Learning from Data with Information Theoretic Criteria II

Jose C. Principe, Ph.D.
Distinguished Professor of Electrical and Biomedical Engineering
and BellSouth Professor

Computational NeuroEngineering Laboratory
University of Florida
http://www.cnel.ufl.edu
principe@cnel.ufl.edu
Outline

• Error Entropy Criterion
• Algorithms for adaptation
  • MEE, MEE-RC, MEE-SIG, NMEE, MEE-FP
• Adaptation of linear filters with divergence
• Fast Renyi’s calculations
From Data to Models

• Optimal Adaptive Models and least squares

Data x → Adaptive System → Output → Data z

Cost function:

\[ J_w(e) = E[(z - f(x, w))^2] \]

\[ \frac{\partial J(e)}{\partial w} = \frac{\partial J(e)}{\partial e} \frac{\partial e}{\partial w} = 0 \]

\[ \frac{\partial J(e)}{\partial w} = \frac{\partial E[e^2]}{\partial e} \frac{\partial e}{\partial w} = -2E[ex] = 0 \]
Model Based Inference

Alternatively, the problem of finding optimal parameters can be framed as model based inference.

The desired response $z$ can be thought as being created by an unknown transformation of the input vector $x$, and the problem is characterized by the joint pdf $p(x,z)$.

The role of optimization is therefore to minimize the Kulback Liebler divergence between the estimated joint and the real one.

$$
\min J(w) = \iint p(x,z) \log \frac{p(x,z)}{\tilde{p}_w(x,z)} \, dx \, dz
$$

If we write $z=f(x)+e$ with the error independent of $x$, then this is equivalent to

$$
\min H_S(e) = -\int p_w(e) \log p_w(e) \, de
$$
Information Theoretic Learning is exactly a set of tools to solve this minimization problem.

Note that this is different from the use of information theory in communications. Here

• We are interested in continuous random variables.
• We can not assume Gaussianity, so need to use nonparametric estimators.
• We are interested in using gradients, so estimators must be smooth.

We will use Parzen estimators. Since for optimization a monotonic function does not affect the result, we will be using the Information Potential instead of Renyi’s entropy most of the time.

\[
\begin{align*}
H_{R_2}(E) &= -\log V(E) \\
V(E) &= E[p(e)]
\end{align*}
\]
Properties of Entropy Learning with Information Potential

- The IP with Gaussian kernels preserves the global minimum/maximum of Renyi’s entropy.

- The global minimum/maximum of Renyi’s entropy coincides with Shannon’s entropy (super-Gaussian).

- Around the global minimum, Renyi’s entropy cost (of any order) has the same eigenvalues as Shannon’s.

- The global minimum degenerates to a line (because entropy is insensitive to the mean).
Error Entropy Criterion

We will use iterative algorithms for optimization of the steepest descent type

\[ w(n+1) = w(n) - \eta \nabla V_2(n) \]

Given a batch of \( N \) samples the IP is

\[ V_2(E) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} G_{\sqrt{2}\sigma}(e_i - e_j) \]

For an FIR the gradient becomes

\[
\nabla_k V_2(n) = \frac{\partial V(e(n))}{\partial w_k} = \frac{\partial V(e(n))}{\partial e(n)} \frac{\partial(e(n))}{\partial w_k} = \\
\frac{2}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} G_{\sigma}(e(n-i) - e(n-j))(e(n-i) - e(n-j)) \left( \frac{\partial y(n-j)}{\partial w_k} - \frac{\partial y(n-i)}{\partial w_k} \right) \\
\nabla_k V_2(n) = \frac{2}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} G_{\sigma}(e(n-i) - e(n-j))(e(n-i) - e(n-j))(x_k(n-j) - x_k(n-i))
\]
Error Entropy Criterion

This can be easily extended to any alpha and any kernel using the expressions of IP.

For the FIR filter we get

\[ \nabla_k V_2(n) = \frac{\alpha - 1}{N^\alpha} \sum_{i=1}^{N} \left( \sum_{j=1}^{N} K(e(n-i) - e(n-j)) \right)^{\alpha-2} \sum_{j=1}^{N} K'_\sigma(e(n-i) - e(n-j))(x_k(n-j) - x_k(n-i)) \]
Comparing Quadratic Entropy and MSE

• IP does not yield a convex performance surface even for the FIR. For adaptation problems that yield zero or small errors, there is a parabolic approximation around the minimum.

