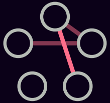


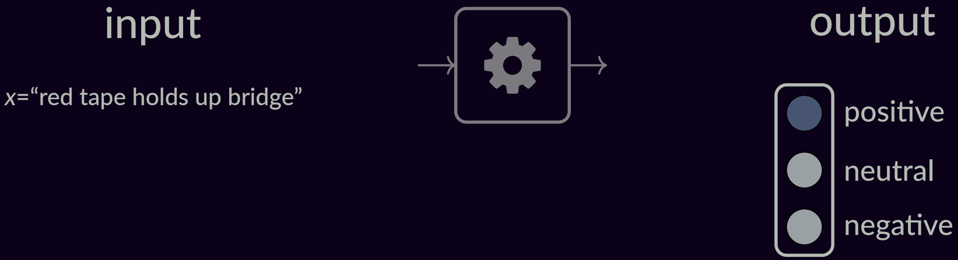
marginalize over sparse distributions when training latent variable models

vlad niculae ltl uva

work with: gonçalo m. correia, wilker aziz, andré martins, mathieu blondel

github.com/deep-spin/sparse-marginalization-lvm https://vene.ro

helpful discrete labels



input

x=“red tape holds up bridge”

output

positive

neutral

negative

classifier: $\Pr(y|x)$

helpful discrete labels

input

x=“red tape holds up bridge”

what if we knew
the newspaper category?

z=

- sports
- government
- technology
- ...

output

- positive
- neutral
- negative

condition on the additional info: $\Pr(y|x, z)$

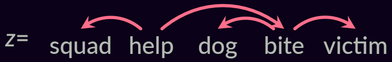
(it is part of the input)

helpful structure

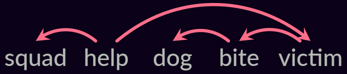
input

x=“squad help dog bite victim”

syntactic analysis



or is it



output

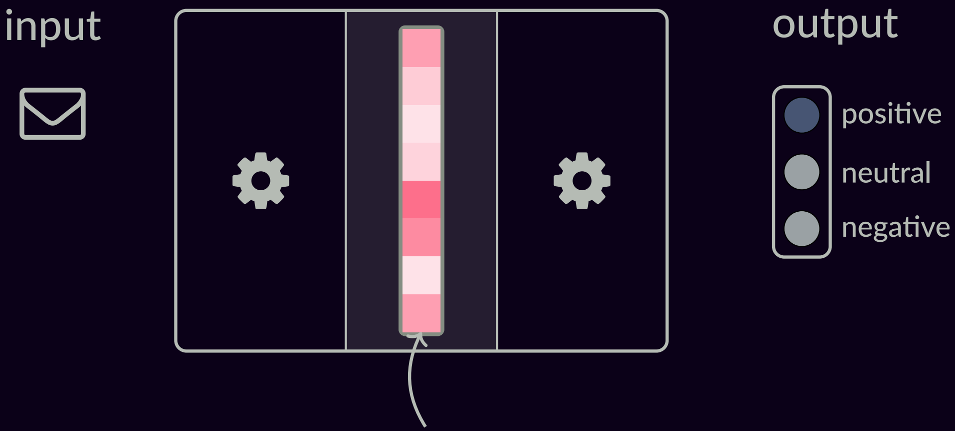
● positive

○ neutral

○ negative

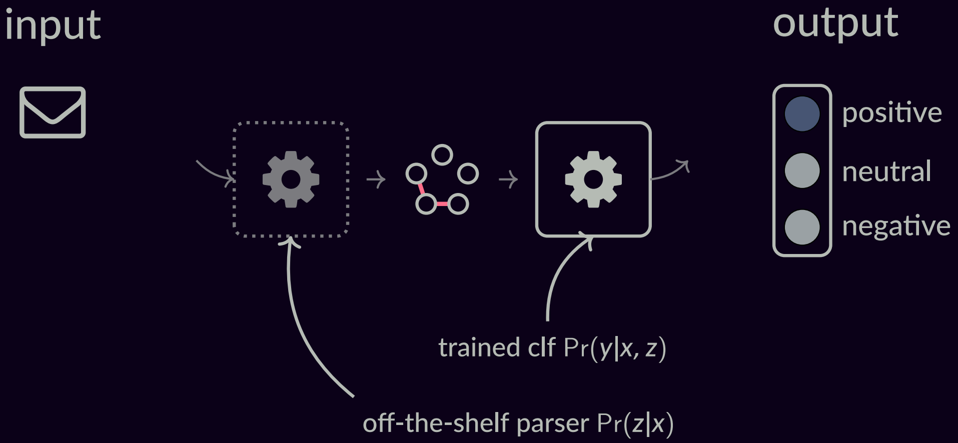
$\Pr(y|x, z).$

deep nets & hope for the best



hope that, somewhere in here,
the ambiguity gets resolved.

pipeline approach



input

output

positive

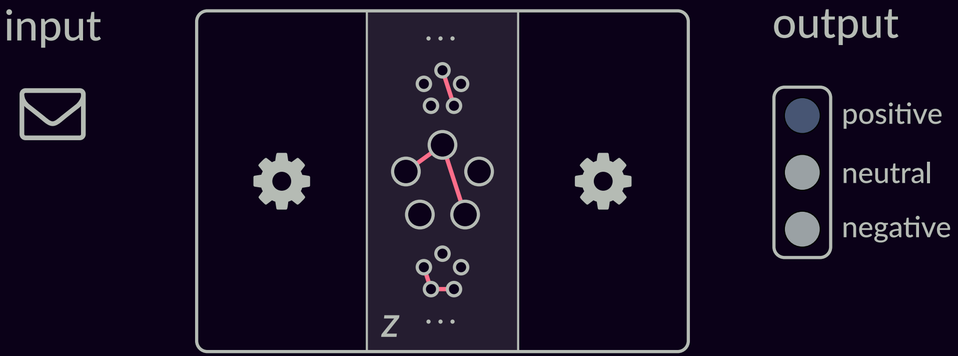
neutral

negative

trained clf $\Pr(y|x, z)$

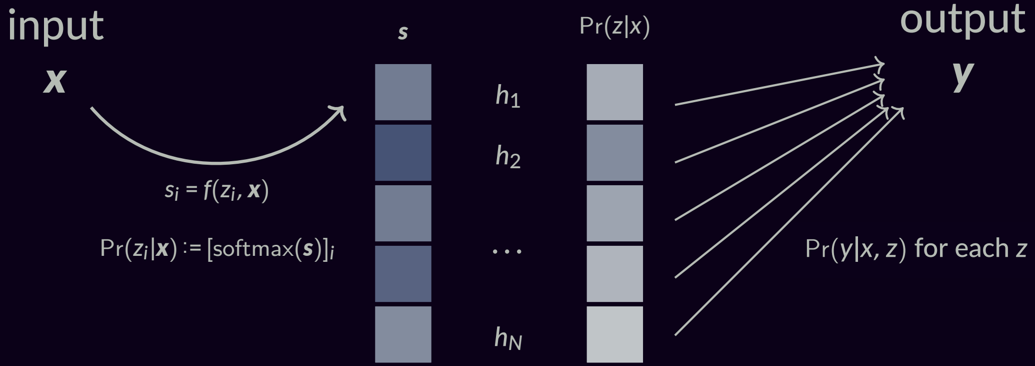
off-the-shelf parser $\Pr(z|x)$

this talk: latent variables



$$\Pr(y|x) = \sum_{z \in \mathcal{Z}} \Pr(z \mid x) \Pr(y \mid x, z).$$

bird's eye view



input

$\boldsymbol{x}$

$s_i = f(z_i, \boldsymbol{x})$

$\Pr(z_i|\boldsymbol{x}) := [\mathrm{softmax}(\boldsymbol{s})]_i$

$\boldsymbol{s}$

h_1

h_2

$\cdots$

h_N

$\Pr(z|x)$

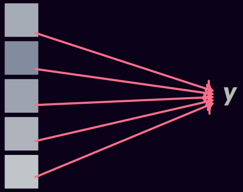
output

$\boldsymbol{y}$

$\Pr(y|x, z)$ for each z

how to learn this

explicit marginalization

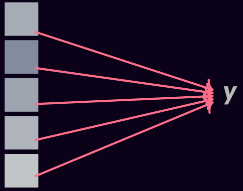


$$\Pr(y|x) = \sum_{z \in \mathcal{Z}} \Pr(z|x) \Pr(y|x, z)$$

exact, but always slow

how to learn this

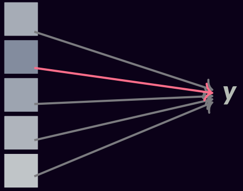
explicit marginalization



$$\Pr(y|x) = \sum_{z \in \mathcal{Z}} \Pr(z|x) \Pr(y|x, z)$$

exact, but always slow

sampling

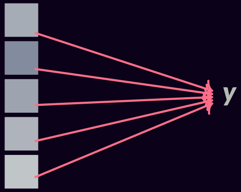


$$\Pr(y|x) = \mathbb{E}_z \Pr(y|x, z)$$
$$\approx \Pr(y|z^+, x)$$

always fast,
but inexact, noisy

how to learn this

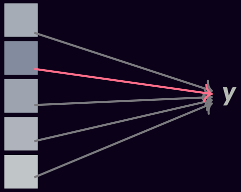
explicit marginalization



$$\Pr(y|x) = \sum_{z \in \mathcal{Z}} \Pr(z|x) \Pr(y|x, z)$$

exact, but always slow

sampling

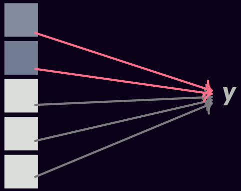


$$\Pr(y|x) = \mathbb{E}_z \Pr(y|x, z)$$

$$\approx \Pr(y|z^+, x)$$

always fast,
but inexact, noisy

this talk: sparse marginalization



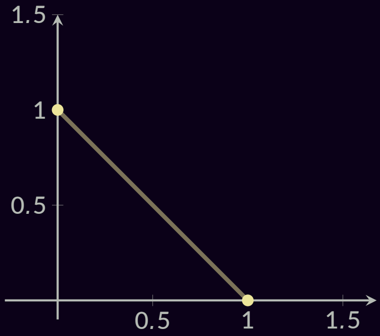
exact and fast,
adaptive acceleration!

the simplex

$$\triangle = \{\boldsymbol{p} \in \mathbb{R}^N : \ \boldsymbol{p} \geq \mathbf{0}, \ \mathbf{1}^\top \boldsymbol{p} = 1\}$$

the simplex

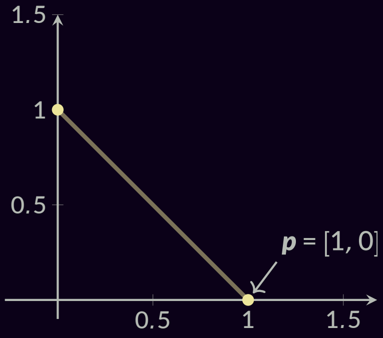
$$\Delta = \{\boldsymbol{p} \in \mathbb{R}^N : \; \boldsymbol{p} \geq \mathbf{0}, \; \mathbf{1}^\top \boldsymbol{p} = 1\}$$



