Review for Exam 1

Erik G. Learned-Miller
Department of Computer Science
University of Massachusetts, Amherst
Amherst, MA 01003

March 26, 2014

Abstract

Here are some things you need to know for the in-class exam. If you cover this material, the handouts posted on the web page, the first 3 problem sets, and the lecture slides, you should be in good shape for the test.

1 Basic notation

You should understand the following types of notation.

- Summation ($\sum$). Example:

$$\sum_{i=1}^{N} X_i = X_1 + X_2 + ... + X_N. \tag{1}$$

- Product ($\prod$). Example:

$$\prod_{i=1}^{N} X_i = X_1 \times X_2 \times ... \times X_N. \tag{2}$$

- When we perform optimization of a function with respect to some changeable parameter, we are usually interested in either the minimum or maximium value of the function itself, or, alternatively, we are interested in the values of the parameter that achieved the maximum or minimum.

  Suppose we are trying to find the maximum value of a function $f(x, y)$ when y is set to the value 3 and x can be any value from some set S. To specify the maximum value of the function we write:

$$\max_{x \in S} \; f(x, y = 3). \tag{3}$$

  We frequently assign this maximum to some variable, as in:

$$F^* = \max_{x \in S} \; f(x, y = 3). \tag{4}$$

- If we are interested in the value of x which gave us this optimum, we use the $\arg\max$ notation:

$$\arg\max_{x \in S} \; f(x, y = 3). \tag{5}$$

  In this case, of course, the maximizing x might be called x^*:

$$x^* = \arg\max_{x \in S} \; f(x, y = 3). \tag{6}$$

2 A bit of vector math

You will need to understand certain basic results about vectors. For the purposes of this class, you can think of a vector as just a collection of numbers, like in Matlab. For example, a vector of length 3, also known as a vector $\mathbf{v}$ in three dimensions, or a 3-dimensional vector, might be something like

$$\mathbf{v} = \begin{bmatrix} 12 \\ -3 \\ 4 \end{bmatrix}.$$

2.1 Vector magnitudes

The magnitude of a vector is a simple concept. You can think about it as the length of the vector, or equivalently, as the distance from the point specified by the vector to the origin of the coordinate system, that is, the distance between the vector and the zero vector. Using the example of **v** defined above, the magnitude of **v** is written

$$||\mathbf{v}|| = \sqrt{12^2 + (-3)^2 + 4^2} = 13.$$

2.2 Unit vectors

A unit vector is just a vector whose length is 1, like

$$\mathbf{w} = \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix}.$$

To convert a vector whose length is *not* 1 into a unit vector that is pointing in the same direction (whose components have the same *relative magnitude* as the original vector), just divide each original component by the magnitude of the vector. For our example **v** from above:

$$\mathbf{u} = \mathbf{v}/13 = \begin{bmatrix} \frac{12}{13} \\ \frac{-3}{13} \\ \frac{4}{13} \end{bmatrix}.$$

You can verify for yourself that **u** has a magnitude of 1.

2.3 Dot products of vectors

The dot product of two vectors is just the sum of the product of the correpsonding components. Let

$$\mathbf{t} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}.$$

Then the dot product of **t** and **v** is

$$\mathbf{t} \cdot \mathbf{v} = (2)(12) + (1)(-3) + (3)(4) = 33.$$

If the components of **v** are $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$, and the components of **t** are $\mathbf{t}_1$, $\mathbf{t}_2$, and $\mathbf{t}_3$, then the dot product can be written

$$\mathbf{t} \cdot \mathbf{v} = \sum_{i=1}^{3} (\mathbf{t}_i)(\mathbf{v}_i).$$

2.4 Angle between unit vectors

When the bases of two vectors are put together (for example, if they are both at the origin), then an angle is formed between them. Let's call it θ. **For unit vectors**, the following formula gives the relationship between the angle θ and the two vectors $\mathbf{u}$ and $\mathbf{v}$:

$$\cos(\theta) = \mathbf{u} \cdot \mathbf{v}. \tag{7}$$

If the two vectors being compared **are not unit vectors**, then they have to be converted to unit vectors before the formula works. Consider two vectors $\mathbf{y}$ and $\mathbf{z}$ that are not unit vectors. To find the angle between them, normalize them first:

$$\cos(\theta) = \frac{\mathbf{y}}{||\mathbf{y}||} \cdot \frac{\mathbf{z}}{||\mathbf{z}||}. \tag{8}$$

Note: these formulas work for vectors with any number of components, whether it is 2, 3, 5, or a hundred thousand components!

