Figure 1: A blurred mystery photo taken at the Ithaca SPCA.

Proj 1: Where Are My Glasses?
(due: 2020-03-06)

1 Introduction

The image in Figure 1 is a blurred version of a picture that I took at the local SPCA. As we will see, a naive approach to de-blurring produces garbage, a characteristic feature of *ill-posed* problems. In order to reconstruct the image, we need to *regularize* the reconstruction, just as we would regularize an ill-posed least squares problem. We will use Tikhonov regularization, as described in class. However, this involves choosing the value of a regularization parameter. Your mission is to investigate the dependence on the parameter, and to investigate three approaches to choosing this parameter (mostly) automatically.

You are given the file blurry.png (the blurred image shown above) and several supporting MATLAB files (Julia and Python versions to be posted). You are responsible for the following deliverables:

- Codes to compute "optimal" values of the regularization parameter λ via the discrepancy principle, the L-curve, and generalized cross-validation.
- A report that addresses the questions posed in the rest of this prompt.

This project is inspired by the project on image deblurring by James G. Nagy and Dianne P. O'Leary ("Image Deblurring: I Can See Clearly Now" in *Computing in Science and Engineering*; Project: Vol. 5, No. 3, May/June 2003, pp. 82-85; Solution: Vol. 5, No. 4, July/August 2003). Other useful references include:

- Hansen, Nagy, O'Leary. *Deblurring Images: Matrices, Spectra, and Filtering*, SIAM 2006.
- Hansen and O'Leary. "The Use of the L-Curve in the Regularization of Discrete Ill-Posed Problems" in *SIAM J. Sci. Comput.*, Vol 14, No. 6, November 1993.
- Golub, Heath, and Wahba. "Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter" in *Technometrics*, Vol. 21, No. 2, May 1979.
- Golub and von Matt. "Generalized Cross-Validation for Large-Scale Problems" in *Journal of Computational and Graphical Statistics*, Vol. 6, No. 1, March 1997.
- Morozov. *Methods for Solving Incorrectly Posed Problems*, Springer-Verlag, 1984.

It should be possible to do this assignment based only on ideas presented in this prompt and in the lectures. However, you are always welcome to use any ideas or code you find in the literature, including in the references noted above, provided you give an appropriate citation.

2 Regularization

Images in MATLAB are represented as three-dimensional arrays of size height $\times$ width $\times$ 3, where the three layers represent the red, green, and blue color intensities at each pixel. The blurred image file in Figure 1 has eight-bit color depth, which means that each of the RGB values is represented as an integer between 0 and 255 (inclusive). Abstractly, we have that for each color plane,

$$V^{\text{blur}} = \text{round}(HV^{\text{orig}})$$

where the round operation represents rounding to the nearest integer and H is a linear blurring operation. The matrices $V \in \mathbb{R}^{n\times 3}$ represent images with n pixels and three colors (though we will always youw the three-index representation in MATLAB). Our blurring operation is a *convolution*, which can be diagonalized by a Fourier transform Z; that is,

$$H = Z^* S Z$$

where Z is a unitary matrix ($Z^*Z = I$) whose action can be applied by the two-dimensional fast Fourier transform and S is a diagonal matrix of eigenvalues. The matrix H is not symmetric, so the eigenvalues are complex!

The simplest reconstruction approach is to simply solve

$$V^{\text{naive}} \approx H^{-1} V^{\text{blur}}.$$

which we can do programmatically by

- Transforming V^{blur} into the Fourier basis: $\hat{V} = ZV^{\text{blur}}$.
- Forming $U = S^{-1}V$ (i.e. scaling each element of V).
- Transforming back: $V^{\text{naive}} = Z^*U$.

A better approach to deblurring the image is *Tikhonov regularization*:

$$V^{\text{tik}}(\lambda) = \text{argmin}_V \, \|HV - V^{\text{blur}}\|_F^2 + \lambda^2 \|V\|_F^2.$$

We will now investigate why this approach is better.

