Nonlinear Classification

CS6120: Natural Language Processing
Northeastern University

David Smith

Making Classifiers Better

- Better features, representations (HW2)
- Better output representations
  - discrete, continuous, ordinal, structured
- Train lots of classifiers and combine them
  - ensemble methods, bagging
- Train later classifiers to fix up earlier ones
  - boosting, stacking

Ensemble Classifiers

Ensemble Classifiers

$$\theta_1 \cdot f_1(x, y) = \psi_1$$

Ensemble Classifiers

$$\theta_1 \cdot f_1(x, y) = \psi_1$$

$$\theta_2 \cdot f_2(x, y) = \psi_2$$

Ensemble Classifiers

$$\theta_1 \cdot f_1(x, y) = \psi_1$$

$$\theta_2 \cdot f_2(x, y) = \psi_2$$

$$\theta_3 \cdot f_3(x, y) = \psi_3$$

Ensemble Classifiers

$$\theta_1 \cdot f_1(x, y) = \psi_1$$

$$\theta_2 \cdot f_2(x, y) = \psi_2$$

$$\theta_3 \cdot f_3(x, y) = \psi_3$$

$$\ldots$$

Ensemble Classifiers

$$\theta_1 \cdot f_1(x, y) = \psi_1$$

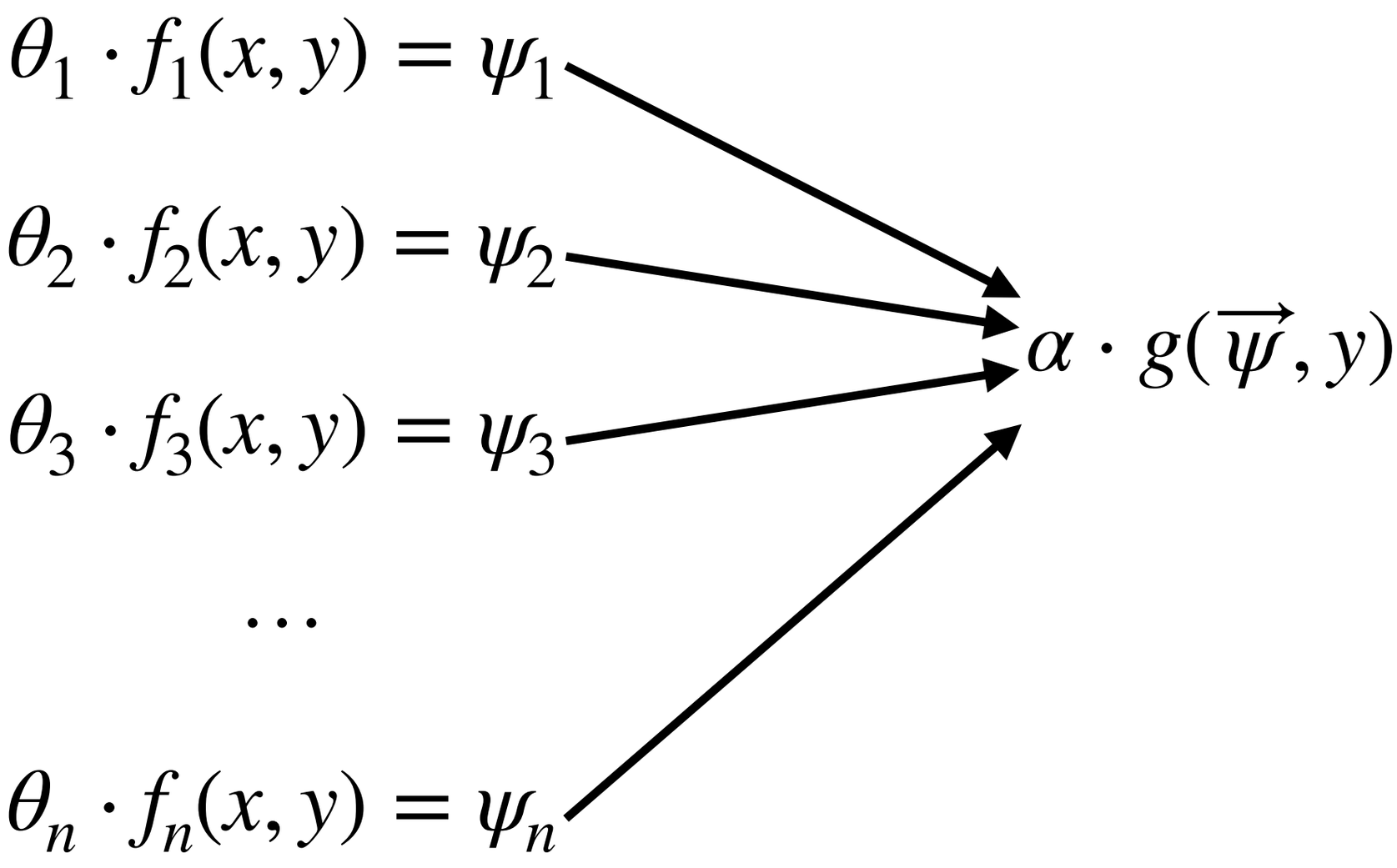
$$\theta_2 \cdot f_2(x, y) = \psi_2$$

$$\theta_3 \cdot f_3(x, y) = \psi_3$$

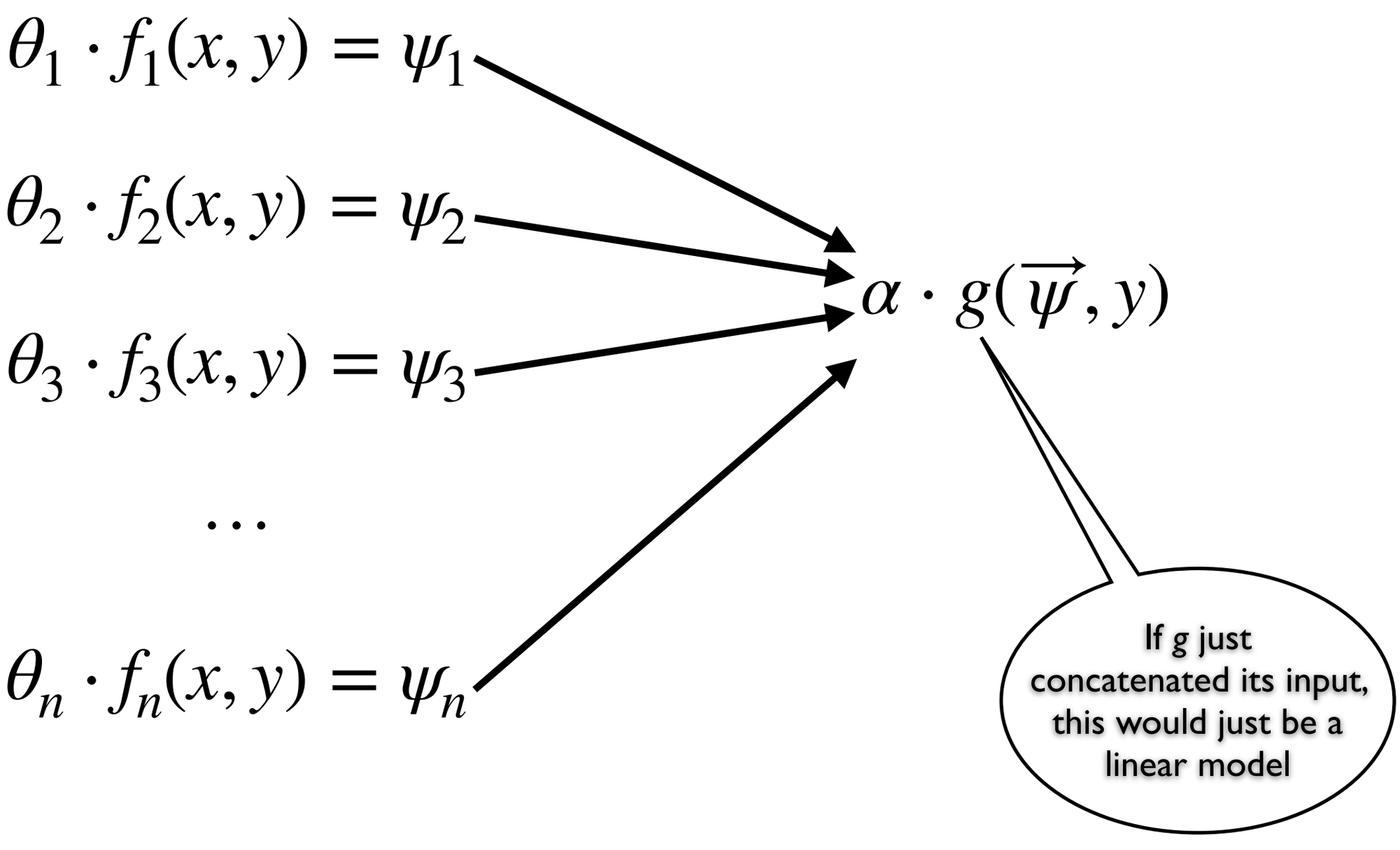
$$\dots$$

$$\theta_n \cdot f_n(x, y) = \psi_n$$

Ensemble Classifiers



Ensemble Classifiers



Stacked Classifiers

Stacked Classifiers

$$\theta_1 \cdot f_1(x, y) = \psi_1$$

Stacked Classifiers

$$\theta_1 \cdot f_1(x, y) = \psi_1$$

$$\theta_2 \cdot f_2(x, y) + \alpha_2 \cdot g(\psi_1, y) = \psi_2$$