$N = 2$

the simplex

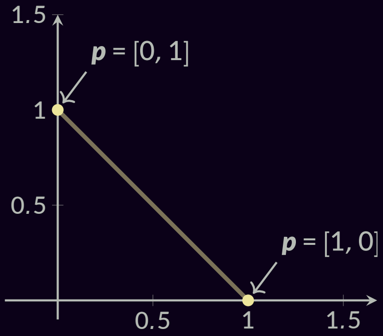
$$\triangle = \{\boldsymbol{p} \in \mathbb{R}^N : \boldsymbol{p} \geq \mathbf{0},\ \mathbf{1}^\top \boldsymbol{p} = 1\}$$



$N = 2$

the simplex

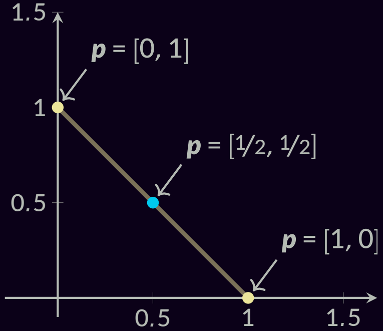
$$\Delta = \{\boldsymbol{p} \in \mathbb{R}^N : \boldsymbol{p} \geq \mathbf{0},\ \mathbf{1}^\top \boldsymbol{p} = 1\}$$



$N = 2$

the simplex

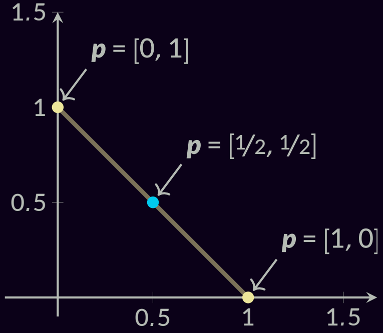
$$\Delta = \{\boldsymbol{p} \in \mathbb{R}^N : \boldsymbol{p} \geq \mathbf{0},\ \mathbf{1}^\top \boldsymbol{p} = 1\}$$



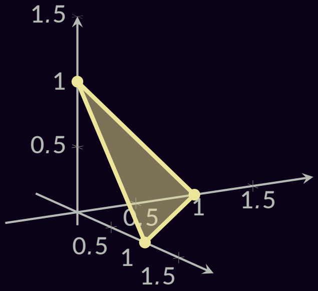
$N = 2$

the simplex

$$\Delta = \{\boldsymbol{p} \in \mathbb{R}^N : \ \boldsymbol{p} \geq \mathbf{0}, \ \mathbf{1}^\top \boldsymbol{p} = 1\}$$



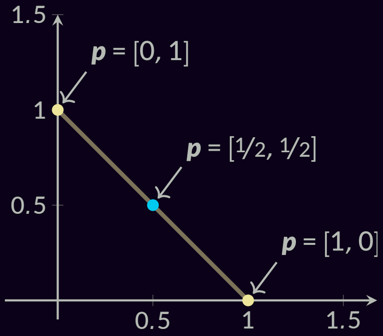
$N = 2$



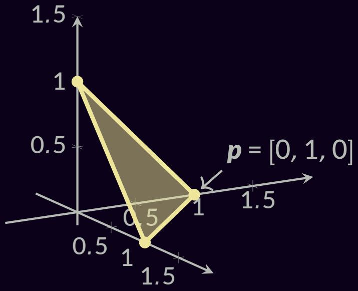
$N = 3$

the simplex

$$\Delta = \{\boldsymbol{p} \in \mathbb{R}^N : \boldsymbol{p} \geq \mathbf{0},\ \mathbf{1}^\top \boldsymbol{p} = 1\}$$



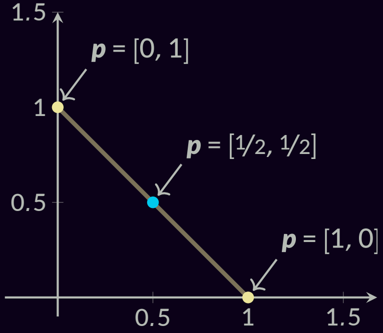
$N = 2$



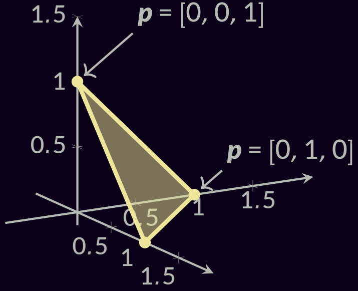
$N = 3$

the simplex

$$\triangle = \{\boldsymbol{p} \in \mathbb{R}^N : \ \boldsymbol{p} \geq \mathbf{0}, \ \mathbf{1}^\top \boldsymbol{p} = 1\}$$



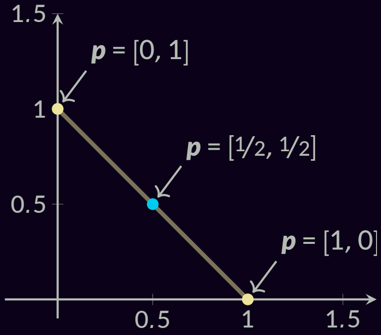
$N = 2$



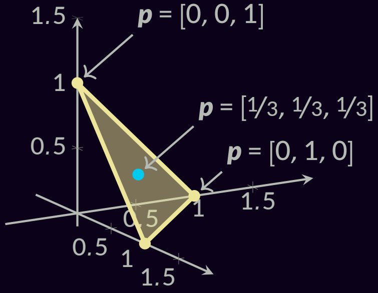
$N = 3$

the simplex

$$\Delta = \{\boldsymbol{p} \in \mathbb{R}^N : \ \boldsymbol{p} \geq \mathbf{0}, \ \mathbf{1}^\top \boldsymbol{p} = 1\}$$



$N = 2$


1.5
$\boldsymbol{p}$ = [0, 0, 1]
1
$\boldsymbol{p}$ = [1⁄3, 1⁄3, 1⁄3]
0.5
$\boldsymbol{p}$ = [0, 1, 0]
1.5
0.5
1
0.5
1
1.5

$N = 3$

second-guessing softmax

the “standard” way to map scores to probabilities
(softmax / gibbs / boltzmann / ... distribution)

$$\Pr(h|x) = \frac{\exp s_i}{\sum_j \exp s_j} > 0$$

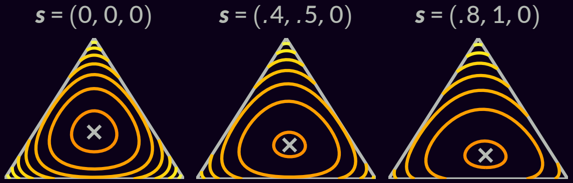
second-guessing softmax

the “standard” way to map scores to probabilities
(softmax / gibbs / boltzmann / ... distribution)

$$\Pr(h|x) = \frac{\exp s_i}{\sum_j \exp s_j} > 0$$

is secretly entropy regularization:

$$\underset{\boldsymbol{p}\in\triangle}{\arg\max}\, \boldsymbol{s}^\top \boldsymbol{p} \underbrace{- \sum_j p_j \log p_j}_{\mathrm{H}(\boldsymbol{p})}$$



softmax

second-guessing softmax

the “standard” way to map scores to probabilities
(softmax / gibbs / boltzmann / ... distribution)

$$\Pr(h|x) = \frac{\exp s_i}{\sum_j \exp s_j} > 0$$

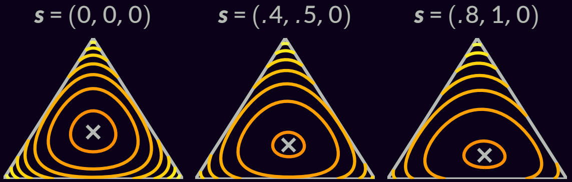
is secretly entropy regularization:

$$\underset{\boldsymbol{p}\in\triangle}{\arg\max}\, \boldsymbol{s}^\top \boldsymbol{p} \underbrace{- \sum_j p_j \log p_j}_{\mathrm{H}(\boldsymbol{p})}$$

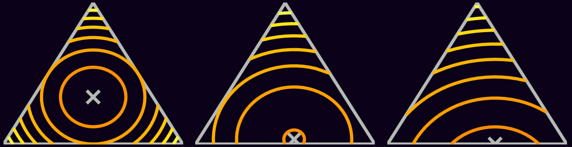
why not try the euclidean norm?

$$\underset{\boldsymbol{p}\in\triangle}{\arg\max}\, \boldsymbol{s}^\top \boldsymbol{p} \underbrace{- {}^1\!/_2 \sum_j p_j^2}_{-{}^1\!/_2 \|p\|_2^2}$$

we have ∼∼∼ sparsity! algorithms! cool name!



softmax



sparsemax (Martins and Astudillo, 2016)

sparsemax

$$\begin{aligned}\mathrm{sparsemax}(\boldsymbol{s}) &= \underset{\boldsymbol{p}\in\triangle}{\arg\max}\, \boldsymbol{p}^\top\boldsymbol{s} - 1/2\|\boldsymbol{p}\|_2^2 \\ &= \underset{\boldsymbol{p}\in\triangle}{\arg\min}\, \|\boldsymbol{p}-\boldsymbol{s}\|_2^2\end{aligned}$$

sparsemax

$$\begin{aligned}\mathrm{sparsemax}(\boldsymbol{s}) &= \underset{\boldsymbol{p}\in\triangle}{\arg\max}\, \boldsymbol{p}^\top \boldsymbol{s} - \tfrac{1}{2}\|\boldsymbol{p}\|_2^2 \\ &= \underset{\boldsymbol{p}\in\triangle}{\arg\min}\, \|\boldsymbol{p} - \boldsymbol{s}\|_2^2\end{aligned}$$

computation:

$$\boldsymbol{p}^\star = [\boldsymbol{s} - \tau\mathbf{1}]_+$$

$$s_i > s_j \Rightarrow p_i \geq p_j$$

expected $\mathcal{O}(d)$ via selection

(Held et al., 1974; Brucker, 1984; Condat, 2016)

sparsemax

$$\begin{aligned}\operatorname{sparsemax}(\boldsymbol{s}) &= \underset{\boldsymbol{p}\in\triangle}{\arg\max}\, \boldsymbol{p}^\top \boldsymbol{s} - 1/2\|\boldsymbol{p}\|_2^2 \\ &= \underset{\boldsymbol{p}\in\triangle}{\arg\min}\, \|\boldsymbol{p} - \boldsymbol{s}\|_2^2\end{aligned}$$

computation:

$$\boldsymbol{p}^\star = [\boldsymbol{s} - \tau\mathbf{1}]_+$$

$$s_i > s_j \Rightarrow p_i \geq p_j$$

expected $\mathcal{O}(d)$ via selection

(Held et al., 1974; Brucker, 1984; Condat, 2016)

backward pass:

$$\boldsymbol{J}_{\text{sparsemax}} = \operatorname{diag}(\boldsymbol{s}) - \frac{1}{|\mathcal{S}|}\boldsymbol{s}\boldsymbol{s}^\top$$

where $\mathcal{S} = \{j : p_j^\star > 0\}$,

$$s_j = [\![j \in \mathcal{S}]\!]$$

(Martins and Astudillo, 2016)

sparsemax

$$\text{sparsemax}(\boldsymbol{s}) = \underset{\boldsymbol{p}\in\triangle}{\arg\max}\, \boldsymbol{p}^\top \boldsymbol{s} - 1/2\|\boldsymbol{p}\|_2^2$$

$$= \underset{\boldsymbol{p}\in\triangle}{\arg\min}\, \|\boldsymbol{p} - \boldsymbol{s}\|_2^2$$

computation:

$\boldsymbol{p}^\star = [\boldsymbol{s} -$

$s_i > s_j \Rightarrow$

expected $\mathcal{O}(d)$

(Held et al., 1974; Brucker, 1984; Condat, 2016)

argmin differentiation

(Colson et al., 2007; Gould et al., 2016)

see also (Amos and Kolter, 2017)

backward pass:

$\text{iag}(\boldsymbol{s}) - \frac{1}{|\mathcal{S}|}\boldsymbol{s}\boldsymbol{s}^\top$

$\{j : p_j^\star > 0\},$

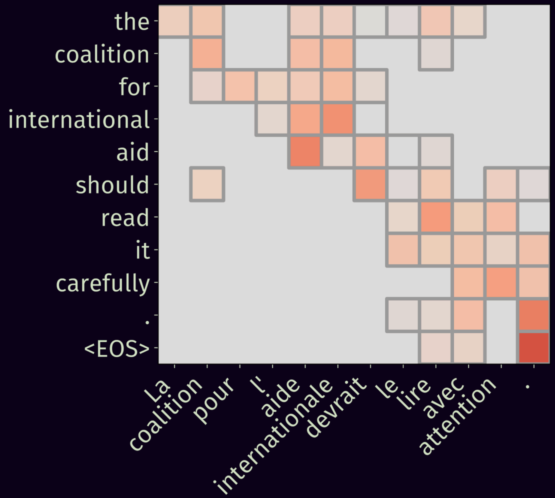
$_j \quad [\![j \in \mathcal{S}]\!]$

(Martins and Astudillo, 2016)

some applications:

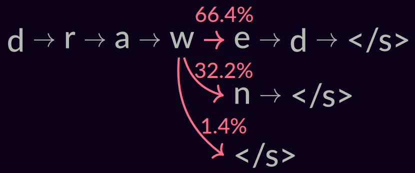
sparse attention

(Martins and Astudillo, 2016; Correia, Niculae, and Martins, 2019)



sparse losses (& seq2seq)

(Blondel et al., 2019; Peters et al., 2019)



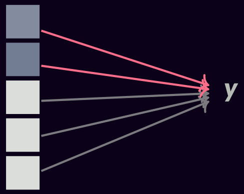
d → r → a → w → e → d → </s>

66.4%

32.2% n → </s>

1.4% </s>

sparsemax enables fast marginalization!



$$
\begin{aligned}
\Pr(y|x) &= \sum_{z\in\mathcal{Z}} \Pr(z|x)\ \Pr(y|z,x) \\
&= \Pr(z_1|x)\ \Pr(y|x,z_1) + \underbrace{\Pr(z_2|x)}_{=0}\ \cancel{\Pr(y|x,z_2)} + \dots \\
&\quad + \Pr(z_i\,|x)\ \Pr(y|x,z_i\,) + \dots + \underbrace{\Pr(z_N|x)}_{=0}\ \cancel{\Pr(y|x,z_N)}
\end{aligned}
$$

saves us from computing $\Pr(y|x,z)$ for many $z \in \mathcal{Z}$!

emergent communication

$$\sum_{z \in \mathcal{Z}} \Pr(z|x) \Pr(x|z, \mathcal{V})$$

- game between two players.
- sender takes x from imagenet,
  and summarizes it in a message z (here: one symbol).
- receiver sees the symbol, and a group of images $\mathcal{V} \ni x$,
  and must pick the intended image.

emergent communication

sender

$$\sum_{z \in \mathcal{Z}} \Pr(z|x) \Pr(x|z, \mathcal{V})$$

receiver

- game between two players.
- sender takes x from imagenet,
  and summarizes it in a message z (here: one symbol).
- receiver sees the symbol, and a group of images $\mathcal{V} \ni x$,
  and must pick the intended image.

emergent communication

sum over
all possible messages
in the vocabulary

$$\sum_{z \in \mathcal{Z}} \Pr(z|x) \Pr(x|z, \mathcal{V})$$

sender

receiver

- game between two players.
- sender takes x from imagenet,
  and summarizes it in a message z (here: one symbol).
- receiver sees the symbol, and a group of images $\mathcal{V} \ni x$,
  and must pick the intended image.

(Lazaridou et al., 2017; Correia, Niculae, Aziz, et al., 2020)

emergent communication

... but make it harder: $|\mathcal{Z}| = 256$, $|\mathcal{V}| = 16$

Method	success (%)	Dec. calls
monte carlo		
sfe	33.05 ±2.84	1
sfe+	44.32 ±2.72	2
nvil	37.04 ±1.61	1
gumbel	23.51 ±16.19	1
st-gumbel	27.42 ±13.36	1
marginalization		

(Lazaridou et al., 2017; Correia, Niculae, Aziz, et al., 2020)

emergent communication

... but make it harder: $|\mathcal{Z}| = 256$, $|\mathcal{V}| = 16$

Method	success (%)	Dec. calls
monte carlo		
sfe	33.05 ±2.84	1
sfe+	44.32 ±2.72	2
nvil	37.04 ±1.61	1
gumbel	23.51 ±16.19	1
st-gumbel	27.42 ±13.36	1
marginalization		
softmax	93.37 ±0.42	256

(Lazaridou et al., 2017; Correia, Niculae, Aziz, et al., 2020)

emergent communication

... but make it harder: $|\mathcal{Z}| = 256$, $|\mathcal{V}| = 16$

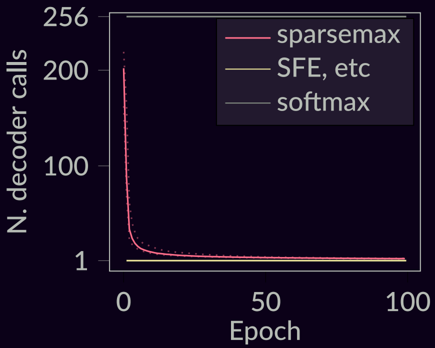
Method	success (%)	Dec. calls
monte carlo		
sfe	33.05 ±2.84	1
sfe+	44.32 ±2.72	2
nvil	37.04 ±1.61	1
gumbel	23.51 ±16.19	1
st-gumbel	27.42 ±13.36	1
marginalization		
softmax	93.37 ±0.42	256
sparsemax	93.35 ±0.50	3.13±0.48

(Lazaridou et al., 2017; Correia, Niculae, Aziz, et al., 2020)

emergent communication

... but make it harder: $|\mathcal{Z}| = 256$, $|\mathcal{V}| = 16$

Method	success (%)	Dec. calls
monte carlo		
sfe	33.05 ±2.84	1
sfe+	44.32 ±2.72	2
nvil	37.04 ±1.61	1
gumbel	23.51 ±16.19	1
st-gumbel	27.42 ±13.36	1
marginalization		
softmax	93.37 ±0.42	256
sparsemax	93.35 ±0.50	3.13±0.48



semi-supervised variational autoencoder

$$\sum_{z \in \mathcal{Z}} \Pr(z|x)\, \ell(x, z)$$

- semi-supervised vae on mnist: z is one of 10 categories
- train with 10% labeled data

semi-supervised variational autoencoder

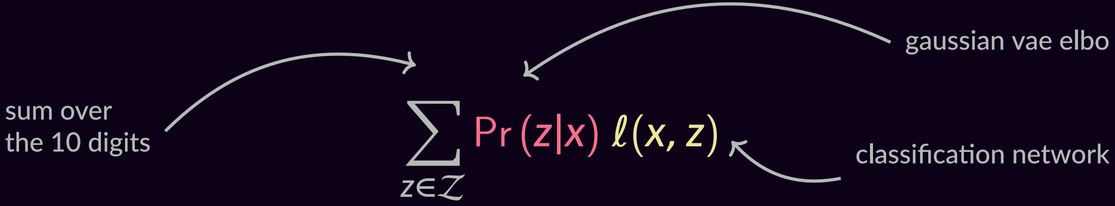
gaussian vae elbo

$$\sum_{z \in \mathcal{Z}} \Pr(z|x)\, \ell(x, z)$$

classification network

- semi-supervised vae on mnist: z is one of 10 categories
- train with 10% labeled data

semi-supervised variational autoencoder



sum over the 10 digits

gaussian vae elbo

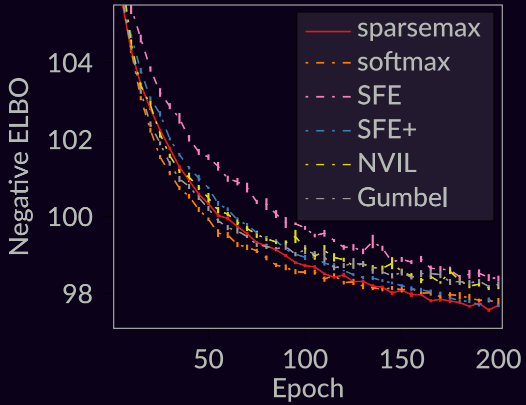
classification network

$$\sum_{z\in\mathcal{Z}} \Pr(z|x)\,\ell(x,z)$$

- semi-supervised vae on mnist: z is one of 10 categories
- train with 10% labeled data

semi-supervised variational autoencoder

method	accuracy (%)	dec. calls
monte carlo		
sfe	94.75±.002	1
sfe+	96.53±.001	2
nvil	96.01±.002	1
gumbel	95.46±.001	1
marginalization		
softmax	96.93±.001	10
sparsemax	96.87±.001	1.01±0.01



limitations

- mostly (and eventually) very sparse.
  worst case: fully dense
- → sparsemax can’t handle structured z

limitations

- mostly (and eventually) very sparse.
  worst case: fully dense
- → sparsemax can't handle structured z

today's solution: top-k sparsemax

$$k\text{-sparsemax}(s) = \underset{\boldsymbol{p}\in\triangle, \|\boldsymbol{p}\|_0\leq k}{\arg\min} \|\boldsymbol{p} - s\|_2^2$$

limitations

- mostly (and eventually) very sparse.
  worst case: fully dense
- → sparsemax can't handle structured z

today's solution: top-k sparsemax

$$k\text{-sparsemax}(s) = \operatorname*{arg\,min}_{\boldsymbol{p}\in\triangle,\, \|\boldsymbol{p}\|_0 \le k} \|\boldsymbol{p} - s\|_2^2$$

