vec{w}-\vec{m}_N)\right\} d\vec{w}$$

$$= \left(\frac{\beta}{2\pi}\right)^{N/2}\left(\frac{\alpha}{2\pi}\right)^{M/2} \exp\left\{-\frac{\beta}{2}\|\vec{t} - \Phi\vec{m}_N\|^2 - \frac{\alpha}{2}\vec{m}_N^T\vec{m}_N\right\} (2\pi)^{M/2} |A|^{-1/2}$$

$$\ln p(\vec{t}|\alpha,\beta) = \frac{M}{2}\ln\alpha + \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \frac{\beta}{2}\|\vec{t} - \Phi\vec{m}_N\|^2 - \frac{\alpha}{2}\vec{m}_N^T\vec{m}_N - \frac{1}{2}\ln|A|$$

suppose $(\beta\Phi^T\Phi)\vec{u}_i = \lambda_i\vec{u}_i$, i.e. λ_i is an eigenvalue of $\beta\Phi^T\Phi$

$\boxed{\alpha = \sum_{i=1}^{M} \frac{\lambda_i}{\vec{m}_N^T\vec{m}_N(\alpha+\lambda_i)}}$ maximizes $\ln p(\vec{t}|\alpha,\beta)$ $\qquad \frac{1}{\beta} = \frac{1}{N - \sum_{i=1}^{M}\frac{\lambda_i}{\alpha+\lambda_i}} \sum_{n=1}^{N}\left\{t_n - \vec{m}_N^T\vec{\phi}(\vec{x}_n)\right\}^2$

procedure

1. choose α,
2. find $\vec{m}_N = \beta S_N \Phi^T\vec{t}$
3. estimate $\alpha_2 = \sum_{i=1}^{M} \frac{\lambda_i}{\vec{m}_N^T\vec{m}_N(\alpha+\lambda_i)}$
4. set $\alpha_1 = \alpha_2$ and repeat

1. choose β_1
2. find $\vec{m}_N = \beta_1 S_N \Phi^T\vec{t}$
3. estimate $\frac{1}{\beta_2} = \frac{1}{N - \sum_{i=1}^{M}\frac{\lambda_i}{\alpha+\lambda_i}} \sum_{n=1}^{N}\left\{t_n - \vec{m}_N^T\vec{\phi}(\vec{x}_n)\right\}^2$
4. set $\beta_1 = \beta_2$ and repeat

Discriminant Functions

goal: classify data, assuming classes are disjoint and therefore separable by <u>decision boundaries</u>

linear classification: data is linearly separable (decision surfaces are (D-1)-dim hyperplanes)

setup: K-coding scheme for K classes (one-hot encoding)

ex: $K=2 \Rightarrow \vec{t} = \begin{bmatrix}1\\0\end{bmatrix}$ or $\begin{bmatrix}0\\1\end{bmatrix}$

<u>discriminant function</u> - simplest approach, assigns $\vec{x}$ to a class

generalized linear model - $y(\vec{x}) = f(\vec{w}^T\vec{x} + w_0)$, $f()$ is a nonlinear function known as an <u>activation function</u>

K=2 case

$y(\vec{x}) = 0$ is the decision boundary

distance from origin to surface is determined by w_0

distance of point to surface is $\frac{|y(\vec{x})|}{\|\vec{w}\|}$

K>2 case

one-versus-one approach: use $K(K-1)/2$ binary classifiers

problem: ambiguous regions

$y_k(\vec{x}) = \vec{w}_k^T \vec{x} + w_{k0}$, $\vec{x} \in C_j$ where $j = \underset{i}{\text{argmax}}\, y_i(x)$

$y_i(x) = y_j(x)$ is the decision boundary between C_i and C_j

least squares: $\vec{y}(\vec{x}) = \tilde{W}^T \tilde{x}$

$$\tilde{W} = \begin{bmatrix} w_{10} & w_{20} & \cdots & w_{K0} \\ \vec{w}_1 & \vec{w}_2 & & \vec{w}_K \end{bmatrix} \in \mathbb{R}^{(D+1)\times K}, \quad \tilde{x} = \begin{bmatrix} 1 \\ \vec{x} \end{bmatrix}, \quad T = \begin{bmatrix} -\vec{t}_1^T- \\ -\vec{t}_2^T- \\ \vdots \\ -\vec{t}_N^T- \end{bmatrix}$$

$\vec{x} \in C_j$ where $j = \underset{i}{\text{argmax}} \underbrace{(\vec{y}(\vec{x}))_i}_{\text{ith entry of } \vec{y}(\vec{x})}$

error $E_D(\tilde{W}) = \frac{1}{2} Tr\{(\tilde{X}\tilde{W} - T)^T (\tilde{X}\tilde{W} - T)\}$

closed form solution $\boxed{\tilde{W} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T T = \tilde{X}^+ T}$

problem: weak against outliers

Probabilistic Discriminative Models

logistic sigmoid $\sigma(a) = \frac{1}{1+\exp(-a)}$

for $K=2$,

$$p(C_1|\vec{x}) = \frac{p(\vec{x}|C_1)p(C_1)}{p(\vec{x}|C_1)p(C_1) + p(\vec{x}|C_2)p(C_2)} = \frac{1}{1+\exp(-a)} = \sigma(a) \quad \text{where } a = \ln \frac{p(\vec{x}|C_1)p(C_1)}{p(\vec{x}|C_2)p(C_2)}$$

for $K>2$

$$p(C_k|\vec{x}) = \frac{p(\vec{x}|C_k)p(C_k)}{\sum_j p(\vec{x}|C_j)p(C_j)} = \underbrace{\frac{\exp(a_k)}{\sum_j \exp(a_j)}}_{\substack{\text{normalized exponential} \\ \text{aka softmax}}} \quad \text{where } a_j = \ln p(\vec{x}|C_j)p(C_j)$$

logistic regression

posterior distribution $p(C_1|\vec{\Phi}) = y(\vec{\Phi}) = \sigma(\vec{w}^T\vec{\Phi})$

