Data Mining & Machine Learning

CS57300
Purdue University

April 10, 2018

Predicting Sequences

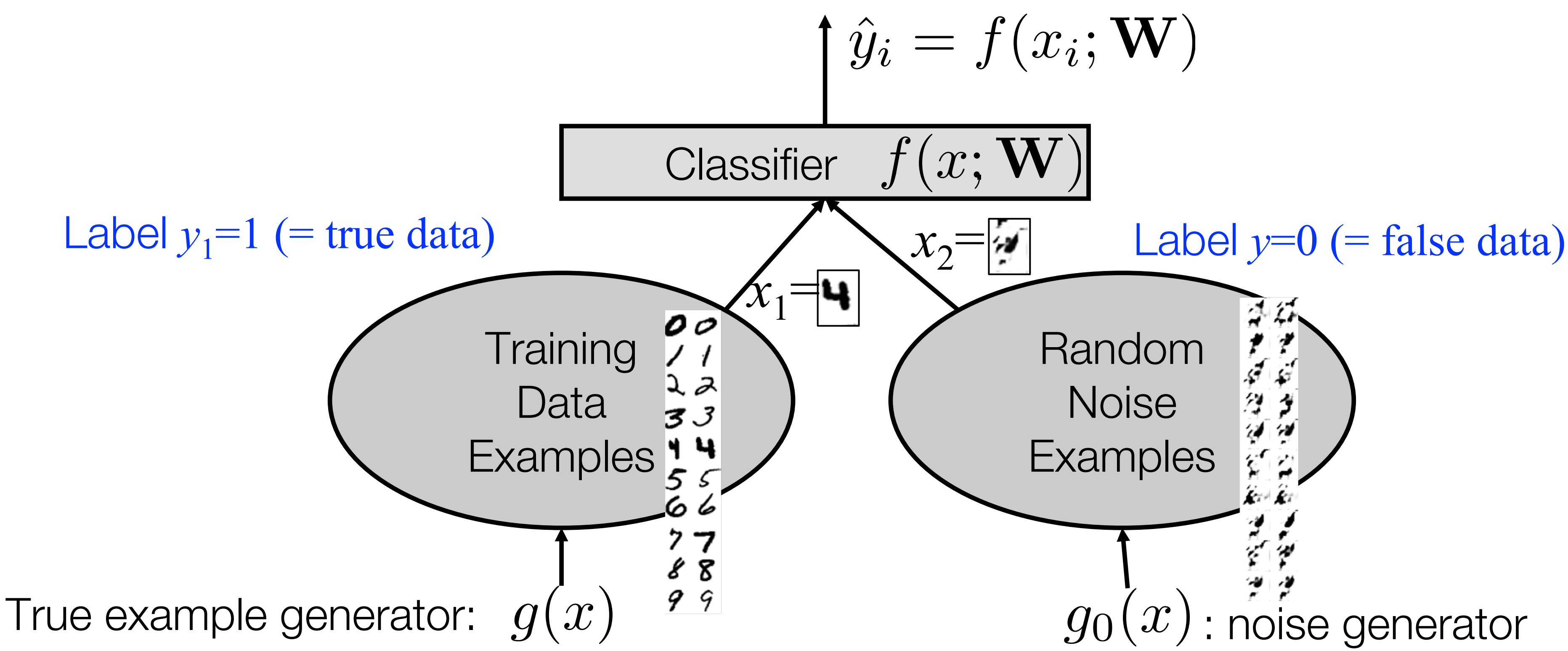
But first, a detour to Noise Contrastive Estimation

Unsupervised Learning as Supervised Learning

- Machine learning methods are much better at classifying examples than generating new ones
  - In classification tasks, we use the exact derivatives to find a solution that maximizes the likelihood
  - In generative tasks, we can only compute an estimate of the derivative
- Because we have better techniques to classify data than to generate data…
  - Can we make generative tasks look more like classification tasks?