Stacked Classifiers

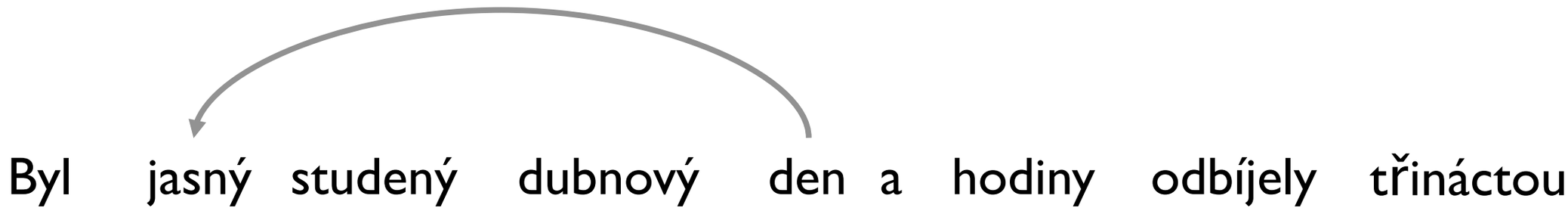
$$\theta_1 \cdot f_1(x, y) = \psi_1$$

$$\theta_2 \cdot f_2(x, y) + \alpha_2 \cdot g(\psi_1, y) = \psi_2$$

$$\ldots$$

Stacked Classifiers

- Example: dependency parsing
  - i.e., predict directed edges

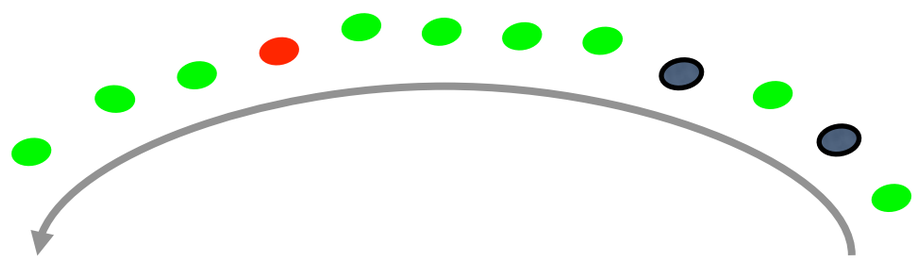


"It was a bright cold day in April and the clocks were striking thirteen"

Stacked Classifiers

- Example: dependency parsing
  - i.e., predict directed edges

score edge w/a linear model

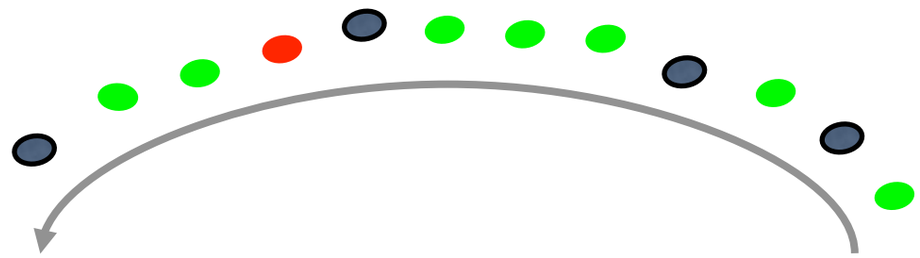


Byl jasný studený dubnový den a hodiny odbíjely třináctou

"It was a bright cold day in April and the clocks were striking thirteen"

Stacked Classifiers

- Example: dependency parsing
  - i.e., predict directed edges

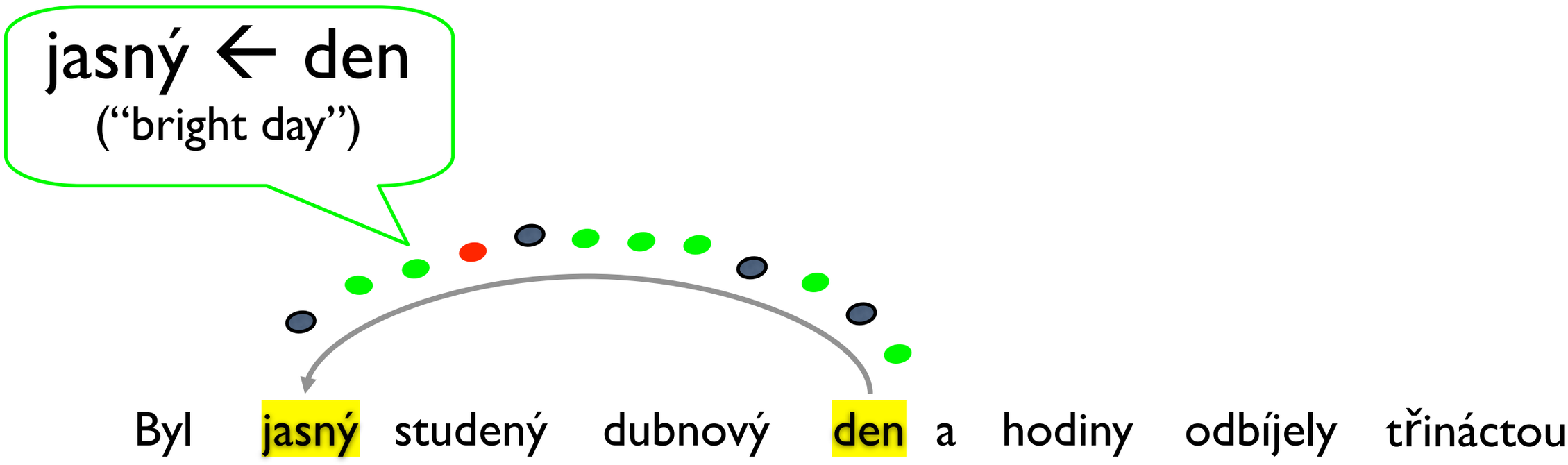


Byl jasný studený dubnový den a hodiny odbíjely třináctou

"It was a bright cold day in April and the clocks were striking thirteen"

Stacked Classifiers

- Example: dependency parsing
  - i.e., predict directed edges

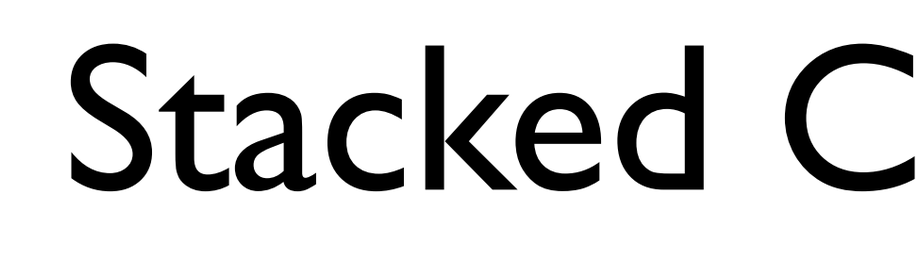


"It was a bright cold day in April and the clocks were striking thirteen"

Stacked Classifiers

- Example: dependency parsing
  - i.e., predict directed edges using PoS tags

jasný ← den
("bright day")



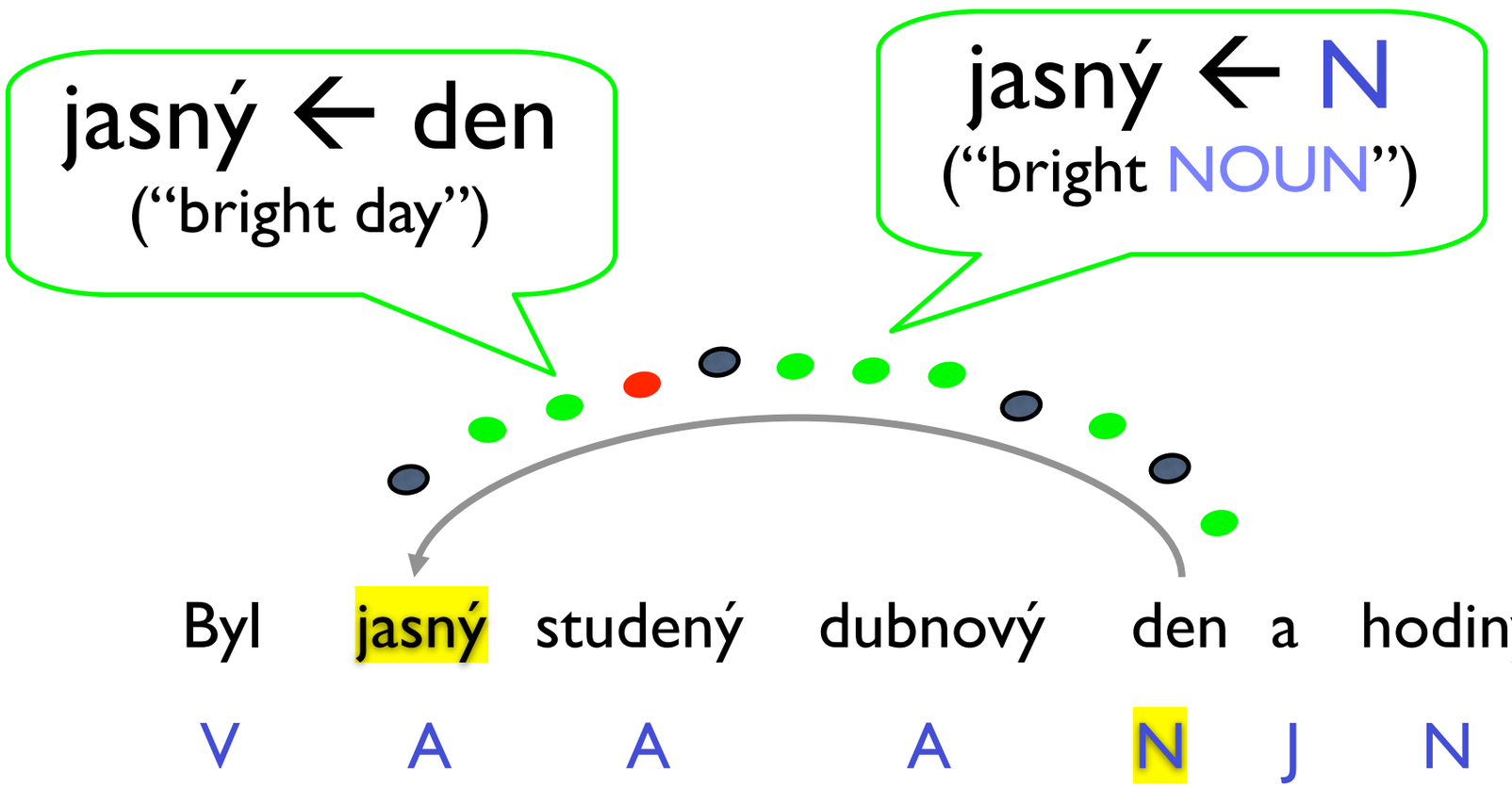
Byl jasný studený dubnový den a hodiny odbíjely třináctou

