Differentiable dynamic programming for structured prediction and attention

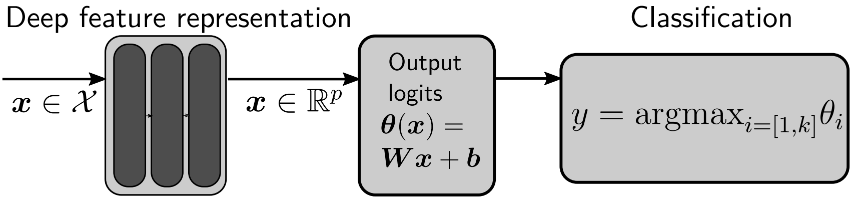
Arthur Mensch, Mathieu Blondel

Inria Parietal, Saclay, France
NTT Communication Science Laboratories, Kyoto, Japan

July 12, 2018

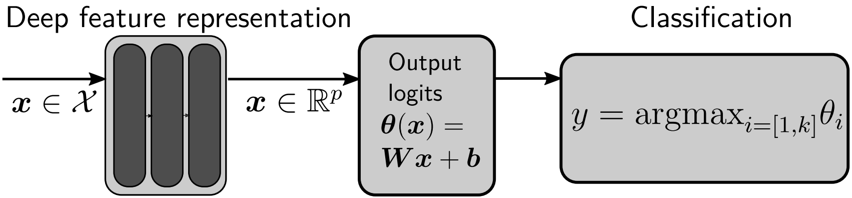
Predictive models: parametrized functions + linear programs

Classification: $\mathcal{Y} = [1, k]$



Predictive models: parametrized functions + linear programs

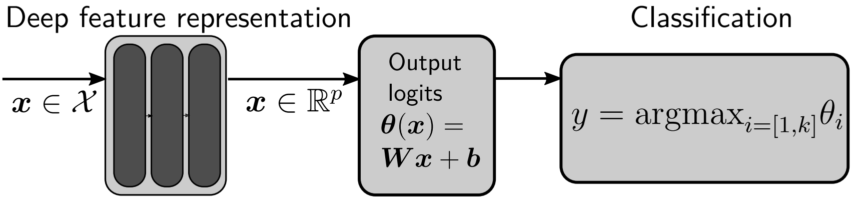
Classification: $\mathcal{Y} = [1, k]$



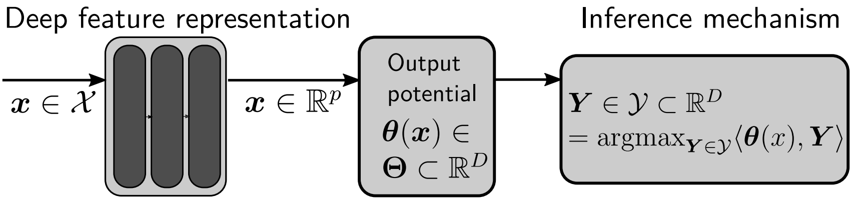
Structured output ? $\mathcal{Y} \subset \mathbb{R}^D$ (edges of a polytope), *e. g.* a tag sequence

Predictive models: parametrized functions + linear programs

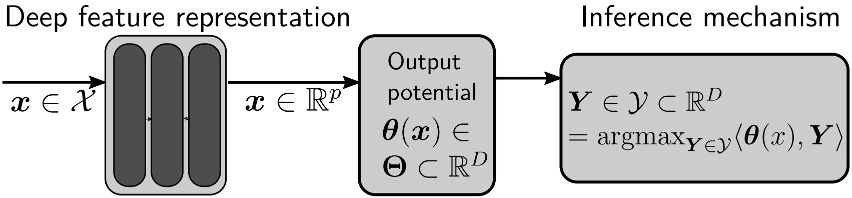
Classification: $\mathcal{Y} = [1, k]$



Structured output ? $\mathcal{Y} \subset \mathbb{R}^D$ (edges of a polytope), *e. g.* a tag sequence

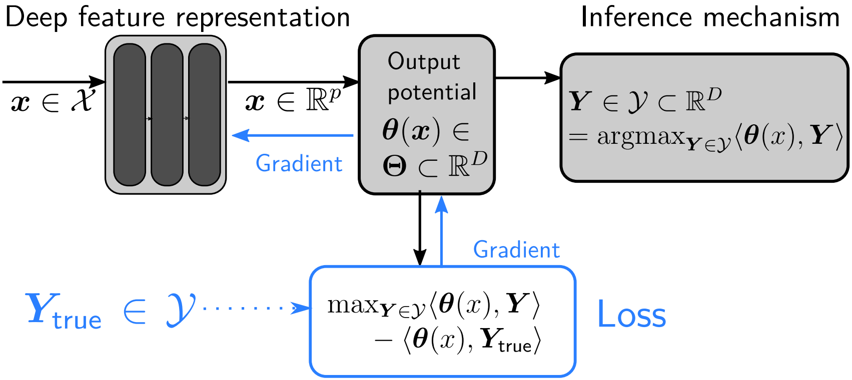


Structure prediction:



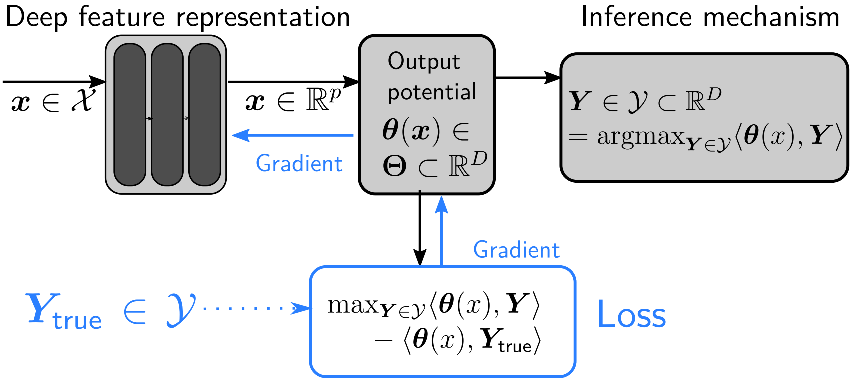
Training

Structure prediction: *Structured perceptron loss*



Training

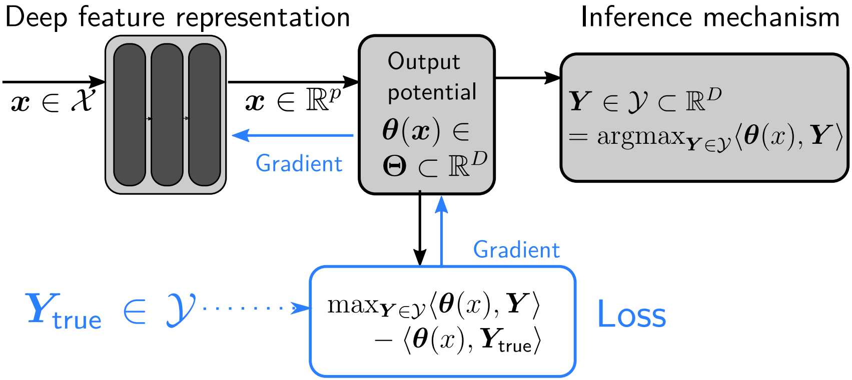
Structure prediction: *Structured perceptron loss*



Backpropagate through the max.

Training

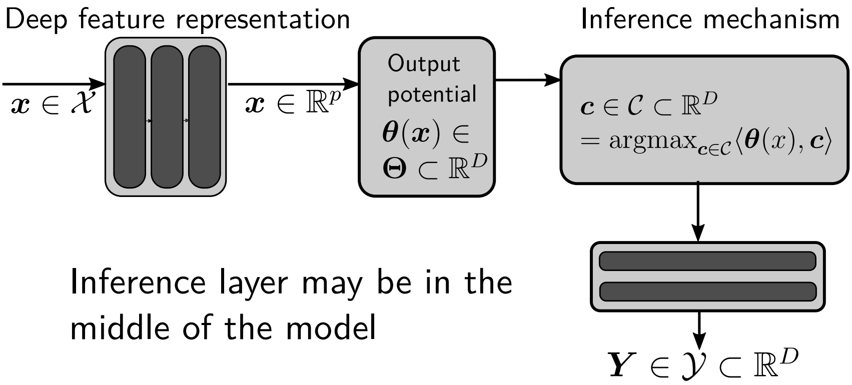
Structure prediction: *Structured perceptron loss*



Backpropagate through the max. Not differentiable everywhere !

Structured prediction as an inner layer

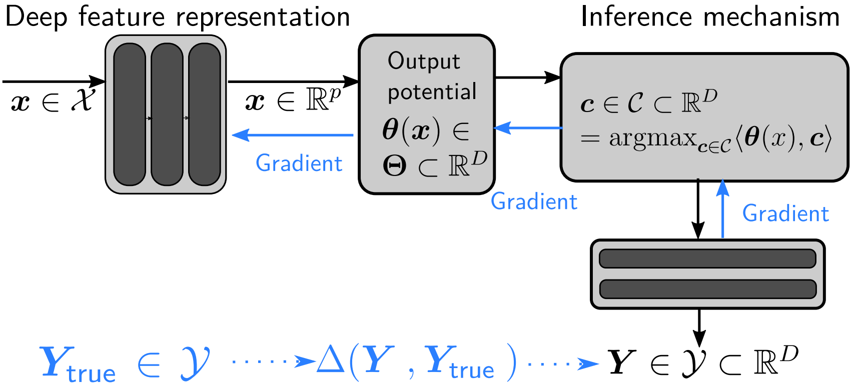
Example: Attention mechanisms, where c are the attention weights.



