Log-linear models (part III)

Lecture, Feb 7

CS 690N, Spring 2017

Advanced Natural Language Processing

http://people.cs.umass.edu/~brenocon/anlp2017/

Brendan O'Connor

College of Information and Computer Sciences

University of Massachusetts Amherst

MaxEnt / Log-Linear models

- **x**: input (all previous words)
- **y**: output (next word)
- **f(x,y)** => R^d feature function [[domain knowledge here!]]
- **v**: R^d parameter vector (weights)

$$p(y|x;v) = \frac{\exp\left(v \cdot f(x,y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(v \cdot f(x,y')\right)}$$

Application to history-based LM:

$$P(w_1..w_T) = \prod_t P(w_t \mid w_1..w_{t-1})$$

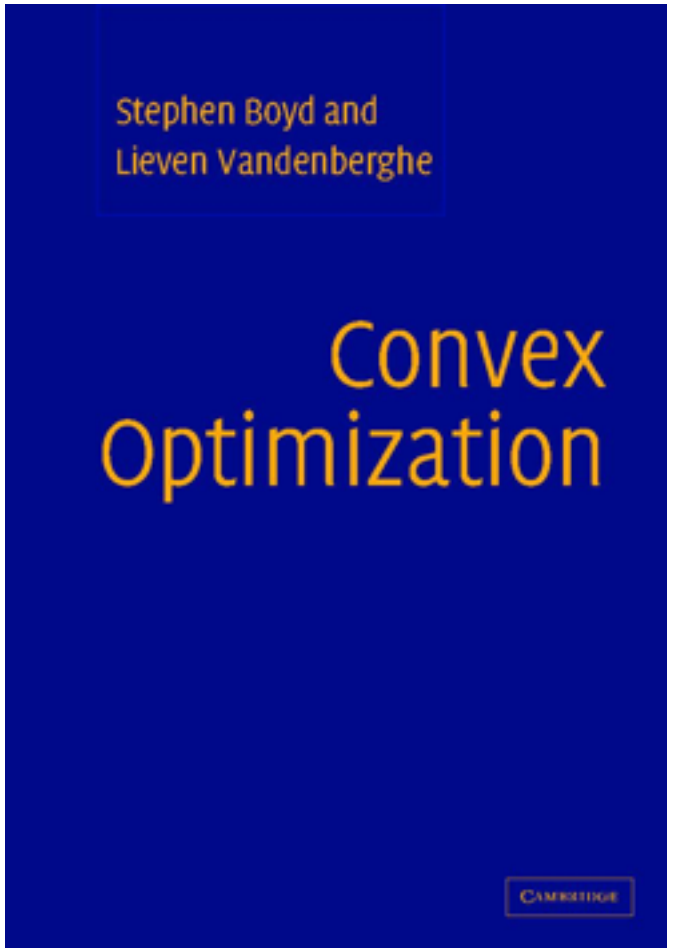
$$= \prod_t \frac{\exp(v \cdot f(w_1..w_{t-1}, w_t))}{\sum_{w \in \mathcal{V}} \exp(v \cdot f(w_1..w_{t-1}, w))}$$

Learning

$$\log p(y|x;v) \quad = \quad v \cdot f(x,y) - \log \sum_{y' \in \mathcal{Y}} \exp\left(v \cdot f(x,y')\right)$$

$$\frac{\partial}{\partial v_j} \log p(y|x;v) \quad =$$

- Gradient at a single example: can it be zero?
- Full dataset gradient: First moments match at the mode
- Log-likelihood is concave
  - At least with regularization, since typically linearly separable
  - Is my function convex?
    Check Boyd and Vandenberghe ch. 3