note: $\frac{d}{da}\sigma(a) = \sigma(a)(1-\sigma(a))$

$$p(\vec{t}|\vec{w}) = \prod_{n=1}^{N} p(t_n|\vec{w})$$
$$= \prod_{n=1}^{N} (1 - p(C_1|\vec{\Phi}_n))^{1-t_n} p(C_2|\vec{\Phi}_n)^{t_n}$$

cross-entropy error function

$$E(\vec{w}) = -\ln p(\vec{t}|\vec{w}) = -\sum_{n=1}^{N} \{ t_n \ln p(C_1|\vec{\Phi}_n) + (1-t_n) \ln(1 - p(C_1|\vec{\Phi}_n)) \}$$
$$= -\sum_{n=1}^{N} \{ t_n \ln \sigma(\vec{w}^T\vec{\Phi}_n) + (1-t_n) \ln(1 - \sigma(\vec{w}^T\vec{\Phi}_n)) \}$$

$$\nabla E(\vec{w}) = -\sum_{n=1}^{N} \{ t_n \frac{1}{\sigma(\vec{w}^T\vec{\Phi}_n)} \sigma(\vec{w}^T\vec{\Phi}_n)(1-\sigma(\vec{w}^T\vec{\Phi}_n))\vec{\Phi}_n + (1-t_n)\frac{1}{1-\sigma(\vec{w}^T\vec{\Phi}_n)}[-\sigma(\vec{w}^T\vec{\Phi}_n)(1-\sigma(\vec{w}^T\vec{\Phi}_n))]\vec{\Phi}_n$$
$$= -\sum_{n=1}^{N} [(t_n - t_n\sigma(\vec{w}^T\vec{\Phi}_n))\vec{\Phi}_n - (\sigma(\vec{w}^T\vec{\Phi}_n) - t_n\sigma(\vec{w}^T\vec{\Phi}_n))\vec{\Phi}_n]$$
$$= -\sum_{n=1}^{N} (t_n - \sigma(\vec{w}^T\vec{\Phi}_n))\vec{\Phi}_n$$
$$= \sum_{n=1}^{N} (\sigma(\vec{w}^T\vec{\Phi}_n) - t_n)\vec{\Phi}_n$$

use iterative algorithm

Maximum Margin Classifier

support vector machines - choose $\vec{w}$ and b to maximize the margin (dist b/w decision boundary and points)

note: distance of $\vec{x}_n$ to decision surface is $\frac{t_n y(\vec{x}_n)}{\|\vec{w}\|} = \frac{t_n(\vec{w}^T\vec{\phi}(\vec{x}_n)+b)}{\|\vec{w}\|}$

maximum margin solution: $\underset{\vec{w},b}{\text{argmax}} \left\{ \frac{1}{\|\vec{w}\|} \min_n \left[t_n(\vec{w}^T\vec{\phi}(\vec{x}_n)+b) \right] \right\}$

note: rescaling $\vec{w}$ doesn't change the distance

WLOG, let margin = 1, i.e. $t_n(\vec{w}^T\vec{\phi}(\vec{x}_n)+b) \geq 1$ for $n=1,\dots,N$

$t_n(\vec{w}^T\vec{\phi}(\vec{x}_n)+b)=1$ if $\vec{x}_n$ is closest to decision surface

maximum margin solution: $\boxed{\underset{\vec{w},b}{\text{argmin}} \frac{1}{2}\|\vec{w}\|^2 \text{ s.t. } t_n(\vec{w}^T\vec{\phi}(\vec{x}_n)+b) \geq 1 \text{ for } n=1,\dots,N}$

use Lagrange multipliers

$L(\vec{w}, b, \vec{a}) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{n=1}^N a_n \{ t_n(\vec{w}^T\vec{\phi}(\vec{x}_n)+b-1) \}$

$\frac{\partial}{\partial \vec{w}} L(\vec{w}, b, \vec{a}) = 0 \Rightarrow \vec{w} = \sum_{n=1}^N a_n t_n \vec{\phi}(\vec{x}_n)$

$\frac{\partial}{\partial b} L(\vec{w}, b, \vec{a}) = 0 \Rightarrow 0 = \sum_{n=1}^N a_n t_n$

$$\boxed{\tilde{L}(\vec{a}) = \sum_{n=1}^N a_n - \frac{1}{2}\sum_{n=1}^N \sum_{m=1}^M a_n a_m t_n t_m k(\vec{x}_n, \vec{x}_m) \quad \text{where } a_n \geq 0 \text{ for } n=1,\dots,N \text{ and } \sum_{n=1}^N a_n t_n = 0} \quad (*)$$

solve $\tilde{L}$ for $\vec{a}$ then recover $\vec{w}$ and b

predict with $y(\vec{x}) = \sum_{n=1}^N a_n t_n k(\vec{x}, \vec{x}_n) + b$

note: the Karush-Kuhn Tucker (KKT) conditions for $(*)$ are

$a_n \geq 0$

$t_n y(\vec{x}_n) - 1 \geq 0$

$a_n \{ t_n y(\vec{x}_n) - 1 \} = 0$

$\Rightarrow$ either $a_n = 0$ or $t_n y(\vec{x}_n) = 1$

any point where $a_n = 0$ doesn't contribute to predictions

points where $t_n y(\vec{x}_n) = 1$ are **support vectors**, i.e. they lie in the maximum margin hyperplane

hard margin - data must be linearly separable

minimize error function $\sum_{n=1}^N E_\infty(y(\vec{x}_n)t_n - 1) + \lambda\|\vec{w}\|^2$

where $E_\infty(z) = \begin{cases} 0 & \text{if } z \geq 0 \text{ and } \lambda > 0 \\ \infty & \text{otherwise} \end{cases}$

soft margin - introduce **slack variable** $\xi_n \geq 0$ to allow some "incorrect" points

$\xi_n = 0$ for correctly classified data, $\xi_n = |t_n - y(\vec{x}_n)|$ for other points

minimize $C\sum_{n=1}^N \xi_n + \frac{1}{2}\|\vec{w}\|^2$ s.t. $t_n(\vec{w}^T\vec{\phi}(\vec{x}_n)+b) \geq 1 - \xi_n$

