C18 Computer Vision Sheet 1

Instructor Andrea Vedaldi (vedaldi@eng.ox.ac.uk)
Web Page http://www.vlfeat.org/~vedaldi/teach.html
Revision HT 2024 (typeset on December 12, 2023)

- For updates, answers and hints, keep in touch with the web page.
- Email if you are puzzled about a question's meaning. The reply, if generally useful, will be placed in the hints section.

Problem 1 Geometric verification

Geometric verification starts with M putative matches $(\mathbf{x}_1, \mathbf{x}'_1), \dots, (\mathbf{x}_M, \mathbf{x}'_M)$ between feature points extracted from two images and searches for the largest subset of them that is compatible with a simple overall image transformation (the subset of inliers).

For simplicity, in this exercise we will assume that image transformations are *similarities* of the type

$$\mathbf{x}' = sR\mathbf{x} + T$$

where $s \in \mathbb{R}_+$ is a scaling factor, R is a 2×2 rotation matrix, and T is a 2D translation. The goal of geometric verification is to find the similarity transformation that fits accurately the largest possible subset of feature matches:

$$(s^*, R^*, T^*) = \operatorname*{argmax}_{s,R,T} |\{i : \|\mathbf{x}'_i - sR\mathbf{x}_i - T\| \leq \epsilon\}|$$

where ϵ is the tolerance for considering a match correct and $|\cdot|$ denotes the cardinality (number of elements) of a set.

This is a highly non-linear problem which is usually solved by using RANSAC. Recall that RANSAC repeats the following two steps:

1. It samples at random a minimal subset of matches, just enough to be able to estimate (s, R, T) from them.
2. It scores the estimate (s, R, T) by counting how many matches agree with it, in the sense specified above.

This is repeated a certain number of times t in the hope that at least one of the randomly-sampled subsets of matches is composed only of inliers. Such a subset should result in the correct similarity transformation being estimated, which in turn should result in a large number of inliers.

Answer the following questions:

1. How many matches $m \ll M$ are needed to estimate a similarity transformation (s, R, T)?
2. Assuming that only a fraction $p = 1/5$ of matches are inliers, estimate the smallest number of RANSAC trials t needed such that a minimal set of inliers is found with probability at least 99%.

3. In practice, with covariant feature detectors we do not have just feature points but more complex features. For example, in the case of SIFT, each feature is a circle with a center $\mathbf{x}$, a radius ρ, and an orientation θ. Explain how RANSAC can be modified in order to take advantage of this information.

Problem 2 Precision and recall

Consider an image retrieval system for a database containing 7 images in total. Given a certain query, the system responds with a ranked list of results whose relevances to the query are, in order of decreasing confidence, $(+1, -1, +1, -1, -1, +1, -1)$ (where $+1$ means relevant and -1 means not relevant).

1. Plot the precision-recall curve and compute the average precision for this result.

2. How would the relevances, precision-recall cure, and average precision change for a query perfectly answered? And for the worst possible answer?

Problem 3 Tensor derivatives

1. The vec operator reshapes a tensor $\mathbf{x}$ into a column vector $\operatorname{vec}\mathbf{x}$. The specific ordering of the tensor elements in the resulting vector is arbitrary, but fixed. In PyTorch, a tensor $\mathbf{x} \in \mathbb{R}^{C\times H\times W}$ can be vectorized using the command `x.view(-1)`. Using PyTorch's convention, consider an element x_{cvu} in $\mathbf{x}$ and find out the index i of the corresponding element $[\operatorname{vec}\mathbf{x}]_i$ in $\operatorname{vec}\mathbf{x}$.

2. Consider a tensor function $\mathbf{y} = f(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^{U_1\times U_2\times U_3}$ and $\mathbf{y} \in \mathbb{R}^{V_1\times V_2\times V_3}$. The vec operator can be used to define the Jacobian of the vectorized function:

$$\frac{d\operatorname{vec} f}{d\operatorname{vec}\mathbf{x}}. \tag{1}$$

   Find out the dimensions of this matrix. Then, find out which derivative is stored at row i and column j, also using PyTorch convention for vectorization.