V A A A N J N V C

"It was a bright cold day in April and the clocks were striking thirteen"

Stacked Classifiers

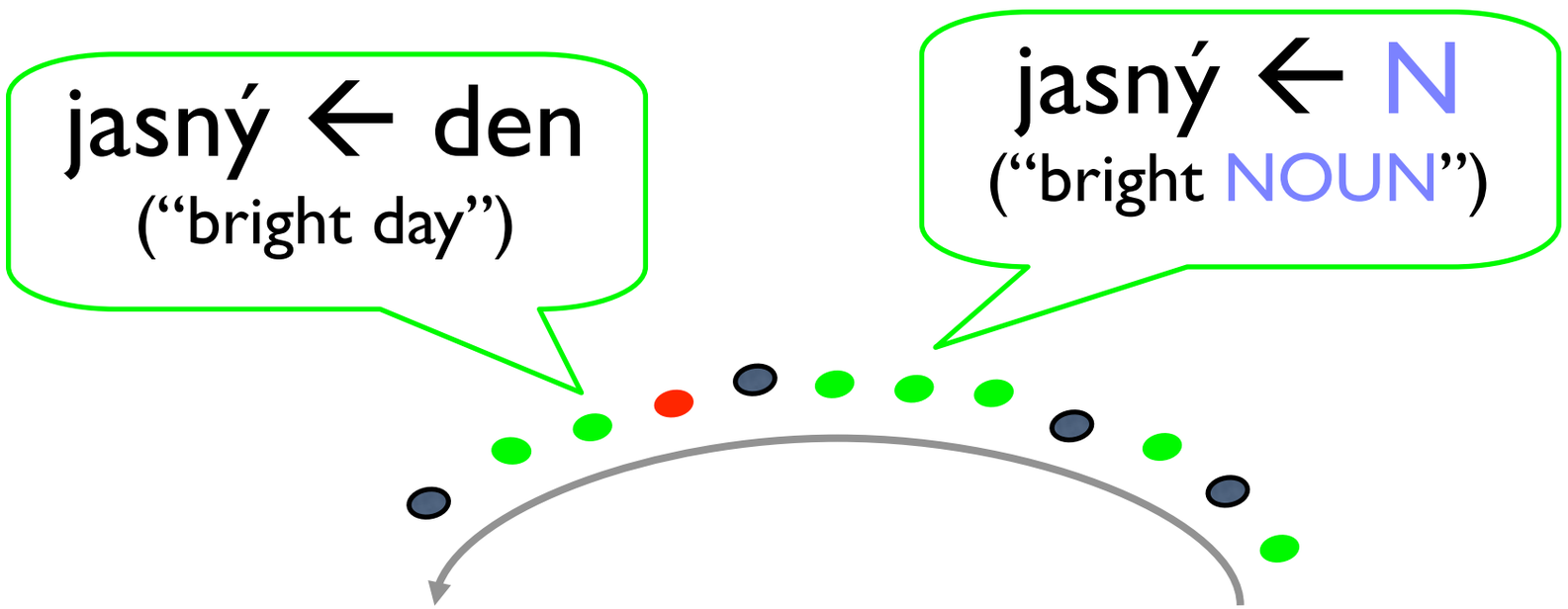
- Example: dependency parsing
  - i.e., predict directed edges using PoS tags



"It was a bright cold day in April and the clocks were striking thirteen"

Stacked Classifiers

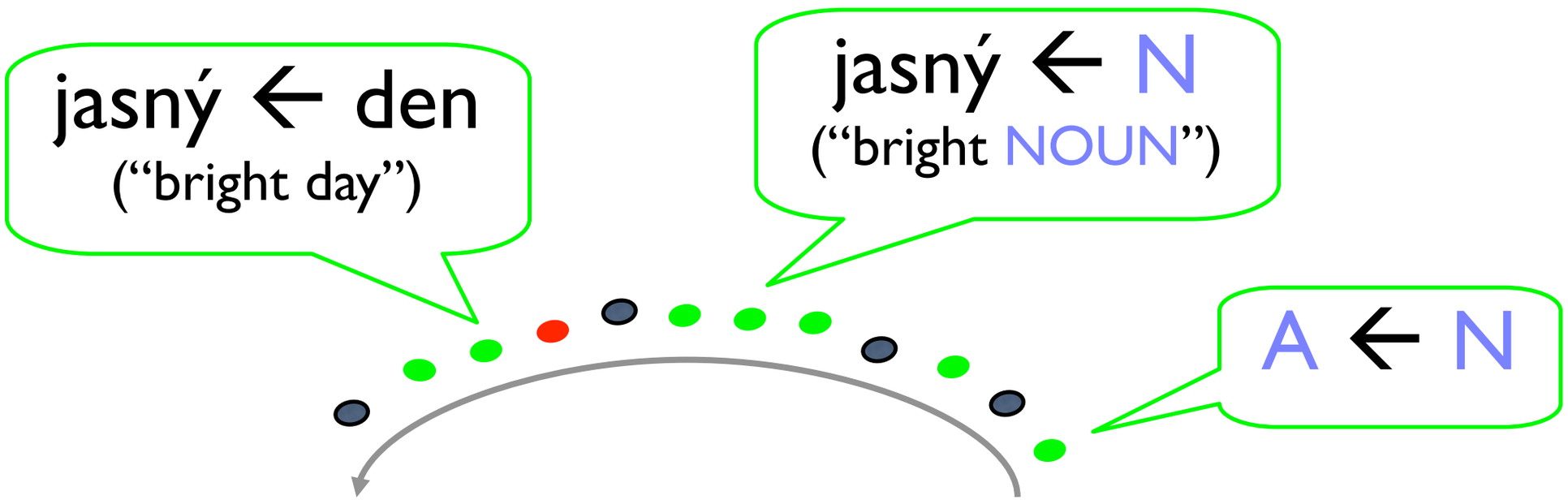
- Example: dependency parsing
  - i.e., predict directed edges using PoS tags



"It was a bright cold day in April and the clocks were striking thirteen"

Stacked Classifiers

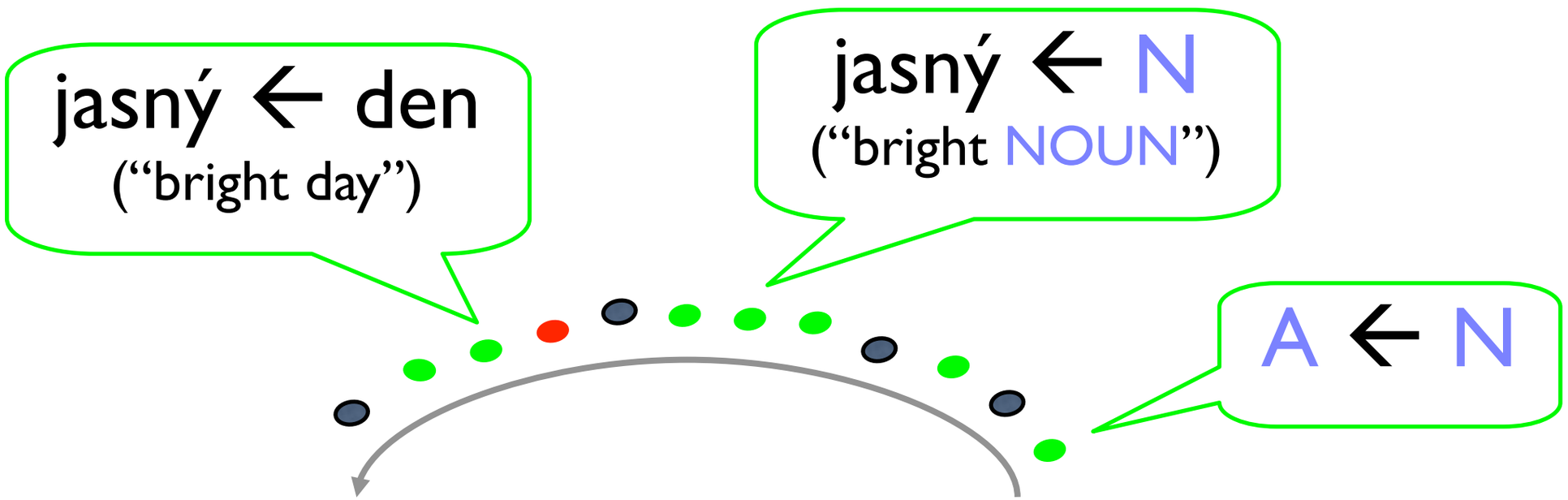
- Example: dependency parsing
  - i.e., predict directed edges using PoS tags



"It was a bright cold day in April and the clocks were striking thirteen"