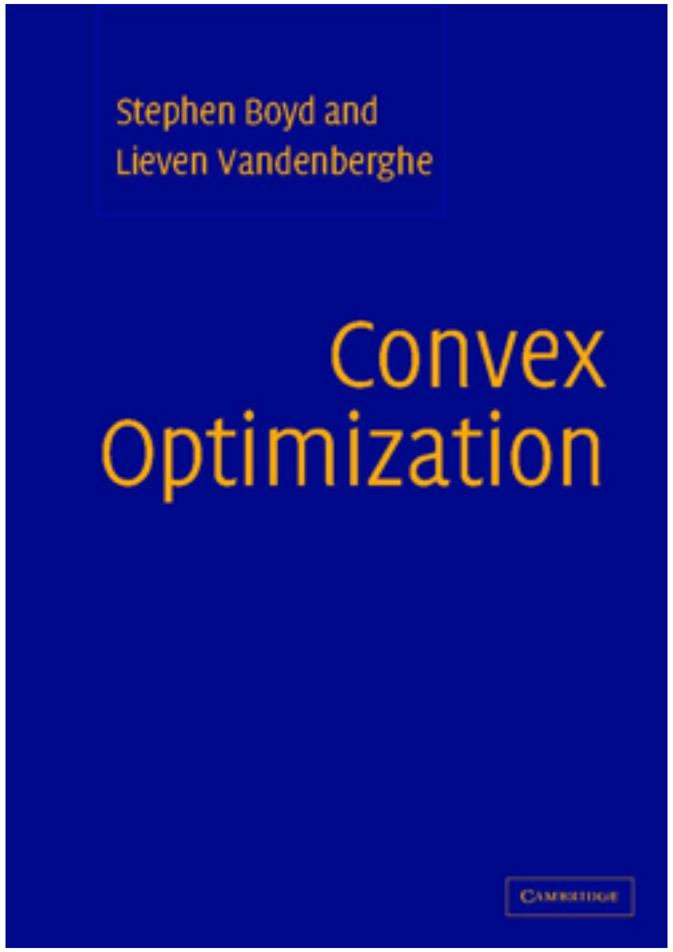
Learning

$$\log p(y|x;v) \quad = \quad v \cdot f(x,y) - \log \sum_{y' \in \mathcal{Y}} \exp\left(v \cdot f(x,y')\right)$$

fun with the chain rule

$$\frac{\partial}{\partial v_j} \log p(y|x;v) \quad =$$

- Gradient at a single example: can it be zero?
- Full dataset gradient: First moments match at the mode
- Log-likelihood is concave
  - At least with regularization, since typically linearly separable
  - Is my function convex?
    Check Boyd and Vandenberghe ch. 3