Perceptron

perceptron: a two class model of the form $y(\vec{x}) = f(\vec{w}^T\vec{\phi}(\vec{x}_n))$ where $f(a) = \begin{cases} +1 & a \geq 0 \\ -1 & a < 0 \end{cases}$

minimize **perceptron criterion** $E_p(\vec{w}) = -\sum_{n \in \mathcal{M}} \vec{w}^T\vec{\phi}(\vec{x}_n) t_n$ where $\mathcal{M}$ is the set of misclassified points

SGD on E_p

1. initialize $\vec{w}^{(0)}$
2. for $t=1,\dots,K$ randomly sample n

   if $\vec{w}^{(t-1)T}\vec{\phi}(\vec{x}_n) < 0$, update $\vec{w}^{(t)} = \vec{w}^{(t-1)} + \eta\vec{\phi}(\vec{x}_n)t_n$

   else, $\vec{w}^{(t)} = \vec{w}^{(t-1)}$

perceptron convergence theorem - if data is linearly separable, perceptron algorithm guaranteed to converge in finite # of steps

Feed-forward Neural Networks

suppose D input vars $x_1, \dots, x_D$; let $x_0 = 1$

each layer consists of multiplication by weight matrix and a nonlinear activation function

in layer 1, $\underbrace{a_j}_{\text{activations}} = \sum_{i=0}^{D} \underbrace{w_{ji}^{(1)}}_{\text{weights}} x_i = \vec{w}_j^{(1)} \vec{x}$ for $j = 1, \dots, M$

use a nonlinear activation function $h()$, hidden units $z_j = h(a_j)$

in layer 2, $\underbrace{a_k}_{\substack{\text{output unit} \\ \text{activations}}} = \sum_{j=0}^{M} w_{kj}^{(2)} z_j$

overall 2-layer network function, sometimes known as multilayer perceptron

$$\boxed{y_k(\vec{x}, \vec{w}) = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i \right) \right)}$$

evaluating the function is forward propagation

note: neural networks are universal approximators

a 2-layer NN can uniformly approximate any cont. func on compact input data to arbitrary accuracy w/ a sufficient number of hidden units and a broad range of activation functions

note: weight-space symmetry guarantees that no NN is unique

ex: if $h()$ is $\tanh()$, and we negate all weights in layer 1, we can negate all the weights in layer 2 to produce the same output (since $\tanh$ is odd)

$$\sum_{j=0}^{M} w_{kj}^{(2)} \tanh\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i \right) = -- \sum_{j=0}^{M} w_{ki}^{(2)} \tanh\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i \right) = \sum_{j=0}^{M} (-w_{kj}^{(2)}) \left(-\tanh\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i \right) \right) = \sum_{j=0}^{M} (-w_{kj}^{(2)}) \tanh\left(\sum_{i=0}^{D} (-w_{ji}^{(1)}) x_i \right)$$

training the model

assume: target has Gaussian distribution, i.e. $p(t \mid \vec{x}, \vec{w}) = \mathcal{N}(t \mid y(\vec{x}, \vec{w}), \beta^{-1})$

training data is i.i.d., i.e. likelihood is $p(\vec{t} \mid X, \vec{w}, \beta) = \prod_{i=1}^{N} p(t_n \mid \vec{x}, \vec{w}) = \prod_{i=1}^{N} \mathcal{N}(t_n \mid y(\vec{x}, \vec{w}), \beta^{-1})$

goal: maximize the log likelihood

$$-\frac{\beta}{2} \sum_{n=1}^{N} \{ y(\vec{x}, \vec{w}) - t_n \}^2 + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi)$$

equivalent to minimizing the sum of squares error

$$E(\vec{w}) = \sum_{n=1}^{N} (y(\vec{x}_n, \vec{w}) - t_n)^2$$

use SGD to find a local minimum

note: $E(\vec{w})$ is not convex, so we might find a local minimum instead of the global minimum

weight-space symmetry $\Rightarrow$ multiple local minima $\Rightarrow$ train the model multiple times and select the best

note: $E(\vec{w}) = \sum_{n=1}^{N} E_n(\vec{w})$ where $E_n(\vec{w}) = (y(\vec{x}_n, \vec{w}) - t_n)^2$

update weights:

1) initialize $\vec{w}^{(0)}$, could be random
2) pick a random x_n
3) compute $\nabla E_n(\vec{w}^{(t)})$
4) $\boxed{\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta \nabla E_n(\vec{w}^{(t)})}$

Back Propagation

1 layer: $E_n = \frac{1}{2}\sum_{k=1}^{K}(y_{nk} - t_{nk})^2 = \frac{1}{2}\sum_{k=1}^{K}(y_k(\vec{x}_n, \vec{w}) - t_{nk})^2 = \frac{1}{2}\sum_{k=1}^{K}\left(\sum_{i=0}^{D}(w_{ki}x_{ni}) - t_{nk}\right)^2$

$\frac{\partial E_n}{\partial w_{ji}} = (y_{nj} - t_{nj})\,x_{ni}$

2 layer: $E_n = \frac{1}{2}\sum_{k=1}^{K}(y_{nk} - t_{nk})^2 = \frac{1}{2}\sum_{k=1}^{K}(y_k(\vec{x}_n, \vec{w}) - t_{nk}) = \frac{1}{2}\sum_{k=1}^{K}\left(\sum_{j=0}^{M}h(a_j) - t_{nk}\right)$

$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}}$

$= \delta_j \frac{\partial a_j}{\partial w_{ji}}$ note: $a_j = \sum_{i=0}^{D} w_{ji}z_i$

$= \boxed{\delta_j z_i}$ $\frac{\partial a_j}{\partial w_{ji}} = z_i$

compute δ_j

for output units (last layer): $\delta_k = y_k - t_k$

for hidden units, use chain rule:

$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial a_j}$ for all $a_k = f(a_j)$ $\quad a_k = \sum_{j=0}^{M}\vec{w}_{kj}h(a_j)$