The largest eigenvalue of the second order approximation of the performance surface is smaller than MSE (approaches zero with large kernel sizes). So stepsizes can be larger for convergence.
Comparing Quadratic Entropy and MSE

• Consider for simplicity the 1-D case, and approximate the Gaussian by its second order Taylor series

\[ G_\sigma(x) = c e^{-x^2/2\sigma^2} \approx c(1 - x^2 / 2\sigma^2) \]

• We can show

\[
\max \hat{V}_{2,\sigma}(e) \approx \frac{1}{N^2} \sum_i \sum_j c(1 - (e_i - e_j)^2 / 2\sigma^2) = c - \frac{c}{2\sigma^2 N^2} \sum_i \sum_j (e_i - e_j)^2
\]

\[
\equiv \min \sum_i \sum_j (e_i - e_j)^2 = 2N \sum_i e_i^2 - 2 \left( \sum_i e_i \right)^2 = 2N \cdot MSE(e) - 2N^2 \bar{\mu}_e^2
\]

When the error is small w.r.t. the kernel size, quadratic entropy training is equivalent to a biased MSE.
Comparing Quadratic Entropy and MSE

- The kernel size produces a dilation in weight space, i.e. it controls the region in weight space where the second order approximation of the entropy cost function is valid.
We have theoretically shown that:

Regardless of the entropy order, increasing the kernel size results in a wider valley around the optimal solution by decreasing the absolute values of the (negative) eigenvalues of the Hessian matrix of the information potential criterion.

The effect of entropy order on the eigenvalues of the Hessian depends on the value of the kernel size.
The batch algorithm just presented is called the Minimum Error Entropy algorithm (MEE).

- The cost function is totally independent of the mapper, so it can be applied generically.
- The algorithm is batch and has a complexity of $O(N^2)$.
- To estimate entropy one needs pairwise interactions.
## Entropy Learning Algorithms

<table>
<thead>
<tr>
<th>MSE Criterion</th>
<th>Error Entropy Criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steepest Descent</td>
<td>Minimum Error Entropy (MEE)</td>
</tr>
<tr>
<td></td>
<td>MEE- RIG Recursive information gradient</td>
</tr>
<tr>
<td>LMS</td>
<td>MEE- SIG Stochastic Information gradient</td>
</tr>
<tr>
<td>LMF</td>
<td>MEE-SAS self adapting step size</td>
</tr>
<tr>
<td>NLMS</td>
<td>NMEE</td>
</tr>
<tr>
<td>RLS</td>
<td>MEE-Fixed point</td>
</tr>
</tbody>
</table>
Suppose that at time $n$, we already have the pdf estimate $f_n(x)$ for $f_X(x)$. Using the new sample $x_{n+1}$, we update this pdf estimate according to

$$f_{n+1}(x) = (1 - \lambda)f_n(x) + \lambda \kappa_\sigma(x - x_{n+1})$$

Which gives (with the initial condition $f_1(x) = \kappa_\sigma(x - x_1)$)

$$\bar{V}_{n+1} = E_X[f_{n+1}(X)] = (1 - \lambda)E_X[f_n(X)] + \lambda E_X[\kappa_\sigma(X - x_{n+1})]$$

$$\approx (1 - \lambda)\bar{V}_n + \frac{\lambda}{M} \sum_{i=n-M+1}^n \kappa_\sigma(x_i - x_{n+1})$$

Practically, the complexity drops to $O(M)$.

Parameters are window length $M$, bandwidth $\sigma$, and forgetting factor $\lambda$. 
MEE-RIG

Increasing $\lambda$

different M

Different kernels
Dropping $E[.]$ and substituting the required pdf by its Parzen estimate over the most recent $M$ samples, at time $k$ our information potential estimate becomes

$$V_\alpha(e(n)) \approx \left( \frac{1}{M} \sum_{i=n-M}^{n-1} \kappa_\sigma(e_n - e_i) \right)^{\alpha-1}$$