Inference layer may be in the middle of the model

Structured prediction as an inner layer

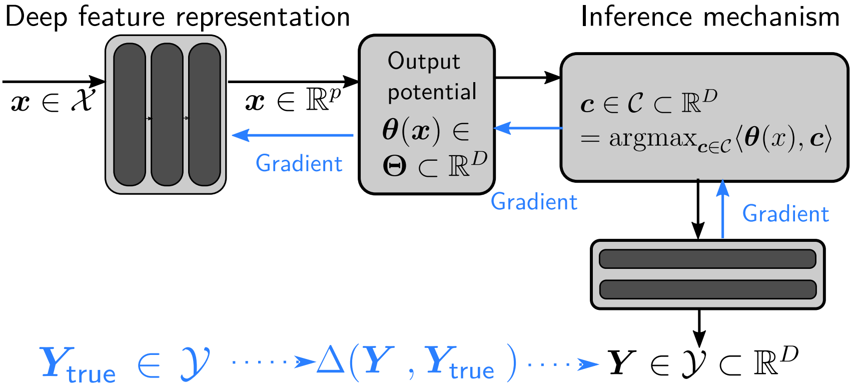
Example: Attention mechanisms, where c are the attention weights.



We need to backpropagate through the argmax.

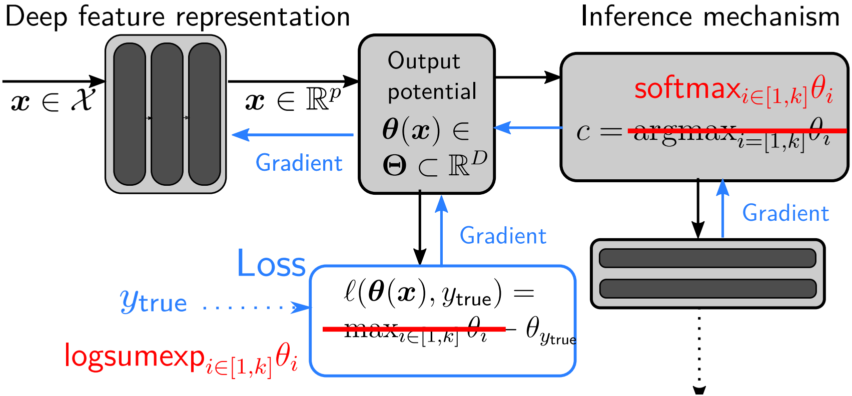
Structured prediction as an inner layer

Example: Attention mechanisms, where c are the attention weights.

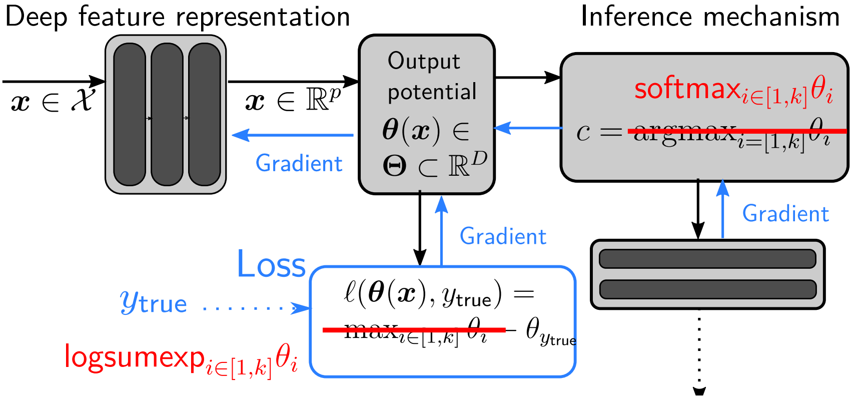


We need to backpropagate through the argmax. Zero derivative !

From max to softmax



From max to softmax



Multinomial loss, softmax attention: **differentiable**

Questions and contributions

- From **max** to **softmax**: Where does this comes from and can we use different smoothing techniques ?

Questions and contributions

- From **max** to **softmax**: Where does this comes from and can we use different smoothing techniques ?
- How to smooth a wide class of **structured prediction** LP problems?

$$\max_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{\theta}(x), \boldsymbol{Y} \rangle \qquad \boldsymbol{Y} \in \mathcal{Y} \subset \mathbb{R}^D = \operatorname*{argmax}_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{\theta}(x), \boldsymbol{Y} \rangle$$

Questions and contributions

- From **max** to **softmax**: Where does this comes from and can we use different smoothing techniques ?
- How to smooth a wide class of **structured prediction** LP problems?

$$\max_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{\theta}(x), \boldsymbol{Y} \rangle \qquad \boldsymbol{Y} \in \mathcal{Y} \subset \mathbb{R}^D = \operatorname*{argmax}_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{\theta}(x), \boldsymbol{Y} \rangle$$

Inference mechanisms often rely on a **dynamic programming** algorithm

Contribution: differentiable dynamic programing

- Smooth **max** layers to design new structured losses
- Differentiable **argmax** layers for inner inference mechanisms

Contributions

Generic framework for differentiable structured prediction:

- Regularizing the max operators with strongly convex penalties.
- May output **sparse** continuous outputs

Applications:

- End-to-end audio to score alignment
- Named entity recognition with sparse predictions
- Block sparse attention mechanisms

Extends and ground in theory: [LeCun et al., 2006, Lample et al., 2016, Kim et al., 2017, Cuturi and Blondel, 2017], etc.

