Akash Tadwai
Indian Institute of Technology Hyderabad
Deep Learning for Vision
ES18BTECH11019

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Assignment-II

Akash Tadwai - ES18BTECH11019

February 24, 2021

1 RANSAC

Let **w** be the inlier ratio,

- As the number of degrees of freedom for homography is 8.
  we need atleast $n = \lceil d/2 \rceil$ points $= 4$ points.
- The probability that all the n points are inliers is w^n.
- The probability that atleast one of n points is an outlier is 1-w^n
- The probability that the algorithm never selects all the n points which are all inliers (Algorithm fails) is $(1 - w^n)^k$, where k is the number of iterations.

In this question, $w = 0.5$, $(1 - w^n)^k = 1 - 0.95$, we have to find k
Solving the equation $(1 - w^n)^k = 1 - 0.95$

$$
\begin{aligned}
\left(1 - \left(\frac{1}{2}\right)^4\right)^k &= 0.05 \\
\Rightarrow \left(\frac{15}{16}\right)^k &= \frac{1}{20} \\
\Rightarrow k &= \left\lceil \frac{log 20}{log \frac{16}{15}} \right\rceil \\
\Rightarrow k &= \lceil 46.41 \rceil \Rightarrow k \quad = 47.
\end{aligned}
$$

Akash Tadwai
Indian Institute of Technology Hyderabad
Deep Learning for Vision
ES18BTECH11019

2 Computing $\frac{\partial f}{\partial W_{ij}^1}$

Using the following notation for the rest of problem

$$w^3 = \begin{bmatrix} w_1^3 \\ w_2^3 \end{bmatrix} W^2 = \begin{bmatrix} W_{11}^2 & W_{12}^2 \\ W_{21}^2 & W_{22}^2 \end{bmatrix} W^1 = \begin{bmatrix} W_{11}^1 & W_{12}^1 \\ W_{21}^1 & W_{22}^1 \end{bmatrix}$$

We know that,

$$\frac{\partial}{\partial x}\sigma(x) = \sigma(x) * [1 - \sigma(x)]$$

Since $f(x) =< w^3, h^2 >$. From this we can find the derivative of f with respect to h_i^2 as follows:

$$\frac{\partial f}{\partial h_i^2} = w_i^3$$

Calculating the derivative of h_i^2 with respect to h_i^1:

$$\begin{aligned} h^2 &= \sigma(W^2 h^1) \\ h_i^2 &= \sigma(\sum_l W_{il}^2 h_l^1) \end{aligned}$$

$$\begin{aligned} \frac{\partial h_j^2}{\partial h_i^1} &= \sigma(\sum_l W_{jl}^2 h_l^1) * [1 - \sigma(\sum_l W_{jl}^2 h_l^1)] * W_{ji}^2 \\ &= h_j^2 * (1 - h_j^2) * W_{ji}^2 \end{aligned}$$

Using these derivatives, we can calculate derivative of f with respect to h_i^1 as follows:

$$\begin{aligned} \frac{\partial f}{\partial h_i^1} &= \sum_k \frac{\partial f}{\partial h_k^2} * \frac{\partial h_k^2}{\partial h_i^1} \\ &= \sum_k w_k^3 * h_k^2 * (1 - h_k^2) * W_{ki}^2 \end{aligned}$$

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

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

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

Akash Tadwai
Indian Institute of Technology Hyderabad
Deep Learning for Vision
ES18BTECH11019

Using all these equations, we can calculate $\frac{\partial f}{\partial W_{ij}^1}$ as follows:

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

3 Vectorisation

The equation is $\Delta_{ij}^{(2)} = \Delta_{ij}^{(2)} + \delta_i^{(3)} * a_j^{(2)}$.
We can vectorize it as, $\Delta_{ij}^{(2)} = \Delta_{ij}^{(2)} + \delta^{(3)} (a^{(2)})^T$.

4 Number of Weights and Biases

- **Number of Weights:** As there are two weight matrices of dimensions, $M \times d$ and $c \times M$, there are total of $\mathbf{M \times d + c \times M}$ weights.
- **Number of Biases:** As each neuron in the hidden and output layer has a bias associated with it. There are total of M and c biases in the hidden and output layer respectively. Hence there are total of $\mathbf{M + c}$ biases
- **Number of Derivatives: M+c**
  Explanation:
  Let the weights between input and hidden layer be W_1, weights between hidden and output layer be W_2.
  Number of independent derivatives in hidden layer :
  $$\frac{dE}{dW_1} = \delta^2 (a^1)^T$$
  As there are M nodes in hidden layer, number of independent derivatives = M.

