Machine Learning for Computer Vision, MVA 2015-2016

Instructor: Iasonas Kokkinos, iasonas.kokkinos[at]ecp.fr

Programming Assignment 1 (3 Points/20) Deliverable: 3/4/16

1 Linear versus logistic regression (.7/3)

The file `week_1_a.m` contains code to compare the robustness of linear and logistic regression. Linear regression has already been implemented for you. Please go through the relevant part of the code, and the relevant comments, before advancing to the remainder.

As discussed in class, the merit function (or, criterion, C) driving logistic regression equals:

$$C(\mathbf{w}) = \sum_{i=1}^{N} y^i \log(g(< \mathbf{x}^i, \mathbf{w} >) + (1 - y^i) \log(1 - g(< \mathbf{x}^i, \mathbf{w} >)). \quad (1)$$

Maximize this criterion for the provided dataset. Making reference to the slides of Lecture 2, this requires computing first and second order differentials, corresponding to $J(w)$ and $H(w)$, respectively.

1 (.2/.7) Write the code to compute $J(\mathbf{w})$ and $H(\mathbf{w})$.

2 (.2/.7) Use these to implement the Newton-Raphson algorithm. Consider that convergence is achieved when $|\mathbf{w}^t - \mathbf{w}^{t-1}|_2 < .001|\mathbf{w}^t|_2$

3 (.2/.7) Plot the merit function, $C(\mathbf{w})$, of Eq. 1 as a function of Newton-Raphson iteration.

4 (.1/.7) Compare the robustness of the fitted boundaries for the two experiments.

If your code is working right, you should be getting the results shown below, when implementing steps [3] and [4] respectively.

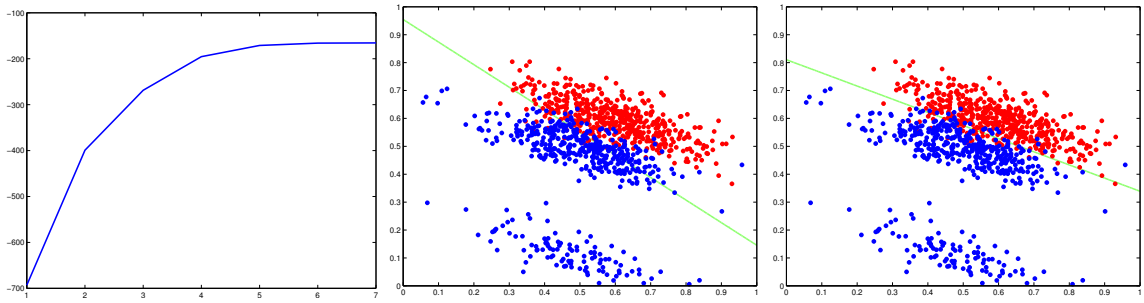

Figure 1: Left: $C(w)$ as a function Newton-Raphson iteration; middle: decision boundary for linear regression; right: decision boundary for logistic regression

2 Logistic Regression and Regularization (2.3/3)

In the previous Section you were asked to maximize the logistic regression objective. Now we turn to regularized logistic regression. As a first step,

(.5/2.3) Find how the $J(\mathbf{w})$ and $H(\mathbf{w})$ would change if one adds an l_2 regularization term:[1]

$$C(\mathbf{w}) \quad = \quad \sum_{i=1}^{N} y^i \log(g(<\mathbf{x}^i, \mathbf{w}>) + (1-y^i)\log(1-g(<\mathbf{x}^i, \mathbf{w}>)) - \lambda\|\mathbf{w}\|_2^2, \quad \text{where} \|\mathbf{w}\|_2^2 = \sum_j \mathbf{w}_j^2$$

Adapt the code you wrote for the previous Section so that it maximizes this merit for a given λ. Verify that your new code properly optimizes the objective above.

This brings us to the task of setting λ by using cross-validation. The dataset we used in the previous Section is linearly separable in two dimensions, and as such too simple to require regularization. We therefore turn to a harder dataset, as in the file `week_1_b.m`

As you can see the data there is no longer linearly separable. We therefore use a nonlinear embedding with harmonic functions. The code for constructing the embedding is provided to you as `embedding.m`. You need to implement cross-validation to pick the right value of λ. A big part of the code is prepared for you, you need to fill in the gaps, as outlined below.

