CS 221: Section 2

Learning with Sequential Inputs/Outputs

...

Basics of Recurrent Neural Networks

Autumn 2017 Course Staff

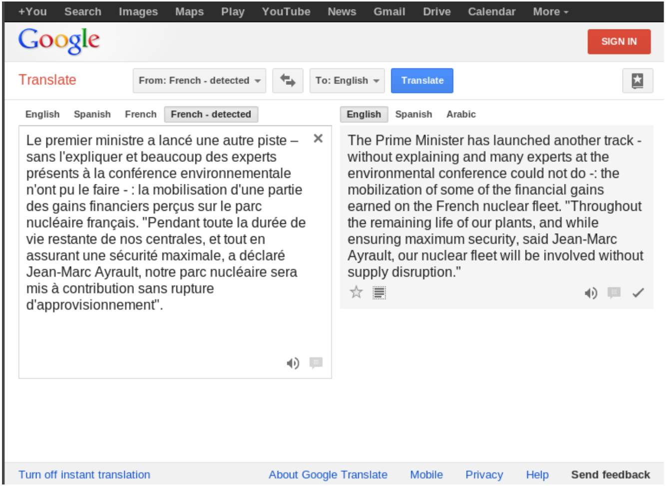
Outline

Outline

Medical Diagnosis

Input (x_t): medical history
Output (y_t): next emergency department visit

Text Translation



Input (x_t): French words in sentence
Output (y_t): translated English words

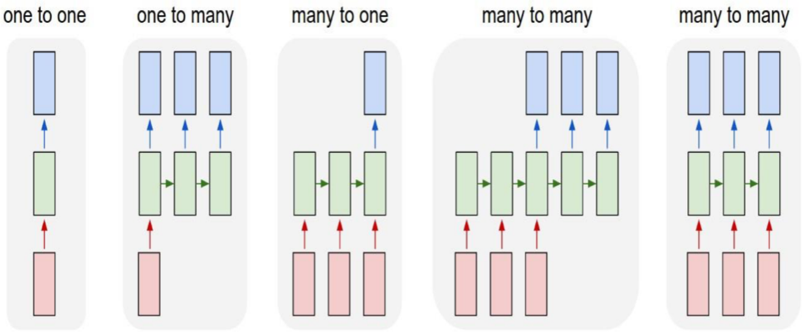
Image Captioning

Input: image
Output: sentence of words

Stock Prediction

Input (x_t): historical stock prices
Output (y_t): stock price today

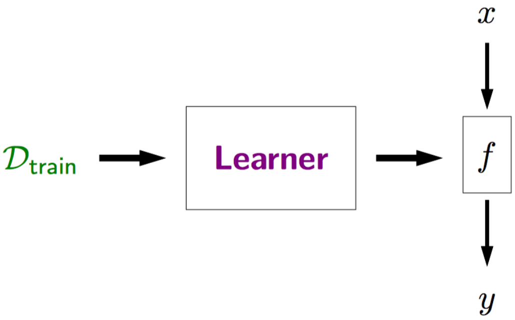
Sequences of Data



Adapted from Fei-Fei Li's CS 231N slides.

Outline

ML Framework



1. Obtain data $\mathcal{D}_{\text{train}}$.
2. Model: choose f to capture relationship between x and y.
3. Loss: specify a loss which tells how right or wrong your prediction is
4. Optimize: run gradient descent to find the best set of weights for your model

Developing a Stock Price Model

Goal: predict what the stock price will be today.

Idea: stock price today depends on prices in the past.

Data: Let x_t be the price on day t. We collect x_t from August, 2016 to August, 2017. Let y_t be the predicted price for day t.

Developing a Stock Price Model

Model: assume a linear relationship.

$$y_t = w_0 + w_1 x_{t-1} + w_2 x_{t-2} + w_3 x_{t-3}$$

In terms of the notation from lecture, we have

$$y_t = w^\top \phi_t, \qquad \phi_t = \left[1, x_{t-1}, x_{t-2}, x_{t-3}\right]^\top$$

Loss: squared loss between predicted and actual stock price.

$$L(x_t, y_t) = (x_t - y_t)^2 = (x_t - w^\top \phi_t)^2$$

Initial Stock Price Model

Benefits

- Training: just ordinary least squares [linear regression].
- Interpretability: linear combination of features.

Problems

- Scalability: We have to manually specify the number of days in the past to include as features.
- Features: hard to specify complex relation between past prices using linear model. For example, want to use $\mathbb{1}\{x_{t-4} - x_{t-5}\}$.

Long Past

Can we repackage the model to keep track of the history more easily?

Let's track h_t as our knowledge to predict time t. Suppose we set

$$\begin{aligned} h_t &= ah_{t-1} + bx_{t-1} \\ h_{t-1} &= ah_{t-2} + bx_{t-2} \\ &\vdots \end{aligned}$$

If we do this cleverly, we can have h_t account for all past information!