- non-convex but easy: sparsemax over the k highest scores (Kyrillidis et al., 2013)
- top-k oracle available for some structured problems.
- certificate: if at least one of the top-k z gets $\Pr(z|x) = 0$, **k-sparsemax = sparsemax**!
  starts with bias, sheds the bias along the way

bit-vector variational autoencoder

$$\sum_{z \in \{0,1\}^D} q(z|x)\, \ell(x, z)$$

bit-vector variational autoencoder

for elbo: $\ell(x, z) = -\log \frac{\Pr(x,z)}{q(z|x)}$

$$\sum_{z \in \{0,1\}^D} q(z|x)\, \ell(x, z)$$

posterior approx / inference network

- vae where z is a collection of D bits

bit-vector variational autoencoder

for elbo: $\ell(x, z) = -\log \frac{\Pr(x,z)}{q(z|x)}$

exponentially large sum

$$\sum_{z\in\{0,1\}^D} q(z|x)\,\ell(x,z)$$

posterior approx / inference network

- vae where z is a collection of D bits

bit-vector vae

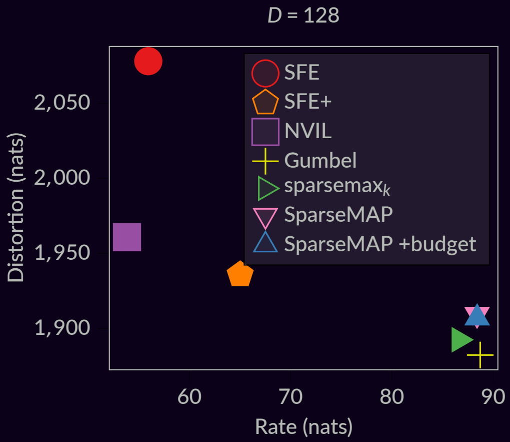
test nll (bits/dim), lower is better

method	D = 32	D = 128
monte carlo		
sfe	3.74	3.77
sfe+	3.61	3.59
nvil	3.65	3.60
gumbel	3.57	3.49
marginalization		
softmax/sparsemax	–	
top-k sparsemax	3.62	3.61

bit-vector vae

test nll (bits/dim), lower is better

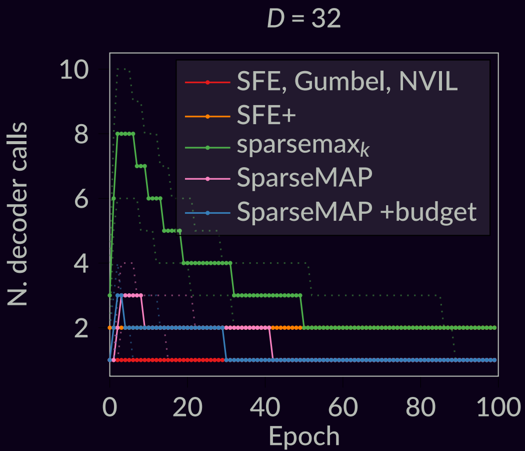
method	*D* = 32	*D* = 128
monte carlo		
sfe	3.74	3.77
sfe+	3.61	3.59
nvil	3.65	3.60
gumbel	3.57	3.49
marginalization		
softmax/sparsemax	–	
top-*k* sparsemax	3.62	3.61



bit-vector vae

test nll (bits/dim), lower is better

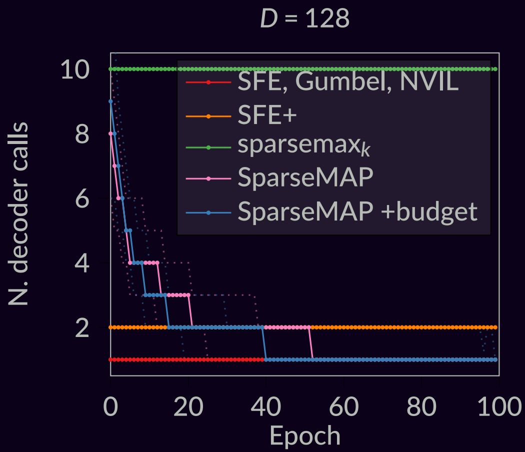
method	D = 32	D = 128
monte carlo		
sfe	3.74	3.77
sfe+	3.61	3.59
nvil	3.65	3.60
gumbel	3.57	3.49
marginalization		
softmax/sparsemax	–	
top-*k* sparsemax	3.62	3.61



bit-vector vae

test nll (bits/dim), lower is better

method	*D* = 32	*D* = 128
monte carlo		
sfe	3.74	3.77
sfe+	3.61	3.59
nvil	3.65	3.60
gumbel	3.57	3.49
marginalization		
softmax/sparsemax	–	
top-*k* sparsemax	3.62	3.61



take home message

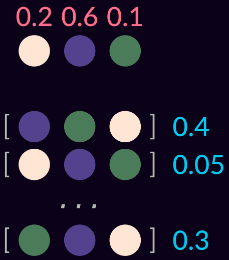
marginalize over sparse distributions
when training latent variable models

github.com/deep-spin/sparse-marginalization-lvm 🏠 vene.ro

take home message

marginalize over sparse distributions
when training latent variable models

discrete and structured

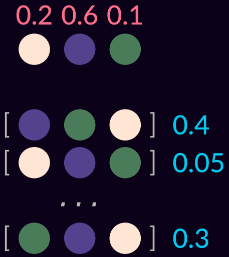


github.com/deep-spin/sparse-marginalization-lvm vene.ro

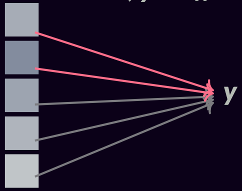
take home message

marginalize over sparse distributions
when training latent variable models

discrete and structured



deterministic, yet efficient

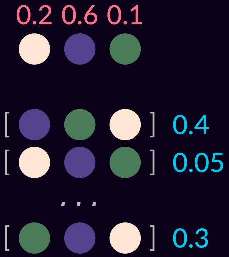


github.com/deep-spin/sparse-marginalization-lvm vene.ro

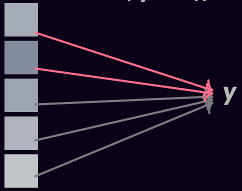
take home message

marginalize over sparse distributions
when training latent variable models

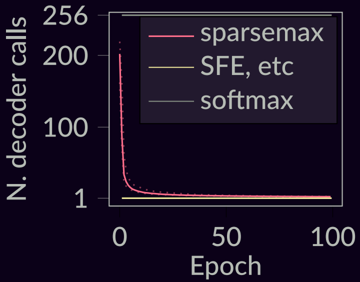
discrete and structured



deterministic, yet efficient



adaptive, as needed



github.com/deep-spin/sparse-marginalization-lvm vene.ro

Extra slides

Acknowledgements

This work was supported by the European Research Council (ERC StG DeepSPIN 758969) and by the Fundação para a Ciência e Tecnologia through contract UID/EEA/50008/2013.

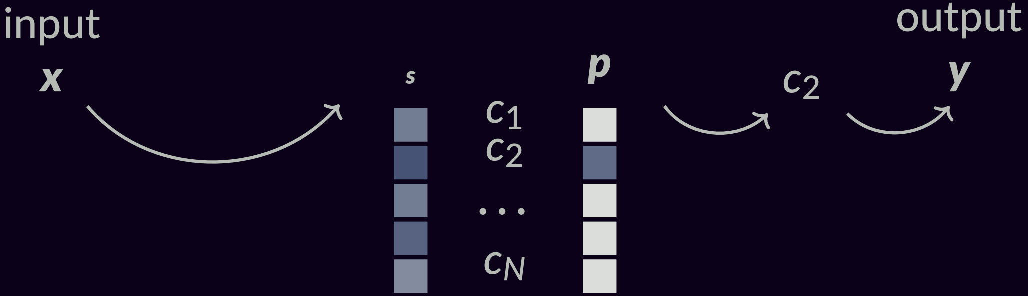
Some icons by Dave Gandy and Freepik via flaticon.com.

Structured Prediction

finally

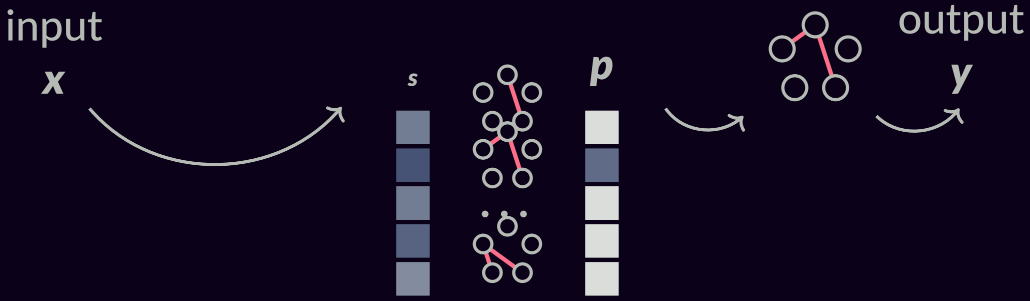
Structured Prediction

is essentially a (very high-dimensional) argmax



Structured Prediction

is essentially a (very high-dimensional) argmax



input

x

s

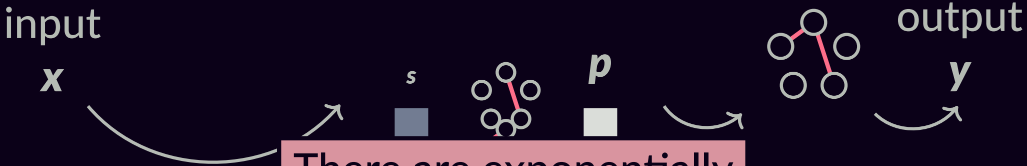
p

output

y

Structured Prediction

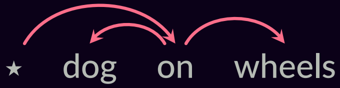
is essentially a (very high-dimensional) argmax



There are exponentially many structures

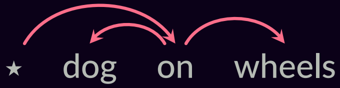
($\boldsymbol{s}$ cannot fit in memory!)

