

MLISP: Machine Learning in Signal Processing

Problem set 3

Prof. V. I. Morgenshtern

Problem 1: Expected maximum of iid Gaussians random variables

Let z_1, \dots, z_N be independent identically distributed (i.i.d.) random variables with $z_i \sim \mathcal{N}(0, \sigma^2)$. Let $y = \max_i z_i$. Prove that $\mathbb{E}[y] \leq \sigma \sqrt{2 \log N}$.

Hints:

- Calculate the moment generating function of z_i , and prove that $\mathbb{E}[\exp(tz_i)] = \exp(t^2\sigma^2/2)$.
- Use the Jensen's inequality and the union bound to show $\exp(t\mathbb{E}[y]) \leq \sum_{i=1}^N \mathbb{E}[\exp(tz_i)]$.
- Take the log in the bound derived above, optimize the bound over the choice of t to finish the proof.

Problem 2: Phoneme classification [Ref: Elements of statistical learning]

Write a program to classify the phoneme data using logic regression on raw features and logistic regression with splines. Your goal is to reproduce results in Lecture 7, Section 3. The phone data is in `phoneme_data.txt` and the description of the data is in `phoneme_info.txt` file.

- Extract the data corresponding to 'aa' and 'ao' phonemes.
- Plot the data.
- Split the data into the training and the test set.
- Fit the logistic regression to the raw data. In this case your output variables are binary, corresponding to 'aa' or 'ao' phonemes, your input variable are the $p = 256$ dimensional vectors with frequency information \mathbf{x} . Report performance on the train and the test set. Plot the coefficients θ .
- Since there are many correlated features, you should filter them using a smooth basis of cubic splines.
- Decide beforehand on a series of five different choices for the number and position of the knots. Form the $p \times D$ spline matrix \mathbf{H} , where $D \ll p$ is the number of degrees of freedom in the model that is determined by the number and positions of the knots. Your new features are $\mathbf{x}^* = \mathbf{H}^T \mathbf{x}$.
- Fit the spline-regularized logistic regression to each choice.
- Use the test set to make the final selection of the positions of the knots.

- Report performance on the train and the test set. Compare to the unregularized case. Plot the coefficients θ .

Hint: You don't need to write a gradient descent algorithm to fit the model in this problem. Use `cvxpy` instead.

Problem 3: Haar wavelets

1. Show that the Haar wavelet transform of a signal of length $N = 2^J$ can be computed in $O(N)$ computations. Implement the corresponding algorithm.
2. Use your algorithm to compute the Haar wavelet transform of the following functions on the interval $[-0.5, 0.5]$. (The functions should be sampled to have 2^J sampling points).

-

$$f_1(t) = \begin{cases} 1, & t < 1/3 \\ 0, & t \geq 1/3 \end{cases}$$

- $f_2(t) = \sin(t)$.

3. Display the wavelet coefficients for both functions.
4. Make histograms of wavelet coefficient sizes for the two cases. Compare and discuss the approximate sparsity in the wavelet coefficients of the two functions.
5. Set 50 percent of the smallest wavelet coefficients for both functions to zero. Take the inverse wavelet transform and display the resulting functions.

Problem 4: Regularization and bias variance trade-off [Ref: Stanford CS229 class]

In this exercise, your goal is to implement regularized linear regression to predict the amount of water flowing out of a dam using the change of water level in a reservoir. In the file `cross_validation.py` you are given the dataset containing historical records on the change in the water level in a reservoir, x , and the amount of water flowing out of the dam, y . This dataset is divided into three parts:

- A training set that your model will learn on: `X`, `y`
- A cross validation set for determining the regularization parameter: `Xval`, `yval`.
- A test set for evaluating performance. These are “unseen” examples which your model did not see during training: `Xtest`, `ytest`

1. Visualize the training data on a scatter plot.
2. Recall that regularized linear regression has the following cost function:

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2n} \sum_{j=1}^p \theta_j^2$$

where n is the number of training examples, p is the number of features, λ is the regularization parameter. Note that you should not regularize the θ_0 term. We start by considering only the single original feature, the water level, and, therefore, $h_{\theta}(x) = \theta_0 + \theta_1 x$. Calculate the gradient of $J(\theta)$ with respect to θ .