If we substitute in the gradient equation

$$\frac{\partial \hat{V}_\alpha(e(n))}{\partial w_k} = -\frac{(\alpha - 1)}{M^\alpha} \left( \sum_{i=n-M}^{n-1} \kappa_\sigma(e_n - e_i) \right)^{\alpha-2} \left[ \sum_{i=n-M}^{n-1} \kappa'_\sigma(e_n - e_i)(x_k(i) - x_k(n)) \right]$$

For $\alpha=2$,

$$\frac{\partial \hat{V}_\alpha(e(n))}{\partial w_k} = -\frac{1}{M^2} \left[ \sum_{i=n-M}^{n-1} G_\sigma(e_n - e_i)(e_n - e_i)(x_k(i) - x_k(n)) \right]$$

We have shown that for the linear case, SIG converges in the mean to the true value of the gradient. So it has the same properties as LMS.
The SIG has another very interesting property. It can be easily applied to Renyi’s entropy and yields an estimator that is on average the gradient of Shannon’s entropy.

\[
\frac{\partial \hat{H}_{\alpha,n}}{\partial w_k} = \frac{1}{1-\alpha} \left( \frac{1}{M} \sum_{i=n-M}^{n-1} \kappa_\sigma(e_n-e_i) \right)^{-2} \left( \frac{1}{M} \sum_{i=n-M}^{n-1} \kappa'_\sigma(e_n-e_i)(x_k(i)-x_k(n)) \right) = \frac{n-1}{\sum_{i=n-M}^{n-1} \kappa_\sigma(e_n-e_i)} \sum_{i=n-M}^{n-1} \kappa'_\sigma(e_n-e_i) (x_k(n)-x_k(i))
\]

In fact,

\[
H_S(e) = -E[\log f_e(e)] \quad \hat{H}_{S,n}(e) \approx -E\left[ \log \left( \frac{1}{L} \sum_{i=n-L}^{n-1} \kappa_\sigma(e_n-e_i) \right) \right] \quad \frac{\partial \hat{H}_{S,n}}{\partial w_k} = E \left[ \frac{n-1}{\sum_{i=n-L}^{n-1} \kappa_\sigma(e_n-e_i)} \sum_{i=n-L}^{n-1} \kappa'_\sigma(e_n-e_i) (x_k(n)-x_k(i)) \right]
\]
SIG for Supervised Linear Filters

- We can derive a “LMS like” algorithm using Renyi’s entropy.

\[ w_{k+1} = w_k - \eta \left( \frac{\partial H_\alpha(X)}{\partial w} \right)_k \]

- For \( \alpha = 2 \), Gaussian kernels, \( G'_\sigma(x) = -x G_\sigma(x) / \sigma^2 \) and \( M=1 \) we get,

\[ \left( \frac{\partial H_2(X)}{\partial w} \right)_k = - \frac{1}{\sigma^2} (e_k - e_{k-1})(x_k - x_{k-1}) \]
Relation between SIG and Hebbian learning

- For Gaussian kernels, $M=1$, the expression to maximize output Entropy on a linear combiner becomes very simple also

\[
\frac{\partial H_\alpha(y_k)}{\partial w} = (y_k - y_{k-1}) \cdot \kappa_\sigma(y_k - y_{k-1}) \cdot (x_k - x_{k-1})
\]

\[
= \frac{1}{\sigma^2} (y_k - y_{k-1}) \cdot (x_k - x_{k-1})
\]

We see that SIG gives rise to a sample by sample adaptation rule that is like Hebbian between consecutive samples!
Does SIG work?

Consider:
- Random variable, $d = 2$
- x-axis is uniform
- y-axis is Gaussian
- sample covariance is the identity matrix
Does SIG work?

- Generated 50 samples of the 2D distribution from the previous slide, \( y = w_1 x_1 + w_2 x_2 \) \((w_1^2 + w_2^2 = 1)\)

- PCA would converge to any direction but SIG consistently found the 90 degree direction!