Factorization Into Parts



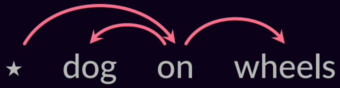
★ dog on wheels

Factorization Into Parts



	dog	on	wheels
∗→	○	●	○
dog→		○	○
on→	●		●
wheels→	○	○	

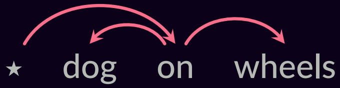
Factorization Into Parts



	dog	on	wheels
*→	○	●	○
dog→		○	○
on→	●		●
wheels→	○	○	

TREE

Factorization Into Parts

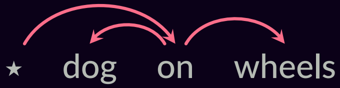


	dog	on	wheels
*→	○	●	○
dog→		○	○
on→	●		●
wheels→	○	○	

TREE

$\boldsymbol{a}_y$ = [010 100 001]

Factorization Into Parts



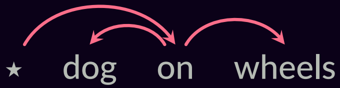
$$\boldsymbol{A} = \left[\begin{array}{r|cccccc}
\star\to\text{dog} & 1 & & 0 & 0 & & \\
\text{on}\to\text{dog} & 0 & & 1 & 1 & & \\
\text{wheels}\to\text{dog} & 0 & & 0 & 0 & & \\
\hline
\star\to\text{on} & 0 & & 1 & 1 & & \\
\text{dog}\to\text{on} & 1 & \ldots & 0 & 0 & \ldots & \\
\text{wheels}\to\text{on} & 0 & & 0 & 0 & & \\
\hline
\star\to\text{wheels} & 0 & & 0 & 0 & & \\
\text{dog}\to\text{wheels} & 0 & & 1 & 0 & & \\
\text{on}\to\text{wheels} & 1 & & 0 & 1 & &
\end{array}\right]
\quad
\boldsymbol{\eta} = \left[\begin{array}{r}
.1 \\ .2 \\ -.1 \\ \hline .3 \\ .8 \\ .1 \\ \hline -.3 \\ .2 \\ -.1
\end{array}\right]$$

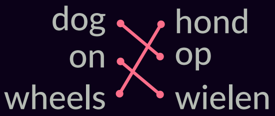
	dog	on	wheels
⁎→	○	●	○
dog→		○	○
on→	●		●
wheels→	○	○	

TREE

$\boldsymbol{a}_y$ = [010 100 001]

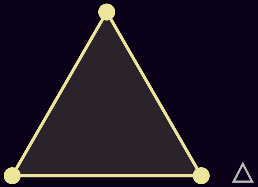
Factorization Into Parts

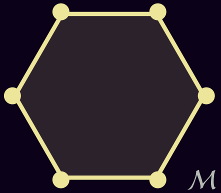




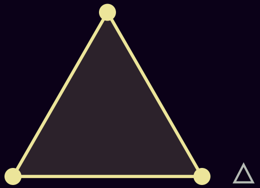
$$A = \begin{array}{r|ccccc}
\star\to\text{dog} & 1 & & 0 & 0 & \\
\text{on}\to\text{dog} & 0 & & 1 & 1 & \\
\text{wheels}\to\text{dog} & 0 & & 0 & 0 & \\
\hline
\star\to\text{on} & 0 & & 1 & 1 & \\
\text{dog}\to\text{on} & 1 & \dots & 0 & 0 & \dots \\
\text{wheels}\to\text{on} & 0 & & 0 & 0 & \\
\hline
\star\to\text{wheels} & 0 & & 0 & 0 & \\
\text{dog}\to\text{wheels} & 0 & & 1 & 0 & \\
\text{on}\to\text{wheels} & 1 & & 0 & 1 &
\end{array}
\qquad
\eta = \left[\begin{array}{r}
.1 \\ .2 \\ -.1 \\ \hline .3 \\ .8 \\ .1 \\ \hline -.3 \\ .2 \\ -.1
\end{array}\right]$$

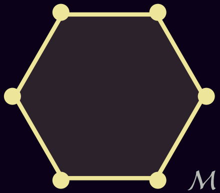
$$A = \begin{array}{r|cccccc}
\text{dog—hond} & 1 & & 0 & 0 & \\
\text{dog—op} & 0 & & 1 & 1 & \\
\text{dog—wielen} & 0 & & 0 & 0 & \\
\hline
\text{on—hond} & 0 & & 0 & 0 & \\
\text{on—op} & 1 & \dots & 0 & 0 & \dots \\
\text{on—wielen} & 0 & & 1 & 1 & \\
\hline
\text{wheels—hond} & 0 & & 1 & 0 & \\
\text{wheels—op} & 0 & & 0 & 0 & \\
\text{wheels—wielen} & 1 & & 0 & 1 &
\end{array}
\qquad
\eta = \left[\begin{array}{r}
.1 \\ .2 \\ -.1 \\ \hline .3 \\ .8 \\ .1 \\ \hline -.3 \\ .2 \\ -.1
\end{array}\right]$$


$\triangle$

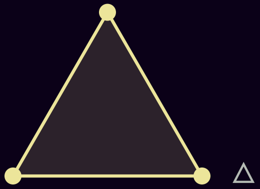

$\mathcal{M}$

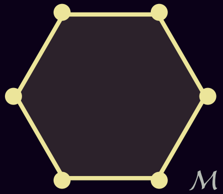
$$\mathcal{M} := \operatorname{conv}\left\{\boldsymbol{a}_h : h \in \mathcal{H}\right\}$$



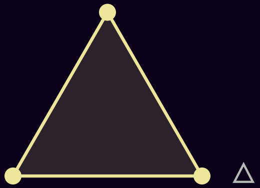


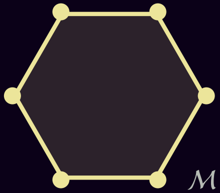
$$\begin{aligned}\mathcal{M} &:= \operatorname{conv}\left\{\boldsymbol{a}_h : h \in \mathcal{H}\right\}\\ &= \left\{\boldsymbol{A}\boldsymbol{p} : \boldsymbol{p} \in \Delta\right\}\end{aligned}$$



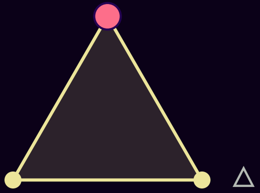


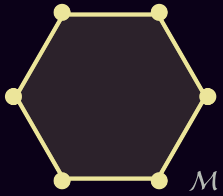
$$\begin{aligned}\mathcal{M} &:= \operatorname{conv}\left\{\boldsymbol{a}_h : h \in \mathcal{H}\right\}\\ &= \left\{\boldsymbol{A}\boldsymbol{p} : \boldsymbol{p} \in \Delta\right\}\\ &= \left\{\mathbb{E}_{H\sim \boldsymbol{p}}\, \boldsymbol{a}_H : \boldsymbol{p} \in \Delta\right\}\end{aligned}$$





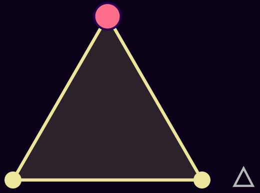
● **argmax** $\arg\max_{\boldsymbol{p} \in \triangle} \boldsymbol{p}^\top \boldsymbol{s}$

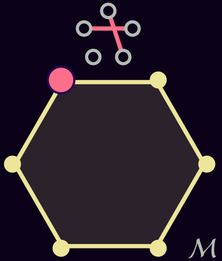




argmax $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s}$

MAP $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta}$



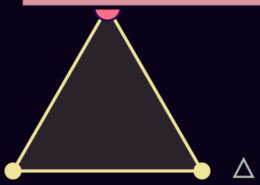


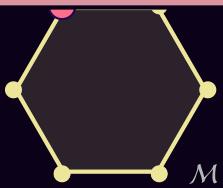
argmax $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s}$

MAP $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta}$

e.g. dependency parsing → **Chu-Liu/Edmonds**

matching → **Kuhn-Munkres**

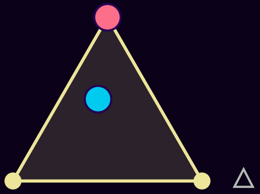


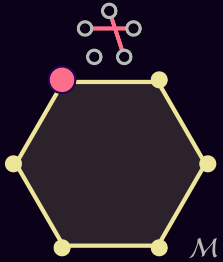


● **argmax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s}$

● **softmax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s} + \mathrm{H}(\boldsymbol{p})$

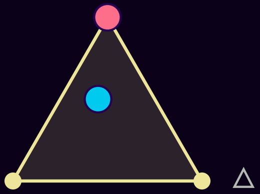
MAP $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta}$ ●

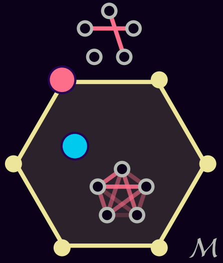




● **argmax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s}$ **MAP** $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta}$ ●

● **softmax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s} + \mathrm{H}(\boldsymbol{p})$ **marginals** $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} + \widetilde{\mathrm{H}}(\boldsymbol{\mu})$ ●





● **argmax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s}$

● **softmax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s} + \mathrm{H}(\boldsymbol{p})$

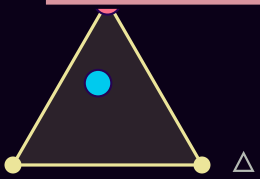
MAP $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta}$ ●

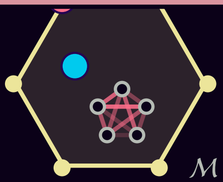
marginals $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} + \widetilde{\mathrm{H}}(\boldsymbol{\mu})$ ●

e.g. sequence labeling → forward-backward

(Rabiner, 1989)

As attention: (Kim et al., 2017)





● **argmax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s}$

● **softmax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s} + \mathrm{H}(\boldsymbol{p})$

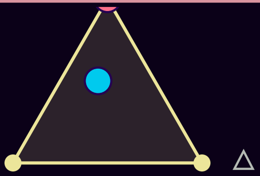
MAP $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta}$ ●

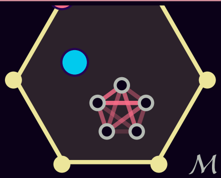
marginals $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} + \widetilde{\mathrm{H}}(\boldsymbol{\mu})$ ●

e.g. dependency parsing → **the Matrix-Tree theorem**

(Koo et al., 2007; D. A. Smith and N. A. Smith, 2007; McDonald and Satta, 2007)

As attention: (Liu and Lapata, 2018)





● **argmax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s}$

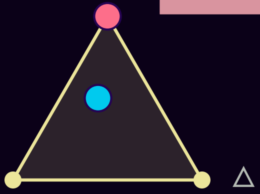
● **softmax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s} + \mathrm{H}(\boldsymbol{p})$

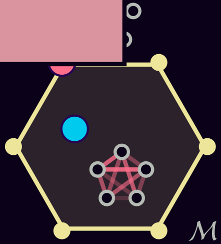
MAP $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta}$ ●

marginals $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} + \widetilde{\mathrm{H}}(\boldsymbol{\mu})$ ●

e.g. matchings → **#P-complete!**

(Taskar, 2004; Valiant, 1979)





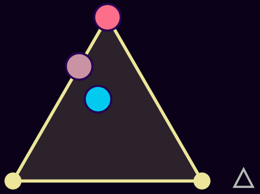
● **argmax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s}$

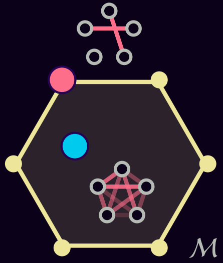
● **softmax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s} + \mathrm{H}(\boldsymbol{p})$

● **sparsemax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s} - 1/2\|\boldsymbol{p}\|^2$

MAP $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta}$ ●

marginals $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} + \widetilde{\mathrm{H}}(\boldsymbol{\mu})$ ●





(Niculae, Martins, Blondel, and Cardie, 2018)

● **argmax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s}$

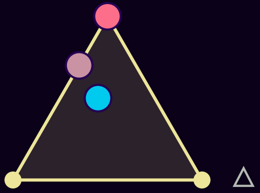
● **softmax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s} + \mathrm{H}(\boldsymbol{p})$

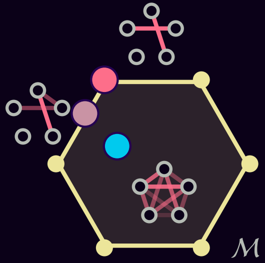
● **sparsemax** $\arg\max_{\boldsymbol{p}\in\triangle} \boldsymbol{p}^\top \boldsymbol{s} - 1/2\|\boldsymbol{p}\|^2$

