Lab: RBF networks and polynomials

Miguel Á. Carreira-Perpiñán

The objective of this lab is for you to explore the behavior of radial basis function (RBF) networks and polynomials, both for nonlinear regression, and apply them to some datasets. The TA will first demonstrate the results of the algorithms on several datasets, and then you will replicate those results and further explore the datasets with the algorithms.

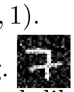
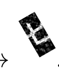
We provide you with the following:

- The scripts `lab09_rbf.m`, `lab09_rbf_MNIST.m` and `lab09_poly.m` set up the problem (toy dataset or MNIST for RBFs, toy dataset for polynomials) and plot various figures.
- The functions `rbftrain.m` and `polytrain.m` train a nonlinear regressor (from D dimensions to D' dimensions) using a RBF network or a polynomial, respectively (for the polynomial we use $D = D' = 1$). See also the functions `sqdist.m`, `rbf.m` and `polyf.m`.

I Datasets

Firstly, construct your own toy datasets to visualize the result easily and understand the algorithms. Take the input instances $\{\mathbf{x}_n\}_{n=1}^N$ in $\mathbb{R}$ or $\mathbb{R}^2$ and the labels $\{y_n\}_{n=1}^N$ in $\mathbb{R}$. Generate a noisy sample from a known function, e.g. $y_n = f(x_n) + \epsilon_n$ where $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$ and $f(x) = ax + b$ or $f(x) = \sin x$.

Then, try the MNIST dataset of handwritten digits, with instances $\mathbf{x} \in \mathbb{R}^D$ (where $D = 784$). Create a ground-truth mapping $\mathbf{y} = \mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$ as follows (as in the previous lab about gradient descent and linear models):

- A random mapping with output dimension D', e.g. take $\mathbf{A}_{D' \times D}$ and $\mathbf{b}_{D' \times 1}$ with elements in $\mathcal{N}(0, 1)$.
- A mapping that rotates, scales, shifts and possibly clips the input image $\mathbf{x}$ and adds noise to it, e.g.  $\rightarrow$ . This makes it easy to visualize the result, since the desired output $\mathbf{f}(\mathbf{x})$ for an image $\mathbf{x}$ should look like $\mathbf{x}$ but transformed accordingly.

II RBF networks for nonlinear regression

Review We minimize the least-squares error

$$E\left(\{\mathbf{w}_h, \boldsymbol{\mu}_h\}_{h=1}^H, \sigma\right) = \frac{1}{2} \sum_{n=1}^N \left\|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n)\right\|^2 + \lambda \sum_{h=1}^H \left\|\mathbf{w}_h\right\|^2 \tag{1}$$

where $\lambda \geq 0$ is a regularization user parameter which controls the smoothness of $\mathbf{f}$, and $\mathbf{f}$ is an RBF network:

$$\mathbf{f}(\mathbf{x}) = \sum_{h=1}^H \mathbf{w}_h \phi_h(\mathbf{x}) \qquad \phi_h(\mathbf{x}) = \exp\left(-\tfrac{1}{2}\left\|\mathbf{x} - \boldsymbol{\mu}_h\right\|^2 / \sigma^2\right) \qquad \mathbf{x} \in \mathbb{R}^D,\ \mathbf{f}(\mathbf{x}) \in \mathbb{R}^{D'} \tag{2}$$

where the radial basis functions $\{\phi_h(\cdot)\}_{h=1}^H$ are (proportional to) Gaussians with centroids $\{\boldsymbol{\mu}_h\}_{h=1}^H \subset \mathbb{R}^D$ and common width σ, and $\{\mathbf{w}_h\}_{h=1}^H \subset \mathbb{R}^{D'}$ are weights. We take $\phi_1(\mathbf{x}) \equiv 1$ if we want to use a *bias*.

We will train RBF networks in an approximate but simple and fast way (requiring no iterative optimization or initial weight values) as follows:

1. Set the centroids $\{\boldsymbol{\mu}_h\}_{h=1}^H$ in an unsupervised way using only the input points $\{\mathbf{x}_n\}_{n=1}^N$, by simply selecting H points at random. If you want, you can further refine this by running k-means using those points as initial centroids.
2. Set the width σ by hand to some reasonable value (we will cross-validate it, see below).