Dynamic programming

Dynamic programming solve the structure prediction problem

$$\mathrm{LP}(\boldsymbol{\theta}) \triangleq \max_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{\theta}, \boldsymbol{Y} \rangle$$

by splitting the combinatorial set $\mathcal{Y} \subset \mathbb{R}^D$ into **sets of smaller dimensions**

- Compute $\mathrm{LP}(\boldsymbol{\theta})$ in linear time $\mathcal{O}(D)$ vs exponential naive resolution

Dynamic programming

Dynamic programming solve the structure prediction problem

$$\mathrm{LP}(\boldsymbol{\theta}) \triangleq \max_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{\theta}, \boldsymbol{Y} \rangle$$

by splitting the combinatorial set $\mathcal{Y} \subset \mathbb{R}^D$ into **sets of smaller dimensions**

- Compute $\mathrm{LP}(\boldsymbol{\theta})$ in linear time $\mathcal{O}(D)$ vs exponential naive resolution

Also provide the **argmax** in $\mathcal{O}(D)$:

$$\operatorname*{argmax}_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{\theta}, \boldsymbol{Y} \rangle$$

Dynamic programming

Dynamic programming solve the structure prediction problem

$$\mathrm{LP}(\boldsymbol{\theta}) \triangleq \max_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{\theta}, \boldsymbol{Y} \rangle$$

by splitting the combinatorial set $\mathcal{Y} \subset \mathbb{R}^D$ into **sets of smaller dimensions**

- Compute $\mathrm{LP}(\boldsymbol{\theta})$ in linear time $\mathcal{O}(D)$ vs exponential naive resolution

Also provide the **argmax** in $\mathcal{O}(D)$:

$$\operatorname*{argmax}_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{\theta}, \boldsymbol{Y} \rangle$$

Examples:

- Viterbi algorithm for infering tag sequences
- Dynamic time warping algorithm for infering alignment matrices

Generic (max, $+$) DP is best path finding

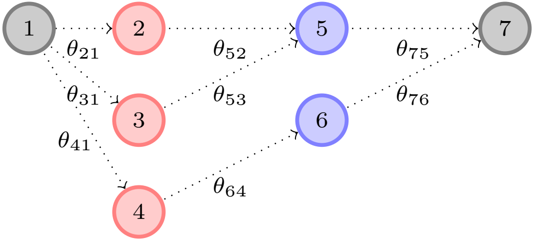
Directed acyclic graph

- $G = (\mathcal{N}, \mathcal{E})$, with 1 root and 1 leaf, nodes numbered in topo. order $[1, N]$
- Edge (i, j) has weight $\theta_{i,j}$ — j parent, i child. $\boldsymbol{\theta} \in \mathbb{R}^{n \times n}$ incidence matrix
- Path $\boldsymbol{Y} \in \mathcal{Y} \subset \{0, 1\}^{N \times N}$: $y_{i,j} = 1$ iff (i, j) is taken

Single path value: $\langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle$

Highest score among all paths

$$\mathrm{LP}(\boldsymbol{\theta}) = \max_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle$$



Maximum value computation (finding the max)

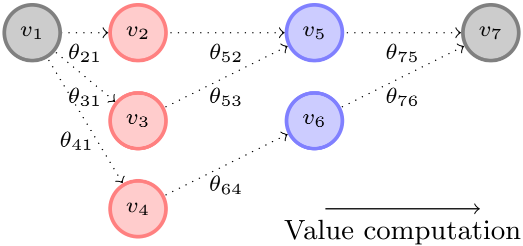
- **Max value from 1 to i**

$$v_i(\boldsymbol{\theta}) = \max_{j \in \mathcal{P}_i} \theta_{i,j} + v_j(\boldsymbol{\theta})$$

- One pass over the graph

$$(v_1 = 0, v_2, \ldots, v_n \triangleq \mathrm{DP}(\boldsymbol{\theta}))$$

= **Bellman equation**



Maximum value computation (finding the max)

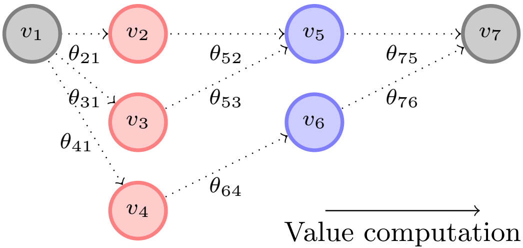
- **Max value from 1 to i**

$$v_i(\boldsymbol{\theta}) = \max_{j \in \mathcal{P}_i} \theta_{i,j} + v_j(\boldsymbol{\theta})$$

- One pass over the graph

$$(v_1 = 0, v_2, \dots, v_n \triangleq \mathrm{DP}(\boldsymbol{\theta}))$$

= **Bellman equation**



The DP recursion solves the linear problem [Bellman, 1958]

$$\mathrm{DP}(\boldsymbol{\theta}) = \mathrm{LP}(\boldsymbol{\theta}) = \max_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle$$

Best path computation (finding the argmax)

What if we want to find the LP solution (a.k.a. perform inference ?)