Stacked Classifiers

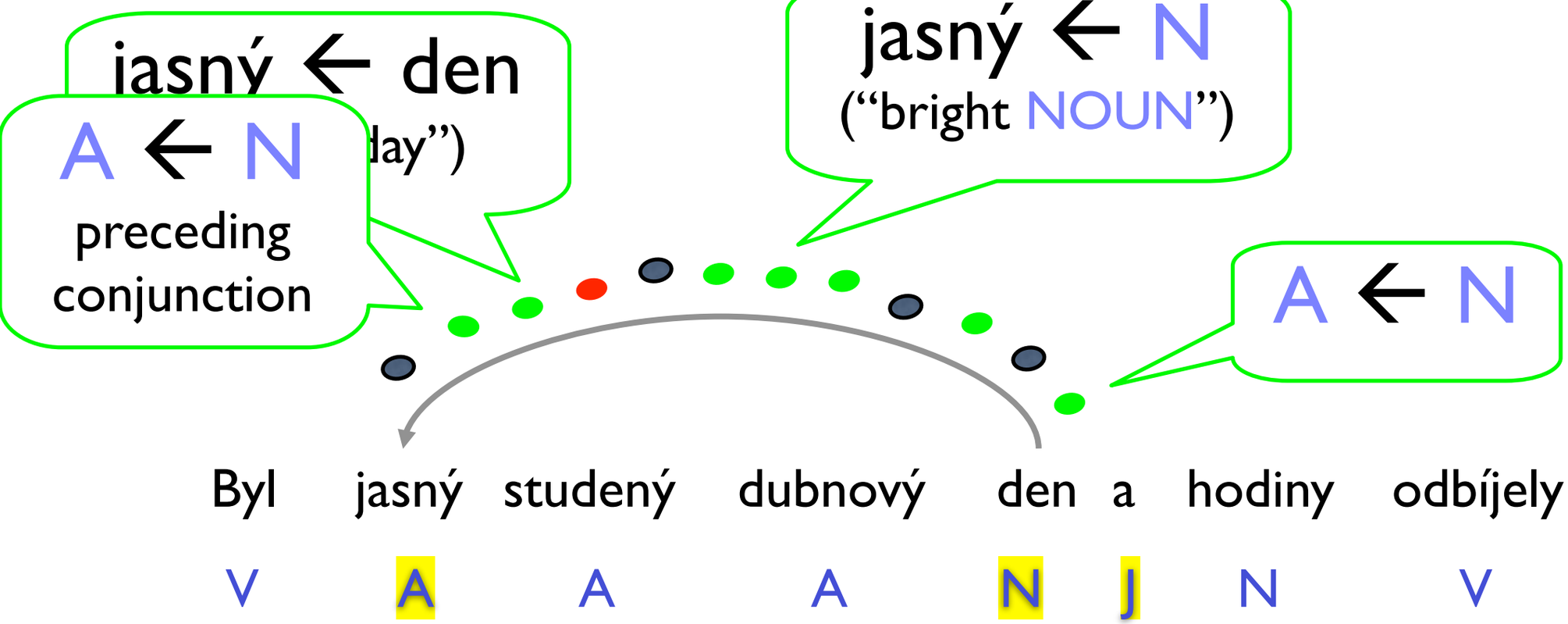
- Example: dependency parsing
  - i.e., predict directed edges using PoS tags



"It was a bright cold day in April and the clocks were striking thirteen"

Stacked Classifiers

- Example: dependency parsing
  - i.e., predict directed edges using PoS tags



“It was a bright cold day in April and the clocks were striking thirteen”

Why Nonlinear Models?

- Theoretical advances in **deep learning**: gradient-based optimization of complicated computational graphs
- In NLP, **word embeddings** give a distributed (non-one-hot) representation of words
- Practical advances in **GPU** hardware and other systems issues

Other Nonlinear Models?

- **Kernel methods** generalize nearest-neighbor classifiers, project data into higher dimensions
- **Decision trees and forests**
- **Boosting** and other ensemble methods
- Not covered in this course

A Stacked Classifier

- Predict features **z** from input **x** w/LogReg

$$\Pr(z_k = 1 \mid \boldsymbol{x}; \boldsymbol{\Theta}^{(x \to z)}) = \sigma(\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x}) = (1 + \exp(-\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x}))^{-1}$$

- Predict output **y** from features **z** w/LogReg

$$\Pr(y = j \mid \boldsymbol{z}; \boldsymbol{\Theta}^{(z \to y)}, \boldsymbol{b}) = \frac{\exp(\boldsymbol{\theta}_j^{(z \to y)} \cdot \boldsymbol{z} + b_j)}{\sum_{j' \in \mathcal{Y}} \exp(\boldsymbol{\theta}_{j'}^{(z \to y)} \cdot \boldsymbol{z} + b_{j'})}$$

A Stacked Classifier

- Predict features **z** from input **x** w/LogReg

$$\Pr(z_k = 1 \mid \boldsymbol{x}; \boldsymbol{\Theta}^{(x \to z)}) = \sigma(\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x}) = (1 + \exp(-\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x}))^{-1}$$

- Predict output **y** from features **z** w/LogReg

$$\Pr(y = j \mid \boldsymbol{z}; \boldsymbol{\Theta}^{(z \to y)}, \boldsymbol{b}) = \frac{\exp(\boldsymbol{\theta}_j^{(z \to y)} \cdot \boldsymbol{z} + b_j)}{\sum_{j' \in \mathcal{Y}} \exp(\boldsymbol{\theta}_{j'}^{(z \to y)} \cdot \boldsymbol{z} + b_{j'})}$$

$$\mathrm{p}(\boldsymbol{y} \mid \boldsymbol{z}; \boldsymbol{\Theta}^{(z \to y)}, \boldsymbol{b}) = \mathrm{SoftMax}(\boldsymbol{\Theta}^{(z \to y)} \boldsymbol{z} + \boldsymbol{b})$$

Stacking w/Hidden **z**

- Use input text **x** to predict *probability* of **z**

$$\boldsymbol{z} = \sigma(\boldsymbol{\Theta}^{(x \rightarrow z)} \boldsymbol{x})$$

- Use continuous **z** to predict output **y**

$$\mathrm{p}(y \mid \boldsymbol{x}; \boldsymbol{\Theta}^{(z \rightarrow y)}, \boldsymbol{b}) = \mathrm{SoftMax}(\boldsymbol{\Theta}^{(z \rightarrow y)} \boldsymbol{z} + \boldsymbol{b})$$

One Linear Model

1	0.0001	a
2	-0.00063	aardvark
3	0.047	able
…	…	…
V	0.000004	zyxt

$$\theta_1^{(x \to z)}$$

Many Linear Models

1	0.0001	…	0.0026	a
2	-0.00063	…	-0.004	aardvark
3	0.047	…	0.078	able
…	…	…	…	…
V	0.000004	…	-0.0000293	zyxt
	$\theta_1^{(x \to z)}$		$\theta_{K_z}^{(x \to z)}$	

Transposed Parameter Matrix

	1	2	3	…	V
$\theta_1^{(x\to z)}$	0.0001	-0.00063	0.047	…	0.000004
	…	…	…	…	…
$\theta_{K_z}^{(x\to z)}$	0.0026	-0.004	0.078	…	-0.0000293
	a	aardvark	able	…	zyxt

$$\Theta^{(x\to z)}$$

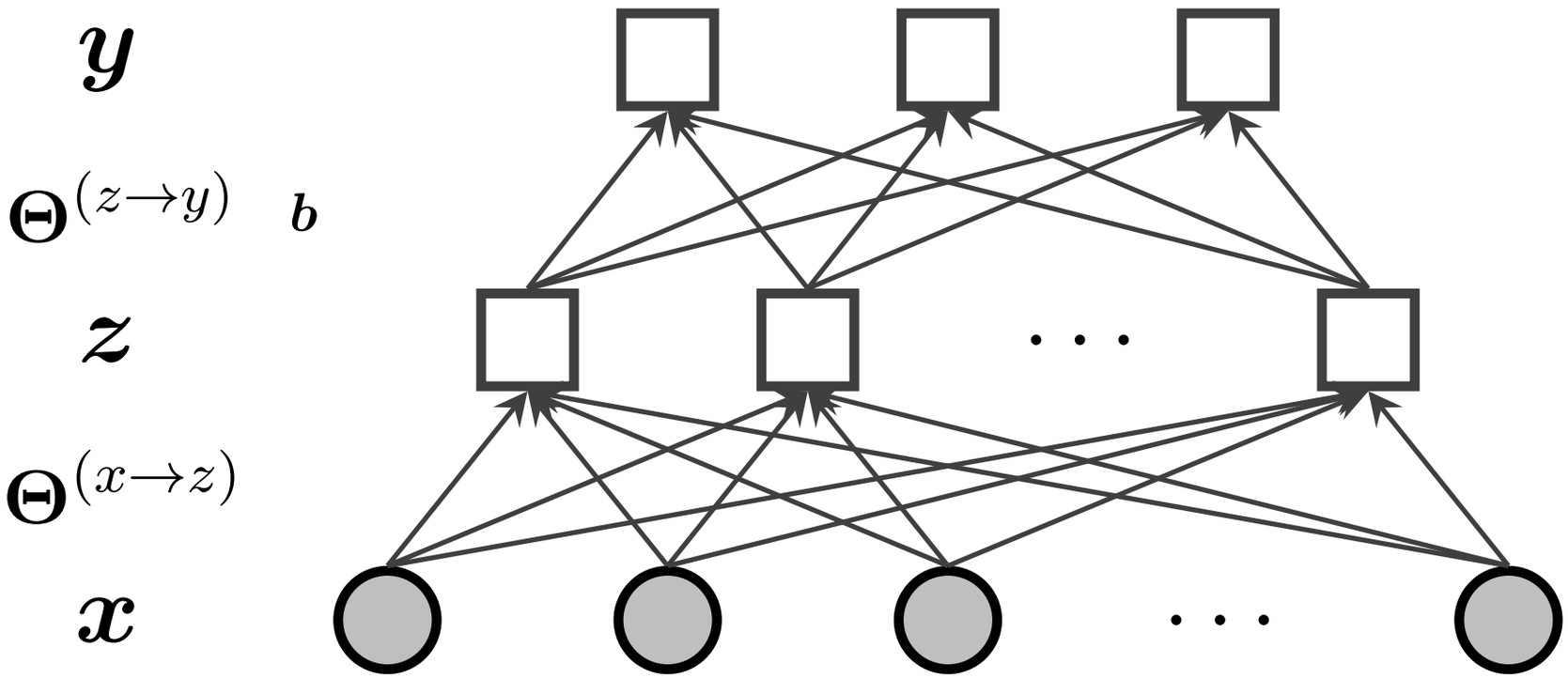
Transposed Parameter Matrix

	1	2	3	…	V
$\theta_1^{(x\to z)}$	0.0001	-0.00063	0.047	…	0.000004
	…	…	…	…	…
$\theta_{K_z}^{(x\to z)}$	0.0026	-0.004	0.078	…	-0.0000293
	a	aardvark	able	…	zyxt