2.5 Vector that maximizes the dot product

Given a unit vector $\mathbf{v}$, which other unit vector maximizes the dot product with $\mathbf{v}$? To answer this question very simply, first ask, "What angle θ maximizes the cosine function (from equation 7)?" The answer is that the cosine function is maximized when $\theta = 0$ and $\cos(0) = 1$. Thus, to maximize the dot product between two unit vectors, the angle between them should be 0! This implies that the unit vector which maximizes the dot product with a unit vector $\mathbf{v}$ is the vector $\mathbf{v}$ itself! In other words:

$$\mathbf{v} \cdot \mathbf{v} \geq \mathbf{v} \cdot \mathbf{w},$$

for any unit vector $\mathbf{w}$.

2.6 Exercises

Here are simple questions to test your knowledge:

1. What's the magnitude of a unit vector?
2. What's the maximum possible value of the dot product between two unit vectors? (Answer: it's the same as the maximum value of the cosine function.)
3. What's the maximum possible value of the dot product between two vectors if their magnitudes are 2 and 5 respectively. (Answer: multiply both sides of equation 8 by the magnitudes of $\mathbf{y}$ and $\mathbf{z}$ and reason from there.)

2.7 Images as vectors

Consider a grayscale image that has 100 rows and 100 columns. There are 10,000 pixels in the image. You can think of this as a 10,000 component vector. Now consider two images I and J. If you think of them as two 10,000 component vectors, you can compute the dot product between them. If we normalize the two vectors, and take their dot product, what is the maximum possible value? (Answer: 1.0, the maximum of the cosine function.)

Suppose we have taken an image I, turned it into a vector, and normalized it, by dividing each element of the vector by the magnitude of I. What other normalized image vector would maximize the dot product with the normalized I? The answer, drawing from section 2.5, is *the same image*.

3 Image Formation

- Understand the basics of the electromagnetic spectrum, that every point on the spectrum represents light of a particular wavelength (and hence frequency). You should know at least one type of light that has longer wavelengths than visible light (e.g., infrared), and one type of light that has shorter wavelengths (e.g., ultraviolet).
- Know about the visible spectrum. Why is it visible? (Answer: it stimulates the photoreceptors in our eyes, whereas other wavelengths do not.)
- Know the approximate wavelengths of the visible spectrum: 380nm (violet) to 760 nm (red).
- A laser of a single wavelength will appear as one of the basic "rainbow" colors on the spectrum. For example, a 550nm laser will appear green.
- Most natural lights are composed of a broad distribution of wavelengths. Example: white light from a lightbulb has a broad distribution of wavelengths.
- We perceive light through our rod and cone cells (3 types). The color we perceive is only a function of the relative stimulation of our 3 types of cone cells.
- The cone cells are sensitive to short wavelengths ("blue"), medium wavelengths ("green") and long wavelengths ("red").
- Colors such as yellow are perceived when multiple types of cones cells (in this case red and green) are stimulated together.
- Understand the linearity of light (see slides from lecture 2 and problem set 2).

- Understand the inverse square law of light propagation. If you are at a distance d from a point light source, and you double your distance to $2d$, the amount of light falling on a unit area drops by a factor of 4. Why? (Answer: because the percentage of the sphere of light that the surface is "catching" drops of as the square of the distance from the center of the sphere.)
- Understand how a pinhole camera works. Why doesn't it need a lens? Answer, because each point in the image is only exposed to light from a single point in the world, so it is naturally in focus.
- Given the dimensions of a pinhole camera, be able to determine the size of the image of an object given it's size and distance from the camera. To do this, you use similar triangles and solve for the height of the object, as done in class.
- Know the thin lens law as described in the lecture slides.
- Know that to an ideal lens whose cross sectional area is 100 times larger than a pinhole essentially lets in 100 times as much light as the pinhole, so a photo taken with this lens could produce the same brightness image in about $\frac{1}{100}$th of the time.

4 Basic probability

- Know all the material from the *Review of Basic Probability* handout.
- Particularly focus on marginalization and Bayes' rule.
- Suppose we are trying to classify images into one of several classes c_1, c_2, and c_3, and to do it we are using feature values f_1 and f_2. Know what is meant by the *likelihoods of the classes*, the *priors of the classes*, and the *posterior probabilty* for each class.
- Marginalization: If you are given a probability distribution for each pair of possible values of $Prob(X, Y)$, then you should be able to compute $Prob(X)$ and $Prob(Y)$ for each possible value of X and Y.
- Know the definition of statistical independence: For two random variables X and Y, X and Y are statistically independent if and only if $P(X|Y) = P(X)$ for all X and Y. Or alternatively, $P(X, Y) = P(X)P(Y)$. Be able to derive one of these formulas from the other (multiply both sides by $P(Y)$.
- Conditional probability. Be able to compute $P(X|Y)$ from $P(X, Y)$ and $P(Y)$.
- Be able to given reasonable estimates of $P(X)$, $P(X|Y)$, and $P(X, Y)$ from raw data.