Best path computation (finding the argmax)

What if we want to find the LP solution (a.k.a. perform inference ?)

The argmax is computable using backpropagation = backtracking

Danskin theorem [Danskin, 1966]

$$\partial \mathrm{DP}(\boldsymbol{\theta}) = \partial_{\boldsymbol{\theta}}(\boldsymbol{\theta} \to \max_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle)) = \mathrm{conv}(\operatorname*{argmax}_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle)$$

- When the argmax is unique: $\partial_{\boldsymbol{\theta}} \mathrm{DP}(\boldsymbol{\theta}) = \operatorname{argmax}_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle$

Best path computation (finding the argmax)

What if we want to find the LP solution (a.k.a. perform inference ?)

The argmax is computable using backpropagation = backtracking

Danskin theorem [Danskin, 1966]

$$\partial \mathrm{DP}(\boldsymbol{\theta}) = \partial_{\boldsymbol{\theta}}(\boldsymbol{\theta} \to \max_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle)) = \mathrm{conv}(\operatorname*{argmax}_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle)$$

- When the argmax is unique: $\partial_{\boldsymbol{\theta}} \mathrm{DP}(\boldsymbol{\theta}) = \mathrm{argmax}_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle$

Dynamic programming layers

We define:

- **Max** layer: $\boldsymbol{\theta} \to \mathrm{DP}(\boldsymbol{\theta}) = \max_{\boldsymbol{Y}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle$
- **Argmax** layer: $\boldsymbol{\theta} \to \partial_{\boldsymbol{\theta}} \mathrm{DP}(\boldsymbol{\theta}) \sim \mathrm{argmax}_{\boldsymbol{Y}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle$

Operator regularization

Obstacles to end-to-end training

- Max layer $\boldsymbol{\theta} \to \mathrm{DP}(\boldsymbol{\theta})$ is not differentiable everywhere
- Argmax layer $\boldsymbol{\theta} \to \partial\mathrm{DP}(\boldsymbol{\theta})$ is piecewise constant / not defined

Operator regularization

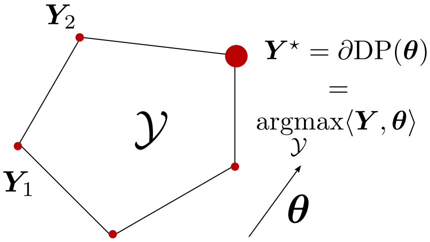
Obstacles to end-to-end training

- Max layer $\boldsymbol{\theta} \to \mathrm{DP}(\boldsymbol{\theta})$ is not differentiable everywhere
- Argmax layer $\boldsymbol{\theta} \to \partial\mathrm{DP}(\boldsymbol{\theta})$ is piecewise constant / not defined

Culprit is the Bellman recursion

$$\boldsymbol{x} \in \mathbb{R}^d \to \max(\boldsymbol{x}) \in \mathbb{R}$$

- Not differentiable everywhere
- Piecewise linear (null Hessian)



Hard geometry

Operator regularization

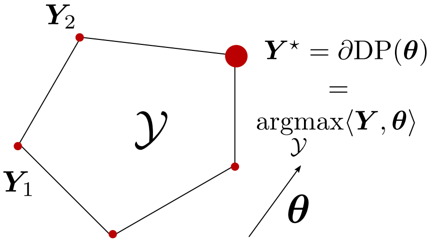
Obstacles to end-to-end training

- Max layer $\boldsymbol{\theta} \to \mathrm{DP}(\boldsymbol{\theta})$ is not differentiable everywhere
- Argmax layer $\boldsymbol{\theta} \to \partial\mathrm{DP}(\boldsymbol{\theta})$ is piecewise constant / not defined

Culprit is the Bellman recursion

$$\boldsymbol{x} \in \mathbb{R}^d \to \max(\boldsymbol{x}) \in \mathbb{R}$$

- Not differentiable everywhere
- Piecewise linear (null Hessian)



Hard geometry

Solution: smooth the maximum operator

Max smoothing

$\Omega : \mathbb{R} \to \mathbb{R}$ strongly-convex function. $\boldsymbol{x} \in \mathbb{R}^d$. Δ^d: d-dim simplex.

Smoothed max operator [Moreau, 1965, Nesterov, 2005]

$$\max_\Omega(\boldsymbol{x}) = \max_{\boldsymbol{y} \in \Delta^d} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \sum_{i=1}^d \Omega(y_i)$$

Max smoothing

$\Omega : \mathbb{R} \to \mathbb{R}$ strongly-convex function. $\boldsymbol{x} \in \mathbb{R}^d$. Δ^d: d-dim simplex.