3. Implement the gradient descent algorithm to fit the model, test that it converges for $\lambda = 0$. Plot the resulting linear fit on top of the training data scatter plot.

Hint: If you want to make the problem simpler, you can use `cvxpy` to fit the model, instead of gradient descent.

4. Next, your goal is to plot the learning curve. Set $\lambda = 0$. The training set contains 12 data points. Fit the model by using only the first two data points ($n = 2$). Recall that the error for a dataset is defined as:

$$J_{\text{error}}(\theta) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2$$

Calculate J_{error} of the fitted model on this two-data-points-large training set (this is the training error). Record the result. Calculate the J_{error} of the fitted model on the whole validation set. Record the result. Next repeat the processes by using the first three data points for training ($n = 3$), then the first four data points ($n = 4$) and so on. Plot the training error, as a function of the number of data points used. This is the learning curve. Plot J_{train} on the validation set (this is the validation error) as the function of the number of data points used. If you did everything correctly, you should observe the following. The training error increases as the number of training examples grows. The validation error decreases as the number of training examples grows. The validation error is higher than the training error.

5. You can observe that the validation error is always high and the training error becomes high when the number of training examples is increased. This reflects a high bias problem in the model – the linear regression model is too simple and is unable to fit our dataset well. In this part of the exercise, you will address this problem by adding more features, using the polynomial regression. Our hypothesis has the form:

$$h_{\theta}(x) = \theta_0 + \theta_1 \cdot (\text{water level}) + \theta_2 \cdot (\text{water level})^2 + \dots + \theta_p \cdot (\text{water level})^p.$$

For this part of the exercise, you will be using a polynomial of degree $p = 8$. Proceed as before to fit this model without regularization. Plot the learning and validation curves. Also plot the fitted polynomial curve on top of the training data scatter plot.

Important tip: To make the problem numerically well conditioned, it is crucial to normalize the features before fitting the polynomial model above. This is because, for example with $x = 40$ will now have a feature $x_8 = 40^8 = 6.5 \cdot 10^{12}$ which will make numerical algorithms unstable. Rescale the features by subtracting the mean and dividing by the standard deviation for each feature separately. Don't forget to account for this rescaling when working with the validation set and the test set.

When you plot the learning curve, you should observe that the training error is extremely small. The validation error is large. This means that the model has very high bias and essentially fits the noise.

6. To avoid this problem, you can now use regularization. Fit the regularized regression for $\lambda = \{0, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 10\}$. Plot the learning and the validation curves. Use the validation set to select the best value for λ , i.e. the value that makes the validation error (when the model is fitted on the whole dataset with $n = 12$) as small as possible. Plot the resulting polynomial fit on top of the training data scatter plot for the optimal λ and for $\lambda = 100$. Discuss your results.
7. Run prediction with the best possible λ on the test set. Calculate error J_{error} on the test set. Make sure that the error on the test set is not too much larger than the error on the validation set. This means that your model is well regularized and you can have confidence that it will perform well on future data.

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 signals 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 transform 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: Debugging machine learning algorithms (exam practice)

Our goal is to classify melons into ‘good’ melons (+1) and ‘bad’ melons (-1). Melons can be distinguished based only on their color and smell. Let \mathcal{H} be the set of all circles in \mathbb{R}^2 . Each circle $h \in \mathcal{H}$ defines the classification rule: all melons whose (smell, color) is inside the circle are classified as ‘good’, all melons whose (smell, color) is outside the circle are classified as ‘bad’. Each circle h

is defined by (r, w_1, w_2) , the radius and the coordinates of the center. The labeled training sample of melons is

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^n, \quad \mathbf{x}_i \in \mathbb{R}^2, y_i \in \{1, -1\}$$

where $\mathbf{x}_i = [x_{i1}, x_{i2}]^T$ are the features (smell, color) and y_i are the labels (+1) for ‘good’, (-1) for ‘bad’. We used the following criterion to determine the optimal circle $h^* \in \mathcal{H}$ for prediction:

$$(r^*, w_1^*, w_2^*) = \arg \min_{r, w_1, w_2} \sum_{i=1}^n \exp(-y_i [r^2 - ((x_{i1} - w_1)^2 + (x_{i2} - w_2)^2)]).$$

Unfortunately h^* we found did not meet the expectations, as it misclassified a non-negligible proportion of the melons used at test time. Your goal is to analyze the problem according to the additional instructions below.

