MLISP: Machine Learning in Signal Processing

Problem set 4

Prof. V. I. Morgenshtern

Problem 1: Reduction of convex optimization problems to standard form

Consider the least squares problem with linear constraints:

$$\begin{array}{ll} \min\limits_{\mathbf{x}} & \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2 \\ \text{subject to} & \mathbf{l} \preceq \mathbf{x} \preceq \mathbf{u} \end{array}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m, \mathbf{l}, \mathbf{u} \in \mathbb{R}^n$ and $\preceq$ denotes the component-wise inequality. Show how to express this problem as a quadratic programming problem:

$$\begin{array}{ll} \min\limits_{\mathbf{x}} & \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{P}\mathbf{x} + \mathbf{c}^\mathsf{T}\mathbf{x} + d \\ \text{subject to} & \mathbf{G}\mathbf{x} \preceq \mathbf{h} \\ & \mathbf{A}\mathbf{x} = \mathbf{b} \end{array}$$

where $\mathbf{P}$ is a symmetric positive semidefinite matrix.

cvxpy package for python and cvx package for Matlab perform similar reductions.

Problem 2: Chernoff bound

A basic question in probability, statistics, and machine learning is the following: given a random variable Z with expectation $\mathbb{E}[Z]$, how likely is Z to be close to its expectation? And more precisely, how close is it likely to be? In mathematical language we are interested in bounding the probabilities of large deviations $\mathbb{P}(Z \geq \mathbb{E}[Z] + t)$ and $\mathbb{P}(Z \leq \mathbb{E}[Z] - t)$ for $t \geq 0$.

The *indicator function* $I_v(X)$ for a real-valued random variable X is defined as

$$I_v(X) = \begin{cases} 1, & \text{if} \quad X \geq v \\ 0, & \text{if} \quad X < v. \end{cases}$$

1. Argue that for $s \geq 0$
$$e^{sX} \geq e^{sv} I_v(X).$$
To do so, consider the two cases $X \geq v$ and $X < v$ separately.
2. Show that
$$\mathbb{E}[e^{sX}] \geq e^{sv}\,\mathbb{P}[X \geq v],$$
and hence
$$\mathbb{P}[X \geq v] \leq \min_{s\geq 0}\left(e^{-sv}\,\mathbb{E}[e^{sX}]\right). \tag{1}$$
The inequality (1) is called the *Chernoff Bound.*

3. Now assume that X is Gaussian with zero mean and unit variance. The probability $\mathbb{P}[X \geq v]$ satisfies

$$\mathbb{P}[X \geq v] = \underbrace{\frac{1}{\sqrt{2\pi}} \int_{v}^{\infty} e^{-\frac{t^2}{2}} dt}_{Q(v)} \qquad v \geq 0$$

Use (1) to show that $Q(v) \leq e^{-v^2/2}$. This is a standard bound on the Q-function, which we will need in class.

Hint: Write out the expectation $\mathbb{E}[e^{sX}]$, and use the fact that a PDF always integrates to 1.

4. Let Q be a χ^2 distributed random variable with k degrees of freedom. Use the same idea as above to obtain the following tail bound:

$$\mathbb{P}[k - Q > t] \leq \left(1 + \frac{t}{k-t}\right)^{-k/2} e^{t/2}$$

for all $t > 0$.

Set $k = 4$. Plot the CDF of the χ^2 distributed random variable with k degrees of freedom. The bound on the CDF you have derived above and the bound derived in [B. Laurent and P. Massart, 'Adaptive estimation of a quadratic functional by model selection', 2000, p. 1325]:

$$\mathbb{P}[k - Q > t] \leq \left[-\frac{t^2}{4k}\right].$$

Compare the results.

Problem 3: Cost function of logistic regression is convex

Recall that to fit logistic regression we minimize the negative log likelihood

$$l(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \left[y^{(i)} \log h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}))\right]$$

where

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}^\mathsf{T}\boldsymbol{\theta}}}. \tag{2}$$

and the labels $y^{(i)} \in \{0, 1\}$. Show that $l(\boldsymbol{\theta})$ is a convex function of $\boldsymbol{\theta}$.

Hints:

- Show that $-\sum_{i=1}^{n} y^{(i)} \log h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$ and $-\sum_{i=1}^{n}(1 - y^{(i)}) \log(1 - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}))$ are both convex.
- Calculate the Hessian matrices and show that these matrices are positive semidefinite.

Problem 4: Introduction to cvxpy: inpainting

The goal of this exercise is to learn how to use cvxpy for a simple image processing problem: inpainting.

- Install cvxpy
- Read the image of Lena, `lena512.png` and `lena512_corrupted.png`. Display both.

The goal is to reconstruct the original image based on the corrupted version and the locations of corruptions.