Smoothed max operator [Moreau, 1965, Nesterov, 2005]

$$\max_\Omega(\boldsymbol{x}) = \max_{\boldsymbol{y} \in \Delta^d} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \sum_{i=1}^d \Omega(y_i)$$

Properties:

- Consistent smoothing: $\max_0(\boldsymbol{x}) = \max(\boldsymbol{x})$
- Twice differentiable almost everywhere with non-zero Hessian

Dynamic programming regularization

What we have at hand

1. **Smooth max:** $\max_\Omega(\boldsymbol{x}) = \max_{\boldsymbol{y} \in \Delta^d} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \sum_{i=1}^d \Omega(y_i)$
2. **Bellman recursion:** $v_i = \max_{j \in \mathcal{P}_i} \theta_{i,j} + v_j, \quad \mathrm{DP}(\boldsymbol{\Theta}) \triangleq v_N$

Dynamic programming regularization

What we have at hand

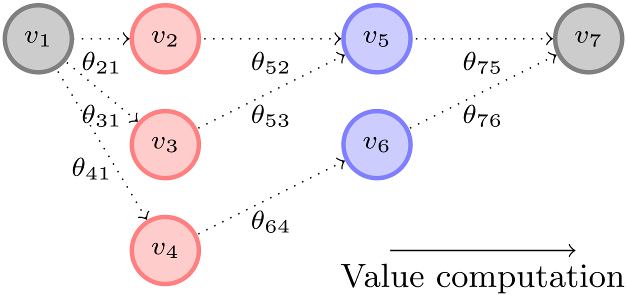
1. **Smooth max:** $\max_\Omega(\boldsymbol{x}) = \max_{\boldsymbol{y}\in\Delta^d} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \sum_{i=1}^d \Omega(y_i)$
2. **Bellman recursion:** $v_i = \max_{j\in\mathcal{P}_i} \theta_{i,j} + v_j, \quad \mathrm{DP}(\boldsymbol{\Theta}) \triangleq v_N$

Bottom-up construction

For all $i \in [N]$:

$$v_i(\boldsymbol{\theta}) = \max{}_\Omega(\theta_{i,j} + v_j)_{j\in\mathcal{P}_i}$$

$$\mathrm{DP}_\Omega(\boldsymbol{\theta}) \triangleq v_N(\boldsymbol{\theta})$$



Regularized best-path: $\nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$

From max to smoothed max:

$$\boldsymbol{Y}(\boldsymbol{\theta}) = \partial \mathrm{DP}(\boldsymbol{\theta}) \Longrightarrow \boldsymbol{Y}_\Omega(\boldsymbol{\theta}) \triangleq \nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$$

Regularized best-path: $\nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$

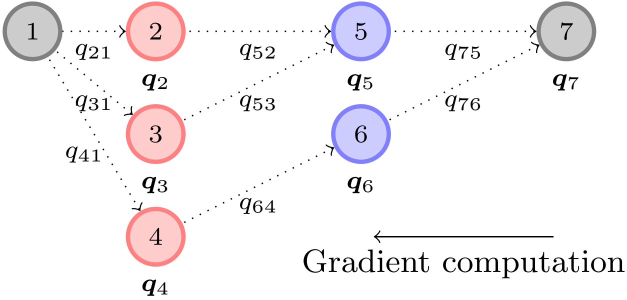
From max to smoothed max:

$$\boldsymbol{Y}(\boldsymbol{\theta}) = \partial \mathrm{DP}(\boldsymbol{\theta}) \Longrightarrow \boldsymbol{Y}_\Omega(\boldsymbol{\theta}) \triangleq \nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$$

Computed with backpropagation

Requirements: Gradients of Bellman equations

$$\boldsymbol{q}_i = \nabla \max{}_\Omega(\theta_{i,j} + v_j)_{j \in \mathcal{P}_i}$$



Entropy and sparsity-inducing ℓ_2^2 regularization

Entropy: $\Omega(x) = \gamma x \log(x) \longrightarrow$ *Softmax* operator

$$\max_\Omega(\boldsymbol{x}) = \log(Z), \text{ where } Z = \textstyle\sum_j \exp(x_j/\gamma)$$

$$\nabla \max_\Omega(\boldsymbol{x}) = (\exp(x_i/\gamma)/Z)_{i \in \mathbb{R}^d}$$

Entropy and sparsity-inducing ℓ_2^2 regularization

Entropy: $\Omega(x) = \gamma x \log(x) \longrightarrow$ *Softmax* operator

$$\max_\Omega(\boldsymbol{x}) = \log(Z), \text{ where } Z = \sum_j \exp(x_j/\gamma)$$

$$\nabla \max_\Omega(\boldsymbol{x}) = (\exp(x_i/\gamma)/Z)_{i \in \mathbb{R}^d}$$