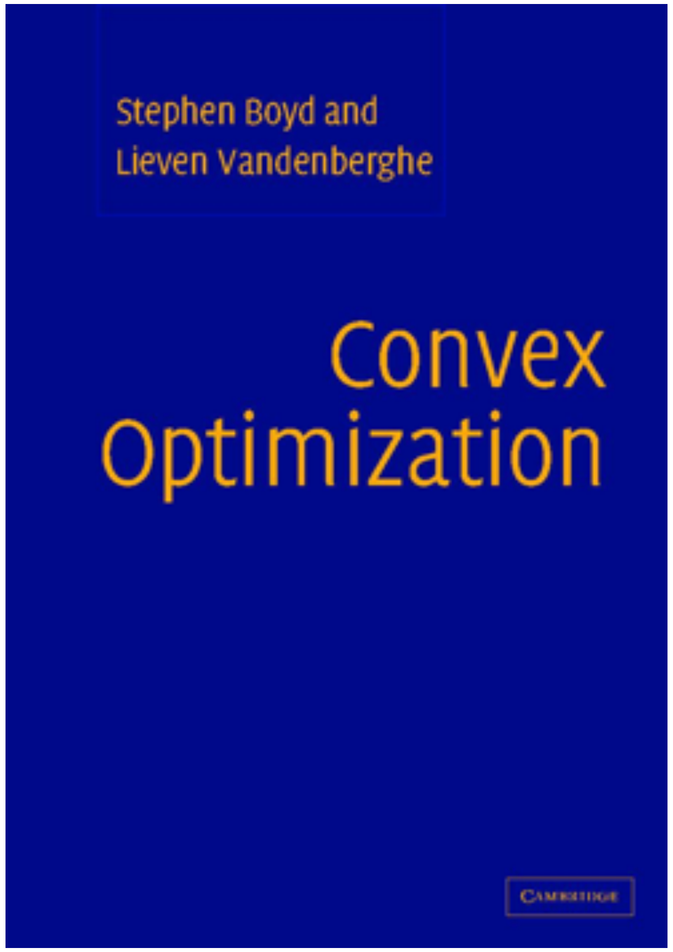
Learning

$$\log p(y|x;v) = v \cdot f(x,y) - \log \sum_{y' \in \mathcal{Y}} \exp\left(v \cdot f(x,y')\right)$$

fun with the chain rule

$$\frac{\partial}{\partial v_j} \log p(y|x;v) = f_j(x,y) - \sum_{y'} p(y'|x;v) f_j(x,y')$$

- Gradient at a single example: can it be zero?
- Full dataset gradient: First moments match at the mode
- Log-likelihood is concave
  - At least with regularization, since typically linearly separable
  - Is my function convex?
    Check Boyd and Vandenberghe ch. 3



Learning

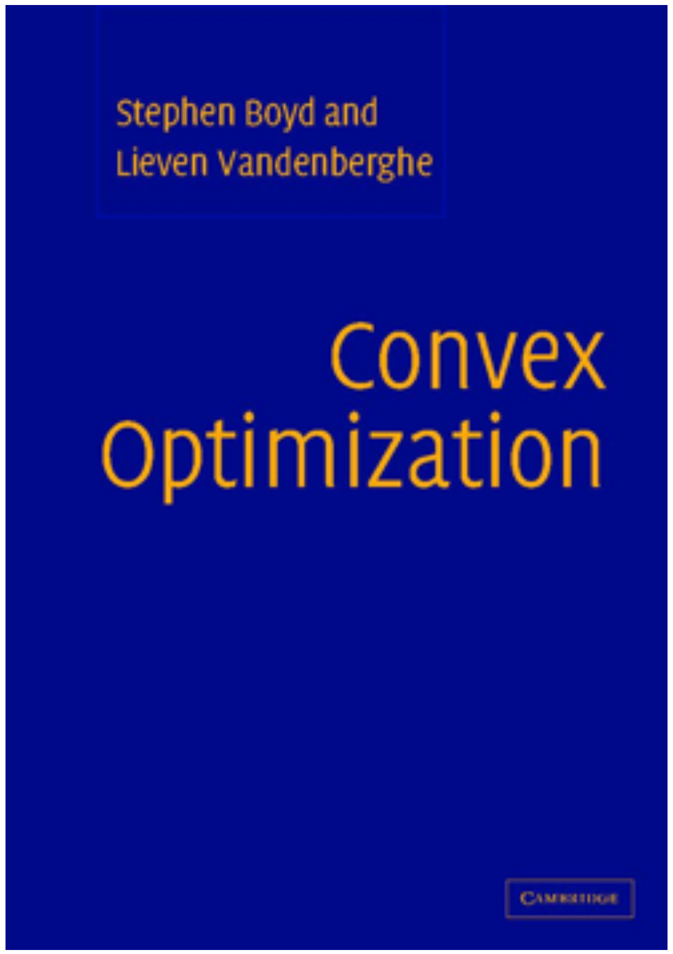
$$\log p(y|x;v) \quad = \quad v \cdot f(x,y) - \log \sum_{y' \in \mathcal{Y}} \exp\left(v \cdot f(x,y')\right)$$

fun with the chain rule

$$\frac{\partial}{\partial v_j} \log p(y|x;v) \quad = \quad f_j(x,y) \quad - \quad \sum_{y'} p(y'|x;v) f_j(x,y')$$

Feature in data? Feature in posterior?

