Vinta Reethu
Deep Learning for Vision
Indian Institute of Technology Hyderabad
Assignment-1

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Assignment-1

Vinta Reethu - ES18BTECH11028

February 26, 2021

1 RANSAC :

Let **w** is the inlier ratio, **n** is the number of points, **k** be the number of iterations.
It is given that 50% of the initial matches that are correct. Therefore $w = 50/100 = 1/2$
The number of degrees of freedom for homography(3d) is **8**.
Therefore, we need at least

$$n = \lceil d/2 \rceil$$
$$n = 4$$

- For one iteration the probability that at least one of the n points is incorrect $= 1 - w^n$.
- Hence for **k** iterations the probability that at least one of the n points is incorrect $= (1-w^n)^k$.

Substituting the values from above we get,

$$(1 - w^n)^k = 1 - 0.95$$
$$\left(1 - \left(\frac{1}{2}\right)^4\right)^k = 0.05$$
$$\left(1 - \frac{1}{16}\right)^k = 0.05$$
$$\left(\frac{15}{16}\right)^k = \frac{1}{20}$$
$$k = \left\lceil \frac{log20}{log\frac{16}{15}} \right\rceil$$
$$k = \lceil 46.41 \rceil$$
$$k = 47.$$

Therefore rounding of the nearest integer we obtain number of iterations(k) $=$ **47**.

Vinta Reethu
Deep Learning for Vision
Indian Institute of Technology Hyderabad
Assignment-1

2 3-layer neural network :

Let us simply the derivative as :

$$\frac{\partial f}{\partial W_{ij}^1} = \sum_k \frac{\partial f}{\partial h_k^1} * \frac{\partial h_k^1}{\partial W_{ij}^1} \tag{1}$$

Given that $f(x) = < w^3, h^2 >$. The derivative of f with respect to h_i^2 is

$$\frac{\partial f}{\partial h_i^2} = \frac{\partial (w^3 h^2)}{\partial h_i^2} = w_i^3$$

Now calculating the individual terms in the above equation.

$$\frac{\partial f}{\partial h_k^1} = \sum_j \frac{\partial f}{\partial h_j^2} * \frac{\partial h_j^2}{\partial h_k^1} \tag{2}$$

In the above formula we have already calculated the first term, let's calculate the 2nd term :

$$\begin{aligned} h^2 &= \sigma(W^2 h^1) \\ h_k^2 &= \sigma(\sum_l W_{kl}^2 h_l^1) \\ \frac{\partial h_j^2}{\partial h_k^1} &= \sigma(\sum_l W_{jl}^2 h_l^1) * [1 - \sigma(\sum_l W_{jl}^2 h_l^1)] * W_{jk}^2 \\ &= h_j^2 * (1 - h_j^2) * W_{jk}^2 \end{aligned}$$

Substituting this in equation *(2)*:

$$\begin{aligned} \frac{\partial f}{\partial h_k^1} &= \sum_j \frac{\partial f}{\partial h_j^2} * \frac{\partial h_j^2}{\partial h_k^1} \\ &= \sum_j w_j^3 * h_j^2 * (1 - h_j^2) * W_{jk}^2 \end{aligned}$$

Calculating the derivative of h_i^1 with respect to W_{ij}^1:

$$\begin{aligned} h^1 &= \sigma(W^1 x) \\ h_i^1 &= \sigma(\sum_j W_{ij}^1 x_j) \end{aligned}$$

$$\begin{aligned} \frac{\partial h_i^1}{\partial W_{ij}^1} &= \sigma(\sum_j W_{ij}^1 x_j) * [1 - \sum_j W_{ij}^1 x_j] * x_j \\ &= h_i^1 * (1 - h_i^1) * x_j \end{aligned}$$

Substituting all these in equation *(1)* we get:

$$\frac{\partial f}{\partial W_{ij}^1} = \sum_k \frac{\partial f}{\partial h_k^1} * \frac{\partial h_k^1}{\partial W_{ij}^1}$$
$$= \frac{\partial f}{\partial h_i^1} * h_i^1 * [1 - h_i^1] * x_j$$
$$= \left(\sum_k w_k^3 * h_k^2 * (1 - h_k^2) * W_{ki}^2 \right) * h_i^1 * (1 - h_i^1) * x_j$$

3 3-layer neural network gradient of the cost function :

The update in the vector form is given as

$$\Delta^{(2)} := \Delta^{(2)} + \delta^3 \left(a^{(2)}\right)^T$$

4 Neural network

Total weights in the network = Number of weights between input layer and hidden layer + Number of weights between hidden layer and output layer.
Number of weights between input layer and hidden layer = d*M
Number of weights between hidden layer and output layer = M*c
Total weights in the network = d*m+m*c = M*(d+c)

Total number of bias in the network = Number of bias units in hidden layer + Number of bias units in output layer.
Number of bias units in hidden layer = M
Number of bias units in output layer = c
Total number of bias in the network = M+c

Total number of independent derivatives = Number of independent derivatives in hidden layer + Number of independent derivatives in output layer.
Let the weights between input and hidden layer be $W1$, weights between hidden and output layer be $W2$.
Number of independent derivatives in hidden layer :
Here we just have to compute

$$\frac{dE}{dW_1} = \delta^2 (a^1)^T$$

As there are M nodes in hidden layer, number of independent derivatives = M.

Number of independent derivatives in hidden layer :
Here we just have to compute

$$\frac{dE}{dW_2} = \delta^3 (a^2)^T$$

As there are c nodes in hidden layer, number of independent derivatives = c.
Total number of independent derivatives = M+c

5 Generalized least squares

Consider a model in which the target data has the form:

$$\mathbf{y}_n = f\left(\mathbf{x}_n; \mathbf{w}\right) + \epsilon_{\mathbf{n}}$$

where ϵ_n is drawn from a zero mean Gaussian distribution having a fixed covariance matrix Σ.