3. Let $g \circ f$ be the composition of two scalar functions $z = g(y)$ and $y = f(x)$ (hence x, y and z are all scalars). Show that

$$\frac{d(g\circ f)}{dx} = \frac{dg}{dy}\cdot\frac{df}{dx}.$$

   Now assume that $\mathbf{z} = g(\mathbf{y})$ and $\mathbf{y} = f(\mathbf{x})$ are vector functions (in the sense that $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ are all vectors). Show that:

$$\frac{d(g\circ f)}{d\mathbf{x}} = \frac{dg}{d\mathbf{y}}\cdot\frac{df}{d\mathbf{x}}.$$

   Finally, assume that g and f are tensor functions (in the sense that $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ are all tensors). Show that

$$\frac{d\operatorname{vec}(g\circ f)}{d\operatorname{vec}\mathbf{x}} = \frac{d\operatorname{vec} g}{d\operatorname{vec}\mathbf{y}}\cdot\frac{d\operatorname{vec} f}{d\operatorname{vec}\mathbf{x}}.$$

Problem 4 Back-propagation algorithm

1. Consider a two-layer neural network $g \circ f$ and let $\mathbf{x}$ be the input tensor, $\mathbf{y} = f(\mathbf{x})$ the intermediate tensor and $z = g(\mathbf{y})$ the output tensor. Show that:

$$\frac{d\,\mathrm{vec}(g \circ h)}{d\,\mathrm{vec}\,\mathbf{x}} = \frac{d\,\mathrm{vec}\,g}{d\,\mathrm{vec}\,\mathbf{y}} \cdot \frac{d\,\mathrm{vec}\,f}{d\,\mathrm{vec}\,\mathbf{x}} \tag{2}$$

2. Assume that the dimensions of the input and intermediate tensors $\mathbf{x}$ and $\mathbf{y}$ are $C \times H \times W$, where $H = W = 16$ and $C = 256$ and that the output tensor $z \in \mathbb{R}$ is a scalar (usually the value of the loss). Find the size of the Jacobian matrices in eq. (2) and find out how much memory is required to store them in IEEE single precision format.

3. Let $\mathbf{p}$ by a fixed tensor that has the same size as $\mathbf{y}$. We define the backward version of f as:

$$\mathbf{p}' = f^{\mathrm{BP}}(\mathbf{x}; \mathbf{p}) = \frac{d\langle \mathbf{p}, f(\mathbf{x})\rangle}{d\mathbf{x}}$$

   Here the symbol $\langle \cdot, \cdot \rangle$ denotes the inner product between the two tensor arguments. Find the size of the tensor $\mathbf{p}'$.

4. Show how $f^{\mathrm{BP}}(\mathbf{x}; \mathbf{p})$ can be used to compute eq. (2) without having to store explicitly the large Jacobian matrix $d\,\mathrm{vec}\,f / d\,\mathrm{vec}\,\mathbf{x}$.

5. Given a deep neural network $h = f_n \circ f_{n-1} \circ \cdots \circ f_1$ where $\mathbf{x}_i = f_i(\mathbf{x}_{i-1})$ are all tensor functions show that:

$$\frac{d\,\mathrm{vec}\,h}{d\,\mathrm{vec}\,\mathbf{x}_0} = \frac{d\,\mathrm{vec}\,f_n}{d\,\mathrm{vec}\,\mathbf{x}_{n-1}} \cdot \frac{d\,\mathrm{vec}\,f_{n-1}}{d\,\mathrm{vec}\,\mathbf{x}_{n-2}} \cdots \frac{d\,\mathrm{vec}\,f_1}{d\,\mathrm{vec}\,\mathbf{x}_0}. \tag{3}$$

   Then, extending the reasoning above, show that, provided that all layers have a backward function implementation, one can compute the overall derivative of eq. (2) without storing any exceedingly large matrix in memory. In particular, under the assumptions that all tensors but the last one have dimensions $C \times H \times W$ as above, compute the amount of memory required to run this algorithm.