$= \sum_k \delta_k \vec{w}_{kj} h'(a_j)$ $\quad \frac{\partial a_k}{\partial a_j} = \vec{w}_{kj}h'(a_j)$

$= \boxed{h'(a_j)\sum_k \delta_k \vec{w}_{kj}}$

then update weights with $\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta\nabla E_n(\vec{w}^{(t)})$

Principal Component Analysis (PCA)

applications: dimension reduction
data visualization
feature extraction
topic modeling
etc

other names: Karhunen-Loève theorem (signal processing)
proper orthogonal decomposition (mechanical engineering)
Eckart-Young theorem (linear algebra)
empirical orthogonal functions (meteorological science)

idea: orthogonally project data onto a lower dimensional subspace

maximize variance of projections, or
minimize average projection distance
] same algorithm

setup: data $\{x_i\}_{i=1}^{N} \in \mathbb{R}^D$

goals: project data onto an M-dimensional subspace, $M < D$, called the principal subspace

assumptions: $M = 1$ (generalize with induction)

data is centered, ie. $\bar{x} = \frac{1}{N}\sum_{i=1}^{N}\vec{x}_i = \vec{0}$ where $\bar{x}$ is the sample mean

1) write errors between points and their projections onto principal subspace (a line through the origin since M=1)

projection $= (\vec{v}_1^T\vec{x}_n)\vec{v}_1$

mean of projections is $\vec{0}$

the error between points and projections

$\|\vec{x}_n - (\vec{v}_1^T\vec{x}_n)\vec{v}_1\|^2 = [\vec{x}_n - (\vec{v}_1^T\vec{x}_n)\vec{v}_1]^T[\vec{x}_n - (\vec{v}_1^T\vec{x}_n)\vec{v}_1]$

$= \vec{x}_n^T\vec{x}_n - 2(\vec{v}_1^T\vec{x}_n)\vec{v}_1^T\vec{x}_n + (\vec{v}_1^T\vec{x}_n)^2\vec{v}_1^T\vec{v}_1$

$= \|\vec{x}_n\|^2 - 2(\vec{v}_1^T\vec{x}_n)^2 + (\vec{v}_1^T\vec{x}_n)^2\|\vec{v}_1\|^2$

$= \|\vec{x}_n\|^2 - (\vec{v}_1^T\vec{x}_n)^2$

2) choose basis vector(s) to minimize the mean error

mean error = $\boxed{\frac{1}{N}\sum_{n=1}^{N}\|\vec{x}_n - (\vec{v}_1^T\vec{x}_n)\vec{v}_1\|^2 = \frac{1}{N}\sum_{n=1}^{N}\left[\|\vec{x}_n\|^2 - (\vec{v}_1^T\vec{x}_n)^2\right]}$

note: minimize mean error $\frac{1}{N}\sum_{n=1}^{N}\left[\|\vec{x}_n\|^2 - (\vec{v}_1^T\vec{x}_n)^2\right]$

$\Leftrightarrow$ maximize $\frac{1}{N}\sum_{n=1}^{N}(\vec{v}_1^T\vec{x}_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(\vec{v}_1^T\vec{x}_n - \underbrace{\vec{v}_1^T\bar{x}}_{0})^2$

$\Leftrightarrow$ maximize sample variance of projections

note: $\frac{1}{N}\sum_{n=1}^{N}(\vec{v}_1^T\vec{x}_n - \vec{v}_1^T\bar{x}_n)^2 = \vec{v}_1^T\underbrace{\left[\frac{1}{N}\sum_{n=1}^{N}(\vec{x}_n-\bar{x})(\vec{x}_n-\bar{x})^T\right]}_{\text{sample covariance matrix}}\vec{v}_1$

constrain $\|\vec{v}_1\| = 1$ so objective is bounded

let $f(\vec{v}_1) = \vec{v}_1^T S \vec{v}_1$ and $g(\vec{v}_1) = \vec{v}_1^T\vec{v}_1 - 1$

$\nabla f(\vec{v}_1) = \lambda_1 \nabla g(\vec{v}_1)$

$2S\vec{v}_1 = 2\lambda_1\vec{v}_1$

$S\vec{v}_1 = \lambda_1\vec{v}_1$

to maximize, λ_1 has to be the largest eigenvalue

PCA algorithm

1) form sample covariance matrix $S = \frac{1}{N}\sum_{n=1}^{N}(\vec{x}_n-\bar{x})(\vec{x}_n-\bar{x})^T$

2) compute eigenvectors associated to the M largest eigenvalues of S, $\{\vec{v}_i\}_{i=1}^{M}$

3) compute coordinates of the projection of the centered data $(\vec{x}_n-\bar{x})$ onto the principal subspace

basis: $\{\vec{v}_i\}_{i=1}^{M}$

coefficients: $\vec{v}_i^T(\vec{x}_n-\bar{x})$

the variance captured by the projection is $\sum_{j=1}^{M}\lambda_j$

the missed variance, or <u>distortion</u>, is $\sum_{j=M+1}^{D}\lambda_j$

<u>Probabilistic PCA</u>

let $\vec{z}$ be a latent variable corresponding to the principal subspace

prior: $p(\vec{z}) = \mathcal{N}(\vec{z}\,|\,\vec{0}, I)$

conditional dist $p(\vec{x}|\vec{z}) = \mathcal{N}(\vec{x}\,|\,W\vec{z}+\vec{\mu}, \sigma^2 I)$

note: $\vec{z}$ "generates" $\vec{x}$ because $\vec{x} = W\vec{z} + \vec{\mu} + \vec{\varepsilon}$ where $\vec{\varepsilon} \sim \mathcal{N}(\vec{\varepsilon}\,|\,\vec{0}, \sigma^2 I)$

find $p(\vec{x}) = \int p(\vec{x}|\vec{z})\,p(\vec{z})\,d\vec{z}$

$\mathbb{E}[\vec{x}] = \mathbb{E}[W\vec{z}+\vec{\mu}+\vec{\varepsilon}] = W\mathbb{E}[\vec{z}] + \vec{\mu} + \mathbb{E}[\vec{\varepsilon}] = \vec{\mu}$