  Number of independent derivatives in hidden layer :
  $$\frac{dE}{dW_2} = \delta^3 (a^2)^T$$
  As there are c nodes in hidden layer, number of independent derivatives = c.
  Total: $\mathbf{M + c}$

Akash Tadwai
Indian Institute of Technology Hyderabad
Deep Learning for Vision
ES18BTECH11019

5 Generalized Least Squares

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

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

We can express our uncertainty over the value of the target variable using a probability distribution. For this purpose, we shall assume that, given the value of x_n, the corresponding value of y_n has a Gaussian distribution with a mean equal to the value $f(x_n, w)$, Now the probability distribution of t (target) can be written as,

$$p(t \mid \mathbf{x}, \mathbf{w}, \Sigma) = \mathcal{N}\left(t \mid f(\mathbf{x}, \mathbf{w}), \Sigma\right)$$

Now considering a data set of inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with corresponding target values $t_1, \ldots, t_N$. Making the assumption that these data points are drawn independently from the distribution gaussian distribution described above, we obtain the following expression for the likelihood function, which is a function of the adjustable parameters $\mathbf{w}$ and Σ, in the form

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

As x will always appear in the set of conditioning variables, and so from now on we will drop the explicit x from expressions such as $p(\mathbf{t} \mid \mathbf{x}, \mathbf{w}, \Sigma)$ in order to keep the notation uncluttered. Taking the logarithm of the likelihood function, and making use of the standard form $\left(\mathcal{N}\left(x \mid \mu, \sigma^2\right) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}\right)$ for the univariate Gaussian, we have

$$\begin{aligned} \ln p(\mathbf{t} \mid \mathbf{w}, \Sigma) &= \sum_{n=1}^{N} \ln \mathcal{N}\left(t_n \mid f(\mathbf{x}, \mathbf{w}), \Sigma\right) \\ &= \frac{N}{2} \ln \Sigma^{-1} - \frac{N}{2} \ln(2\pi) - \Sigma^{-1} E_D(\mathbf{w}) \end{aligned}$$

where the sum-of-squares error function is defined by

$$\boxed{E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \left\{t_n - f(\mathbf{x}, \mathbf{w})\right\}^2}$$

Having written down the likelihood function, we can use maximum likelihood to determine $\mathbf{w}$ and Σ. Consider first the maximization with respect to $\mathbf{w}$. we know 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 given by $E_D(\mathbf{w})$. The gradient of the log likelihood function takes the form

Akash Tadwai
Indian Institute of Technology Hyderabad
Deep Learning for Vision
ES18BTECH11019

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

where we have written $f(x, w)$ as $w^T \phi(x)$ where $\phi(x)$ is some basis function.
Setting this gradient to zero gives

$$0 = \sum_{n=1}^{N} t_n \phi\left(\mathbf{x}_n\right)^{\mathrm{T}} - \mathbf{w}^{\mathrm{T}} \left(\sum_{n=1}^{N} \phi\left(\mathbf{x}_n\right) \phi\left(\mathbf{x}_n\right)^{\mathrm{T}}\right)$$

Solving for **w** we obtain

$$\boxed{\mathbf{w}_{\mathrm{MLE}} = \left(\mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^{\mathrm{T}} \mathbf{t}}$$

6 Weight Space Symmetries

6.1 Scale Symmetry

- **Answer:** During backpropagation the weights will be scaled by the same rate γ, and hence leading to Vanishing Gradient Problem.
- **Explanation:** Assuming that the all the incoming weights to a hidden layer are scaled by γ and outgoing weights are scaled by $\frac{1}{\gamma}$. While updating the weights, the corresponding Δ terms will be as follows:
- Considering gradients accumulated during back propagation. Let they be (δ_l, δ_l') before and after scaling at the hidden layer. Similarly, for preceding and successive layer's accumulated errors can be given by $(\delta_{l-1}, \delta_{l-1}')$ and $(\delta_{l+1}, \delta_{l+1}')$.

$$\begin{aligned}
\Delta_l' &= \delta_{l+1}' * a_l' \\
&= \delta_{l+1} * \gamma * a_l \quad \left(\delta_{l+1} = \delta_{l+1}', \delta_l = \frac{1}{\gamma} * \delta_l', \delta_{l-1} = \delta_{l-1}'\right) \\
&= \gamma * \Delta_l
\end{aligned}$$

- Hence the change term is also is scaled by γ, if γ is very small, the gradients vanish and else if γ is large, the gradients will explode leading to vanishing gradient problem.
- In both of these cases, neural network does not converge.

Akash Tadwai
Indian Institute of Technology Hyderabad
Deep Learning for Vision
ES18BTECH11019

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

6.2 Permutation Symmetry

- For **M** hidden units, any given weight vector will belong to a set of **M**! equivalent weight vectors associated with this inter- change symmetry, corresponding to the **M**! different orderings of the hidden units. The network will therefore have an overall weight-space symmetry factor of **M**!.
- If there are **l** such layers there are total, $(M!)^l$ permutations of the weights which can give same output.
- As the Neural Network loss function is not convex there are many local minima encountered during training. Hence one needs to be careful while initializing weights for good performance.

******THE END******