3. Given the centroids and width, the values $\phi_h(\mathbf{x}_n)$ are fixed, and the weights $\{\mathbf{w}_h\}_{h=1}^H$ are determined by optimizing E, which reduces to a simple linear regression. We solve the linear system[1]:

$$\left(\mathbf{\Phi}\mathbf{\Phi}^T + \lambda\mathbf{I}\right)\mathbf{W} = \mathbf{\Phi}\mathbf{Y}^T \Leftrightarrow \mathbf{W} = \left(\mathbf{\Phi}\mathbf{\Phi}^T + \lambda\mathbf{I}\right)^{-1}\mathbf{\Phi}\mathbf{Y}^T \tag{3}$$

where $\mathbf{\Phi}_{H\times N} = (\phi_h(\mathbf{x}_n))_{hn}$, $\mathbf{W}_{H\times D'} = (\mathbf{w}_1,\dots,\mathbf{w}_H)^T$ and $\mathbf{Y}_{D'\times N} = (\mathbf{y}_1,\dots,\mathbf{y}_N)$. The linear system solution can be done in Matlab with `linsolve` or the "\" operator (you can also use `inv` but this is numerically more costly and less stable).

There are 3 hyperparameters for the user to set: the number of basis functions H, the width σ, and the regularization parameter λ. We set them by cross-validation using a grid search. For example, we can use $H \in \{3, 5, 10, 50\}$, $\sigma \in \{2^{-2}, 2^0, 2^2, 2^4\}$ and $\lambda \in \{0, 10^{-5}, 10^{-3}, 10^{-1}\}$ (the actual values will depend on your problem, particularly for σ). We train an RBF network (on the training set) for each combination of values of (H, σ, λ) and pick the one with lowest error on a validation set.

Exploration: toy problem We visualize the results with the following plots:

- The dataset (y_n vs x_n) and the RBF network $f(x)$.
- The training and validation error of several RBF networks trained using different hyperparameter values (H, σ, λ).

Questions to consider:

- How does the RBF network look like if we vary one of the hyperparameters keeping the rest fixed, that is:
  - if you increase H?
  - if you increase σ?
  - if you increase λ?

  And, how does this affect the training and the validation error? For example, what value of H (or λ, or σ) gives the lowest training error? How about the lowest validation error?
- What is the value of the RBF network $f(x)$ for a point x that is far from any training point $x_1, \dots, x_N$?
- For a given point x, how many hidden units (i.e., BFs) have nonnegligible value $\phi_h(x)$?
- What gives lower error: using a random subset of points as centroids for the RBFs, or running k-means?

Exploration: MNIST Select a small enough subset of MNIST as training inputs (using the whole dataset will be slow). Apply the ground-truth linear transformation to the data to generate the output labels $\mathbf{y} = \mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$. Then proceed as with the toy dataset to compute the optimal solution exactly by picking a value for (H, σ, λ), picking H training images as basis function centroids, and solving the linear system to get the weights. Then, consider the same questions as with the toy dataset. Note that the appropriate values for σ, etc. may now be significantly different. We visualize the results with the following plots:

- The training and validation error of several RBF networks trained using different hyperparameter values (H, σ, λ).
- For each RBF network:
  - We plot each centroid $\boldsymbol{\mu}_h$, $h = 1, \dots, H$, as a grayscale image.
  - If using as linear mapping the rotation/shift/scale/clip transformation, which produces as output a (possibly smaller) image $\mathbf{y}_n$, we plot the following for a few sample images: the input image $\mathbf{x}_n$, and the output $\mathbf{y}_n$. Compare with the result produced by the true linear mapping.

---

[1] Compare it with the case of linear regression (in a previous lab):

$$\text{Objective function: } E(\mathbf{W}) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{W}\mathbf{x}_n\|^2 \qquad \text{Normal equations: } (\mathbf{X}\mathbf{X}^T)\mathbf{W} = \mathbf{X}\mathbf{Y}^T \Leftrightarrow \mathbf{W} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{Y}^T.$$

The only difference is that now the inputs are $\mathbf{\Phi}_{H\times N}$ instead of $\mathbf{X}_{D\times N}$, and that we have the extra term on λ.

III Polynomial regression

Review We minimize the same error as in eq. (1) but using the canonical basis for polynomials as basis functions (in 1D):

$$f(x) = \sum_{h=0}^{H} w_h \phi_h(x) \qquad \phi_h(x) = x^h \qquad x \in \mathbb{R},\ f(x) \in \mathbb{R}. \tag{4}$$

Since the basis functions have no parameters (width, centroids), $\boldsymbol{\phi}(x) = (1, x, x^2, \dots, x^H)^T$ is fixed given x, so finding the optimal weights can be solved exactly as in eq. (3) by solving a linear regression. The hyperparameters H and λ are set by cross-validation as before.

Exploration: toy problem As in the RBF network case.