Let us use the Gaussian distribution to model this. That is let y_n has a Gaussian distribution with a mean equal to the value $f(x_n, w)$.

As a result, the probability distribution of the target(t) is

$$p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \Sigma) = \prod_{n=1}^{N} \mathcal{N}\left(t_n \mid f(\mathbf{x_n}, \mathbf{w}), \Sigma\right)$$

Now applying log on both sides we obtain,

$$\ln p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \Sigma) = \sum_{n=1}^{N} \ln \mathcal{N}\left(t_n \mid f(\mathbf{x_n}, \mathbf{w}), \Sigma\right)$$

Substituting the Gaussian distribution as,

$$\mathcal{N}(t_n \mid f(\mathbf{x}, \mathbf{w}), \Sigma) = \frac{1}{(2\pi\Sigma)^{1/2}} \exp\left\{-\frac{1}{2\Sigma}(x - f(\mathbf{x}, \mathbf{w}))^2\right\}$$

$$\ln p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \Sigma) = \frac{N}{2} \ln \Sigma^{-1} - \frac{N}{2} \ln(2\pi) - \frac{\Sigma^{-1}}{2} \sum_{n=1}^{N} \{t_n - f(\mathbf{x_n}, \mathbf{w})\}^2$$

Let us re-write $f(x, w) = w^T \phi(x)$, where $\phi(x)$ is a function of x.

$$\ln p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \Sigma) = \frac{N}{2} \ln \Sigma^{-1} - \frac{N}{2} \ln(2\pi) - \frac{\Sigma^{-1}}{2} \sum_{n=1}^{N} \left\{t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}\left(\mathbf{x}_n\right)\right\}^2$$

Let the sum of squares error be given by

$$\nabla \ln p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \Sigma) = \frac{d}{dw}(-\Sigma^{-1} E_D(w))$$

$$E_D(w) = \frac{1}{2} \sum_{n=1}^{N} \left\{t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}\left(\mathbf{x}_n\right)\right\}^2$$

On differentiating the log maximum likelihood wrt to w, we can see that maximization of the likelihood function under a conditional Gaussian noise distribution for a linear model is equivalent to minimizing a sum-of-squares error function.

$$\begin{aligned}
\nabla \ln p(\mathbf{t} \mid \mathbf{X}, \mathbf{w}, \Sigma) &= \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}\left(\mathbf{x}_n\right) \right\} \boldsymbol{\phi}\left(\mathbf{x}_n\right)^{\mathrm{T}} \\
0 &= \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}\left(\mathbf{x}_n\right) \right\} \boldsymbol{\phi}\left(\mathbf{x}_n\right)^{\mathrm{T}} \\
0 &= \sum_{n=1}^{N} t_n \phi\left(\mathbf{x}_n\right)^{\mathrm{T}} - \mathbf{w}^{\mathrm{T}} \left(\sum_{n=1}^{N} \phi\left(\mathbf{x}_n\right) \phi\left(\mathbf{x}_n\right)^{\mathrm{T}} \right)
\end{aligned}$$

Separating out w terms we obtain,

$$\begin{aligned}
\sum_{n=1}^{N} t_n \phi\left(\mathbf{x}_n\right)^{\mathrm{T}} &= \mathbf{w}^{\mathrm{T}} \left(\sum_{n=1}^{N} \phi\left(\mathbf{x}_n\right) \phi\left(\mathbf{x}_n\right)^{\mathrm{T}} \right) \\
\mathbf{w}_{\mathbf{MLE}} &= \left(\Phi^{\mathrm{T}} \Phi\right)^{-1} \Phi^{\mathrm{T}} \mathbf{t}
\end{aligned}$$

6 Weights symmetry

a) Scale symmetry

Since the weights are scaled by the same rate γ, during the back propagation all of them will be scaled by the same rate. This leads to the **vanishing gradient problem**.

- Let us assume that the incoming weights to a hidden layer are scaled by γ and outgoing weights are scaled by $\frac{1}{\gamma}$.
- Let us consider a hidden layer l, the updating of weights happens in the following manner,

$$\begin{aligned}
\Delta_l' &= \delta_{l+1}' * a_l' \\
&= \delta_{l+1} * \gamma * a_l \\
&= \gamma * \Delta_l
\end{aligned}$$

  Note that δ_l, δ_l' are before and after scaling at the hidden layer.

- If the value of γ is very high, then the gradients will explode in very less time. If the value of γ is very small, the gradients will vanish. In both these situations the network weights will not converge.

b) Permutation Symmetry

- If we interchange the values of all of the weights, bias leading both into and out of a particular hidden unit with the corresponding values of the weights, bias associated with a different hidden unit. This clearly leaves the network input–output mapping function unchanged, but it corresponds to a different choice of weight vector.

Vinta Reethu
Deep Learning for Vision
Indian Institute of Technology Hyderabad
Assignment-1

- For M hidden units, any given weight vector will belong to the set of $\mathbf{M!}$ equivalent weight vectors aligned with this interchange symmetry, referring to the $\mathbf{M!}$ numerous ordering of the hidden units.
- The network will therefore have an overall weight-space symmetry factor of $\mathbf{M!} * \mathbf{2^M}$.
- If there are t such layers, there are $(\mathbf{M!})^\mathbf{t}$ different ways in which they give the same output.
- We need to be careful while initialising the network weights, as the neural network itself is not convex and can attain many local minima.