Learn a Classifier to Distinguishing Noise from Data

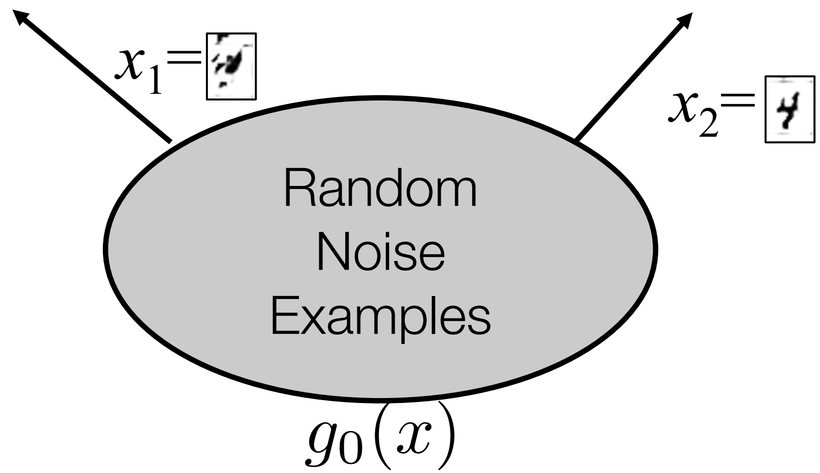
- This is key idea behind ***noise contrastive estimation*** (*NCE*): make generative tasks look like classification tasks
  - Pioneered by Hastie, Tibshirani, Friedman in The Elements of Statistical Learning in 2008, Section 14.2.4 “Unsupervised Learning as Supervised Learning”
  - The idea is quite simple, consider the task of learning to distinguish noise from the data, i.e., search for a good classifier $f(x;\mathbf{W})$



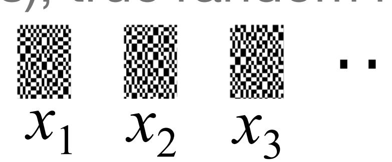
Generation Task Using Classifier $f(x;\mathbf{W})$

- Now use the learned classifier (which distinguishes noise from the data) to generate new data… but how?
  - Naïve approach: Generate examples from the random noise… whatever gets classified as “real data” will be our generated examples

$\hat{y}_1 = f(x_1;\mathbf{W}) \approx 0$ $\hat{y}_2 = f(x_2;\mathbf{W}) \approx 1$



  - What is the problem with this naïve approach?
    - In very high dimensions (e.g., images), true random noise will not generate any interesting examples



Go to iPython notebook

Back to sequences

Sequences

- In this lecture we will focus on word sequences (a.k.a. text)
- The techniques we see are applicable to any type of sequence

- A sequence is a succession of elements from a set (likely finite)
- We will write a sequence of n elements as $x_1, \ldots, x_n$
- The temporal ordering is key to learning the sequence

PURDUE UNIVERSITY

Word Sequences

"Rank-3" word embedding

- Embed words w.r.t. their sentences
  - "Bring me a constant woman to her husband"
  - "Forgetting, like a good man your late censure" of his wife

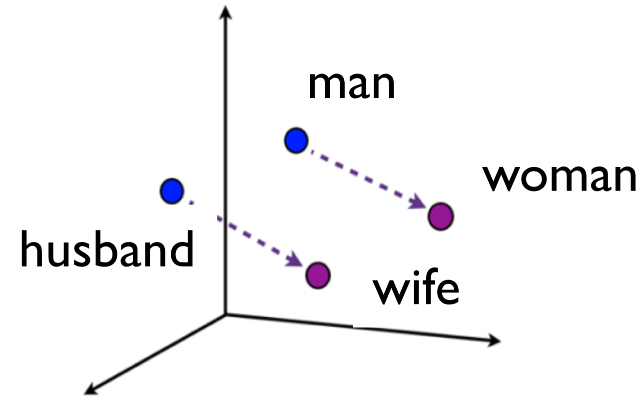
www.tensorflow.org

- Under the Markov assumption
  - P[bring, me, a, constant, woman, to, her, husband] = P[bring]P[me | bring] P[a | me] P[constant| a] … P[husband | her]

  - Go to ipython notebook

Word Sequences & Embeddings

"Rank-3" word embedding



www.tensorflow.org

Male-Female

- Embed words w.r.t. their sentences
  - "Bring me a constant woman to her husband"
  - "Forgetting, like a good man your late censure" of his wife

- Initial idea by (Chen et al. 2012) was named Latent Markov Embedding, rediscovered by (Mikolov et al. 2013) named **word2vec**

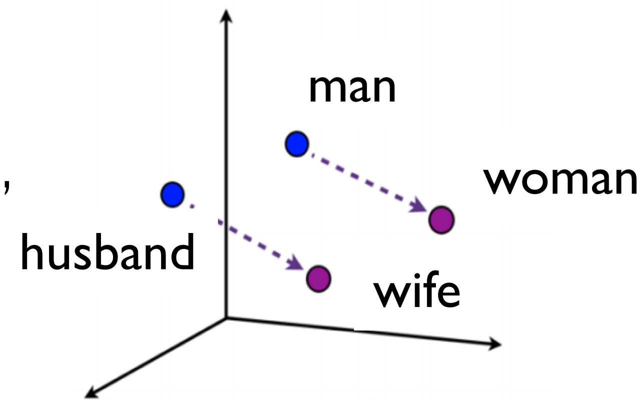
- Main difference:
  - Application: (Mikolov et al. 2013) paid attention to the composition of latent vectors in sentences
  - Otherwise, techniques equivalent