5 Classification

- Understand Bayes' rule. Practice it many times so you are fluid with it. Try making up some tables of $p(x|c)$ where x is a feature value and c is a class, and then try to compute $p(c|x)$. You will also need, of course, $p(c)$ the prior probabilty of a class. *If you don't understand this, you better get help before the test.*
- If you *estimate* the posteriors for each class ($p(c|x)$) and you pick the class with the highest posterior, this is known as *maximum a posteriori* classification, or MAP classification. MAP classification is not ALWAYS the best strategy when you have *estimated* the posteriors.
- However, if you have the *exact*, true posteriors for the classes, then choosing the class with the highest posterior yields a *Bayes Optimal Classifier*, which is a classifier with the minimum probability of error. This smallest possible probability of error is also known as the Bayes error rate, or just the Bayes error.
- Given the true likelihoods (not the ones estimated from data) for each class, and the true priors (not the ones estimated from data), one can calculate the exact posteriors ($p(c|x)$), and hence, build a Bayes optimal classifier, using Bayes' rule and MAP classification. (MAKE SURE YOU UNDERSTAND THIS!)

6 Supervised learning

- Know the definition of supervised learning.
- Know how nearest neighbor classification works.
- Know how k-nearest neighbor classification works.
- What happens if my training data in supervised learning comes from one probability distribution $p(x, y)$ and my test data comes from a very different distribution $q(x, y)$. Answer: The performance of my classifier may be arbitrarily bad, since I trained on the "wrong data". Example: My training data consists of red apples (class A) and red pears (class B). My test data consists of green apples and green pears. I cannot expect to get the test examples correct (although I may get lucky and use shape as a feature instead of color, but that would just be luck.)
- What is the fundamental problem with estimating a distribution over entire images? Describe the trade-offs between using smaller parts of the image to classify images (like a single pixel) and using larger parts, like mutliple pixels or even the entire image. (difficulty of estimation versus amount of information in the features).

7 Basic Matlab familiarity

- Know how to turn the values in a matlab matrix into a vector (use the colon operator).
- Know how to transpose a matrix and what this means (you can use either the quote operator (') or use the transpose command. This can used to turn a row vector into a column vector or vice versa.
- Understand the structure of a "grayscale" or "scalar-valued" image. It is simply a 2-dimensional array of numbers.
- Understand the structure of a "color" image: it is a 3-dimensional array of numbers in which the first "layer" represents red, the second layer represents green, and the third represents blue.
- IMPORTANT: Understand that a *grayscale* or *scalar-valued* image can be rendered in color using a *look-up-table*, which is essentially a scheme for doing color-by-number. That is, for each value in the image, the computer looks up the red-green-blue (RGB) color that should be used for that particular number.
- Know the repmat command in matlab. Know that it can be used to avoid doing for loops, and that for loops are slow in matlab.

8 Image comparison

- Understand the "sum of squared differences" between two scalar-valued images I and J:

  $$SSD(I,J) = \sum_{i=1}^{\#ofpixels} (I_i - J_i)^2.$$

  This is the same thing as the "Euclidean distance" except that the Eucildean distance has a square root:

  $$\begin{aligned} D_{Euclid}(I,J) &= \sqrt{\sum_{i=1}^{\#ofpixels} (I_i - J_i)^2} \\ &= \left(\sum_{i=1}^{\#ofpixels} (I_i - J_i)^2 \right)^{\frac{1}{2}}. \end{aligned}$$

- Understand the "sum of absolute differences":

$$\begin{aligned} D_{abs}(I,J) &= \sum_{i=1}^{\#ofpixels} |I_i - J_i| \\ &= \left(\sum_{i=1}^{\#ofpixels} |I_i - J_i| \right)^{\frac{1}{1}}. \end{aligned}$$

  Understand why one might want to use one instead of the other (sum of squared differences weighs larger errors more heavily, on average).

