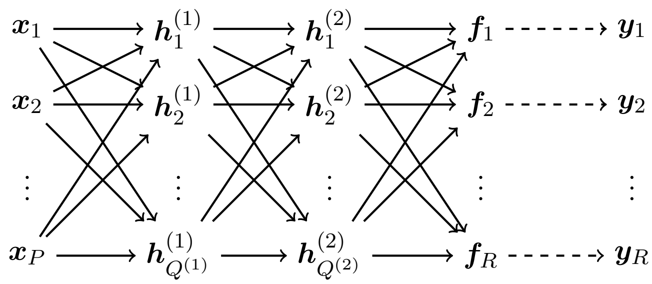
DEEP LEARNING TUTORIAL

MAZIAR RAISSI

1. Outline. This is a short tutorial on the following topics in Deep Learning: Neural Networks, Recurrent Neural Networks, Long Short Term Memory Networks, Variational Auto-encoders, and Conditional Variational Auto-encoders. The full code for this tutorial can be found in `https://github.com/maziarraissi/DeepLearningTutorial`.

2. Neural Networks [2]. Consider the following deep neural network with two hidden layers.



Here, $\boldsymbol{x}_p \in \mathbb{R}^{N\times 1}$ denotes dimension $p = 1,\dots,P$ of the input data $\boldsymbol{X} = [\boldsymbol{x}_1,\dots,\boldsymbol{x}_P] \in \mathbb{R}^{N\times P}$. The first hidden layer is given by

$$\boldsymbol{h}_q^{(1)} = h(\sum_{p=1}^{P} a_{pq}^{(1)}\boldsymbol{x}_p + b_q^{(1)}), \quad q = 1,\dots,Q^{(1)}, \tag{2.1}$$

where $\boldsymbol{h}_q^{(1)} \in \mathbb{R}^{N\times 1}$ denotes dimension $q = 1,\dots,Q^{(1)}$ of the matrix $\boldsymbol{H}^{(1)} = [\boldsymbol{h}_1^{(1)},\dots,\boldsymbol{h}_{Q^{(1)}}^{(1)}] \in \mathbb{R}^{N\times Q^{(1)}}$. In matrix-vector notations we obtain $\boldsymbol{H}^{(1)} = h(\boldsymbol{X}A^{(1)} + b^{(1)})$ with $A^{(1)} = [a_{pq}^{(1)}]_{p=1,\dots,P,q=1,\dots,Q^{(1)}}$ being a $P \times Q^{(1)}$ matrix of multipliers and $b^{(1)} = [b_1^{(1)},\dots,b_{Q^{(1)}}^{(1)}] \in \mathbb{R}^{1\times Q^{(1)}}$ denoting the bias vector. Here, $h(x)$ is the activation function given explicitly by $h(x) = \tanh(x)$. Similarly, the second hidden layer $\boldsymbol{H}^{(2)} \in \mathbb{R}^{N\times Q^{(2)}}$ is given by $\boldsymbol{H}^{(2)} = h(\boldsymbol{H}^{(1)}A^{(2)} + b^{(2)})$ where $A^{(2)}$ is a $Q^{(1)} \times Q^{(2)}$ matrix of multipliers and $b^{(2)} \in \mathbb{R}^{1\times Q^{(2)}}$ denotes the bias vector. The output of the neural network $\boldsymbol{F} \in \mathbb{R}^{N\times R}$ is given by $\boldsymbol{F} = \boldsymbol{H}^{(2)}A + b$ with $A \in \mathbb{R}^{Q^{(2)}\times R}$ and $b \in \mathbb{R}^{1\times R}$. Moreover, we assume a Gaussian noise model

$$\boldsymbol{y}_r = \boldsymbol{f}_r + \boldsymbol{\epsilon}_r, \quad r = 1,\dots,R, \tag{2.2}$$

with mutually independent $\boldsymbol{\epsilon}_r \sim \mathcal{N}(\boldsymbol{0}, \sigma_r^2\boldsymbol{I})$. Letting $\boldsymbol{y} := [\boldsymbol{y}_1;\dots;\boldsymbol{y}_R] \in \mathbb{R}^{NR\times 1}$, we obtain the following likelihood

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \Sigma \otimes \boldsymbol{I}), \tag{2.3}$$

where $\Sigma = \mathrm{diag}(\sigma_1^2,\dots,\sigma_S^2)$ and $\boldsymbol{f} := [\boldsymbol{f}_1;\dots;\boldsymbol{f}_R] \in \mathbb{R}^{NR\times 1}$. One can train the parameters of the neural network by minimizing the resulting negative log likelihood.