$\text{cov}[\vec{x}] = \mathbb{E}[(W\vec{z}+\vec{\varepsilon})(W\vec{z}+\vec{\varepsilon})^T]$

$= W\mathbb{E}[\vec{z}\vec{z}^T]W^T + 2\mathbb{E}[\vec{z}^T W^T\vec{\varepsilon}] + \mathbb{E}[\vec{\varepsilon}\vec{\varepsilon}^T]$

$= WW^T + 2\mathbb{E}[\text{tr}(\vec{\varepsilon}\vec{z}^T W^T)] + \sigma^2 I$

$= WW^T + 2\,\text{tr}(\underbrace{\mathbb{E}[\vec{\varepsilon}\vec{z}^T]}_{\text{0 bc cov of 2 ind r.v.s}}W^T) + \sigma^2 I$

$= WW^T + \sigma^2 I$

$\Rightarrow \boxed{p(\vec{x}) = \mathcal{N}(\vec{x}\,|\,\vec{\mu}, WW^T + \sigma^2 I)}$

recall: when $p(\vec{x}) = \mathcal{N}(\vec{x}|\vec{\mu}, \Lambda^{-1})$ is a marginal Gaussian
and $p(\vec{y}|\vec{x}) = \mathcal{N}(\vec{y}|A\vec{x}+\vec{b}, L^{-1})$ is a conditional Gaussian
then $p(\vec{y}) = \mathcal{N}(\vec{y}|A\vec{\mu}+\vec{b}, L^{-1}+A\Lambda^{-1}A^T)$
and $p(\vec{x}|\vec{y}) = \mathcal{N}(\vec{x}|\Sigma\{A^TL(\vec{y}-\vec{b})+\Lambda\vec{\mu}\}, \Sigma)$
where $\Sigma = (\Lambda+A^TLA)^{-1}$

find posterior: $p(\vec{z}|\vec{x}) = \mathcal{N}(\vec{z}|(I+W^T\frac{1}{\sigma^2}IW)^{-1}\{W\frac{1}{\sigma^2}I(\vec{x}-\vec{\mu})+I\vec{0}\}, (I+W^T\frac{1}{\sigma^2}IW)^{-1})$
$= \mathcal{N}(\vec{z}|M^{-1}W^T(\vec{x}-\vec{\mu}), \sigma^2M^{-1})$
where $M = W^TW+\sigma^2I$

likelihood $p(X|\vec{\mu}, W, \sigma^2) = \prod_{n=1}^{N}\mathcal{N}(\vec{x}_n|\vec{\mu}, WW^T+\sigma^2I)$

log likelihood $\ln p(X|\vec{\mu}, W, \sigma^2) = \sum_{n=1}^{N}[-\frac{D}{2}\ln(2\pi)-\frac{1}{2}\ln|WW^T+\sigma^2I|-\frac{1}{2}(\vec{x}_n-\vec{\mu})^T(WW^T+\sigma^2I)^{-1}(\vec{x}_n-\vec{\mu})]$

$\vec{\mu}_{ML} = \bar{x}$

$W_{ML} = U_M(L_M-\sigma^2I)^{1/2}R$
- U_M: cols are M eigenvectors
- L_M: diagonal matrix of M largest eigenvalues
- R: orthogonal matrix

$\sigma^2_{ML} = \frac{1}{D-M}\sum_{i=M+1}^{D}\lambda_i$

$\mathbb{E}[\vec{z}|\vec{x}] = M^{-1}W_{ML}^T(\vec{x}-\bar{x})$ in latent space maps to $W_{ML}\mathbb{E}[\vec{z}|\vec{x}]+\vec{\mu}_{ML}$ in data space
where $M = W^TW+\sigma^2I$

Kernel PCA

given: $\{\vec{x}_n\} \in \mathbb{R}^D$ of N observations

principal components are given by the eigenvectors of the data covariance matrix $S = \frac{1}{N}\sum_{n=1}^{N}\vec{x}_n\vec{x}_n^T$
$S\vec{u}_i = \lambda_i\vec{u}_i$ with $\|\vec{u}_i\| = 1$

consider a nonlinear transformation $\vec{\Phi}: \mathbb{R}^D \to \mathbb{R}^M$

each data point $\vec{x}_n$ is projected onto $\vec{\Phi}(\vec{x}_n)$ in feature space

assume $\sum_{n=1}^{N}\vec{\Phi}(\vec{x}_n) = 0$

sample covariance matrix in feature space is $C = \frac{1}{N}\sum_{n=1}^{N}\vec{\Phi}(\vec{x}_n)\vec{\Phi}(\vec{x}_n)^T$

principal components are given by the eigen vector equations
$C\vec{v}_i = \lambda_i\vec{v}_i$ (but solving this directly would be $O(M^3)$)

note: $\lambda_i\vec{v}_i = \frac{1}{N}\sum_{n=1}^{N}\vec{\Phi}(\vec{x}_n)[\vec{\Phi}(\vec{x}_n)^T\vec{v}_i]$, so $\vec{v}_i$ is a linear combination of $\vec{\Phi}(\vec{x}_n)$

write $\vec{v}_i = \sum_{n=1}^{N}a_{in}\vec{\Phi}(\vec{x}_n)$

then $\frac{1}{N}\sum_{n=1}^{N}\vec{\Phi}(\vec{x}_n)\vec{\Phi}(\vec{x}_n)^T\vec{v}_i = \lambda_i\vec{v}_i$

$\frac{1}{N}\sum_{n=1}^{N}\vec{\Phi}(\vec{x}_n)\vec{\Phi}(\vec{x}_n)^T\sum_{m=1}^{N}a_{im}\vec{\Phi}(\vec{x}_m) = \lambda_i\sum_{n=1}^{N}a_{in}\vec{\Phi}(\vec{x}_n)$

$\vec{\Phi}(\vec{x}_\ell)^T\frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{N}\vec{\Phi}(\vec{x}_n)\vec{\Phi}(\vec{x}_n)^T\vec{\Phi}(\vec{x}_m)a_{im} = \vec{\Phi}(\vec{x}_\ell)^T\lambda_i\sum_{n=1}^{N}\vec{\Phi}(\vec{x}_n)a_{in}$