Word feature columns interpretable as **word embeddings**

$$\Theta^{(x\to z)}$$

A Feedforward Neural Network!



$$z = \sigma(\Theta^{(x \to z)} x)$$

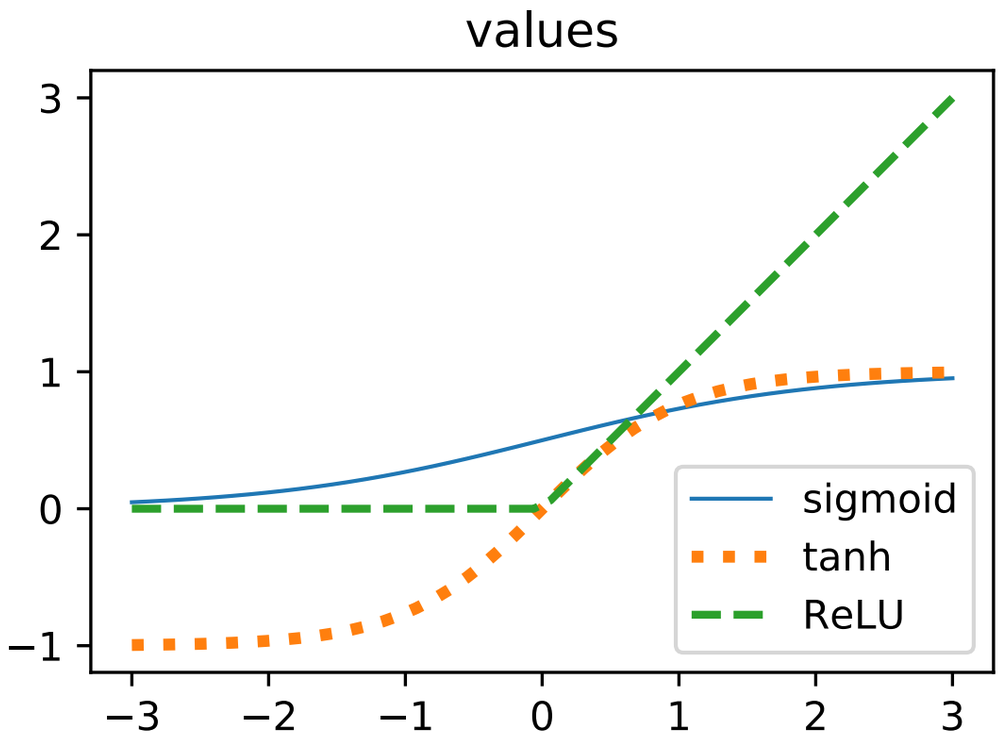
$$\mathrm{p}(y \mid x; \Theta^{(z \to y)}, b) = \mathrm{SoftMax}(\Theta^{(z \to y)} z + b)$$

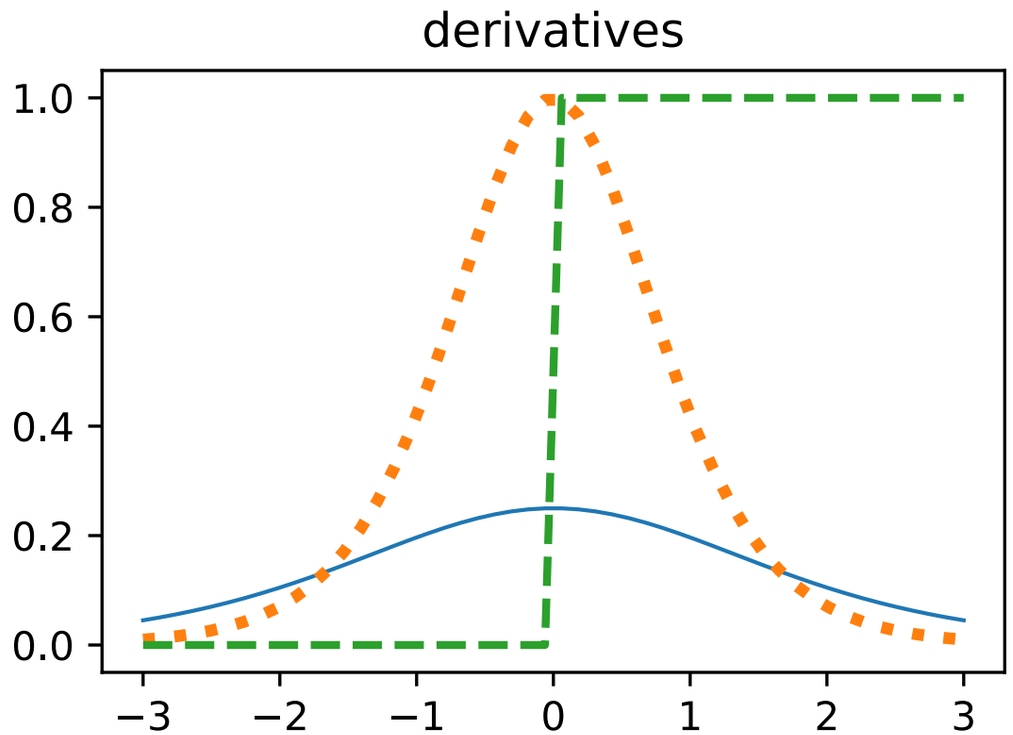
A *universal function approximator* given arbitrarily wide z, but deeper might be easier to learn than wider.

What ~~Classifiers~~ Functions to Stack?

$$\boldsymbol{z} = \sigma(\boldsymbol{\Theta}^{(x \to z)} \boldsymbol{x})$$

$$\mathrm{p}(y \mid \boldsymbol{x}; \boldsymbol{\Theta}^{(z \to y)}, \boldsymbol{b}) = \mathrm{SoftMax}(\boldsymbol{\Theta}^{(z \to y)} \boldsymbol{z} + \boldsymbol{b})$$