Chen, S., Moore, J. L., Turnbull, D., & Joachims, T. (2012). Playlist prediction via metric embedding. In ACM *SIGKDD.*

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In NIPS

Word2vec Embeddings

"Rank-3" word embedding



www.tensorflow.org

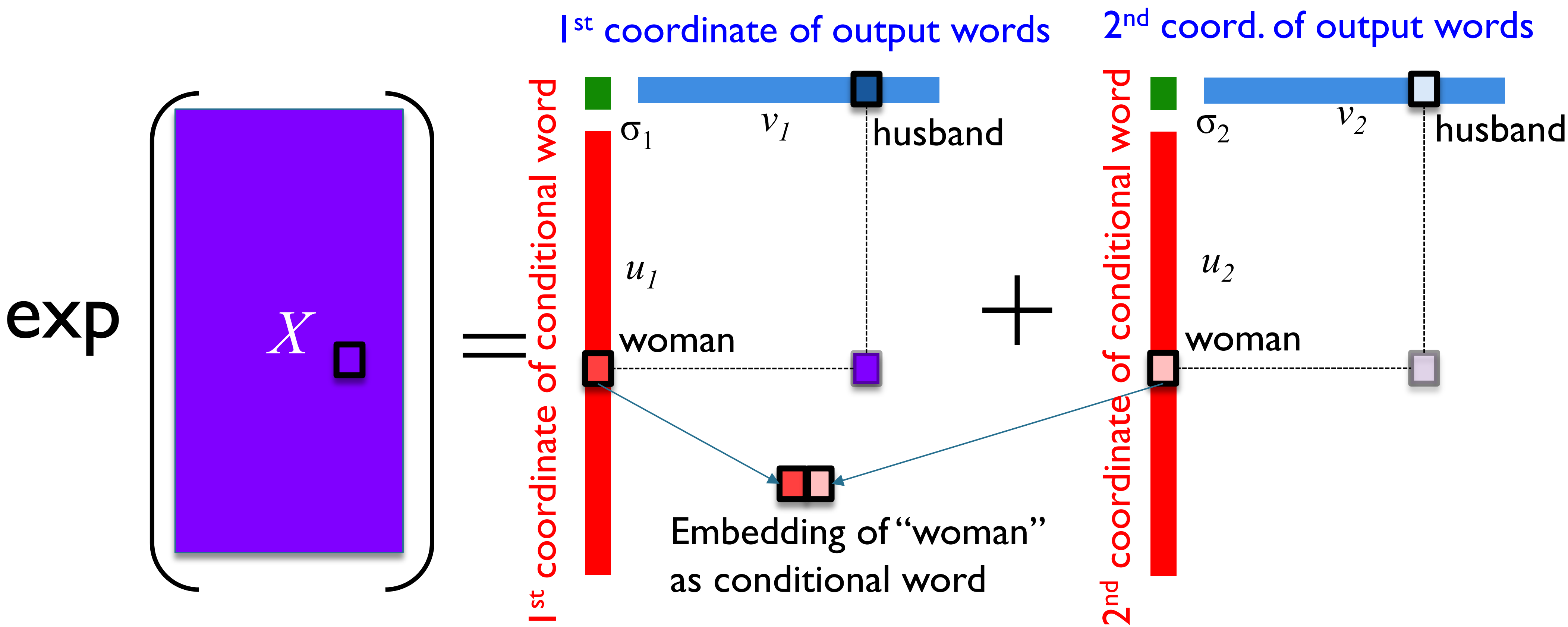
Male-Female

How the math works

- "Bring me a constant woman to her husband"
- "Forgetting, like a good man your late censure" of his wife

- Conditional bag-of-words assumption (SKIPGRAM): All words are independent given the target word
  - P[bring, constant, husband | woman] = P[bring | woman] P[constant | woman] P[husband | woman]
  - P[forgetting, like, good, late, censure, wife | man] = P[forgetting | man] P[like | man] P[good | man] P[late | man] P[censure | man] P[wife | man]

Word2vec Type Embeddings III



$$P[\text{husband}|\text{woman}] = \frac{\exp\left(\sum_{i=1}^{k} \sigma_i u_{(i,\text{woman})} v_{(i,\text{husband})}\right)}{\sum_{w \in \text{AllWords}} \exp\left(\sum_{i=1}^{k} \sigma_i u_{(i,\text{woman})} v_{(i,w)}\right)}$$

The machine learning challenge is not summing over all words