Vision as Bayesian Inference

Learning Deep Network

\[ I \xrightarrow{w_1} H_1 \xrightarrow{w_2} H_2 \xrightarrow{w_3} O \quad \text{I: Input} \]

\[ O: \text{Output} \]

Example: \[ O = \text{ReLu}(W_3H_2) \]
\[ H_2 = \text{ReLu}(W_2H_1) \]
\[ H_1 = \text{ReLu}(W_1I) \]

More generally, \[ O = O(W_3, H_2) \]
\[ H_2 = H_2(W_2, H_1) \]
\[ H_1 = H_1(W_1, I) \]

Only require that the functions \( O(\cdot, \cdot), H_2(\cdot, \cdot), H_1(\cdot, \cdot) \) are differentiable

The functions can be composed together to give an output \( O = O(W, I), W = (W_1, W_2, W_3) \)
Loss Function \( \mathcal{L}(O(W, I), T) = L \)

- \( O(W, I) \): the output of network
- \( T \): the ground truth
- Dataset: \( \{(I_n, T_n) : n = 1, \ldots, N\} \)

To train the deep network, we need to compute the derivatives \( \frac{d}{dW} \mathcal{L}(O(W, I), T) \)

**Batch Mode \( \Rightarrow \) Steepest descent (SD)**

\[
W^{t+1} = W^t - \eta_t \frac{1}{N} \sum_{n=1}^{N} \frac{d}{dW} \mathcal{L}(O(W, I_n), T_n)
\]

**Online Learning \( \Rightarrow \) Stochastic Gradient Descent (SGD)**

At time step, select example \( n(t) \) at random

\[
W^{t+1} = W^t - \eta_t \frac{d}{dW} \mathcal{L}(O(W, I_{n(t)}), T_{n(t)})
\]

This is old fashioned SGD. In practice at time \( t \), select a subset \( N(t) \) of the data

\[
W^{t+1} = W^t - \eta_t \frac{1}{|N(t)|} \sum_{n \in N(t)} \frac{d}{dW} \mathcal{L}(O(W, I_n), T_n)
\]
Loss Function

**Note** The loss function $\mathcal{L}(O(W, I), T)$ is a non-convex function of $W$

⇒ Convergence cannot be guaranteed.

We will discuss later why it is non-convex

SGD has a stochastic property which enables it to avoid some local minima in the loss function.

There are theoretical results, informally called ‘Robbins-Monro theory’, which can even guarantee convergence to the global minimum of the loss function provided certain conditions apply.

**But** these results do not apply to deep networks
Vision as Bayesian Inference

We need to compute \( \frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial W_2}, \frac{\partial L}{\partial W_3} \)?

To do this, we use the chain rule of differentiation.

→ This is called back propagation.

To compute \( \frac{\partial L}{\partial W_3} \):

\[
\frac{\partial L}{\partial W_3} = \frac{\partial}{\partial W_3} \mathcal{L}(O(H_2, W_3), T) = \frac{\partial L}{\partial O} \cdot \frac{\partial O}{\partial W_3}
\]

To compute \( \frac{\partial L}{\partial W_2} \), recall that \( H_2 = H_2(H_1, W_2) \)

First compute \( \frac{\partial L}{\partial H_2} = \frac{\partial L}{\partial O} \cdot \frac{\partial O}{\partial H_2} \)

\[
\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial H_2} \cdot \frac{\partial H_2}{\partial W_2}
\]

To compute \( \frac{\partial L}{\partial W_1} \), recall that \( H_1 = H_1(I, W_1) \)

First compute \( \frac{\partial L}{\partial H_1} = \frac{\partial L}{\partial H_2} \cdot \frac{\partial H_2}{\partial H_1} \)

\[
\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial H_1} \cdot \frac{\partial H_1}{\partial W_1}
\]

Lecture 07-05
Key Idea

Computing the derivatives exploits the compositionality of the function $O = O(W, I)$