- Gradient at a single example: can it be zero?
- Full dataset gradient: First moments match at the mode
- Log-likelihood is concave
  - At least with regularization, since typically linearly separable
  - Is my function convex?
    Check Boyd and Vandenberghe ch. 3



Gradient descent

- Batch gradient descent (doesn't work well by itself)
- Most commonly used alternatives
  - LBFGS (adaptive version of batch GD)
    - Call a library implementation with gradient callback
  - SGD, one example at a time
    - and adaptive variants: Adagrad, Adam, etc.
    - Intuition
    - Issue: Combining per-example sparse updates with regularization updates
      - Lazy updates
      - Occasional regularizer steps (easy to implement)

- stopped here on 2/7

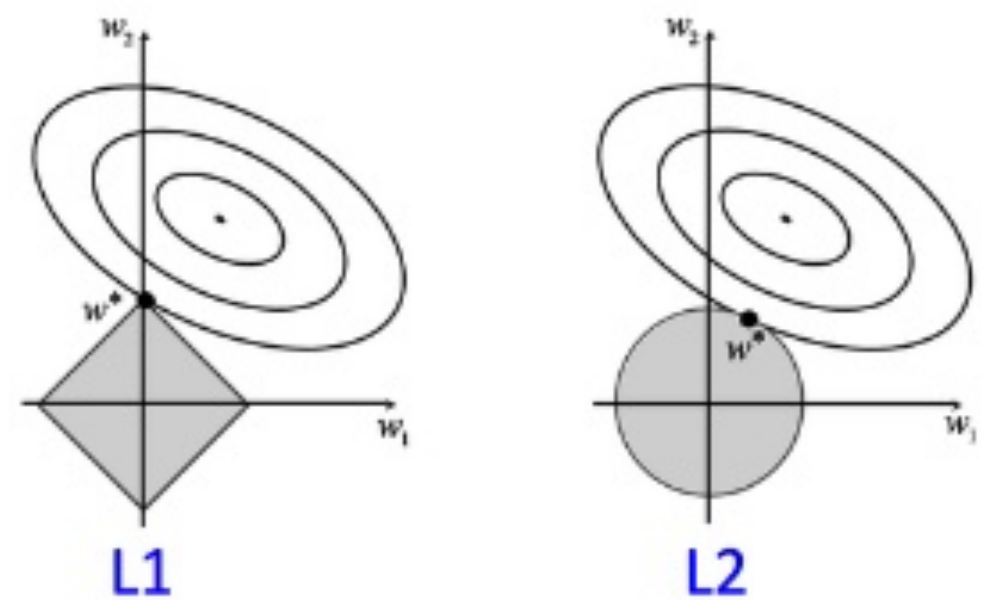
Engineering

- Sparse dot products are crucial!
- Lots and lots of features?
  - Millions to billions of features: performance often keeps improving!
  - Features seen only once at training time typically help
  - Feature name=>number mapping is the problem; the parameter vector is fine
- Feature hashing: make e.g. N(u,v,w) mapping random with collisions (!)
  - Accuracy loss low since features are rare. Works well, great for large-scale data (memory usage constant!)
  - Practically: use a fast string hashing function (e.g. murmurhash or Python's internal one)