2.1. Illustrative Example. Figure 2.1 depicts a neural network fit to a synthetic dataset generated by random perturbations of a simple one dimensional function.

3. Recurrent Neural Networks [2]. Let us consider a time series dataset of the form $\{\boldsymbol{y}_t : t = 1,\dots,T\}$. We can employ the following recurrent neural network

$$\begin{tikzcd}
 & & \hat{\boldsymbol{y}}_t \\
\boldsymbol{h}_{t-2} \arrow[r, "W"] & \boldsymbol{h}_{t-1} \arrow[r, "W"] & \boldsymbol{h}_t \arrow[u, "{V, c}"'] \\
 & \boldsymbol{y}_{t-2} \arrow[u, "{U, b}"'] & \boldsymbol{y}_{t-1} \arrow[u, "{U, b}"']
\end{tikzcd}$$

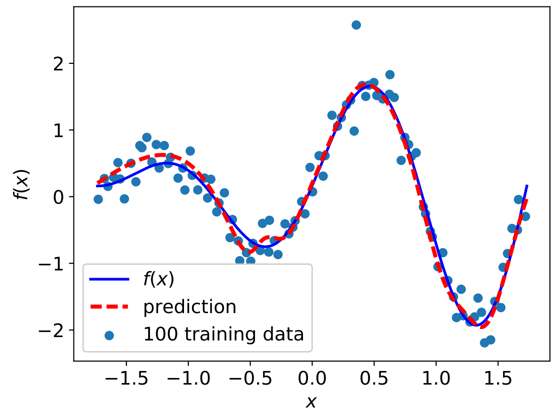


FIG. 2.1. *Neural network fitting a simple one dimensional function.*

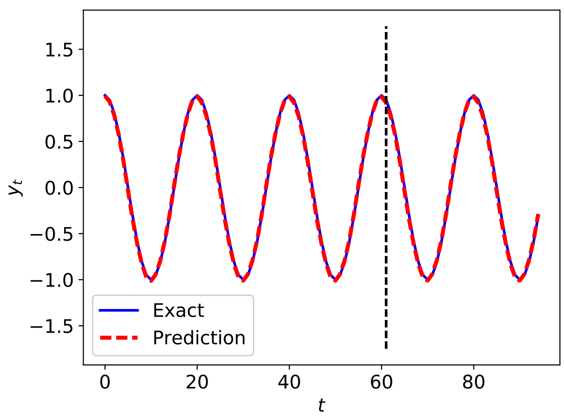


FIG. 3.1. *Recurrent neural network predicting the dynamics of a simple sine wave.*

to model the next value $\hat{\boldsymbol{y}}_t$ of the variable of interest as a function of its own lagged values $\boldsymbol{y}_{t-1}$ and $\boldsymbol{y}_{t-2}$; i.e., $\hat{\boldsymbol{y}}_t = f(\boldsymbol{y}_{t-1}, \boldsymbol{y}_{t-2})$. Here, $\hat{\boldsymbol{y}}_t = \boldsymbol{h}_t V + c$, $\boldsymbol{h}_t = \tanh(\boldsymbol{h}_{t-1}W + \boldsymbol{y}_{t-1}U + b)$, $\boldsymbol{h}_{t-1} = \tanh(\boldsymbol{h}_{t-2}W + \boldsymbol{y}_{t-2}U + b)$, and $\boldsymbol{h}_{t-2} = \boldsymbol{0}$. The parameters U, V, W, b, and c of the recurrent neural network can be trained my minimizing the mean squared error

$$\mathcal{MSE} := \frac{1}{T-2} \sum_{t=3}^{T} |\boldsymbol{y}_t - \hat{\boldsymbol{y}}_t|^2.$$

3.1. Illustrative Example. Figure 3.1 depicts a recurrent neural network (with 5 lags) learning and predicting the dynamics of a simple sine wave.