$\frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{N}k(\vec{x}_\ell, \vec{x}_n)k(\vec{x}_n, \vec{x}_m)a_{im} = \lambda_i\sum_{n=1}^{N}k(\vec{x}_\ell, \vec{x}_n)$

where $k(\vec{x}_n, \vec{x}_m) = \vec{\Phi}(\vec{x}_n)^T\vec{\Phi}(\vec{x}_m)$

this is a system of linear equations, so we can write it as $K^2\vec{a}_i = N\lambda_iK\vec{a}_i$

find $\vec{a}_i$ by solving $\boxed{K\vec{a}_i = \lambda_iN\vec{a}_i}$

note: $\|\vec{v}_i\| = 1 \Rightarrow 1 = \vec{v}_i^T\vec{v}_i = (\sum_{n=1}^{N}a_{in}\vec{\Phi}(\vec{x}_n))^T(\sum_{m=1}^{N}a_{im}\vec{\Phi}(\vec{x}_m)) = \vec{a}_i^TK\vec{a}_i = \lambda_iN\vec{a}_i^T\vec{a}_i$

note: we can recover $\vec{v}_i$ with $\vec{a}_i$ and get the principal components

note: to find the projection of a new point $\vec{x}$

coefficients $y_i(\vec{x}) = \vec{\Phi}(\vec{x})^T \vec{v}_i = \sum_{n=1}^{N} a_{in} \vec{\Phi}(\vec{x})^T \vec{\Phi}(\vec{x}_n) = \sum_{n=1}^{N} a_{in} k(\vec{x}, \vec{x}_n)$

applications of PCA

data compression - storing data directly = ND space

storing compressed data = D + NM + MD space

store $\underbrace{\sum_{j=1}^{D} (\vec{v}_i^T \vec{x}) \vec{v}_i}_{\text{D space}}$, $\underbrace{\vec{v}_i^T \vec{x}_n}_{\text{NM space}}$, and $\underbrace{\vec{v}_i}_{\text{MD space}}$ for $i = 1, \ldots, M$

saves space if $N \gg M$ or $D \gg M$

data visualization - project onto $\mathbb{R}^2$ or $\mathbb{R}^3$ and plotting

data whitening - transform new data s.t. it has mean 0 and unit covariance

stack eigenvector equations into $SU = UL$

$U \in \mathbb{R}^{D \times D}$ has columns $\vec{u}_i$

$L \in \mathbb{R}^{D \times D}$ is a diagonal matrix s.t. $L_{ii} = \lambda_i$

transform data using $\vec{y}_n = L^{-1/2} U^T (\vec{x}_n - \bar{x})$

$\sum_{n=1}^{N} \vec{y}_n = \sum_{n=1}^{N} L^{-1/2} U^T (\vec{x}_n - \bar{x})$

$= L^{-1/2} U^T (\sum_{n=1}^{N} \vec{x}_n - \bar{x})$

$= L^{-1/2} U^T \vec{0} = \vec{0}$

$\Rightarrow$ mean = 0

$\frac{1}{N} \sum_{n=1}^{N} (\vec{y}_n - \vec{0})(\vec{y}_n - \vec{0})^T = \frac{1}{N} \sum_{n=1}^{N} L^{-1/2} U^T (\vec{x}_n - \bar{x})(\vec{x}_n - \bar{x})^T U L^{-1/2}$

$= L^{-1/2} U^T [\frac{1}{N} \sum_{n=1}^{N} (\vec{x}_n - \bar{x})(\vec{x}_n - \bar{x})^T] U L^{-1/2}$

$= L^{-1/2} U^T S U L^{-1/2}$

$= L^{-1/2} U^T U L L^{-1/2}$

$= L^{-1/2} L L^{-1/2} = I$

$\Rightarrow$ unit covariance

PCA for higher-dimensional data

if $N < D$, we can compute eigenvalues and eigenvectors of $S \in \mathbb{R}^{N \times N}$ instead of $S \in \mathbb{R}^{D \times D}$

let $X \in \mathbb{R}^{N \times D}$ where the nth row is $(\vec{x}_n - \vec{x})^T$

then covariance matrix $S = \frac{1}{N} X^T X$

the eigenvalue equations are $\frac{1}{N} X^T X \vec{u}_i = \lambda_i \vec{u}_i$

$\frac{1}{N} X X^T (X \vec{u}_i) = \lambda_i (X \vec{u}_i)$

$\frac{1}{N} X X^T (\vec{v}_i) = \lambda_i \vec{v}_i$

so $XX^T \in \mathbb{R}^{N \times N}$ has the same eigenvalues λ_i as $X^T X \in \mathbb{R}^{D \times D}$

and the eigenvectors differ by a linear transformation $\vec{v}_i = X \vec{u}_i$

K-Means Clustering

$\{\vec{\mu}_k\}_{k=1}^{K} \in \mathbb{R}^D$ represents the center of each cluster

$r_{nk} \in \{0, 1\}$ is a binary indicator for if the nth point belongs to the kth cluster

i.e. $r_{nk} = \begin{cases} 1 & \text{if } \vec{x}_n \in \text{cluster } k \\ 0 & \text{otherwise} \end{cases}$

then the distortion (error measure) is $J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \| \vec{x}_n - \vec{\mu}_k \|^2$