![Gaussian and Uniform distributions]
MEE-SAS

• The largest value of the IP is \( V(0) \). Therefore define a new cost

\[
J_{\text{MEE-SAS}}(e) = \min_w [V(0) - V(e)]^2.
\]

• Since \( V(0) - V(e) \) is always of the same sign, it does not change the direction of the gradient but can be used to accelerate the search.

\[
w(n+1) = w(n) + \mu [V(0) - V(e)] \nabla V(e) \\
= w(n) + \mu(n) \nabla V(e)
\]

• We have shown that the minimum is preserved, and that the curvature of the cost changes from forth to second power.
The point where the curvature changes can be analytically computed

\[ v_k = \pm \sqrt{\frac{V(0) - V_0(\epsilon) - 1}{3\lambda_k}} \]
If we plot the difference in gradients we can see this effect.

\[ \sigma = 0.1, \text{grad}[J(MEE)] - \text{grad}[J(MEE-SAS)] \]

\[ \sigma = 0.35, \text{grad}[J(MEE)] - \text{grad}[J(MEE-SAS)] \]

\[ \sigma = 0.6, \text{grad}[J(MEE)] - \text{grad}[J(MEE-SAS)] \]
MEE-SAS

Tracking parameters in a nonstationary environment

- Weight Error Power
- Weight Track

Graphs showing the performance of MEE, MEE-SAS, and Switching MEE-SAS over iterations.
The criterion of NMEE is formulated as constrained optimization

$$\min _{w(n+1)-w(n)} \| \nabla e V(e_p(n)) \|_2$$

subject to $V(0) = V(e_p(n)) = 0$

Solving using the Lagrange multipliers we obtain

$$w(n+1) = w(n) + \eta \frac{e_a(n) \nabla V(e_p(n))}{(\nabla V(e_p(n))^T u(n)}$$

Practically we have to use the a priori error.
The Lagrangian has a much more complex form than for NLMS:

\[
\lambda = \frac{4\sigma^2 Le_a(n)}{\sum_{i=n-L}^{n-1} \kappa(e_p(n) - e_p(i))(e_p(n) - e_p(i)) \{\|u(n)\|^2 - u^T(i)u(n)\}}
\]

We can see the dependence on the input power, but also a function of the kernel size and an extra error term.

We showed that this makes NMEE insensitive to power, but also to kernel size and faster than MEE.
Normalized MEE

Normalized MEE

Number of Iterations

Weight Error Power

MEE, \( P=1 \) (step-size=0.1)
NMEE, \( P=1 \) (step-size=1.0)
MEE, \( P=10 \) (step-size=0.01)
NMEE, \( P=10 \) (step-size=1.0)
NLMS, \( P=1 \) (step-size=0.15)
NLMS, \( P=10 \) (step-size=0.15)
Normalized MEE

Changes in adaptation for different kernel sizes.
Fixed Point MEE

This is a second order adaptation method. The stationary point is found as

\[ \frac{\partial V(e)}{\partial w_k} \bigg|_{w_i = w_*} = 0 \]

Changing it to the fixed point form \( w^* = F(w^*) \) yields

\[ w_{k+1} = F(w_k) \]

\[ = \left( \frac{1}{2k^2\sigma^2} \sum_{i=1}^{k} \sum_{j=1}^{k} K_{\sigma \sqrt{2}} (e_j - e_i) \cdot [u_j - u_i] \cdot [u_j - u_i]^T \right)^{-1} \cdot \left( \frac{1}{2k^2\sigma^2} \sum_{i=1}^{k} \sum_{j=1}^{k} K_{\sigma \sqrt{2}} (e_j - e_i) \cdot [d_j - d_i] \cdot [u_j - u_i] \right) \]

\[ = R_E(w_k)^{-1} P_E(w_k) \]

We showed that it converges when the initial point is close to the optimum.
Fixed Point MEE

When combined with the recursive estimator of the IP, a practical algorithm is obtained.

\[
\bar{R}_E (w_k) = (1 - \lambda) \bar{R}_E (w_{k-1}) + \frac{\lambda}{L} \sum_{i=k-L}^{k-1} K \sigma \sqrt{2} (e_k - e_i) [u_k - u_i] [u_k - u_i]^T
\]

\[
\bar{P}_E (w_k) = (1 - \lambda) \bar{P}_E (w_{k-1}) + \frac{\lambda}{L} \sum_{i=k-L}^{k-1} K \sigma \sqrt{2} (e_k - e_i) [d_k - d_i] [u_k - u_i]
\]