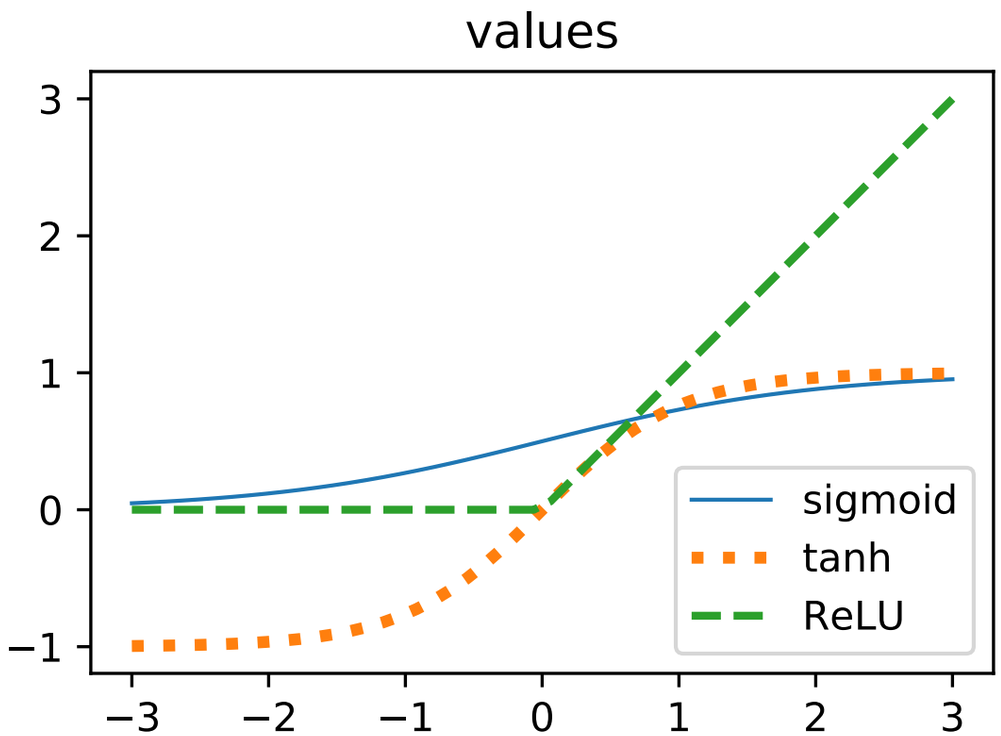


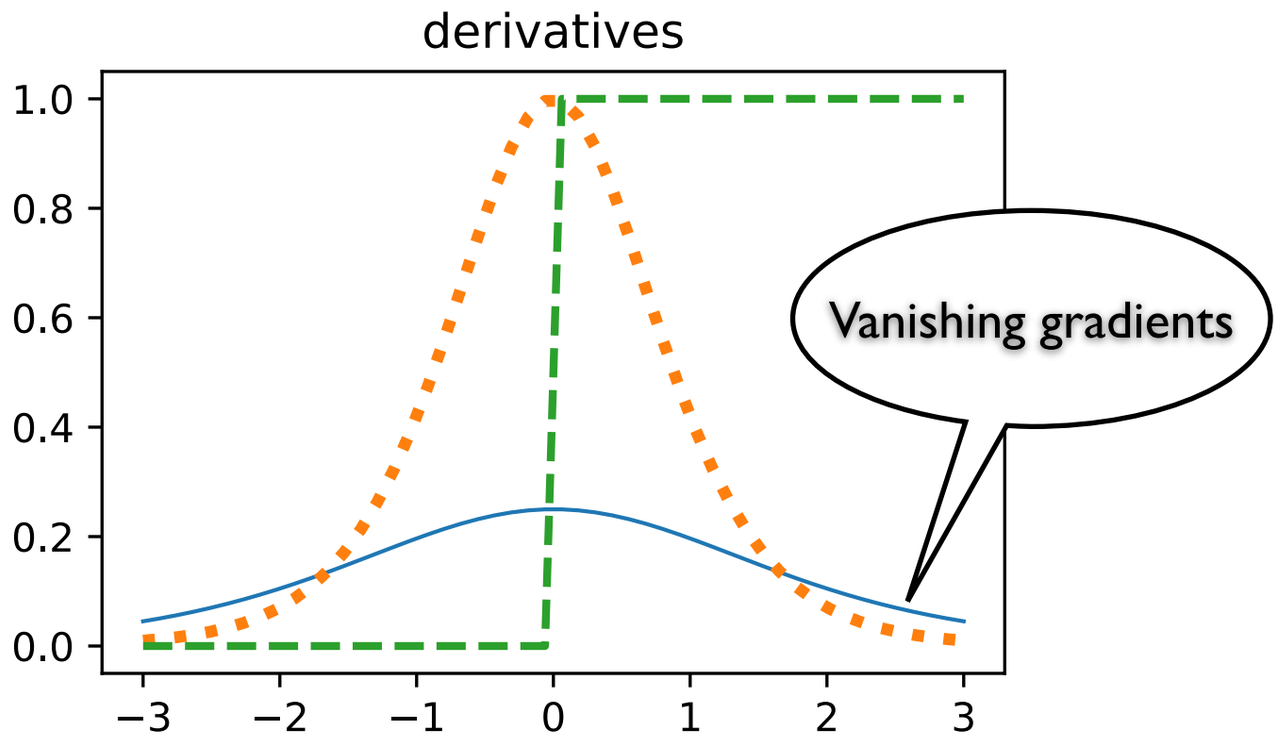


What ~~Classifiers~~ Functions to Stack?

$$\boldsymbol{z} = \sigma(\boldsymbol{\Theta}^{(x \to z)} \boldsymbol{x})$$

$$\mathrm{p}(y \mid \boldsymbol{x}; \boldsymbol{\Theta}^{(z \to y)}, \boldsymbol{b}) = \mathrm{SoftMax}(\boldsymbol{\Theta}^{(z \to y)} \boldsymbol{z} + \boldsymbol{b})$$

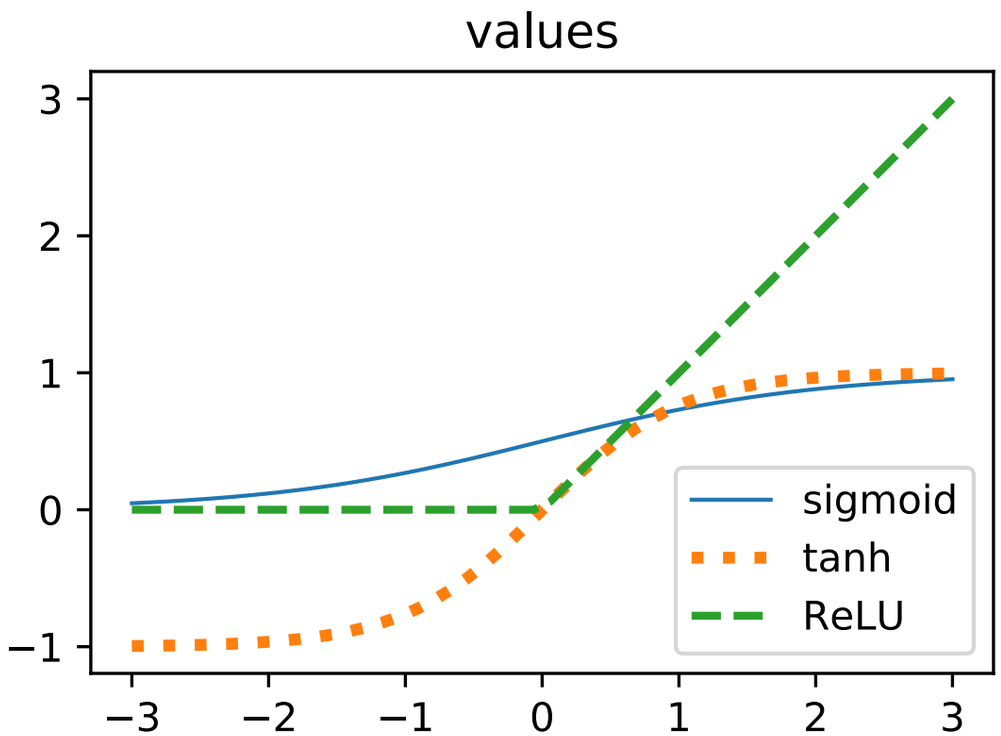


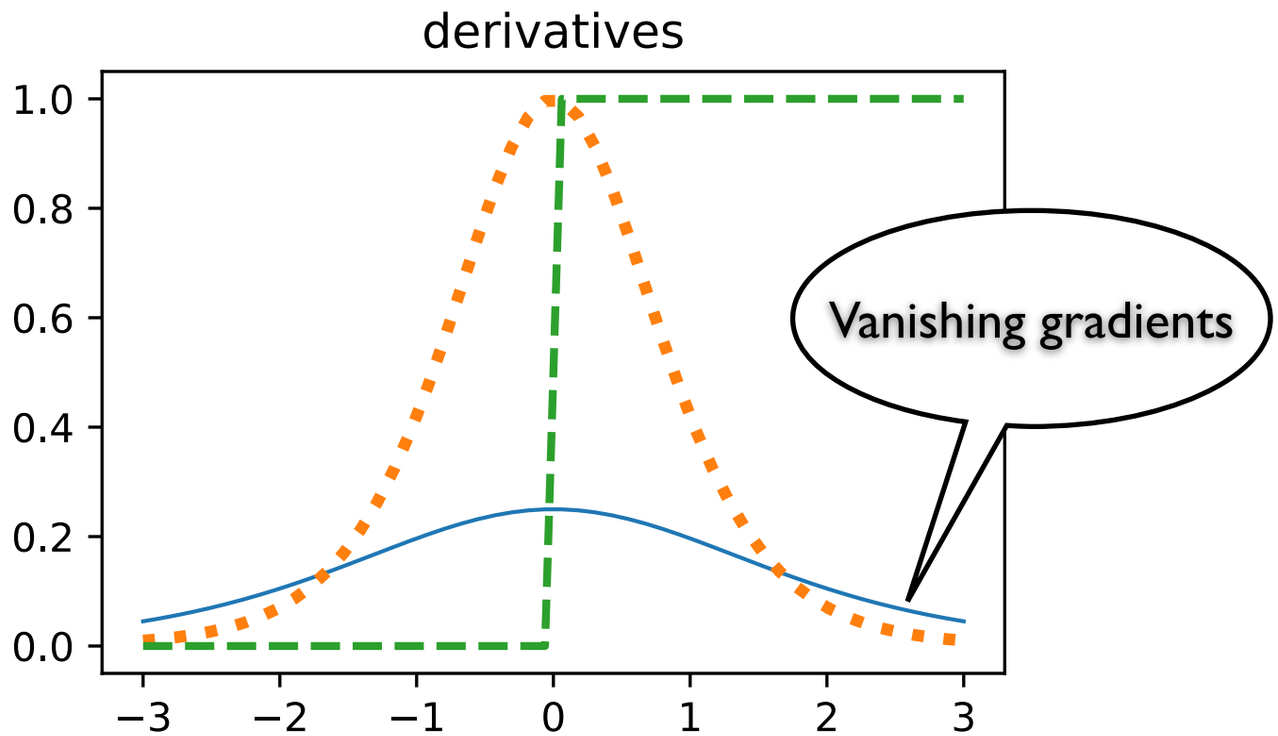


What ~~Classifiers~~ Functions to Stack?

$$\boldsymbol{z} = \sigma(\boldsymbol{\Theta}^{(x \to z)} \boldsymbol{x})$$

$$\mathrm{p}(y \mid \boldsymbol{x}; \boldsymbol{\Theta}^{(z \to y)}, \boldsymbol{b}) = \mathrm{SoftMax}(\boldsymbol{\Theta}^{(z \to y)} \boldsymbol{z} + \boldsymbol{b})$$