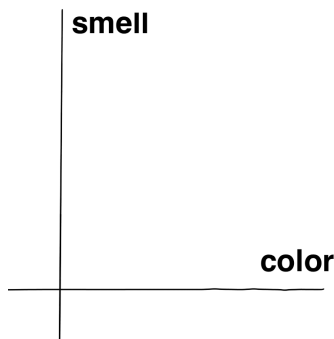
1. Scenario 1: h^* had a very low training error.

- Give a possible explanation for the large test error of h^* .
- Draw a training set, the prediction function h^* , and the true distribution (if needed) that demonstrates your explanation.
- Propose a solution to mitigate this problem that is consistent with your drawing.

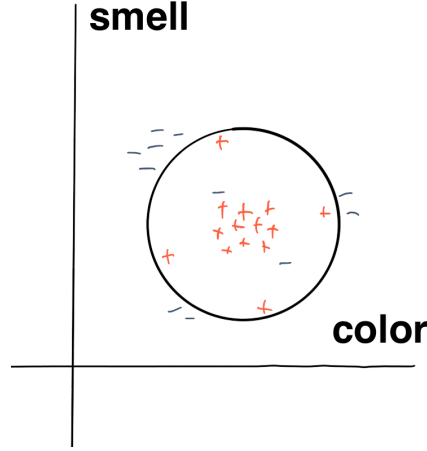


2. Scenario 2: The sample size was large, and h^* had training error of 0.4.

- Give a possible explanation for the large test error of h^* .
- Draw a training set, the prediction function h^* , and the true distribution (if needed) that demonstrates your explanation.
- Propose a solution to mitigate this problem that is consistent with your drawing.



3. Suppose that with the current training set we obtain the following solution:



In the figure, ‘+’ stands for good melons and ‘-’ stands for bad melons. Assume that it is OK to mistakenly classify some of the good melons as bad ones, but it is absolutely not allowed to classify the bad melons as good ones. We decide to modify our training procedure to address this requirement. We use regularization to do that:

$$(r^*, w_1^*, w_2^*) = \arg \min_{r, w_1, w_2} \sum_{i=1}^n \exp(-y_i[r^2 - ((x_{i1} - w_1)^2 + (x_{i2} - w_2)^2)]) + \lambda \Omega(r, w_1, w_2)$$

where $\Omega(r, w_1, w_2)$ is the regularizer.

- Suggest a suitable regularizer.
- Write down its mathematical form.
- Draw a regularized solution.
- What is the effect of increasing λ on the solution?

Problem 7: Haar wavelets (exam practice)

Consider the function $f : [0, 1] \rightarrow \mathbb{R}$ defined as

$$f(x) = \begin{cases} 2, & x \in [0, \frac{1}{2}) \\ 4, & x \in [\frac{1}{2}, 1] \end{cases}$$

Your goal is to find the approximation to $f(x)$ in the subspace $V_0 \oplus W_0 \oplus W_1$. Please restrict all computations to the interval $[0, 1]$ and follow the step-by-step instructions.

1.
 - What are the dimensions of the spaces V_0, W_0, W_1 ?
 - Plot the basis functions of V_0, W_0, W_1 .
2. Express the projection of the function f onto the subspace $V_0 \oplus W_0 \oplus W_1$ in terms of the wavelet coefficients and the basis functions. Justify in details why this decomposition is true.

Hint: You do not need to compute the coefficients explicitly here. This is done in the next subtask.

3. Compute the Haar wavelet coefficients, representing the projection of f onto W_1 .
4. Give an example of a function for which Haar wavelet coefficients corresponding to W_1 are zero.