4. Long Short Term Memory (LSTM) Networks [2]. A long short term memory (LSTM) network replaces the units $\boldsymbol{h}_t = \tanh(\boldsymbol{h}_{t-1}W + \boldsymbol{y}_{t-1}U + b)$ of a recurrent neural network with

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{s}_t),$$

where

$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{h}_{t-1}W_o + \boldsymbol{y}_{t-1}U_o + b_o\right),$$

is the output gate and

$$\boldsymbol{s}_t = \boldsymbol{f}_t \odot \boldsymbol{s}_{t-1} + \boldsymbol{i}_t \odot \widetilde{\boldsymbol{s}}_t,$$

is the cell state. Here,

$$\widetilde{\boldsymbol{s}}_t = \tanh\left(\boldsymbol{h}_{t-1}W_s + \boldsymbol{y}_{t-1}U_s + b_s\right).$$

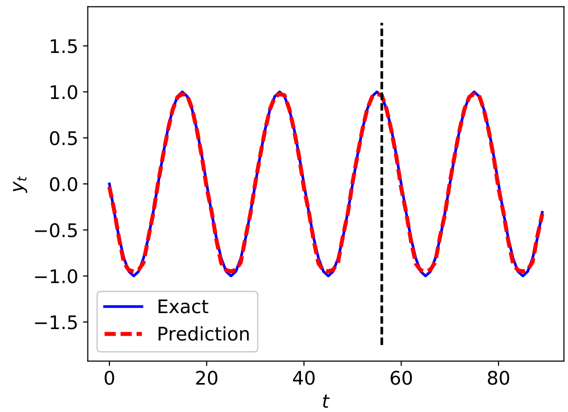


FIG. 4.1. *Long short term memory network predicting the dynamics of a simple sine wave.*

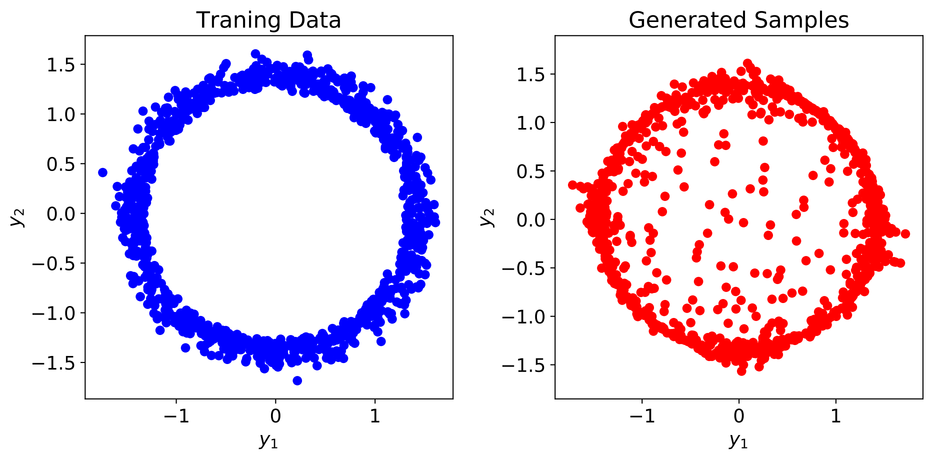


FIG. 4.2. *Training data and samples generated by a variational auto-encoder.*

Moreover,

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{h}_{t-1} W_i + \boldsymbol{y}_{t-1} U_i + b_i\right)$$

is the external input gate while

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{h}_{t-1} W_f + \boldsymbol{y}_{t-1} U_f + b_f\right),$$

is the forget gate.

4.1. Illustrative Example. Figure 4.1 depicts a long short term memory network (with 10 lags) learning and predicting the dynamics of a simple sine wave.

5. Variational Auto-encoders [1]. Let us start by the prior assumption that

$$p(\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z}|\boldsymbol{0}, I\right),$$

where $\boldsymbol{z}$ is a latent variable. Moreover, let us assume

$$p(\boldsymbol{y}|\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{y}|\mu_2(\boldsymbol{z}), \Sigma_2(\boldsymbol{z})\right),$$

where $\mu_2(\boldsymbol{z})$ and $\Sigma_2(\boldsymbol{z})$ are modeled as deep neural networks. Here, $\Sigma_2(\boldsymbol{z})$ is constrained to be a diagonal matrix. We are interested in minimizing the negative log likelihood $-\log p(\boldsymbol{y})$, where $p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}$. However, $-\log p(\boldsymbol{y})$ is not analytically tractable. To deal with this issue, one could employ a variational distribution $q(\boldsymbol{z}|\boldsymbol{y})$ and compute the following Kullback-Leibler divergence; i.e.,

$$\mathbb{KL}\left[q(\boldsymbol{z}|\boldsymbol{y}) \;||\; p(\boldsymbol{z}|\boldsymbol{y})\right] = \int \log \frac{q(\boldsymbol{z}|\boldsymbol{y})}{p(\boldsymbol{z}|\boldsymbol{y})} q(\boldsymbol{z}|\boldsymbol{y}) d\boldsymbol{z} = \int \left[\log q(\boldsymbol{z}|\boldsymbol{y}) - \log p(\boldsymbol{z}|\boldsymbol{y})\right] q(\boldsymbol{z}|\boldsymbol{y}) d\boldsymbol{z}.$$