MAP $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta}$ ●

marginals $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} + \widetilde{\mathrm{H}}(\boldsymbol{\mu})$ ●

SparseMAP $\arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} - 1/2\|\boldsymbol{\mu}\|^2$ ●





Algorithms for SparseMAP

$$\boldsymbol{\mu}^\star = \arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} - 1/2\|\boldsymbol{\mu}\|^2$$

Algorithms for SparseMAP

$$\boldsymbol{\mu}^\star = \arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} - \tfrac{1}{2}\|\boldsymbol{\mu}\|^2$$

linear constraints
(alas, exponentially many!)

quadratic objective

Algorithms for SparseMAP

$$\boldsymbol{\mu}^\star = \arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} - 1/2\|\boldsymbol{\mu}\|^2$$

linear constraints
(alas, exponentially many!)

quadratic objective

Conditional Gradient

(Frank and Wolfe, 1956; Lacoste-Julien and Jaggi, 2015)

Algorithms for SparseMAP

$$\boldsymbol{\mu}^\star = \arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} - 1/2\|\boldsymbol{\mu}\|^2$$

linear constraints

(alas, exponentially many!)

quadratic objective

Conditional Gradient

(Frank and Wolfe, 1956; Lacoste-Julien and Jaggi, 2015)

- select a new corner of $\mathcal{M}$

Algorithms for SparseMAP

$$\boldsymbol{\mu}^\star = \arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} - 1/2\|\boldsymbol{\mu}\|^2$$

linear constraints
(alas, exponentially many!)

quadratic objective

Conditional Gradient

(Frank and Wolfe, 1956; Lacoste-Julien and Jaggi, 2015)

- select a new corner of $\mathcal{M}$

$$\boxed{\boldsymbol{a}_{y^\star} = \arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \underbrace{(\boldsymbol{\eta} - \boldsymbol{\mu}^{(t-1)})}_{\tilde{\boldsymbol{\eta}}}}$$

Algorithms for SparseMAP

$$\boldsymbol{\mu}^\star = \arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} - 1/2\|\boldsymbol{\mu}\|^2$$

linear constraints
(alas, exponentially many!)

quadratic objective

Conditional Gradient

(Frank and Wolfe, 1956; Lacoste-Julien and Jaggi, 2015)

- select a new corner of $\mathcal{M}$
- update the (sparse) coefficients of $\boldsymbol{p}$
  - Update rules: vanilla, away-step, pairwise

Algorithms for SparseMAP

$$\boldsymbol{\mu}^\star = \arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} - 1/2\|\boldsymbol{\mu}\|^2$$

linear constraints
(alas, exponentially many!)

quadratic objective

Conditional Gradient

(Frank and Wolfe, 1956; Lacoste-Julien and Jaggi, 2015)

- select a new corner of $\mathcal{M}$
- update the (sparse) coefficients of ***p***
  - Update rules: vanilla, away-step, pairwise
  - Quadratic objective: **Active Set**

    (Nocedal and Wright, 1999, Ch. 16.4 & 16.5)

    (Wolfe, 1976; Vinyes and Obozinski, 2017)

Algorithms for SparseMAP

$$\boldsymbol{\mu}^\star = \arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top\boldsymbol{\eta} - 1/2\|\boldsymbol{\mu}\|^2$$

linear constraints
(alas, exponentially many!)

quadratic objective

Conditional Gradient

(Frank and Wolfe, 1956; Lacoste-Julien and Jaggi, 2015)

- select a new corne
- update the (sparse
  - Update rules: var
  pairwise
  - Quadratic objecti
  (Nocedal and Wright, 1999, Ch. 16.4 & 16.5)
  (Wolfe, 1976; Vinyes and Obozinski, 2017)

Active Set achieves
finite & **linear** convergence!

Algorithms for SparseMAP

$$\boldsymbol{\mu}^\star = \arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} - 1/2\|\boldsymbol{\mu}\|^2$$

linear constraints
(alas, exponentially many!)

quadratic objective

Conditional Gradient

(Frank and Wolfe, 1956; Lacoste-Julien and Jaggi, 2015)

- select a new corner of $\mathcal{M}$
- update the (sparse) coefficients of $\boldsymbol{p}$
  - Update rules: vanilla, away-step, pairwise
  - Quadratic objective: **Active Set**

    (Nocedal and Wright, 1999, Ch. 16.4 & 16.5)

    (Wolfe, 1976; Vinyes and Obozinski, 2017)

Backward pass

$\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\eta}}$ is sparse

Algorithms for SparseMAP

$$\boldsymbol{\mu}^\star = \arg\max_{\boldsymbol{\mu}\in\mathcal{M}} \boldsymbol{\mu}^\top \boldsymbol{\eta} - 1/2\|\boldsymbol{\mu}\|^2$$

linear constraints
(alas, exponentially many!)

quadratic objective

Conditional Gradient

(Frank and Wolfe, 1956; Lacoste-Julien and Jaggi, 2015)

- select a new corner of $\mathcal{M}$
- update the (sparse) coefficients of $\boldsymbol{p}$
  - Update rules: vanilla, away-step, pairwise
  - Quadratic objective: **Active Set**

  (Nocedal and Wright, 1999, Ch. 16.4 & 16.5)

  (Wolfe, 1976; Vinyes and Obozinski, 2017)

Backward pass

$\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\eta}}$ is sparse

computing $\left(\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\eta}}\right)^\top \boldsymbol{dy}$
takes $\mathcal{O}(\dim(\boldsymbol{\mu})\,\mathrm{nnz}(\boldsymbol{p}^\star))$

Algorithms for SparseMAP

$$\boldsymbol{\mu}^\star = \underset{\boldsymbol{\mu}\in\mathcal{M}}{\arg\max}\, \boldsymbol{\mu}^\top \boldsymbol{\eta} - 1/2\|\boldsymbol{\mu}\|^2$$

linear constraints
(alas, exponentially many!)

quadratic objective

Conditi **pass**

(Frank and Wolfe, 1956

- select a new c
- update the (sparse) coefficients of $\boldsymbol{p}$
  - Update rules: vanilla, away-step, pairwise
  - Quadratic objective: **Active Set**

  (Nocedal and Wright, 1999, Ch. 16.4 & 16.5)

  (Wolfe, 1976; Vinyes and Obozinski, 2017)

Completely modular: just add MAP

rse

computing $\left(\frac{\partial\boldsymbol{\mu}}{\partial\boldsymbol{\eta}}\right)^\top \boldsymbol{dy}$
takes $\mathcal{O}(\dim(\boldsymbol{\mu})\,\mathrm{nnz}(\boldsymbol{p}^\star))$

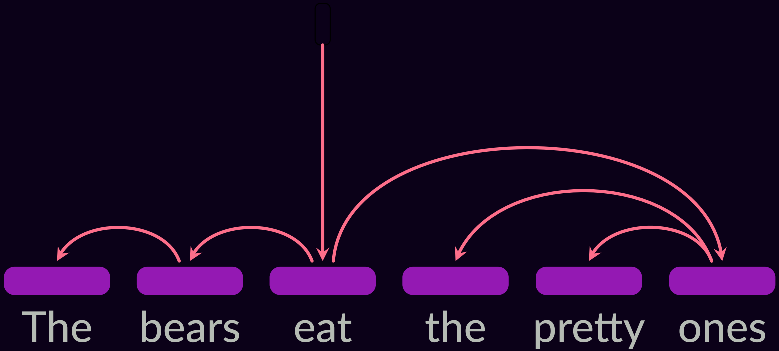
Dependency TreeLSTM

(Tai et al., 2015)

The bears eat the pretty ones

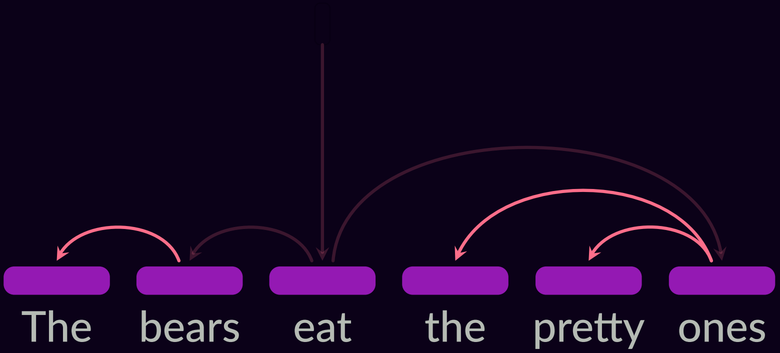
Dependency TreeLSTM

(Tai et al., 2015)



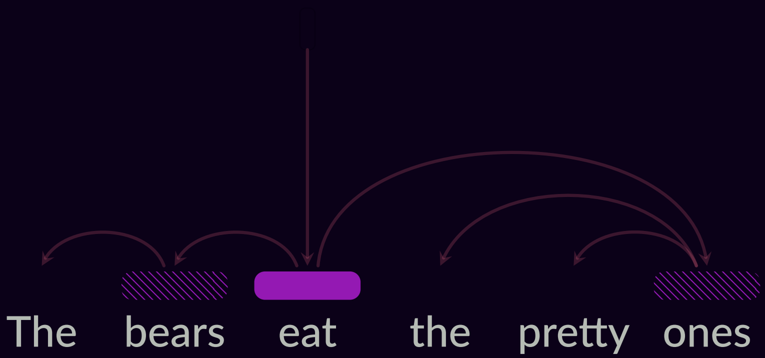
Dependency TreeLSTM

(Tai et al., 2015)



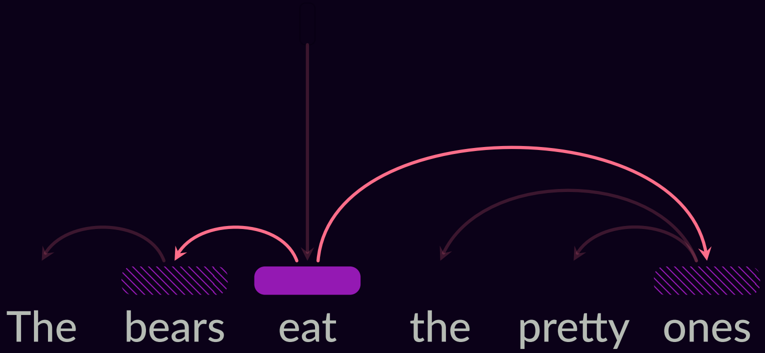
Dependency TreeLSTM

(Tai et al., 2015)



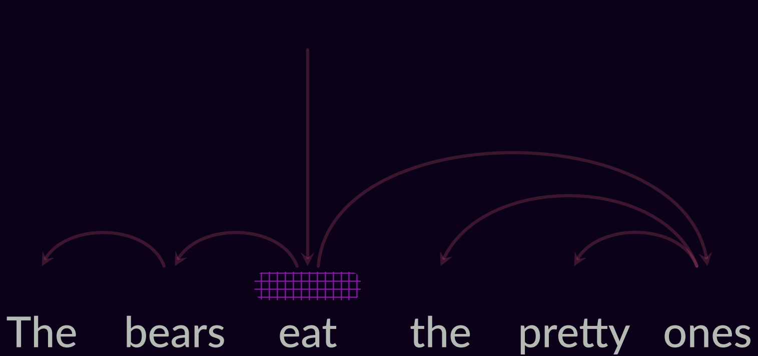
Dependency TreeLSTM

(Tai et al., 2015)