Tasks

1. Generate a deblurred image by running the following commands

```
obj = p1setup();
imnaive = p1tikhonov(obj, 0);
image(imnaive);
```

   Ideally, this naive deblurring approach should reproduce the original image. Does it? What happens if you set $\lambda = 0.01$, i.e.

```
imtik = p1tikhonov(obj, 0.01);
image(imtik);
```

Include both images in your report — you will want to use the MATLAB print command.

2. Look at the code in p1tikhonov. Using your understanding of the normal equations and of the eigenvalue decomposition of H, explain why this code does actually solve the Tikhonov minimization problem.

3. Because of rounding, the blurred image actually looks like

$$V^{\text{blur}} = HV^{\text{orig}} + E$$

where E is a backward error. What is the maximum error per pixel (assuming no rounding error in applying H)? What is a natural bound on $\|E\|_F^2$?

4. Using the decomposition $H = Z^*SZ$, what are the singular values of H? The eigenvalues of H are in the obj.s vector — what is the smallest singular value?

5. Based on the error bound on $\|E\|_F^2$ and the value of the smallest singular value, provide a bound on $\|V^{\text{orig}} - V^{\text{naive}}\|_F$. Compare to $\|V^{\text{blur}}\|_F$, and argue that the bad behavior of the naive deblurring approach is completely predictable.

3 Choosing the parameter

The unsatisfactory feature of Tikhonov regularization is that there is a free parameter λ, and so far we have no guidance as to how to find it. In the remainder of this project, we consider three different approaches to finding an "optimal" λ according to three different criteria.

3.1 Morozov's discrepancy principle

Morozov's discrepancy principle says roughly that we should choose λ so that the residual

$$\rho(\lambda) = \|HV^{\text{tik}}(\lambda) - V^{\text{blur}}\|_F$$

is about the same size as the difference between V^{blur} and the "correct" right hand side. Usually this distance is difficult to estimate (which often makes the discrepancy principle hard to apply in practice), but in our case we already have a good estimate.

Tasks

1. Write a routine to compute λ such that $\rho(\lambda)$ is approximately $\sqrt{n/4}$ where n is the number of pixels (this corresponds to assuming the rounded-off quantity is uniformly distributed):

```
function [lambda] = p1discrepancy(obj, lmin, lmax)
```

   where lmin and lmax are user-defined lower and upper bounds on the value of the regularization parameter λ. If you would like, default values of 10^{-4} and 1 span a good range. You should feel free to re-use p1tikhonov if you would like; you may also use p1applyH, which applies the blurring operator to an image. You may use MATLAB's fzero — or just write a bisection routine.

2. What is the optimal λ by this criterion? Show the image for the computed regularization parameter.

3. On a log-log scale, plot $\rho(\lambda)$ against λ over a "reasonable" range of values around the recommended regularization parameter. Use a dashed horizontal line to indicate $\sqrt{n/4}$, and mark somehow the point on the curve associated with the desired value of λ. Are there any visually distinctive features of the plot that suggest this is around the right value?

3.2 Generalized cross-validation

The *generalized cross-validation criterion* involves minimizing

$$G(\lambda) = \frac{N\rho(\lambda)^2}{\Big(\mathrm{tr}(I - H\hat{H}^\dagger(\lambda))\Big)^2},$$

where $\hat{H}^\dagger(\lambda)$ is the solution operator for the Tikhonov regularized problem ($\hat{H}^\dagger(\lambda) = (H^*H + \lambda^2 I)^{-1}H^*$) and N is the number of unknowns (in this case, three times the number of pixels). Minimizing G is easy given the SVD of H. In fact, we know how to compute the SVD of H in this problem, but in other settings this is not so easy. So we will seek a different approach.