Adding Non-Linearity

Let's model a more complex relationship b/w the present and past.

Add some non-linear function; examples:

- sigmoid [logistic unit from lecture]
- relu [rectified linear unit]

We end up with:

$$h_t = f_{\text{nonlinear}}(ah_{t-1} + bx_{t-1})$$
$$y_t = g_{\text{nonlinear}}(ch_t)$$

Outline

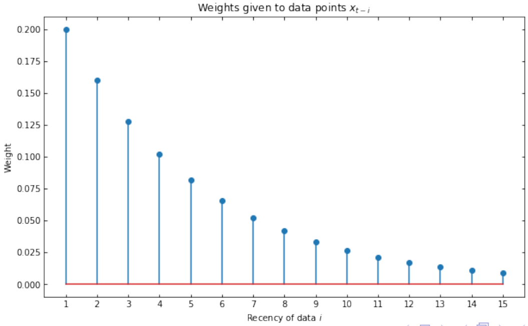
Example RNN Model

Let's model stock price as a weighted moving average.

$$h_t = f(h_{t-1}, x_{t-1}) \qquad h_t = (1-\alpha)h_{t-1} + \alpha x_{t-1}$$
$$y_t = g(h_t) \qquad y_t = h_t$$

Why is this a reasonable model? If we expand it out, we find

$$y_t = h_t = \alpha x_{t-1} + \alpha(1-\alpha)x_{t-2} + \alpha(1-\alpha)^2 x_{t-3} + \dots$$



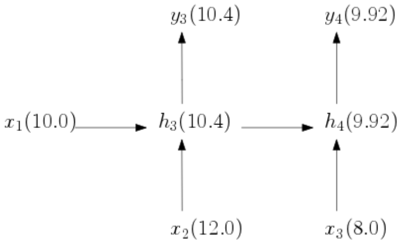
Running forward and backward

Let's model stock price as a weighted moving average.

$$h_t = f(h_{t-1}, x_{t-1}) \qquad h_t = (1-\alpha)h_{t-1} + \alpha x_{t-1}$$
$$y_t = g(h_t) \qquad y_t = h_t$$

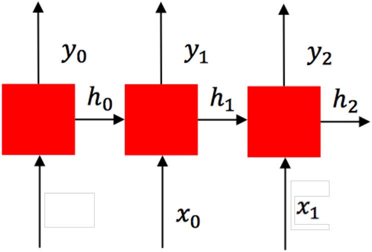
Suppose we're given $x = \begin{bmatrix} 10 & 12 & 8 & 12 \end{bmatrix}^\top$, and $\alpha = 0.2$.

[boardwork]



Basics of RNNs

We've covered a specific case of the more general RNN:



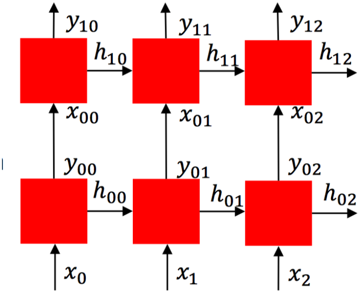
Adapted from John Canny's CS 294-129 slides.

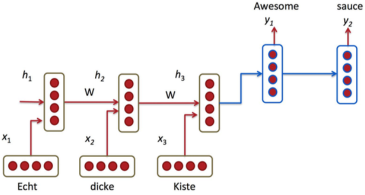
So: RNN is neural network with recurrent feature over time.

$$h_t = f(h_{t-1}, x_{t-1})$$
$$y_t = g(h_t)$$

Expanding RNNs

It's common to stack layers to train higher-level features. You can also have an input RNN (encoder) and an output RNN (decoder).

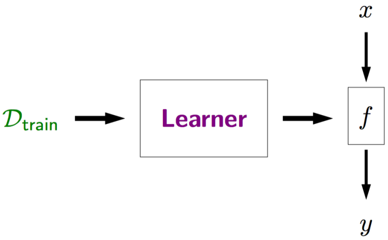




Adapted from John Canny's CS 294-129 slides and Stanford's CS 224N slides.

ML Framework

Takeaway. An RNN is just another choice of how we want to capture the relationship between X and Y! It can be combined with other RNN's or other models like a lego piece.



1. Obtain data $\mathcal{D}_{\text{train}}$.
2. Model: choose f to capture relationship between x and y.
3. Loss: specify a loss which tells how right or wrong your prediction is
4. Optimize: run gradient descent to find the best set of weights for your model

Outline

Additional resources

If you're interested in using RNNs for your project, we recommend the following resources:

1. The Unreasonable Effectiveness of RNNs (Karpathy 2015)
2. Deep Learning Textbook Chapter on RNNs (Goodfellow et al 2016)
3. WildML RNN Tutorial With Code (Britz 2015)
4. CS 224N Lecture Notes (Instructors Chris Manning and Richard Socher)
5. Ask Us!!