Feature selection

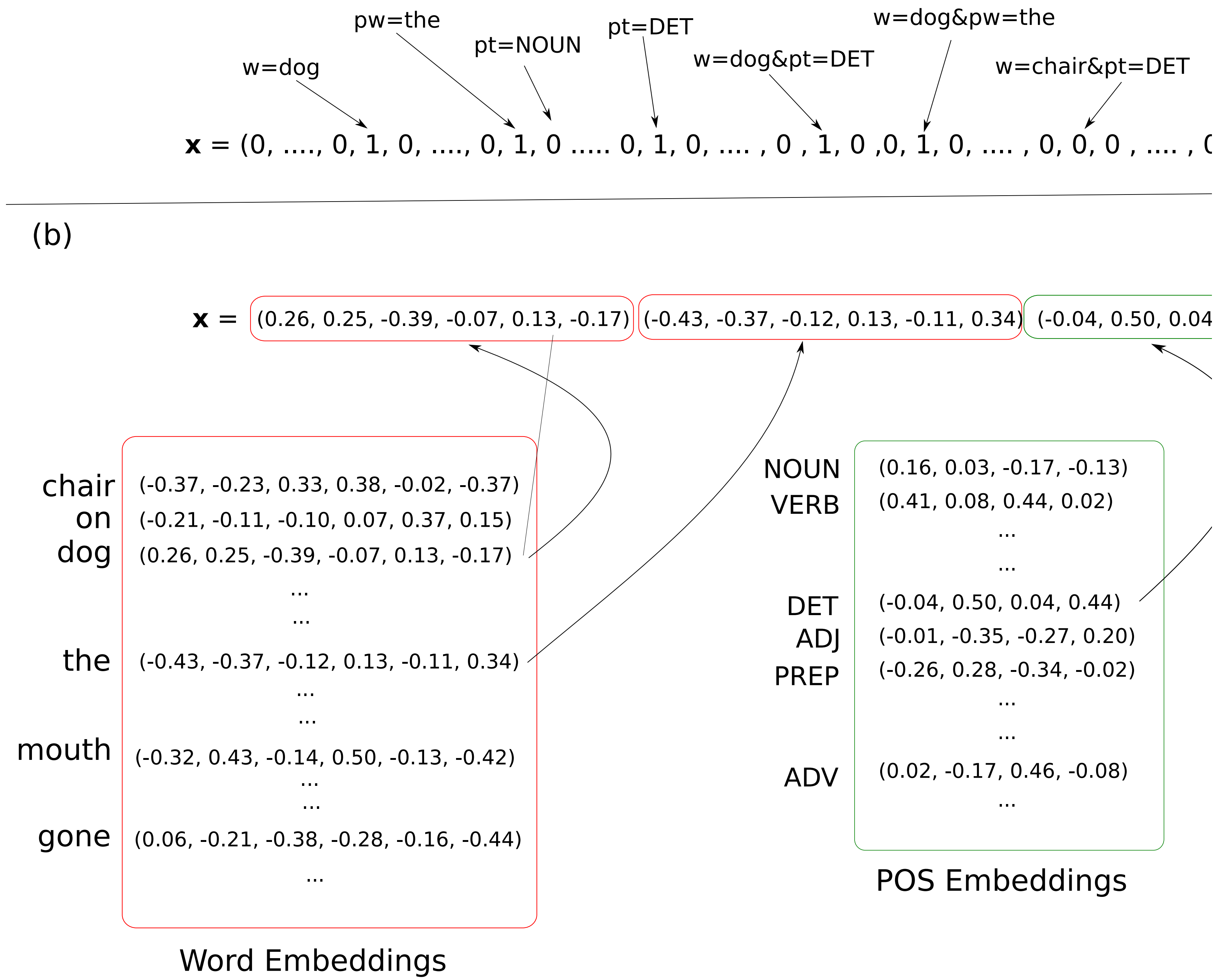
- Offline feature selection
  - Count cutoffs: computational, not performance benefits
  - Predictive value: mutual info. / info. gain / chi-square
- L1 regularization: encourages θ sparsity