ℓ_2^2 **penalty:** $\Omega(x) = \gamma x^2$ *Sparsemax* [Martins and Astudillo, 2016]

$$\nabla \max_\Omega(\boldsymbol{x}) = \boldsymbol{P}_{\Delta^d}(\boldsymbol{x}/\gamma)$$

Sparse: ℓ_2 projection on simplex

Differentiable DP properties

$\mathrm{DP}_\Omega(\boldsymbol{\theta})$ properties

- $\boldsymbol{\theta} \to \mathrm{DP}_\Omega(\boldsymbol{\theta})$ is convex.
- $\mathrm{DP}_\Omega(\boldsymbol{\theta}) = \mathrm{LP}_\Omega(\boldsymbol{\theta})$ **if and only if** $\Omega = -\gamma H(\boldsymbol{\theta})$

$$\mathrm{LP}_\Omega(\boldsymbol{\theta}) \triangleq \max{}_\Omega \left(\langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle\right)_{\boldsymbol{Y} \in \mathcal{Y}} = \max_{\boldsymbol{p} \in \triangle^D} \left\langle \boldsymbol{p}, (\langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle)_{\boldsymbol{y} \in \mathcal{Y}} \right\rangle - \Omega(\boldsymbol{p})$$

In this (only) case:

- Local Bellman regularization = full LP regularization

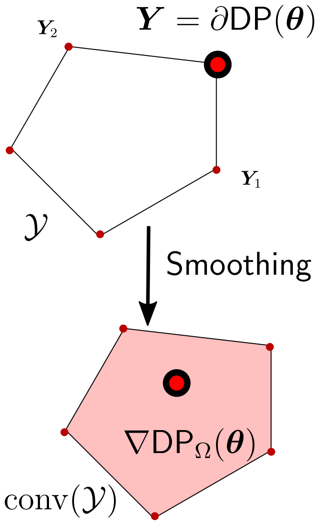
Relaxed gradient properties

Probabilistic interpretation

We can define a distribution $\mathcal{D}_\Omega$ on the set of paths $\mathcal{Y}$ such that

$$\nabla \mathsf{DP}_\Omega(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{D}_\Omega}[\boldsymbol{Y}] \in \mathsf{conv}(\mathcal{Y})$$

Predicted path probabilities: $p_{\boldsymbol{\theta},\Omega}(\boldsymbol{Y})$



Relaxed gradient properties

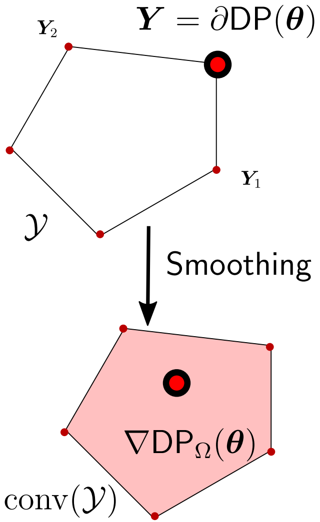
Probabilistic interpretation

We can define a distribution $\mathcal{D}_\Omega$ on the set of paths $\mathcal{Y}$ such that

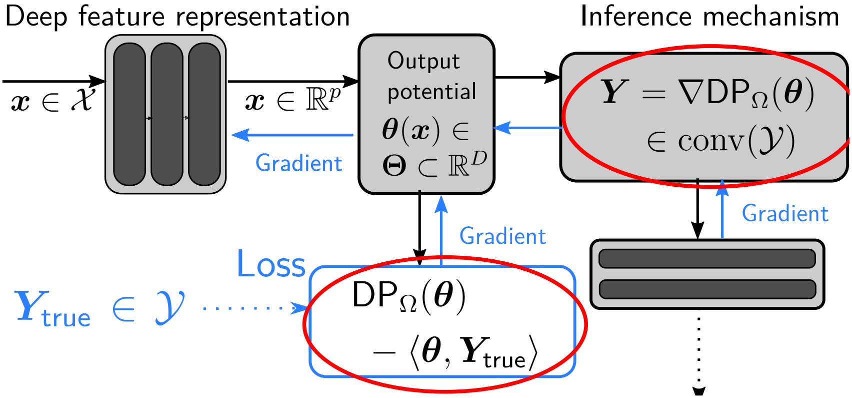
$$\nabla \mathrm{DP}_\Omega(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{D}_\Omega}[\boldsymbol{Y}] \in \mathrm{conv}(\mathcal{Y})$$

Predicted path probabilities: $p_{\boldsymbol{\theta},\Omega}(\boldsymbol{Y})$

- **Negentropy:** Gibbs distribution: $p_{\boldsymbol{\theta},\Omega}(\boldsymbol{Y}) \propto \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle$
- ℓ_2^2: $\mathcal{D}_\Omega$ has a small support $\to \nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$ is sparse