The troublesome term is the trace that appears in the denominator, but it turns out that we can estimate this trace using the fact that if z is any

random vector with independent entries of mean zero and variance one, then

$$\mathrm{tr}(I - H\hat{H}^{\dagger}(\lambda)) = \mathbb{E}\left[z^T(I - H\hat{H}^{\dagger}(\lambda))z\right].$$

The *Hutchinson* estimator uses random probe vectors consisting of ± 1 entries to estimate the trace of a large matrix, and Golub and von Matt suggested a procedure that approximates the GCV criterion with minimization of

$$\tilde{G}(\lambda) = \frac{Nm^2\rho(\lambda)^2}{\left(\sum_{l=1}^{m} z_l^T(I - H\hat{H}^{\dagger}(\lambda))z_l\right)^2},$$

where each vector z_l is a ± 1 vector drawn using a pseudo-random number generator.

Questions/tasks

1. Show that
$$\frac{d}{d\lambda}\left(I - H\hat{H}^{\dagger}(\lambda)\right) = 2\lambda(\hat{H}^{\dagger}(\lambda))^*\hat{H}^{\dagger}(\lambda).$$
From this, you will be able to compute derivatives of $\rho(\lambda)^2$ and of $z_\ell^T(I - H\hat{H}^{\dagger}(\lambda))z_\ell$.
Hint: The expression for differentiating a matrix inverse is in your notes!

2. For a given set of probe vectors stored in the columns of a matrix Z in MATLAB, write a routine to evaluate $\tilde{G}(\lambda)$ and the derivative $\tilde{G}'(\lambda)$. You should need to employ multiple factorizations per call:

```
function [G, dG] = p1gcv(obj, lambda, Z)
```

3. Using the p1gcv function as a building block, write an optimizer to minimize the $\tilde{G}$:

```
function [lambda] = p1gcvopt(obj, lmin, lmax, Z)
```

If the argument Z is omitted, form probe vectors with the following code fragment

```
[h,w,~] = size(obj.imblur);
m = 10;
Z = sign(rand(h,w,m)-0.5);
```

You should probably make use of the p1gcv code from the previous question!

4. What is the optimal λ by this criterion? Show the image for the computed regularization parameter.

5. On a log-log scale, plot $\tilde{G}(\lambda)$ against λ over a "reasonable" range of values around the recommended regularization parameter. Mark somehow the point on the curve associated with the desired value of λ. Is the curve "flat" in the neighborhood of the minimum, or is the minimum clearly defined?

3.3 The L-curve

The L-curve is a log-log plot of the residual norm (x-axis) against the solution norm (y-axis) for varying values of λ. It is named because it tends to look like the letter L. The vertical part of the L corresponds to the under-regularized case, where the residual norm changes negligibly with changes in λ, but the solution norm changes dramatically. The horizontal part corresponds to the over-regularized case, where small reductions to the solution norm correspond to increasingly large residuals. The corner of the L is the "sweet spot" where the two balance each other, and we seek to find this by finding λ where the curvature is maximal. You are given a function p1lcurve that computes a point on the L-curve (and the curvature) associated with a given λ; the function p1lcurve_plot sweeps out the curve, returning parallel lists of residual norms, solution norms, logarithmically-spaced values for λ, and curvatures $\kappa(\lambda)$.

Questions/tasks

1. Draw a (log-log) plot of the L-curve, and a semi-log plot showing the curvature as a function of the (log) residual norm. Is there a well-defined corner with a large curvature?

2. Write an optimizer to maximize the curvature:

```
function [lambda] = p1lcurveopt(obj, lmin, lmax)
```

   You may use p1lcurve to compute κ, and you may use fminbnd to do the optimization.

3. What is the optimal λ by this criterion? Show the image for the computed regularization parameter; is this a good result?

4 Notes

1. The type of very ill-conditioned problems seen here occur frequently not only in image reconstruction and inverse problems, but also in solving "first kind" integral equations that occur in mathematical physics. Integral equations of the second kind tend to be much better behaved.

2. We have sadly passed on a discussion of the use of iterative solvers for treating the large linear systems that arise during the computation of the Tikhonov solutions and during the computations involved in determining the regularization parameter.

3. The types of model selection criteria involved here are useful as well for a variety of problems in machine learning — though we are not always so blessed with tricks to make everything efficient!

4. As you might gather from our example, though the methods described in this project are mostly robust (hence popular) any regularization method can be fooled. There is no substitute for checking the answer to see if it looks sensible.