Collection of Formulas for Statistical, Neural and Other Learning Systems

©Mats Gustafsson and Tomas Olofsson,
Uppsala University, Signal and Systems Group

November, 1999

Optimization
Steepest descent

$$\theta_{k+1} = \theta_k - \eta \left.\frac{\partial J}{\partial \theta}\right|_{\theta_k}$$

Constrained optimization: minimize $J(\theta)$ under the constraints

$$c_1(\theta) = 0, \ldots, c_M(\theta) = 0$$

First order condition for local minima:

$$\frac{\partial L}{\partial \theta} = 0, \quad \frac{\partial L}{\partial \lambda_l} = 0, \quad \forall l$$

where

$$L(\theta) = J(\theta) - \sum_l^M \lambda_l c_l(\theta)$$

Differentiation

$$f = A\mathbf{x}, \quad \frac{\partial f}{\partial \mathbf{x}} = A$$

$$f = \mathbf{x}^T A\mathbf{x}, \quad \frac{\partial f}{\partial \mathbf{x}} = (A + A^T)\mathbf{x}$$

$$f = \mathbf{y}^T A\mathbf{x}, \quad \frac{\partial f}{\partial \mathbf{x}} = A^T\mathbf{y}$$

$$f = |A|, \quad \frac{\partial f}{\partial A} = |A|(A^{-1})^T$$

$$f = log|A|, \quad \frac{\partial f}{\partial A} = (A^{-1})^T$$

$$f = \mathbf{x}^T A\mathbf{y}, \quad \frac{\partial f}{\partial A} = \mathbf{x}\mathbf{y}^T$$

Multidimensional Normal Distribution

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|C|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T C^{-1}(\mathbf{x}-\mathbf{m})}$$

Conditional normal distributions. If $\mathbf{x}$ and $\mathbf{y}$ are jointly Gaussian, i.e.,

$$\mathbf{z} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim N\left(\begin{pmatrix} \mathbf{m}_x \\ \mathbf{m}_y \end{pmatrix}, \begin{pmatrix} \mathbf{C}_x & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{C}_y \end{pmatrix} \right)$$

then the distribution for $\mathbf{y}$ conditioned on $\mathbf{x}$ is also Gaussian with mean

$$\mathbf{m} = \mathbf{m}_y + \mathbf{C}_{yx}\mathbf{C}_x^{-1}(\mathbf{x} - \mathbf{m}_x)$$

and covariance matrix

$$\mathbf{C} = \mathbf{C}_y - \mathbf{C}_{yx}\mathbf{C}_x^{-1}\mathbf{C}_{xy}$$

Statistics and Estimation

Maximum likelihood (ML) estimation

$$\theta^* = \arg\max_\theta p(observations|\theta)$$

Maximum a posteriori (MAP) estimation

$$\theta^* = \arg\max_\theta p(\theta|observations)$$

Bayes theorem:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

Minimization of Bayes risk:
Determine the conditional risks:

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1}^{J} \lambda_{ij} P(c_j|\mathbf{x}), \quad i = 1, 2, \ldots, I$$

and choose the action α_k which yields the smallest risk.

Fisher's Linear Discriminant:

$$y = \mathbf{w}^T \mathbf{x}$$

$$\tilde{m}_i = \mathbf{w}^T \mathbf{m}_i, \quad \tilde{s}_i^2 = \sum_{y \in \mathcal{Y}_i} (y - \tilde{m}_i)^2$$

$$S_i = \sum_{\mathbf{x} \in \mathcal{X}_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T$$

$$S_W = S_1 + S_2, \qquad S_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$$

$$J(\mathbf{w}) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2} = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

Unsupervised Learning:
Mixture density:

$$f(\mathbf{x}|\mathbf{w}) = \sum_{j=1}^{J} P(c_j) f(\mathbf{x}|c_j, \mathbf{w}_j)$$

C-means Clustering:

1. Choose the number of classes, J.
2. Choose prototype vectors $\mathbf{w}_j, j = 1, 2, \ldots, J$.
3. Classify each pattern.
4. Recompute estimates based on the classification.

5. Return to step 2 if not consistent.

Principal Component Analysis

$$C\mathbf{w}_j = \lambda_j \mathbf{w}_j, \quad C = E\{(\mathbf{x} - \mathbf{m}_x)(\mathbf{x} - \mathbf{m}_x)^T\}, \quad \mathbf{m}_x = E\{\mathbf{x}\}$$

Karhunen-Loève Transformation and Expansion for patterns from a zero mean distribution:

$$\mathbf{y} = W^T\mathbf{x}, \quad \mathbf{x} = W\mathbf{y}$$

Classical Parzen Windows:
d = Dimension of the pattern space.
N = Number of patterns
V_N = Volume of a hypercube with edge length $h_N, V_N = (h_N)^d$.
k_N = Number of samples inside the hypercube.

$$f(\mathbf{x}) \approx \frac{k_N/N}{V_N} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{V_N}\phi(\frac{\mathbf{x} - \mathbf{x}_n}{h_N})$$

where ϕ is a d-dimensional unit step function.

The Delta Rule (LMS):

$$\begin{aligned} y_j(n) &= \mathbf{w}_j^T\mathbf{x}(n) \\ \delta_j(n) &= d_j(n) - y_j(n) \\ \Delta w_{ji} &= \eta\delta_j(n)x_i(n) \end{aligned}$$

w_{ji}: Weight from node # i to node #j
y_j: Output from node # j
d_j: Desired output
x_i: Component # i of the pattern vector $\mathbf{x}$.

Error Backpropagation:

$$\begin{aligned}
\delta_j(n) &= e_j(n)s'(v_j(n)), \quad e_j(n) = d_j(n) - y_j(n) \quad \text{(output layer)} \\
\delta_j(n) &= s'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \quad \text{(hidden layer)} \\
\Delta w_{ji}(n) &= \eta \delta_j(n) x_i(n) + \alpha \Delta w_{ji}(n-1)
\end{aligned}$$

Linear Regression:

$$\mathbf{y} = A^T \mathbf{x} + \mathbf{e}$$

Ordinary Least Squares Regression:

$$A_{OLS} = (X_N^T X_N)^{-1} X_N^T Y_N$$

where

$$X_N = (\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(N))^T$$

and

$$Y_N = (\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(N))^T$$

Ridge Regression:

$$A_{RR} = (X_N^T X_N + \lambda I)^{-1} X_N^T Y_N$$

Principal Component Regression:

$$A_{PCR} = \sum_{k=1}^{K} \frac{1}{\lambda_k} \mathbf{w}_k \mathbf{w}_k^T X_N^T Y_N$$

Learning Vector Quantization:
Update the winner $\mathbf{w}_j$:

$$\mathbf{w}_j(n+1) = \begin{cases} \mathbf{w}_j(n) + \eta(n)[\mathbf{x} - \mathbf{w}_j(n)] & \text{if right class} \\ \mathbf{w}_j(n) - \eta(n)[\mathbf{x} - \mathbf{w}_j(n)] & \text{otherwise} \end{cases}$$

where $0 < \eta < 1$.