9 Understanding the sizes of sets

- How many distinct 10x10 binary images are there? Answer: 2^{100}.
- How many 1000x1000 color images are there if each color channel (red, green, blue) can take on 256 colors? First, calculate the number of colors per pixel:

$$\begin{aligned} c &= 256 * 256 * 256 && (9) \\ &= 2^8 * 2^8 * 2^8 && (10) \\ &= 2^{24} && (11) \\ &= 2^{20} * 2^4 && (12) \\ &\approx 1,000,000 * 16 && (13) \\ &= 16\ million. && (14) \end{aligned}$$

  Given about 16 million colors per pixel, the number of 1000x1000 images is

$$(16\ million^{1000*1000}) = (16\ million)^{million}.$$

  According to some estimates, there are about 2^{80} particles in the known universe.

- Given 100 x translations of an image, 100 y translations, 1000 rotations, and 500 different scales, how many different images tranformations could I produce? I'll let you do this one yourself.

10 Filtering an image

Given an image I with N pixels and another image J of *exactly the same size*, then the result k of filtering I with J is simply the dot product of the two images:

$$k = \sum_{i=1}^{N} I_i \times J_i,$$

where I_i and J_i are the pixel values in each image.

To filter an image I with a smaller image J, repeat the process described above for each patch of I that is the same size as image J by "sliding" the filter window across each possible position in I.

When you filter a larger image I with a smaller patch J, you will generally end up with a smaller image as a result. For example if I is 7x7 and J is 3x3, then the final image will be 5x5, because there are exactly 25 locations where the entire filter kernel J will fit within the image I. If it is important to make the final image the same size as the original image, one can allow the filter to extend outside the image I. To do this, one has to invent values for the missing pixels outside the image I. Usually, a value of 0 is used.

10.1 Some very common filter kernels

We have mentioned a variety of common filter kernels in class. Here is a review.

10.1.1 Averaging filter

The following filters compute the average brightness value of the patch that they are filtered with:

$$f_{3x3} = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

$$f_{5x5} = \begin{bmatrix} \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \end{bmatrix}$$

The first one averages over a 3x3 neighborhood. The second over a 5x5 neighborhood.

10.1.2 Weighted average filter

Instead of each pixel contributing equally to the average, we can have pixels near the middle of the filter contribute more and pixels near the edge of the filter contribute less, as in

$$f_{Gauss} = \begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix}$$

While you don't need to memorize the specific values of such a filter, things you should notice include

- The values are all positive,
- The values in the middle are largest, decreasing away from the middle,
- The values sum to 1.0,
- They are symmetric about the center.

If you plotted the filter as a surface it would look approximately like a two-dimensional Gaussian (or normal) distribution. This is the kind of filter that one uses to produce the smoothed (or blurred) images used in a Gaussian pyramid (such as the one used in the SIFT descriptor).

10.1.3 Edge filters

Some filters, which look like little "mini-edges" if you show them as an image, are good for detecting edges. The following filters can be used to find horizontal edges in images, vertical edges, and diagonal edges, respectively:

$$f_{vert} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix},$$

$$f_{hor} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix},$$

$$f_{diag} = \begin{bmatrix} 0 & 0.5 & 1 \\ -0.5 & 0 & 0.5 \\ -1 & -0.5 & 0 \end{bmatrix}.$$

10.1.4 Partial derivative filters

In class, we discussed ways of estimating the gradient of an image at each point. This is the direction in which an image is changing the brightness the fastest, and can be written as a vector of the partial derivatives:

$$\nabla I = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix}.$$

To compute a "discrete" approximation to the partial derivatives you can use the following filters:

$$f_{\partial x} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix},$$

and

$$f_{\partial y} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix}.$$

Alternatively, you could use the vertical edge filter (x-derivative) and horizontal edge filter (y-derivative) defined above to compute approximations to the partial derivatives.

10.1.5 The trivial "identity" filter

You should be able to figure out what the following filter does. If not, you don't understand filters yet:

$$f_{ident} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

11 Mutual Information

- Mutual information is a function of the *joint distribution* of two random variables. You do NOT need to memorize the mathematical definition of mutual information. However, if I give you a joint distribution (as a table) and the definition of mutual information, you should be able to compute it. This will include computing the *marginal distributions* from the joint distribution.

- Mutual information is 0 if and only if the random variables of the joint distribution are statistically independent. I want you to be able to show that *if* two random variables are independent, then their mutual information is 0. (You don't have to be able to show that this is the *only way* that the mutual information can be 0.)

- A good feature, whose value in an image is represented by the random variable X should have *high* mutual information with the class label for an image, represented by the random variable C. If it is not, then the feature random variable doesn't give you any information about the class, which means it is a useless feature.

- Mutual information cannot be higher than the amount of information in either random variable. What do I mean by this? Suppose I am doing a classification problem with 4 classes. Then 2 bits of information is enough to tell me which class I'm in. Thus, the mutual information between any feature and the class label cannot be more than 2 bits.