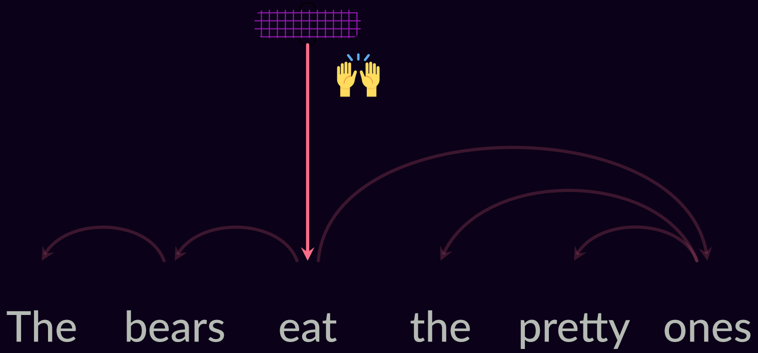
Dependency TreeLSTM

(Tai et al., 2015)



Dependency TreeLSTM

(Tai et al., 2015)



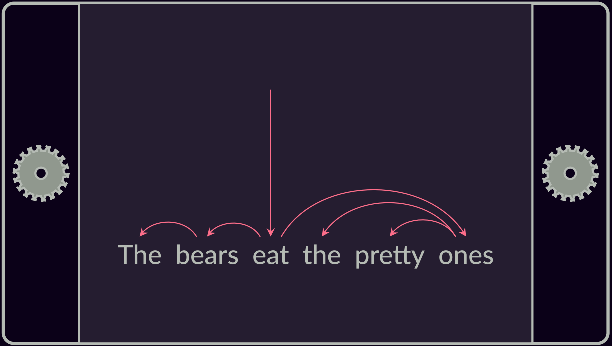
The bears eat the pretty ones

Latent Dependency TreeLSTM

(Niculae, Martins, and Cardie, 2018)

input

x



output

y

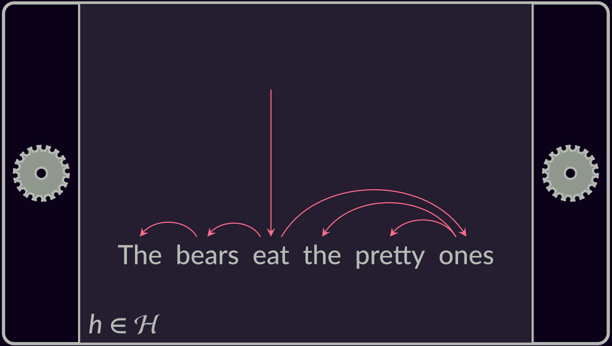
Latent Dependency TreeLSTM

(Niculae, Martins, and Cardie, 2018)

$$p(y|x) = \sum_{h \in \mathcal{H}} p(y \mid h, x)\, p(h \mid x)$$

input

x



output

y

Structured Latent Variable Models

$$p(y \mid x) = \sum_{h \in \mathcal{H}} p \quad (y \mid h, x)\, p \quad (h \mid x)$$

Structured Latent Variable Models

$$p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\phi}}(y \mid h, x)\, p_{\boldsymbol{\pi}}(h \mid x)$$

Structured Latent Variable Models

e.g., a TreeLSTM defined by *h*

$$p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\phi}}(y \mid h, x)\, p_{\boldsymbol{\pi}}(h \mid x)$$

Structured Latent Variable Models

$$p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\phi}}(y \mid h, x)\, p_{\boldsymbol{\pi}}(h \mid x)$$

e.g., a TreeLSTM defined by *h*

latent classifier

Structured Latent Variable Models

sum over
all possible trees

e.g., a TreeLSTM defined by *h*

$$p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\phi}}(y \mid h, x)\, p_{\boldsymbol{\pi}}(h \mid x)$$

latent classifier

Exponentially large sum!

Structured Latent Variable Models

sum over
all possible trees

e.g., a TreeLSTM defined by *h*

$$p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\phi}}(y \mid h, x)\, p_{\boldsymbol{\pi}}(h \mid x)$$

latent classifier

How to define $\boldsymbol{p_{\pi}}$?

idea 1

idea 2

idea 3

Structured Latent Variable Models

sum over
all possible trees

e.g., a TreeLSTM defined by *h*

$$p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\phi}}(y \mid h, x)\, p_{\boldsymbol{\pi}}(h \mid x)$$

latent classifier

How to define $\boldsymbol{p_{\pi}}$?

$$\sum_{h \in \mathcal{H}}$$

idea 1

idea 2

idea 3

Structured Latent Variable Models

sum over
all possible trees

e.g., a TreeLSTM defined by *h*

$$p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\phi}}(y \mid h, x)\, p_{\boldsymbol{\pi}}(h \mid x)$$

latent classifier

How to define $\boldsymbol{p_{\pi}}$?

$$\sum_{h \in \mathcal{H}} \quad \frac{\partial p(y \mid x)}{\partial \boldsymbol{\pi}}$$

idea 1

idea 2

idea 3

Structured Latent Variable Models

sum over
all possible trees

e.g., a TreeLSTM defined by *h*

$$p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\phi}}(y \mid h, x)\; p_{\boldsymbol{\pi}}(h \mid x)$$

latent classifier

How to define $p_{\boldsymbol{\pi}}$?

$$\sum_{h \in \mathcal{H}} \frac{\partial p(y \mid x)}{\partial \boldsymbol{\pi}}$$

idea 1 $p_{\boldsymbol{\pi}}(h \mid x) = 1$ if $h = h^\star$ else 0 argmax

idea 2

idea 3

Structured Latent Variable Models

sum over
all possible trees

e.g., a TreeLSTM defined by *h*

$$p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\phi}}(y \mid h, x)\, p_{\boldsymbol{\pi}}(h \mid x)$$

latent classifier

How to define $\boldsymbol{p_{\pi}}$?

$$\sum_{h \in \mathcal{H}} \qquad \frac{\partial p(y \mid x)}{\partial \boldsymbol{\pi}}$$

idea 1 $p_{\boldsymbol{\pi}}(h \mid x) = 1$ if $h = h^{\star}$ else 0 argmax 😊

idea 2

idea 3

Structured Latent Variable Models

sum over
all possible trees

e.g., a TreeLSTM defined by *h*

$$p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\phi}}(y \mid h, x)\, p_{\boldsymbol{\pi}}(h \mid x)$$

latent classifier

How to define $p_{\boldsymbol{\pi}}$?

			$\sum_{h \in \mathcal{H}}$	$\frac{\partial p(y \mid x)}{\partial \boldsymbol{\pi}}$
idea 1	$p_{\boldsymbol{\pi}}(h \mid x) = 1$ if $h = h^\star$ else 0	argmax	😊	😟
idea 2				
idea 3				

Structured Latent Variable Models

sum over
all possible trees

e.g., a TreeLSTM defined by *h*

$$p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\phi}}(y \mid h, x)\, p_{\boldsymbol{\pi}}(h \mid x)$$

latent classifier

How to define $p_{\boldsymbol{\pi}}$?

			$\sum_{h \in \mathcal{H}}$	$\frac{\partial p(y \mid x)}{\partial \boldsymbol{\pi}}$
idea 1	$p_{\boldsymbol{\pi}}(h \mid x) = 1$ if $h = h^{\star}$ else 0	argmax	😊	😟
idea 2	$p_{\boldsymbol{\pi}}(h \mid x) \propto \exp\big(\text{score}_{\boldsymbol{\pi}}(h; x)\big)$	softmax		
idea 3				

Structured Latent Variable Models

sum over
all possible trees

e.g., a TreeLSTM defined by *h*

$$p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\phi}}(y \mid h, x)\, p_{\boldsymbol{\pi}}(h \mid x)$$

latent classifier

How to define $p_{\boldsymbol{\pi}}$?

			$\sum_{h \in \mathcal{H}}$	$\frac{\partial p(y \mid x)}{\partial \boldsymbol{\pi}}$
idea 1	$p_{\boldsymbol{\pi}}(h \mid x) = 1 \text{ if } h = h^\star \text{ else } 0$	argmax	😊	😟
idea 2	$p_{\boldsymbol{\pi}}(h \mid x) \propto \exp\left(\text{score}_{\boldsymbol{\pi}}(h; x)\right)$	softmax		😊
idea 3				

Structured Latent Variable Models

sum over
all possible trees

e.g., a TreeLSTM defined by *h*

$$p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\phi}}(y \mid h, x)\, p_{\boldsymbol{\pi}}(h \mid x)$$

latent classifier

How to define $p_{\boldsymbol{\pi}}$?

			$\sum_{h \in \mathcal{H}}$	$\frac{\partial p(y \mid x)}{\partial \boldsymbol{\pi}}$
idea 1	$p_{\boldsymbol{\pi}}(h \mid x) = 1$ if $h = h^{\star}$ else 0	argmax	😊	😟
idea 2	$p_{\boldsymbol{\pi}}(h \mid x) \propto \exp\left(\text{score}_{\boldsymbol{\pi}}(h; x)\right)$	softmax	😱	😊
idea 3				

Structured Latent Variable Models

sum over
all possible trees

e.g., a TreeLSTM defined by *h*

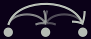
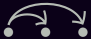
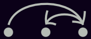
$$p(y \mid x) = \sum_{h \in \mathcal{H}} p_{\boldsymbol{\phi}}(y \mid h, x)\; p_{\boldsymbol{\pi}}(h \mid x)$$

latent classifier

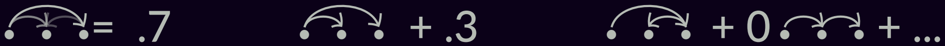
How to define $p_{\boldsymbol{\pi}}$?