The practical importance is that the complexity is reduced from \(O(K^2)\) to \(O(L)\).
Alternatively we can apply the matrix inversion lemma to decrease the computation of $\text{Re}$

$$\bar{R}_E(w_k) = (1 - \lambda) \bar{R}_E(w_{k-1}) + \frac{\lambda}{L} \Pi \cdot \Pi^T$$

$$\Pi = [q_{k-1} \quad q_{k-2} \quad \cdots \quad q_{k-L}] \quad q_i = \sqrt{K_{\sigma \sqrt{2}}(e_k - e_i) \cdot (u_k - u_i)}$$

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$$

$$\Lambda = (1 - \lambda) \bar{R}_E(w_{k-1}) \quad B = \frac{\lambda}{L} \Pi$$

$$C = I \quad D = \Pi^T$$

$$\bar{\Phi}(w_k) = \frac{1}{1 - \lambda} \bar{\Phi}(w_{k-1}) - \frac{\lambda}{(1 - \lambda)^2 L} \bar{\Phi}(w_{k-1}) \Pi \left( I_{L \times L} + \frac{\lambda}{(1 - \lambda) L} \Pi^T \bar{\Phi}(w_{k-1}) \Pi \right)^{-1} \Pi^T \bar{\Phi}(w_{k-1})$$

The matrix inversion takes $O(L^3)$
Fixed Point MEE

As expected the MEE-FP is faster and goes directly to the minimum

![Graph showing weight error power versus number of iterations for MEE (μ=0.02) and FP-MEE (λ=0.2).](image1)

![Contour plot comparing MEE and FP-MEE](image2)
Adaptation of Linear Systems with Divergence

- Exemplify for QMI-ED

\[ V_{ED} = V_J - 2V_C + V_1V_2 = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} v_i(i, j)v_2(i, j) - 2 \frac{1}{N} \sum_{i=1}^{N} v_1(i)v_2(i) + V_1V_2 \]

- Taking the sensitivity with respect to the weights

\[ \frac{\partial V_{ED}}{\partial w_{kj}} = \frac{\partial V_{ED}}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial w_{kj}} = \left( \frac{\partial V_J}{\partial y_j(n)} - 2 \frac{\partial V_C}{\partial y_j(n)} + \frac{\partial V_1V_2}{\partial y_j(n)} \right) \frac{\partial y_j(n)}{\partial w_{kj}} \]

This is a straight extension because the potential fields and their gradients add up.
MEE for Nonlinear Systems

- Consider the error signal. Think of the IPTs as errors of a nonlinear mapper (such as the MLP). How can we train the MLP?
- Use the IF as the Injected error.
- Then apply the Backpropagation algorithm.

\[
\frac{\partial J}{\partial w_{ij}} = \sum_{p=1}^{k} \sum_{n=1}^{N} \frac{\partial J}{\partial y_p(n)} \frac{\partial y_p(n)}{\partial w_{ij}}
\]

This methodology extends naturally to IP.
Global Optimization by Annealing Kernel Size

We have a way to avoid local minima in non-convex performance surfaces:

1. Start with a large kernel size, and adapt to reach minimum.
2. Decrease the kernel size to decrease the bias and adapt again.
3. Repeat until the kernel size is compatible with the data.

Kernel size annealing is equivalent to the method of convolution smoothing in global optimization. Hence, as long as the annealing rate is right (slow enough), the information potential provides a way to avoid local minima and reach the global minimum.
If advanced search methods are needed, care must be taken when extending them to error entropy criterion. Basically the problems are related with the definition of trust regions and the adaptation of the kernel size.

We have studied the scaled conjugate method and the Levenberg Marquadt algorithm and have implemented modifications that lead to consistent results.
The Fast Gauss Transform (FGT) is an example of a fast algorithm to approximately compute matrix (A) vector (d) products where the matrix elements are \[ a_{ij} = \varphi(x_i - x_j) \] with \( \varphi \) a Gaussian function.

The basic idea is to cluster the data and to expand the Gaussian function in Hermite polynomials to divide and conquer the complexity (multi pole method).

\[
\exp\left(-\frac{(y_j - y_i)^2}{4\sigma^2}\right) = \sum_{n=0}^{p-1} \frac{1}{n!} \left(\frac{y_i - y_c}{2\sigma}\right)^n h_n \left(\frac{y_j - y_c}{2\sigma}\right) + \epsilon(p)
\]

\[
h_n(y) = (-1)^n \frac{d^n}{dx^n}\left(\exp(-x^2)\right)
\]
A greedy clustering algorithm called the farthest point is normally used (because it can be computed in $O(kN)$ time, $k$ # of clusters).