A grayscale image, `lena512.png`, is represented as an $m \times n$ matrix of intensities $\mathbf{U}^{\text{orig}}$ (typically between the values 0 and 255). We are given the values U_{ij}^{orig}, for $(i,j) \in \mathcal{K}$, where $\mathcal{K} \subset \{1,\dots,m\} \times \{1,\dots,n\}$ is the set of indices corresponding to known pixel values. Our job is to in-paint the image by guessing the missing pixel values, i.e., those with indices not in $\mathcal{K}$. The reconstructed image will be represented by $\mathbf{U} \in \mathbb{R}^{m,n}$, where U matches the known pixels, i.e., $U_{ij} = U_{ij}^{\text{orig}}$ for $(i,j) \in \mathcal{K}$.

The reconstruction $\mathbf{U}$ is found by minimizing the total variation of $\mathbf{U}$, subject to matching the known pixel values. We will use the l2 total variation, defined as

$$\text{tv}(\mathbf{U}) = \sum_{i=1}^{m-1} \sum_{i=1}^{n-1} \left\| \begin{bmatrix} U_{i+1,j} - U_{ij} \\ U_{i,j+1} - U_{ij} \end{bmatrix} \right\|_2$$

- Compute the set $\mathcal{K}$ by comparing the two images and determining where `lena512` is different from `lena512_corrupted`
- In cvxpy, implement the following convex optimization problem:

$$\begin{aligned} &\min_{\mathbf{U}} \text{tv}(\mathbf{U}) \\ &\text{subject to} \quad U_{ij} = U_{ij}^{\text{orig}} \text{ for } (i,j) \in \mathcal{K} \end{aligned}$$

  Hint: you can use `tv` function of cvxpy
- Solve the problem using the `SCS` solver.
- Display the reconstructed image. How accurate is your reconstruction?
- Why minimizing the tv norm helped you reconstruct the image?
- Substitute tv($\mathbf{U}$) with the Frobenius norm of $\mathbf{U}$. Solve the problem and display your result. Was the reconstruction accurate now?
- Discuss your results and observations.

Problem 5: Wavelet denoising

In class we talked about denoising via wavelet shrinkage at the level $\sigma\sqrt{2\log(n)}$, where σ^2 is the variance of the noise. In this exercise you are provided with four signal that are stored as columns of the text file `clean_signal.txt`. The noisy versions of the same signals are stored as columns of the text file `noisy_signal.txt`, $\sigma = 1$. You can use the `read_data.ipynb` to read the data.

1. Use `pywt` package in Python to compute the wavelet transforms of the noisy and noiseless version signals. Visualize the wavelet coefficients at all levels.

   Hint: `pywt.wavedec` is the function you can use. You are also free to choose the types and levels of the wavelet for this problem.

2. Use the soft thresholding formula derived in class to denoise the signals. Take the inverse wavelet transfrom and visualize the results. Report the l2 norm of the difference between the clean signal and the reconstructed signal. Compare it to the l2 norm of the difference between the clean signal and the noisy signal. By how much were you able to denoise?

3. Form the wavelet transform matrix corresponding to the wavelet transform that you used above. This is a 1000×1000 matrix in this case.

   Hint: To form this matrix, you can generate all the standard basis vectors (1 sparse vectors with exactly one nonzero element) one-by-one and apply `pywt.wavedec` to each of these vectors. This recovers the corresponding wavelet transform matrix column-by-column.

4. Use cvxpy to implement wavelet shrinkage in the form of l1 minimization problem.

5. Report the denoising results in the same way as above. If everything is done correctly, your results should be the same as above.

Problem 6: Basic compressive sensing experiment

1. Choose n sufficiently large, but small enough such that your calculations run fast. For example, $n = 500$ might be a reasonable choice. Let $m \leq n$ be the number of measurements and $\delta = m/n$ be the undersampling ration. Set $\delta = 1/10$ and choose m accordingly. Generate the $m \times n$ measurement matrix $\mathbf{A}$ with i.i.d. entries distributed as $A_{ij} \sim \mathcal{N}(0, 1/m)$. Let s be the sparsity of the signal and $\rho = s/m < 1$ be the ratio between the number of measurements and the sparsity. Set $\rho = 1/5$ and choose s accordingly. Generate the signal $\mathbf{x}_0 \in \mathbb{R}^n$ randomly with s nonzero components. You can choose the locations and the amplitudes of elements of $\mathbf{x}_0$ by some kind of random mechanism, the exact details of this mechanism are not important. Generate the data vector $\mathbf{b} = \mathbf{A}\mathbf{x}_0$.

2. Use cvxpy to solve

$$\begin{aligned} \min_{\mathbf{x}} \quad & \|\mathbf{x}\|_1 \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned}$$

   Denote the solution $\hat{\mathbf{x}}$.