$$\min_{\theta} \; -\log p_\theta(y|x) + \lambda \sum_j |\theta_j|$$



- L1 optimization: convex but nonsmooth; requires subgradient methods

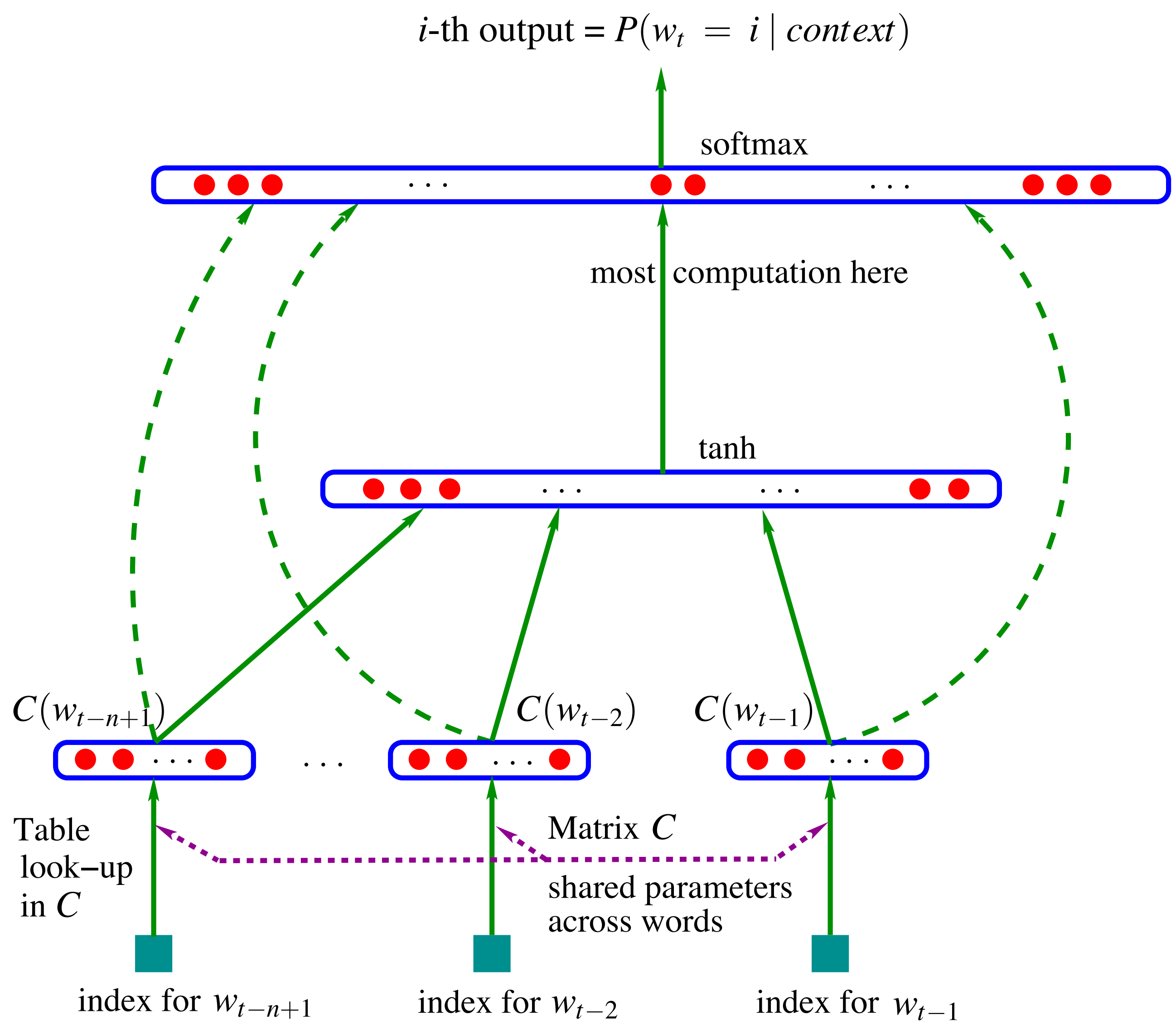
Dense representations



- Saul and Pereira 1997?
- Mnih and Hinton 2007: log-bilinear model

- Bengio et al. 2003: N-gram MLP

$$f(w_t, \cdots, w_{t-n+1}) = \hat{P}(w_t | w_1^{t-1})$$



$$C(i) \in \mathbb{R}^m$$

Word embedding parameters

$$x = (C(w_{t-1}), C(w_{t-2}), \cdots, C(w_{t-n+1}))$$

$$y = b + Wx + U \tanh(d + Hx)$$

$$\hat{P}(w_t | w_{t-1}, \cdots w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}.$$