The information potential for 1 D data becomes

$$V(y) \approx \frac{1}{2\sigma N^2 \sqrt{\pi}} \sum_{j=1}^{N} \sum_{B} \sum_{n=0}^{p-1} \frac{1}{n!} h \left( \frac{y_j - y_{CB}}{2\sigma} \right) C_n(B)$$

Where

$$C_n(B) = \sum_{y_j \in B} \left( \frac{y_j - y_{CB}}{2\sigma} \right)^n$$

And this can be computed in $O(kpN)$ time ($p$ degree of approximation).
Fast Gauss Transform

For multidimensional data, we can define the multidimensional Hermit expansion

\[ h_\alpha(y) = h_{\alpha_1}(y_1)h_{\alpha_2}(y_2) \cdots h_{\alpha_d}(y_d) \]

But this is expensive computationally. Alternatively, we can factorize the Gaussian as

\[
\exp\left( -\frac{\|y_j - y_i\|^2}{4\sigma^2} \right) = \exp\left( -\frac{\|y_j - c\|^2}{4\sigma^2} \right) \exp\left( -\frac{\|y_i - c\|^2}{4\sigma^2} \right) \exp\left( 2\frac{(y_j - c) \cdot (y_i - c)}{4\sigma^2} \right)
\]

And break the cross term using a Taylor series expansion

\[
\exp\left( 2\frac{(y_j - c) \cdot (y_i - c)}{4\sigma^2} \right) = \sum_{\alpha \geq 0} \frac{2^{\vert \alpha \vert}}{\alpha!} \left( \frac{y_j - c}{2\sigma} \right)^\alpha \left( \frac{y_i - c}{2\sigma} \right)^\alpha + \varepsilon(\alpha)
\]

The information potential can be written

\[
V_T(y) \approx \frac{1}{N^2(4\pi\sigma^2)^{d/2}} \sum_{j=1}^{N} \sum_{B} \sum_{\alpha \geq 0} C_\alpha(B) \exp\left( -\frac{\|y_j - c_B\|^2}{4\sigma^2} \right) \left( \frac{y_j - c_B}{2\sigma} \right)^\alpha
\]

\[
C_\alpha(B) = \frac{2^{\vert \alpha \vert}}{\alpha!} \sum_{e_i \in B} \exp\left( -\frac{\|y_i - c_B\|^2}{4\sigma^2} \right) \left( \frac{y_i - c_B}{2\sigma} \right)^\alpha
\]
Example: Fast Gauss Transform

Computation of IP for 2 and 3D data as a function of data samples (k=20, p=5)
Example: Fast Gauss Transform

Computation of IP for 3D data.

\[ \text{Absolute Error} = | \text{IP(direct)} - \text{IP(Hermite or Taylor)} | \]

- Hermite Exp. \( p=3 \)
- Taylor Exp. \( p=3 \)
- Hermite Exp. \( p=6 \)
- Taylor Exp. \( p=6 \)

Hermite Exp. \( p=3 \)
Taylor Exp. \( p=3 \)
Hermite Exp. \( p=6 \)
Taylor Exp. \( p=6 \)
Direct method
Example: Fast Gauss Transform

Computation of a 9th order adaptive filter (1000 samples).
Kullback Leibler Divergence

- KL Divergence measures the “distance” between pdfs (Csiszar and Amari)
  - Relative entropy
  - Cross entropy
  - Information for discrimination

\[
D_k(f, g) = \int f(x) \log \frac{f(x)}{g(x)} dx
\]

*notice similarity to* \(H_S(X) = -\int f_X(x) \log f_X(x) dx\)
How does this compare with PCA

- Two Gaussian distributions

\[
p(x) = \frac{1}{2}(N(x, m_1, \Sigma_1) + N(x, m_2, \Sigma_2))
\]

\[
m_1 = \begin{bmatrix} -1.0 \\ -1.0 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix}
\]

\[
m_2 = \begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix}
\]

MLP
2-4-1