3. Is $\hat{\mathbf{x}} = \mathbf{x}_0$? To test this, calculate $\|\hat{\mathbf{x}} - \mathbf{x}_0\|/\|\mathbf{x}_0\|$. This number should be very small. If $\hat{\mathbf{x}} \neq \mathbf{x}_0$, reduce s and repeat the experiment.

4. If $\hat{\mathbf{x}} = \mathbf{x}_0$, regenerate $\mathbf{x}_0$ while keeping s and $\mathbf{A}$ fixed. Do this many enough times to convince yourself that the recovery is successful ($\hat{\mathbf{x}} = \mathbf{x}_0$) every time.

5. Next, change the matrix $\mathbf{A}$ to be the subsampled discrete Fourier transform (DFT) matrix. Specifically, let $\mathbf{F}$ be the $n \times n$ DFT matrix. Then select m rows of $\mathbf{F}$ randomly and form $\mathbf{A}$ out of the selected set of row. Repeat the exercise for the new $\mathbf{A}$.

Problem 7: Phase transitions in compressed sensing

In this problem we use notations from the previous problem. We have seen in class that the signal sparsity s has to be smaller than the number of measurements m to guarantee succesful reconstruction: $\hat{\mathbf{x}} = \mathbf{x}_0$. It turns out that in compressive sensing there is a very sharp phase transition phenomenon. Concretely, for fixed m, one can determine a critical value $s_{\text{crit}}(m)$. If s is slightly smaller than $s_{\text{crit}}(m)$, compressive sensing reconstruction will be successful for nearly every choice of $\mathbf{A}$ and $\mathbf{x}$. If s is slightly larger than $s_{\text{crit}}(m)$, compressive sensing reconstruction will be unsuccessful for nearly every choice of $\mathbf{A}$ and $\mathbf{x}$. In this exercise you are asked to explore the phase transition phenomenon.

To do this work with Gaussian matrices from the previous exercise. Select n that is large, but small enough so that you can complete the experiment in reasonable time. Consider the δ, ρ plane. Consider a 10×10 regular grid in this plane. A finer grid would be better, if your compute time allows you to complete the experiment. For each point on the grid repeat the process described in the previous exercise at least 10 times. Randomly generate the Gaussian measurement, the sparse signal $\mathbf{x}_0$, the measurement vector $\mathbf{b}$, use cvxpy to obtain $\hat{\mathbf{x}}$. Each time record if the reconstruction has been successful. Calculate the fraction of succesfull reconstructions out of 10 attempts.

At this point you have solved 1000 convex problems: 10 problems for each of 10×10 grid points. Display your results a 10×10 matrix, where each element contains the success fraction of compressed sensing reconstruction for that specific grid point. You should find that the 10×10 grid separates into 2 distinct contiguous regions. In one region (corresponding to small s), compressed sensing reconstruction is always successful, in the other region (corresponding to large s) compressed sensing reconstruction is always unsuccessful. Display your results on a plot.

For more details about the phase transition phenomenon and on how to set up similar experiments on the large scale, see 'Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing' by D. Donoho and J. Tanner and 'Deterministic matrices matching the compressed sensing phase transitions of Gaussian random matrices' by H. Monajemi, S. Jafarpour, M. Gavish, Stat 330/CME 362 Collaboration, and D. Donoho.

Problem 8: Solving underdetermined linear systems of equations (exam practice)

Let $\mathbf{A} = [2\ 1]$ and $\mathbf{x} = [x_1\ x_2]^{\mathsf{T}}$. Consider the underdetermined linear equation $\mathbf{A}\mathbf{x} = 6$.

1. Find all solutions of this equation and draw them.
2. Solve the $l1$-minimization problem:

$$\begin{aligned}&\min_{\mathbf{x}} \|\mathbf{x}\|_1\\ &\text{subject to } \mathbf{A}\mathbf{x} = 6\end{aligned}$$

   Find the exact solution of this problem. Justify that this is indeed the solution.

   Hint: Draw your steps.

3. How sparse is your solution in the previous task? Justify your answer.

Problem 9: Convex optimization problems (exam practice)

1. Is the function $f(x, y) = \max(y - 42, x/2)$ convex, concave, affine or neither of these? Explain your answer rigorously.

Hint: You are allowed to use the rules of disciplined convex programming.

2. Is the following optimization problem convex?

$$\begin{aligned} &\min_{x_1, x_2} x_1^2 + 6x_2^2 \\ &\text{subject to } x_1 + 12x_2 = 18 \end{aligned}$$

   Explain why.

3. Is the following optimization problem convex?

$$\begin{aligned} &\min_{x_1, x_2} x_1 + 12x_2 \\ &\text{subject to } x_1^2 + 6x_2^2 = 18 \end{aligned}$$

   Explain why.

4. Is the following optimization problem convex?

$$\begin{aligned} &\min_{x_1, x_2} x_1 + 12x_2 \\ &\text{subject to } x_1^2 + 6x_2^2 \leq 18 \end{aligned}$$

   Explain why.