K-fold cross-validation procedure (1.8/2.3)

For every value of λ:

- For every round $k \in \{1, \ldots, K\}$:
  - a Split the training data into a parameter estimation set, $\mathcal{S}_k$, and a validation set, $\mathcal{V}_k$.
  - b Find $\mathbf{w}_k^*$ by maximizing the logistic regression objective on $\mathcal{S}_k$:

$$\mathbf{w}_k^* = \text{argmax}_{\mathbf{w}} \sum_{i \in \mathcal{S}_k} y^i \log(g(<\mathbf{x}^i, \mathbf{w}>) + (1-y^i)\log(1-g(<\mathbf{x}^i, \mathbf{w}>)) - \lambda\|\mathbf{w}\|_2^2 \tag{2}$$

  - c Estimate the associated zero-one loss on the validation set $\mathcal{V}_k$:

$$E_{k,\lambda} = \sum_{i \in \mathcal{V}_k} [y^i \neq \text{sign}(\langle \mathbf{w}_k^*, \mathbf{x}^i \rangle)] \tag{3}$$

- Find the cross-validation for λ by averaging the loss over these K rounds:

$$L_\lambda = \frac{1}{K} \sum_{k=1}^{K} E_{k,\lambda} \tag{4}$$

Identify the value of $\lambda^* = \arg\min_\lambda L_\lambda$ that gives the lowest cross-validation loss.
Finally, obtain $\mathbf{w}^*$ as the solution of

$$\mathbf{w}^* = \text{argmax}_{\mathbf{w}} \sum_{i=1}^{N} y^i \log(g(<\mathbf{x}^i, \mathbf{w}>) + (1-y^i)\log(1-g(<\mathbf{x}^i, \mathbf{w}>)) - \lambda^*\|\mathbf{w}\|_2^2, \tag{5}$$

namely use the optimal λ^* and the whole training set.

Include as part of your report the following three items:

[1] Note: $C(\mathbf{w})$ is interpreted as a 'merit function' and as such is maximized. Therefore the regularization term is *subtracted*, favoring low-norm weight vectors.

a The function $L_{o,\lambda}$

b The decision boundary obtained when training with λ^*.

c The number of errors on the test set.

If your code is working right, the answers to [a] and [b] should be similar to the plots provided below:

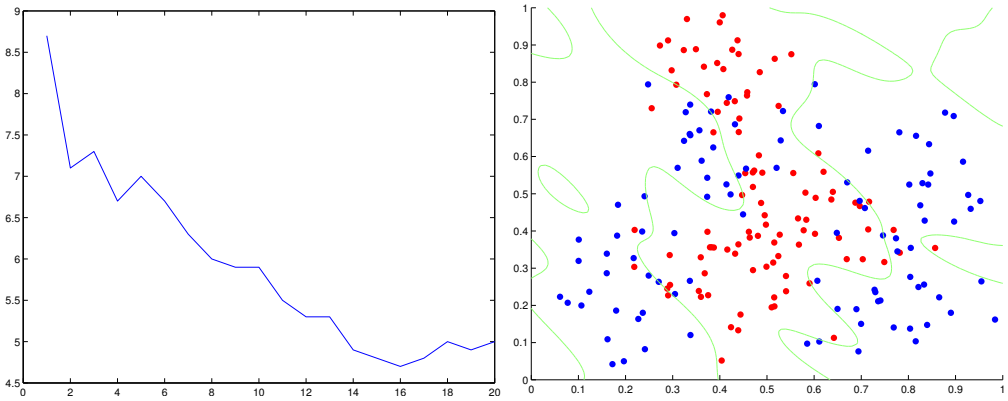

Figure 2: Cross-validation error as a function of λ and decision boundary for λ^*

Appendix

Visualizing 2D functions

This text accompanies the function `visualize.m`, which illustrates matlab's visualization abilities.

To visualize a 2D function in a domain $\mathbf{x} = (h, v) \in\in [-1, 1] \times [-1, 1]$, where h, v are the horizontal and vertical coordinates, respectively, we will sample its value on a regular grid with a step of .01 units, which gives us a 1001×1001 matrix with values:

$$F[i, j] = f((i/100, j/100), w) \tag{6}$$

The function `visualize.m` involves code to

1 Compute this matrix.

2 Visualize it using the functions `surf.m` and `imagesc.m`.

3 Plotting the zero set of f (or, decision boundary) using `contour.m`.

Matlab figure hints

- Once a figure is generated you can save it into a jpeg,eps,png,... file using print. Type `>> help print` for details. A quick example:

```
>> input = [0:.01:1]; plot(input,sin(input));
>> title('Sinusoidal'); xlabel('input'); ylabel('output');
>> print -depsc sinusoid
```

- You can superimpose plots using `hold on`.
- For your convenience you can merge multiple plots in a single figure using `subplot`. Type `help subplot` for details.