Problem 5 Layers and backward mode

This exercise looks at the problem of defining a new layer in a CNN and implementing its forward and backward modes. Consider in particular the layer $\mathbf{y} = f(\mathbf{x})$ which subtracts from each feature channel of $\mathbf{x}$ the corresponding mean:

$$y_{cv} = x_{cv} - \frac{1}{HW} \sum_{u=0}^{(H,W)-1} x_{cu}.$$

Recall that we use index base 0 and we use the multi-index $u = (u_1, u_2)$ for compactness. A related layer, called batch normalisation, is very commonly used in recent CNN architectures to improve their numerical conditioning.

1. Find an analytical expression for the backward function f^{BP}.

2. Write MATLAB or Python functions that implement f and f^{BP}.

3. In writing the MATLAB or Python unction that implements f^{BP}, did you need to explicitly compute the very large Jacobian matrix $d\operatorname{vec} f / d\operatorname{vec} \mathbf{x}$? If so, can you rewrite your implementation to be memory efficient?

4. Computing derivatives is error-prone. Verify your implementation of f and f^{BP} *numerically*. In order to do so, recall that, given a scalar function $g(x)$, its derivative can be approximated by finite differences:
$$\frac{dg}{dx}(x) \approx \frac{g(x+\delta) - g(x)}{\delta}.$$
Use this fact to write a MATALB or Python program that computes an approximation of the output of $f^{\mathrm{BP}}(\mathbf{x}, \mathbf{p})$ by taking the differences of forward evaluations of $f(\mathbf{x})$ (you can use a random but fixed value of $\mathbf{x}$ and $\mathbf{p}$ for this experiment). Then, check that this approximation and f produce nearly the same answer. If this is not the case, then your code is buggy and should be fixed.

Problem 6 Convolutional networks

This problem studies a simple but realistic CNN for classification of small images. The model is known as LeNet and is one of the earliest deep convolutional neural networks, designed for handwritten digit classification. The network takes as input a 28×28-dimensional image of a digit and produces a output a 10-dimensional vector of class probabilities, one for each of 10 possible digits.

LeNet uses four standard layer types: convolution, max pooling, ReLU and softmax cross-entropy loss. The following table shows how these layers are combined and configured:

layer	0	1	2	3	4	5	6	7	8
type	input	conv	mpool	conv	mpool	conv	relu	conv	softmxl
filt. shape	–	$1 \times 5 \times 5$	2×2	$20 \times 5 \times 5$	2×2	$50 \times 4 \times 4$	–	$500 \times 1 \times 1$	–
num. filt.	–	20	–	50	–	500	–	10	–
stride	–	1	2	1	2	1	–	1	–
pad	–	0	0	0	0	0	–	0	–
data shape	$1 \times 28 \times 28$	?	?	?	?	?	?	?	?
data size	3KB	?	?	?	?	?	?	?	?
r.f. size	1	?	?	?	?	?	?	?	?

1. Given that the input tensor $\mathbf{x}_0$ has shape $1 \times 28 \times 281$, find out the shapes of all other tensors $\mathbf{x}_1, \dots \mathbf{x}_8$. Compute also the memory size of each tensor (in bytes) and the memory requirements for backpropagation.

2. The *receptive field* of element $[\mathbf{x}_t]_{cvu}$ of tensor $\mathbf{x}_t$ is the subset of the input image $\mathbf{x}_0$ that can affect $[\mathbf{x}_t]_{cvu}$. Calculate the size of the receptive fields for the elements of tensors $\mathbf{x}_1, \dots, \mathbf{x}_8$ for LeNet.