K-means algorithm

1) fix $\vec{\mu}_k$ and optimize J wrt r_{nk}

$r_{nk} = 1$ for $k = \arg\min_j \|\vec{x}_n - \vec{\mu}_j\|^2$

$r_{nk} = 0$ otherwise

i.e. assign each point to its nearest center

2) fix r_{nk} and optimize J wrt $\vec{\mu}_k$

$$\frac{\partial J}{\partial \vec{\mu}_k} = -2\sum_{n=1}^{N} r_{nk}(\vec{x}_n - \vec{\mu}_k) = 0$$

$$\sum_{n=1}^{N} r_{nk}\vec{x}_n = \sum_{n=1}^{N} r_{nk}\vec{\mu}_k$$

$$\vec{\mu}_k = \frac{\sum_{n=1}^{N} r_{nk}\vec{x}_n}{\sum_{n=1}^{N} r_{nk}} = \frac{\sum_{n=1}^{N} \vec{x}_n}{|C_k|}$$

where $C_k = \{n \mid r_{nk} = 1\} \Rightarrow |C_k|$ = mean of points in C_k

3) repeat

every iteration decreases J and $J \geq 0$ so convergence is guaranteed

might converge to local min

K-mediods: $J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk} v(\vec{x}_n, \vec{\mu}_k)$

K-means application

image segmentation and compression: reduce an image into K colors using $\mathbb{R}^3$ vectors (RGB value of pixels)

not super great because it doesn't consider spatial proximity of pixels

lossy data compression: store the cluster of each data point and the k centers

often called <u>vector quantization</u>

$\vec{\mu}_k$ are codebook vectors

<u>EM Algorithm</u> (General)

X = set of all data

Z = set of all latent variables

Θ = set of all model parameters

log likelihood $\ln p(X|\Theta) = \ln\{\sum_Z p(X, Z|\Theta)\}$

we only observe incomplete data X and not complete data $\{X, Z\}$

cannot maximize likelihood directly $\Rightarrow$ use iterative process

1. initialize Θ^{old}
2. evaluate $p(X|Z, \Theta^{old})$
3. evaluate $\Theta^{new} = \arg\max_\Theta Q(\Theta, \Theta^{old})$

where $Q(\Theta, \Theta^{old}) = \sum_Z p(X|Z, \Theta^{old}) \ln p(X, Z|\Theta)$

4. set $\Theta^{old} \leftarrow \Theta^{new}$, repeat until convergence

Mixtures of Gaussians

let $\vec{z}$ be a 1-of-K coding (aka one-hot coding) vector

$z_k \in \{0,1\}$ for $k=1,\dots,K$ and $\sum_{k=1}^{K} z_k = 1$

suppose $p(z_k=1) = \pi_k$

then $0 \le \pi_k \le 1$, $\sum_{k=1}^{K} \pi_k = 1$, and $p(\vec{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$

assume the conditional is Gaussian: $p(\vec{x} \mid z_k=1) = \mathcal{N}(\vec{x} \mid \vec{\mu}_k, \Sigma_k)$

$$p(\vec{x} \mid \vec{z}) = \prod_{k=1}^{K} \mathcal{N}(\vec{x} \mid \vec{\mu}_k, \Sigma_k)^{z_k}$$

Gaussian mixture model: $p(\vec{x}) = \sum_{\vec{z}} p(\vec{x} \mid \vec{z})\, p(\vec{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\vec{x} \mid \vec{\mu}_k, \Sigma_k)$

log likelihood $= \ln p(X \mid \vec{\pi}, \vec{\mu}, \Sigma) = \sum_{n=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\vec{x}_n \mid \vec{\mu}_k, \Sigma_k)\right)$

unable to solve via the usual ML approach b/c of singularities

define $\gamma(z_k) := p(z_k=1 \mid \vec{x}) = \frac{p(\vec{x}, \vec{z}_k)}{p(\vec{x})} = \frac{p(z_k=1)\,p(\vec{x} \mid z_k=1)}{\sum_{j=1}^{K} p(z_j=1)\,p(\vec{x} \mid z_j=1)} = \frac{\pi_k \mathcal{N}(\vec{x} \mid \vec{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\vec{x} \mid \vec{\mu}_j, \Sigma_j)}$ = "responsibility" that component k takes for "explaining" $\vec{x}$

EM algorithm for GMM

1. initialize $\vec{\mu}_k$, Σ_k, π_k
2. E: evaluate $\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\vec{x}_n \mid \vec{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\vec{x}_n \mid \vec{\mu}_j, \Sigma_j)}$
3. M: re-estimate parameters

$$\vec{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})\, \vec{x}_n$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\vec{x}_n - \vec{\mu}_k)(\vec{x}_n - \vec{\mu}_k)^T \quad ; \quad N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

$$\pi_k = \frac{N_k}{N}$$

4. repeat until convergence

note: $p(X, Z \mid \mu, \Sigma, \pi) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}} \mathcal{N}(\vec{x}_n \mid \vec{\mu}_k, \Sigma_k)^{z_{nk}}$

$$\Rightarrow \ln p(X, Z \mid \mu, \Sigma, \pi) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \{\ln \pi_k + \ln \mathcal{N}(\vec{x}_n \mid \vec{\mu}_k, \Sigma_k)\}$$

by Bayes' Theorem, $p(Z \mid X, \mu, \Sigma, \pi) \propto p(X, Z \mid \mu, \Sigma, \pi)$

$$\Rightarrow \ln p(Z \mid X, \mu, \Sigma, \pi) \propto \ln p(X, Z \mid \mu, \Sigma, \pi)$$

$$\propto \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \{\ln \pi_k + \ln \mathcal{N}(\vec{x}_n \mid \vec{\mu}_k, \Sigma_k)\}$$

$$\mathbb{E}_z[\ln p(X, Z \mid \mu, \Sigma, \pi)] = \mathbb{E}_z\left[\sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \{\ln \pi_k + \ln \mathcal{N}(\vec{x}_n \mid \vec{\mu}_k, \Sigma_k)\}\right]$$

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{E}_z[z_{nk}] \{\ln \pi_k + \ln \mathcal{N}(\vec{x}_n \mid \vec{\mu}_k, \Sigma_k)\}$$

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \{\ln \pi_k + \ln \mathcal{N}(\vec{x}_n \mid \vec{\mu}_k, \Sigma_k)\}$$

maximizing this expectation wrt μ_k, Σ_k, and π_k gives the M step (step 3)

Committee Methods

idea: improve accuracy by aggregating the results of many models

bootstrapping - given a training set $\{X_i\}_{i=1}^N$

generate a bootstrap training set by randomly sampling N points with replacement

train L models on each of L bootstrap training set

boosting - weight each data point, giving more weight to points labeled incorrectly by the previous training iteration