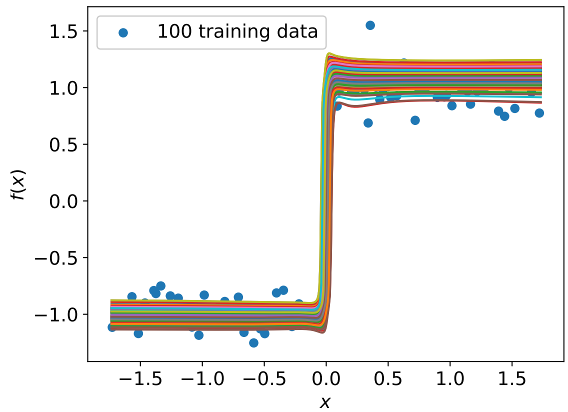


FIG. 6.1. *Training data and samples generated by a conditional variational auto-encoder.*

Using the Bayes rule $p(\boldsymbol{z}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{y})}$, one obtains

$$\mathbb{KL}\left[q(\boldsymbol{z}|\boldsymbol{y}) \mid\mid p(\boldsymbol{z}|\boldsymbol{y})\right] = \int \left[\log q(\boldsymbol{z}|\boldsymbol{y}) - \log p(\boldsymbol{y}|\boldsymbol{z}) - \log p(\boldsymbol{z}) + \log p(\boldsymbol{y})\right] q(\boldsymbol{z}|\boldsymbol{y}) d\boldsymbol{z}.$$

Therefore,

$$\mathbb{KL}\left[q(\boldsymbol{z}|\boldsymbol{y}) \mid\mid p(\boldsymbol{z}|\boldsymbol{y})\right] = \log p(\boldsymbol{y}) + \mathbb{KL}\left[q(\boldsymbol{z}|\boldsymbol{y}) \mid\mid p(\boldsymbol{z})\right] - \int \log p(\boldsymbol{y}|\boldsymbol{z}) q(\boldsymbol{z}|\boldsymbol{y}) d\boldsymbol{z}.$$

Rearranging the terms yields

$$-\log p(\boldsymbol{y}) + \mathbb{KL}\left[q(\boldsymbol{z}|\boldsymbol{y}) \mid\mid p(\boldsymbol{z}|\boldsymbol{y})\right] = -\int \log p(\boldsymbol{y}|\boldsymbol{z}) q(\boldsymbol{z}|\boldsymbol{y}) d\boldsymbol{z} + \mathbb{KL}\left[q(\boldsymbol{z}|\boldsymbol{y}) \mid\mid p(\boldsymbol{z})\right].$$

A variational auto-encoder proceeds by minimizing the terms on the right hand side of the above equation. Moreover, let us assume that

$$q(\boldsymbol{z}|\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{z}|\mu_1(\boldsymbol{y}), \Sigma_1(\boldsymbol{y})\right),$$

where $\mu_1(\boldsymbol{y})$ and $\Sigma_1(\boldsymbol{y})$ are modeled as deep neural networks. Here, $\Sigma_1(\boldsymbol{y})$ is constrained to be a diagonal matrix. One can use

$$\mu_1(\boldsymbol{y}) + \boldsymbol{\epsilon}\Sigma_1(\boldsymbol{y})^{1/2}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, I).$$

to generate samples from $q(\boldsymbol{z}|\boldsymbol{y})$.

5.1. Illustrative Example. Figure 4.2 depicts the training data and the samples generated by a variational auto-encoder.

6. Conditional Variational Auto-encoders. Conditional variational auto-encoders, rather that making the assumption that $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, I)$, start by assuming that

$$p(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z}|\mu_0(\boldsymbol{x}), \Sigma_0(\boldsymbol{x})\right),$$

where $\mu_0(\boldsymbol{x})$ and $\Sigma_0(\boldsymbol{x})$ are modeled as deep neural networks. Here, $\Sigma_0(\boldsymbol{x})$ is constrained to be a diagonal matrix.

6.1. Illustrative Example. Figure 6.1 depicts the training data and the samples generated by a conditional variational auto-encoder.

REFERENCES

[1] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.