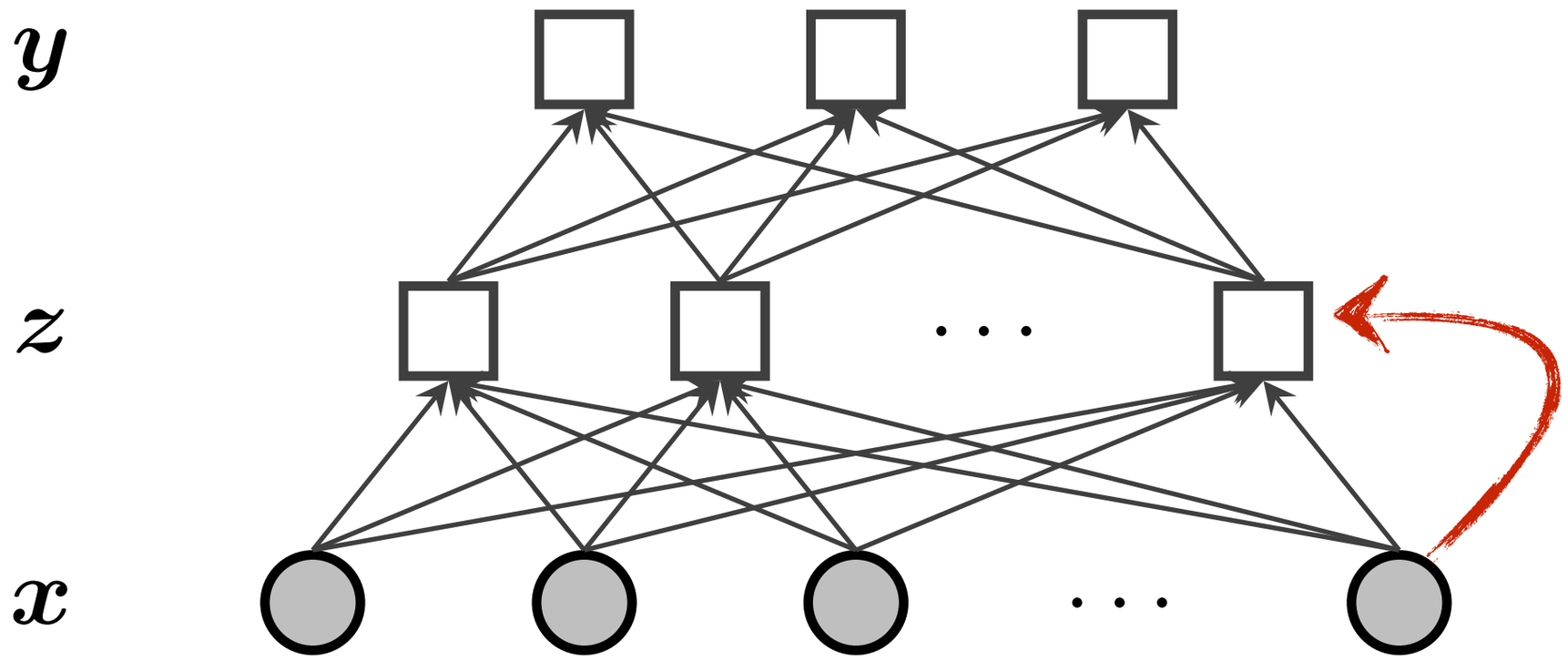


$$\mathrm{ReLU}(a) = \begin{cases} a, & a \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

What ~~Classifiers~~ Functions to Stack?

$$\boldsymbol{z} = \sigma(\boldsymbol{\Theta}^{(x \to z)} \boldsymbol{x})$$

$$\mathrm{p}(y \mid \boldsymbol{x}; \boldsymbol{\Theta}^{(z \to y)}, \boldsymbol{b}) = \mathrm{SoftMax}(\boldsymbol{\Theta}^{(z \to y)} \boldsymbol{z} + \boldsymbol{b})$$



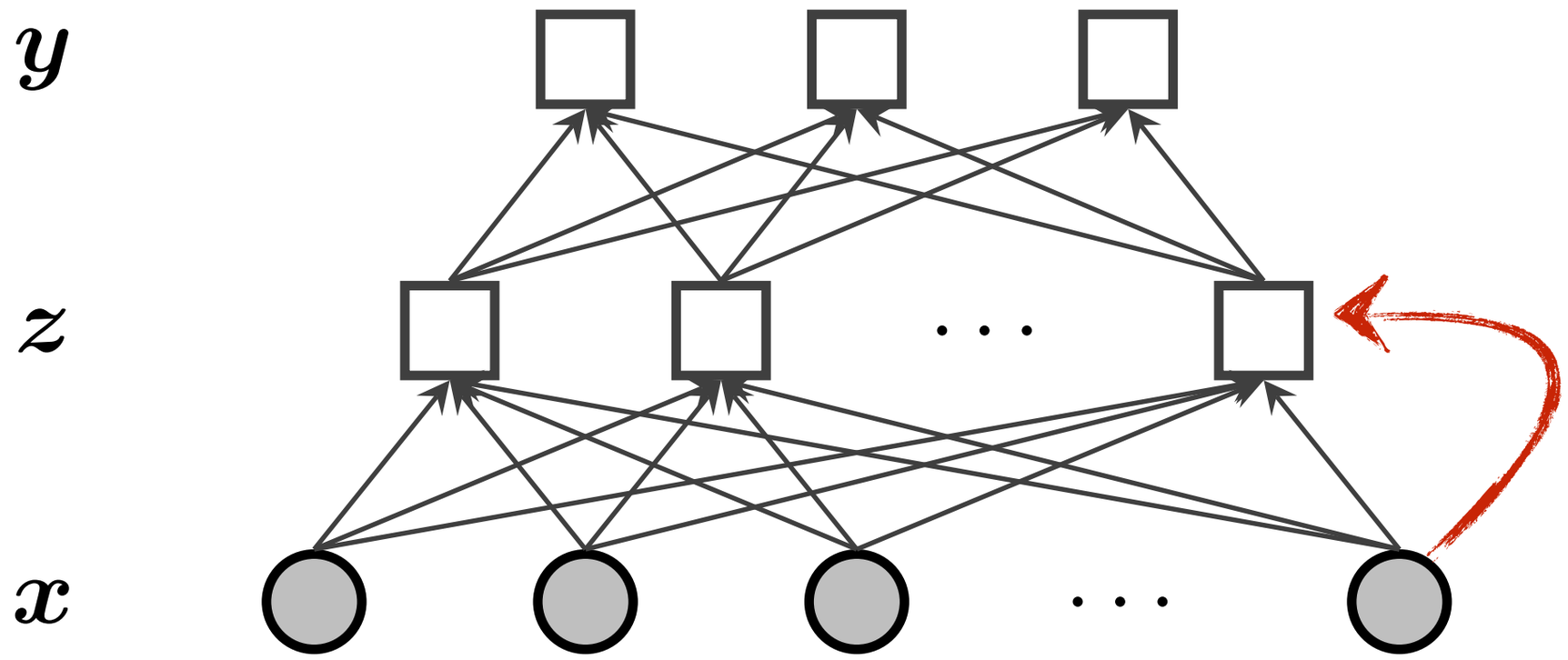
$$\boldsymbol{t} = \sigma(\Theta^{(t)} \boldsymbol{x} + \boldsymbol{b}^{(t)})$$

$$\boldsymbol{z} = \boldsymbol{t} \odot f(\boldsymbol{\Theta}^{(x \to z)} \boldsymbol{x}) + (\mathbf{1} - \boldsymbol{t}) \odot \boldsymbol{x}$$

What ~~Classifiers~~ Functions to Stack?

$$\boldsymbol{z} = \sigma(\boldsymbol{\Theta}^{(x \to z)} \boldsymbol{x})$$

$$\mathrm{p}(y \mid \boldsymbol{x}; \boldsymbol{\Theta}^{(z \to y)}, \boldsymbol{b}) = \mathrm{SoftMax}(\boldsymbol{\Theta}^{(z \to y)} \boldsymbol{z} + \boldsymbol{b})$$



Gating weights

$$\boldsymbol{t} = \sigma(\Theta^{(t)} \boldsymbol{x} + \boldsymbol{b}^{(t)})$$

$$\boldsymbol{z} = \boldsymbol{t} \odot f(\boldsymbol{\Theta}^{(x \to z)} \boldsymbol{x}) + (\mathbf{1} - \boldsymbol{t}) \odot \boldsymbol{x}$$

Loss Functions

Loss Functions

Conditional log-likelihood $$-\mathcal{L} = -\sum_{i=1}^{N} \log \mathrm{p}(y^{(i)} \mid \boldsymbol{x}^{(i)}; \Theta)$$

Loss Functions

Conditional log-likelihood $$-\mathcal{L} = -\sum_{i=1}^{N} \log \mathrm{p}(y^{(i)} \mid \boldsymbol{x}^{(i)}; \Theta)$$

$$\tilde{y}_j \triangleq \Pr(y = j \mid \boldsymbol{x}^{(i)}; \Theta)$$

Cross entropy loss $$-\mathcal{L} = -\sum_{i=1}^{N} \boldsymbol{e}_{y^{(i)}} \cdot \log \tilde{\boldsymbol{y}}$$

Loss Functions

Conditional log-likelihood $$-\mathcal{L} = -\sum_{i=1}^{N} \log \mathrm{p}(y^{(i)} \mid \boldsymbol{x}^{(i)}; \Theta)$$

$$\tilde{y}_j \triangleq \Pr(y = j \mid \boldsymbol{x}^{(i)}; \Theta)$$

Cross entropy loss $$-\mathcal{L} = -\sum_{i=1}^{N} \boldsymbol{e}_{y^{(i)}} \cdot \log \tilde{\boldsymbol{y}}$$

Remember: Entropy $$H(X) = -\sum_{i=1}^{n} p(X = x_i) \lg p(X = x_i)$$

Loss Functions

Conditional log-likelihood $$-\mathcal{L} = -\sum_{i=1}^{N} \log \mathrm{p}(y^{(i)} \mid \boldsymbol{x}^{(i)}; \Theta)$$

$$\tilde{y}_j \triangleq \Pr(y = j \mid \boldsymbol{x}^{(i)}; \Theta)$$

Cross entropy loss $$-\mathcal{L} = -\sum_{i=1}^{N} \boldsymbol{e}_{y^{(i)}} \cdot \log \tilde{\boldsymbol{y}}$$