1. initialize weights to be equal, $w_n^{(1)} = \frac{1}{N}$ for $n = 1, \dots, N$
2. fit a classifier $y_m(\vec{x})$ to the data

   minimize weighted error $J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\vec{x}_n) \neq t_n)$

   where $I(y_m(\vec{x}_n) \neq t_n) = \begin{cases} 1 & \text{if } y_m(\vec{x}_n) \neq t_n \\ 0 & \text{otherwise} \end{cases}$
3. evaluate

   $\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\vec{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$

   then evaluate $\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}$

   note: initially $0 \leq \epsilon \leq \frac{1}{2}$

   $\Rightarrow \infty \geq \alpha \geq 0$

   - → many errors, update lots of weights slightly
   - → fewer errors, update weights aggressively
4. update weights

   $w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(y_m(\vec{x}_n) \neq t_n)\}$
5. repeat 2-4 for $m = 1, \dots, M$
6. final prediction

   $Y_M(\vec{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(\vec{x})\right)$

equivalently, minimize exponential error function wrt weights α_ℓ and base classifiers y_ℓ

$E = \sum_{n=1}^{N} \exp\{-t_n f_m(\vec{x}_n)\}$

where $f_m(\vec{x}_n) = \frac{1}{2} \sum_{\ell=1}^{m} \alpha_\ell y_\ell(\vec{x}_n)$

note: $E = \sum_{n=1}^{N} \exp\{-t_n f_m(\vec{x}_n)\}$

$= \sum_{n=1}^{N} \exp\{-t_n f_{m-1}(\vec{x}_n) - \frac{1}{2} t_n \alpha_m y_m(\vec{x}_n)\}$

$= \sum_{n=1}^{N} w_n^{(m)} \exp\{-\frac{1}{2} t_n \alpha_m y_m(\vec{x}_n)\}$

where $w_n^{(m)} = \exp\{-t_n f_{m-1}(\vec{x}_n)\}$

when minimizing wrt α_m and y_m, we can view $w_n^{(m)}$ as constants

let $\mathcal{T}_m = \{\text{data points correctly classified by } y_m\}$

$\mathcal{M}_m = \{\text{data points incorrectly classified by } y_m\}$

then $E = \sum_{n=1}^{N} w_n^{(m)} \exp\{-\frac{1}{2} t_n \alpha_m y_m(\vec{x}_n)\}$

$= \sum_{n \in \mathcal{T}_m} w_n^{(m)} \exp\{-\frac{\alpha_m}{2} \underbrace{t_n y_m(\vec{x}_n)}_{\text{1 b/c classified correctly}}\} + \sum_{n \in \mathcal{M}_m} w_n^{(m)} \exp\{-\frac{\alpha_m}{2} \underbrace{t_n y_m(\vec{x}_n)}_{\text{-1 bc classified incorrectly}}\}$

$= \exp\{-\frac{\alpha_m}{2}\} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + \exp\{\frac{\alpha_m}{2}\} \sum_{n \in \mathcal{M}_m} w_n^{(m)}$

$= \exp\{-\frac{\alpha_m}{2}\} \sum_{n=1}^{N} w_n^{(m)} I(y_m(\vec{x}_n) = t_n) + \exp\{\frac{\alpha_m}{2}\} \sum_{n=1}^{N} w_n^{(m)} I(y_m(\vec{x}_n) \neq t_n)$

$= \exp\{-\frac{\alpha_m}{2}\} \sum_{n=1}^{N} w_n^{(m)} - \exp\{-\frac{\alpha_m}{2}\} \sum_{n=1}^{N} w_n^{(m)} I(y_m(\vec{x}_n) \neq t_n) + \exp\{\frac{\alpha_m}{2}\} \sum_{n=1}^{N} w_n^{(m)} I(y_m(\vec{x}_n) \neq t_n)$

$= \exp\{-\frac{\alpha_m}{2}\} \sum_{n=1}^{N} w_n^{(m)} + (\exp\{\frac{\alpha_m}{2}\} - \exp\{-\frac{\alpha_m}{2}\}) \sum_{n=1}^{N} w_n^{(m)} I(y_m(\vec{x}_n) \neq t_n)$

when minimizing E wrt y_m, $\exp\{-\frac{\alpha_m}{2}\}\sum_{n=1}^{N} w_n^{(m)}$ is constant

so we can minimize $J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\vec{x}_n) \neq t_n)$ (see boosting 2)

when minizing E wrt a_m,

we get $a_m = \ln\{\frac{1-\epsilon_m}{\epsilon_m}\}$ (see boosting 3)

when updating the weights,

$$w_n^{(m+1)} = w_n^{(m)} \exp\{-\tfrac{1}{2} t_n a_m y_m(\vec{x}_n)\}$$
$$= w_n^{(m)} \exp\{-\tfrac{\alpha_m}{2}(1 - 2I(y_m(\vec{x}_n) \neq t_n))\}$$
$$= w_n^{(m)} \exp\{-\tfrac{\alpha_m}{2}\} \exp\{a_m I(y_m(\vec{x}_n) \neq t_n)\}$$

since $\exp\{-\frac{\alpha_m}{2}\}$ is independent of n, it would affect all weights equally and so can be discarded

thus we get $w_n^{(m+1)} = w_n^{(m)} \exp\{a_m I(y_m(\vec{x}_n) \neq t_n)\}$ (see boosting 4)

note: $t_n y_m(\vec{x}_n) = \begin{cases} 1 & \text{if } t_n = y_m(\vec{x}_n) \\ -1 & \text{if } t_n \neq y_m(\vec{x}_n) \end{cases}$

so $t_n y_m(\vec{x}_n) = 1 - 2I(y_m(\vec{x}_n) \neq t_n)$

where $I(y_m(\vec{x}_n) \neq t_n) = \begin{cases} 1 & \text{if } y_m(\vec{x}_n) \neq t_n \\ 0 & \text{if } y_m(\vec{x}_n) = t_n \end{cases}$