			$\sum_{h \in \mathcal{H}}$	$\frac{\partial p(y \mid x)}{\partial \boldsymbol{\pi}}$
idea 1	$p_{\boldsymbol{\pi}}(h \mid x) = 1$ if $h = h^{\star}$ else 0	argmax	😊	😟
idea 2	$p_{\boldsymbol{\pi}}(h \mid x) \propto \exp\big(\text{score}_{\boldsymbol{\pi}}(h; x)\big)$	softmax	😱	😊
idea 3		SparseMAP	😊	😊

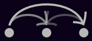
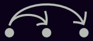
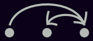
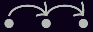
SparseMAP

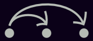
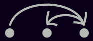
 = .7  + .3 

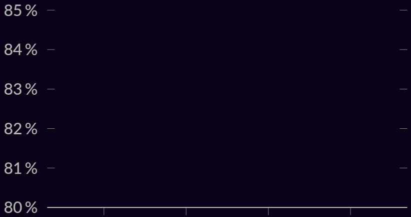
SparseMAP

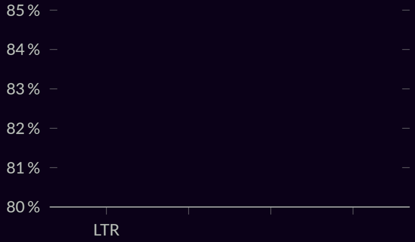


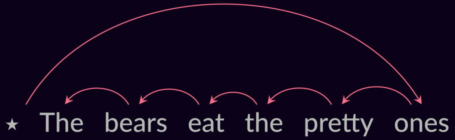
SparseMAP

 $= .7$  $+ .3$  $+ 0$  $+ \ldots$

$p(y \mid x) = .7\, p_{\boldsymbol{\phi}}(y \mid$  $) + .3\, p_{\boldsymbol{\phi}}(y \mid$  $)$

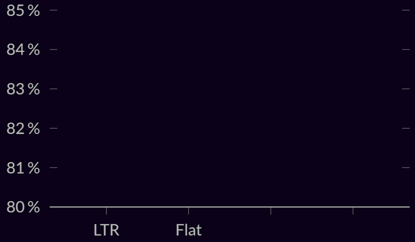

85 %
84 %
83 %
82 %
81 %
80 %

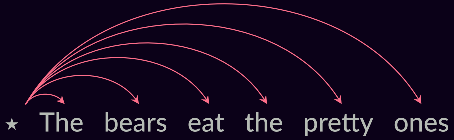




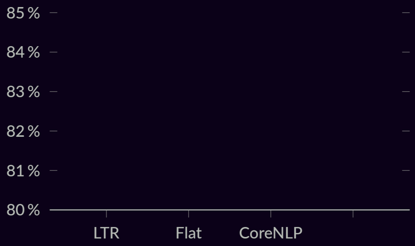
★ The bears eat the pretty ones

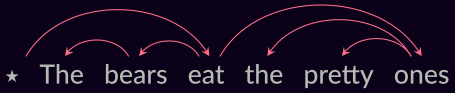
Left-to-right: regular LSTM




★ The bears eat the pretty ones

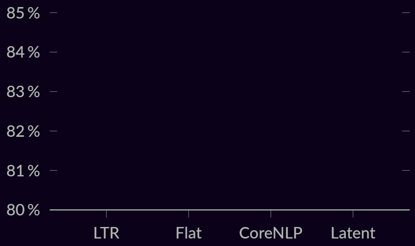
Flat: bag-of-words–like



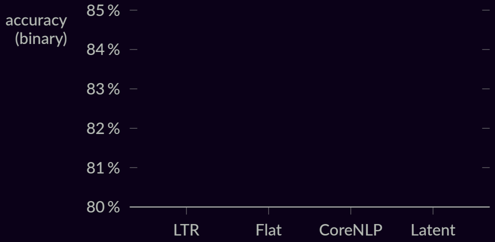


★ The bears eat the pretty ones

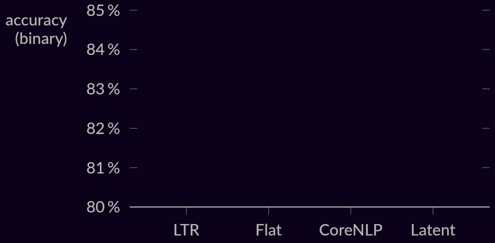
CoreNLP: off-line parser


85 %
84 %
83 %
82 %
81 %
80 %
LTR
Flat
CoreNLP
Latent

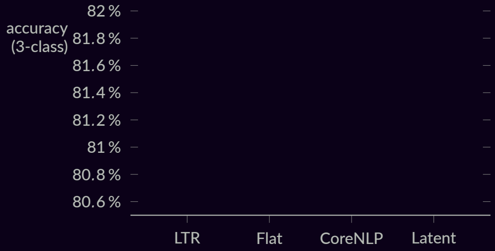
Sentiment classification (SST)



Sentiment classification (SST)



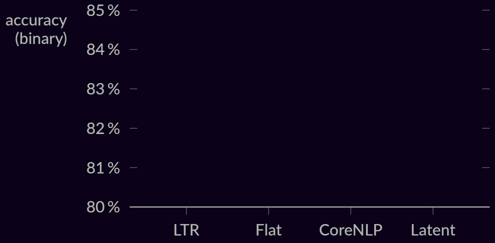
Natural Language Inference (SNLI)



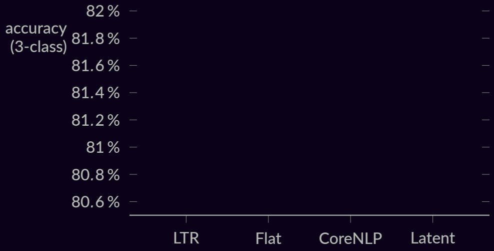
Sentence pair classification (P, H)

$$p(y \mid P, H) = \sum_{h_P \in \mathcal{H}(P)} \sum_{h_H \in \mathcal{H}(H)} p_{\boldsymbol{\phi}}(y \mid h_P, h_H)\, p_{\boldsymbol{\pi}}(h_P \mid P)\, p_{\boldsymbol{\pi}}(h_H \mid H)$$

Sentiment classification (SST)



Natural Language Inference (SNLI)

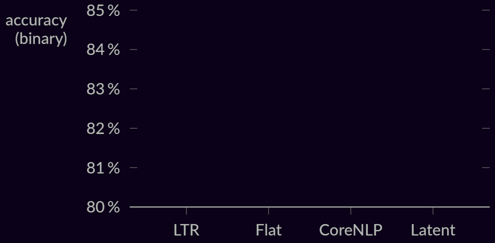


Reverse dictionary lookup

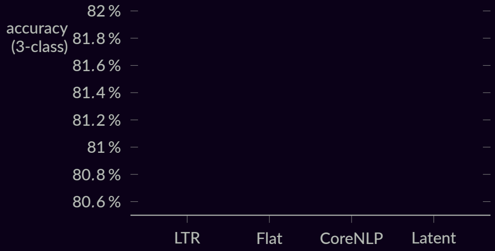
given word description, predict word embedding (Hill et al., 2016)

instead of $p(y \mid x)$, we model $\mathbb{E}_{p_{\boldsymbol{\pi}}} \boldsymbol{g}(x) = \sum_{h \in \mathcal{H}} \boldsymbol{g}(x; h)\, p_{\boldsymbol{\pi}}(h \mid x)$

Sentiment classification (SST)

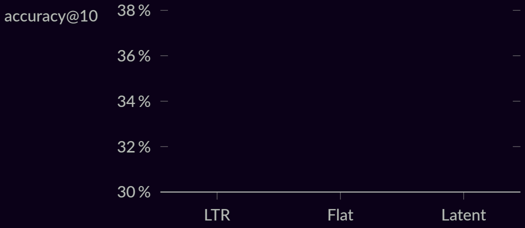


Natural Language Inference (SNLI)

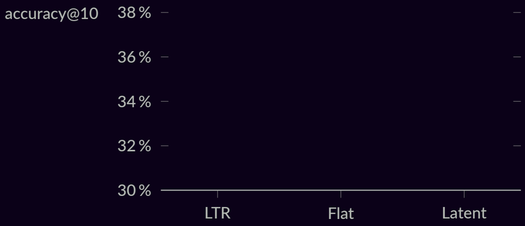


Reverse dictionary lookup

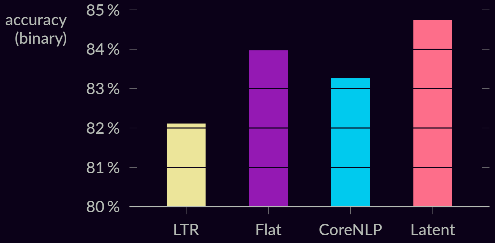
(definitions)



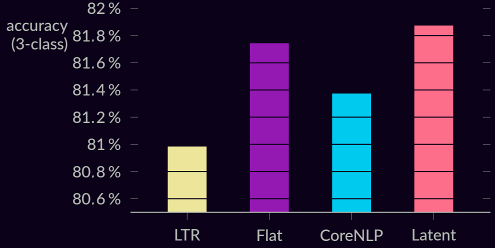
(concepts)



Sentiment classification (SST)

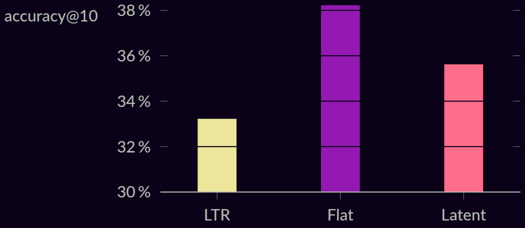


Natural Language Inference (SNLI)

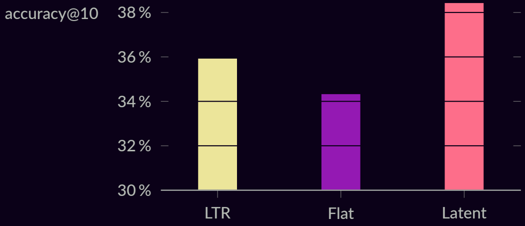


Reverse dictionary lookup

(definitions)

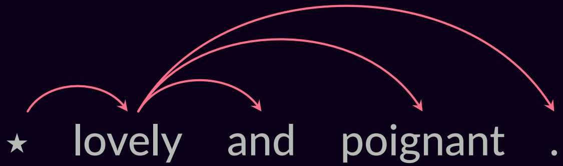


(concepts)



Syntax vs. Composition Order

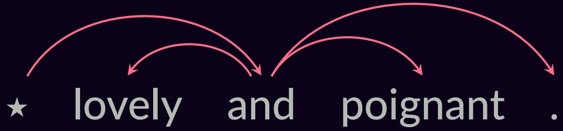
CoreNLP parse, $p = 21.4\%$



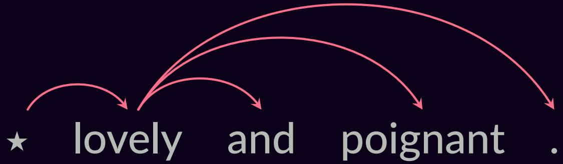
★ lovely and poignant .

Syntax vs. Composition Order

$p = 22.6\%$



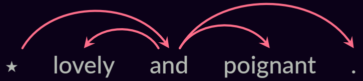
CoreNLP parse, $p = 21.4\%$



. . .

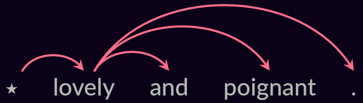
Syntax vs. Composition Order

$p = 22.6\%$



★ lovely and poignant .

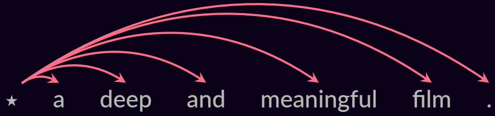
CoreNLP parse, $p = 21.4\%$



★ lovely and poignant .

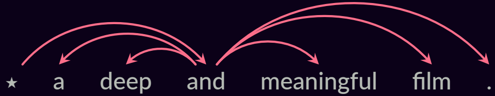
. . .

$p = 15.33\%$



★ a deep and meaningful film .

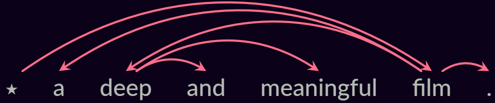
$p = 15.27\%$



★ a deep and meaningful film .

. . .

CoreNLP parse, $p = 0\%$



★ a deep and meaningful film .