Remember: Entropy $$H(X) = -\sum_{i=1}^{n} p(X = x_i) \lg p(X = x_i)$$

Regularization/weight decay $$L = \sum_{i=1}^{N} \ell^{(i)} + \lambda_{z \to y} ||\boldsymbol{\Theta}^{(z \to y)}||_F^2 + \lambda_{x \to z} ||\boldsymbol{\Theta}^{(x \to z)}||_F^2$$

Learning the Parameters

(Forward) computation steps

$$\boldsymbol{z} \leftarrow f(\boldsymbol{\Theta}^{(x \to z)} \boldsymbol{x}^{(i)})$$

$$\tilde{\boldsymbol{y}} \leftarrow \text{SoftMax}\left(\boldsymbol{\Theta}^{(z \to y)} \boldsymbol{z} + \boldsymbol{b}\right)$$

$$\ell^{(i)} \leftarrow -\boldsymbol{e}_{y^{(i)}} \cdot \log \tilde{\boldsymbol{y}},$$

Gradient updates

$$\boldsymbol{b} \leftarrow \boldsymbol{b} - \eta^{(t)} \nabla_{\boldsymbol{b}} \ell^{(i)}$$

$$\boldsymbol{\theta}_k^{(z \to y)} \leftarrow \boldsymbol{\theta}_k^{(z \to y)} - \eta^{(t)} \nabla_{\boldsymbol{\theta}_k^{(z \to y)}} \ell^{(i)}$$

$$\boldsymbol{\theta}_k^{(x \to z)} \leftarrow \boldsymbol{\theta}_k^{(x \to z)} - \eta^{(t)} \nabla_{\boldsymbol{\theta}_k^{(x \to z)}} \ell^{(i)},$$

Gradient Updates

$$\boldsymbol{\theta}_k^{(z\to y)} \leftarrow \boldsymbol{\theta}_k^{(z\to y)} - \eta^{(t)} \nabla_{\boldsymbol{\theta}_k^{(z\to y)}} \ell^{(i)}$$

$$\nabla_{\boldsymbol{\theta}_k^{(z\to y)}} \ell^{(i)} = \left[\frac{\partial \ell^{(i)}}{\partial \theta_{k,1}^{(z\to y)}}, \frac{\partial \ell^{(i)}}{\partial \theta_{k,2}^{(z\to y)}}, \ldots, \frac{\partial \ell^{(i)}}{\partial \theta_{k,K_y}^{(z\to y)}} \right]^\top$$

$$\begin{aligned} \frac{\partial \ell^{(i)}}{\partial \theta_{k,j}^{(z\to y)}} &= -\frac{\partial}{\partial \theta_{k,j}^{(z\to y)}} \left(\boldsymbol{\theta}_{y^{(i)}}^{(z\to y)} \cdot \boldsymbol{z} - \log \sum_{y\in\mathcal{Y}} \exp \boldsymbol{\theta}_y^{(z\to y)} \cdot \boldsymbol{z} \right) \\ &= \left(\Pr(y = j \mid \boldsymbol{z}; \boldsymbol{\Theta}^{(z\to y)}, \boldsymbol{b}) - \delta\left(j = y^{(i)}\right) \right) z_k, \end{aligned}$$

Gradient Updates

$$\boldsymbol{\theta}_k^{(x \to z)} \leftarrow \boldsymbol{\theta}_k^{(x \to z)} - \eta^{(t)} \nabla_{\boldsymbol{\theta}_k^{(x \to z)}} \ell^{(i)}$$

$$\begin{aligned}
\frac{\partial \ell^{(i)}}{\partial \theta_{n,k}^{(x \to z)}} &= \frac{\partial \ell^{(i)}}{\partial z_k} \frac{\partial z_k}{\partial \theta_{n,k}^{(x \to z)}} \\
&= \frac{\partial \ell^{(i)}}{\partial z_k} \frac{\partial f(\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x})}{\partial \theta_{n,k}^{(x \to z)}} \\
&= \frac{\partial \ell^{(i)}}{\partial z_k} \times f'(\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x}) \times x_n
\end{aligned}$$

Gradient Updates

$$\boldsymbol{\theta}_k^{(x \to z)} \leftarrow \boldsymbol{\theta}_k^{(x \to z)} - \eta^{(t)} \nabla_{\boldsymbol{\theta}_k^{(x \to z)}} \ell^{(i)}$$

$$\frac{\partial \ell^{(i)}}{\partial \theta_{n,k}^{(x \to z)}} = \frac{\partial \ell^{(i)}}{\partial z_k} \frac{\partial z_k}{\partial \theta_{n,k}^{(x \to z)}}$$

$$= \frac{\partial \ell^{(i)}}{\partial z_k} \frac{\partial f(\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x})}{\partial \theta_{n,k}^{(x \to z)}}$$

$$= \frac{\partial \ell^{(i)}}{\partial z_k} \times f'(\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x}) \times x_n$$

Specifically for logistic (sigmoid)

$$\frac{\partial \ell^{(i)}}{\partial \theta_{n,k}^{(x \to z)}} = \frac{\partial \ell^{(i)}}{\partial z_k} \times \sigma(\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x}) \times (1 - \sigma(\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x})) \times x_n$$

$$= \frac{\partial \ell^{(i)}}{\partial z_k} \times z_k \times (1 - z_k) \times x_n.$$

Gradient Updates

$$\boldsymbol{\theta}_k^{(x \to z)} \leftarrow \boldsymbol{\theta}_k^{(x \to z)} - \eta^{(t)} \nabla_{\boldsymbol{\theta}_k^{(x \to z)}} \ell^{(i)}$$

$$\frac{\partial \ell^{(i)}}{\partial \theta_{n,k}^{(x \to z)}} = \frac{\partial \ell^{(i)}}{\partial z_k} \frac{\partial z_k}{\partial \theta_{n,k}^{(x \to z)}}$$

$$= \frac{\partial \ell^{(i)}}{\partial z_k} \frac{\partial f(\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x})}{\partial \theta_{n,k}^{(x \to z)}}$$

$$= \frac{\partial \ell^{(i)}}{\partial z_k} \times f'(\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x}) \times x_n$$

Specifically for logistic (sigmoid)

$$\frac{\partial \ell^{(i)}}{\partial \theta_{n,k}^{(x \to z)}} = \frac{\partial \ell^{(i)}}{\partial z_k} \times \sigma(\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x}) \times (1 - \sigma(\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x})) \times x_n$$

No update for features at 0

$$= \frac{\partial \ell^{(i)}}{\partial z_k} \times z_k \times (1 - z_k) \times x_n.$$

Gradient Updates

$$\boldsymbol{\theta}_k^{(x \to z)} \leftarrow \boldsymbol{\theta}_k^{(x \to z)} - \eta^{(t)} \nabla_{\boldsymbol{\theta}_k^{(x \to z)}} \ell^{(i)}$$

$$\frac{\partial \ell^{(i)}}{\partial \theta_{n,k}^{(x \to z)}} = \frac{\partial \ell^{(i)}}{\partial z_k} \frac{\partial z_k}{\partial \theta_{n,k}^{(x \to z)}}$$

$$= \frac{\partial \ell^{(i)}}{\partial z_k} \frac{\partial f(\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x})}{\partial \theta_{n,k}^{(x \to z)}}$$

$$= \frac{\partial \ell^{(i)}}{\partial z_k} f'(\boldsymbol{\theta}_k^{(x \to z)} \cdot \boldsymbol{x}) \times x_n$$

Specifically for logistic (sigmoid)

Vanishing gradients

No update for features at 0

$$\frac{\partial \ell^{(i)}}{\partial \theta_{n,k}^{(x \to z)}} = \frac{\partial \ell^{(i)}}{\partial z_k} \times \sigma(\boldsymbol{\theta}^{(x \to z)} \cdot \boldsymbol{x}) \times (1 - \sigma(\boldsymbol{\theta}^{(x \to z)} \cdot \boldsymbol{x})) \times x_n$$

$$= \frac{\partial \ell^{(i)}}{\partial z_k} \times z_k \times (1 - z_k) \times x_n.$$

Backpropagation

Algorithm 6 General backpropagation algorithm. In the computation graph G, every node contains a function f_t and a set of parent nodes $\boldsymbol{\pi}_t$; the inputs to the graph are $\boldsymbol{x}^{(i)}$.

procedure BACKPROP($G = \{f_t, \boldsymbol{\pi}_t\}_{t=1}^{T}\}, \boldsymbol{x}^{(i)}$)
    $v_{t(n)} \leftarrow x_n^{(i)}$ for all n and associated computation nodes $t(n)$.
    for $t \in$ TOPOLOGICALSORT(G) **do** ▷ Forward pass: compute value at each node
        if $|\pi_t| > 0$ **then**
            $v_t \leftarrow f_t(v_{\pi_{t,1}}, v_{\pi_{t,2}}, \ldots, v_{\pi_{t,N_t}})$
    $g_{\text{objective}} = 1$ ▷ Backward pass: compute gradients at each node
    for $t \in$ REVERSE(TOPOLOGICALSORT(G)) **do**
        $g_t \leftarrow \sum_{t':t \in \boldsymbol{\pi}_{t'}} g_{t'} \times \nabla_{v_t} v_{t'}$ ▷ Sum over all t' that are children of t, propagating the gradient $g_{t'}$, scaled by the local gradient $\nabla_{v_t} v_{t'}$
    return $\{g_1, g_2, \ldots, g_T\}$

Backpropagation

- Automatic differentiation by
  - Program transformation, or
  - Recording *tap*e of forward computations, then reversing and differentiating
- Static vs. dynamic computation graphs
  - E.g., graphs for different length inputs
- Applied both in NN, HMMs, dynamic programs
- Contrast w/forward-mode expectation semiring