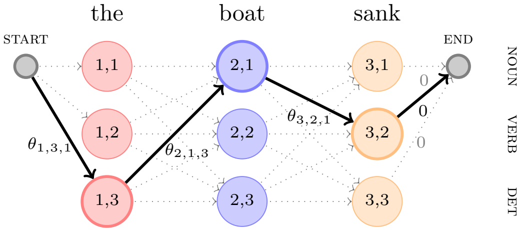


Differentiable dynamic programming layers



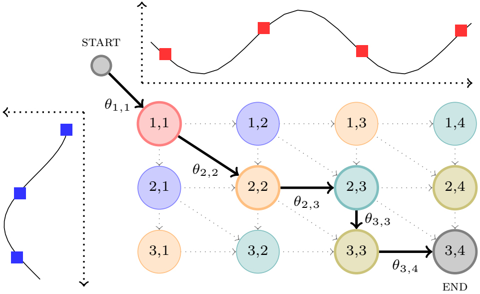
Applications

Viterbi



$$\langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle = \theta_{1,3,1} + \theta_{2,1,3} + \theta_{3,2,1}$$

Dynamic time warping



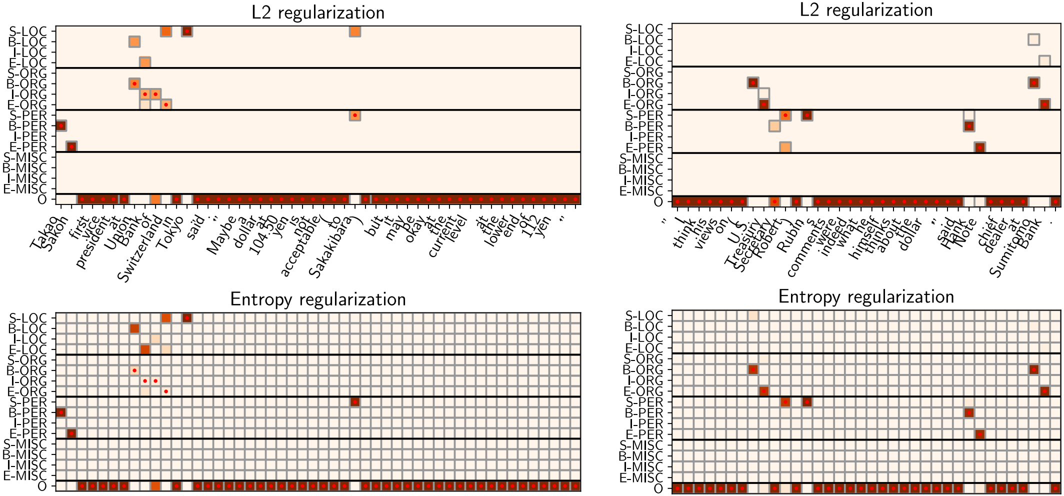
$$\langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle = \theta_{1,1} + \theta_{2,2} + \theta_{2,3} + \theta_{3,3} + \theta_{3,4}$$

$$\nabla \mathrm{Vit}_\Omega : \mathbb{R}^{T \times S \times S} \to \mathbb{R}^{T \times S \times S}$$

$$\nabla \mathrm{DTW}_\Omega : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$$

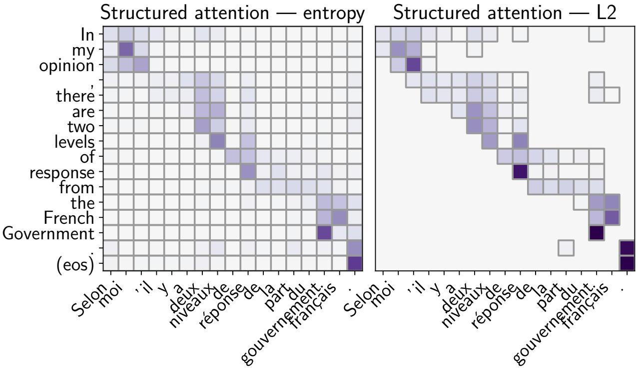
K-best set predictions in named entity recognition

$\Omega = \ell_2^2$: output k-best subset of $\boldsymbol{Y}$ such that $p_{\boldsymbol{\theta},\Omega}(\boldsymbol{Y}) \neq 0$



Structured attention — Neural machine transation

- Compute the attention vector $\boldsymbol{c}$ by marginalizing a 2 state linear-chain CRF.
- Use Vit_Ω, with sparse marginal computation $\Omega = \ell_2^2$.
- Versus simple softmax in original version



Similar BLEU scores WMT14 1M

Attention model	fr→en	en→fr
Softmax	**27.96**	**28.08**
CRF + entropy	**27.96**	27.98
CRF + ℓ_2^2 reg.	27.21	27.28

Conclusion

General framework to put dynamic programming algorithms into arbitrary networks

- Efficient and stable algorithms
- Flexibility of regularization (sparse output)

Experiments

- ℓ_2/entropy have similar performance
- More interpretable outputs / k-best sets with sparsity

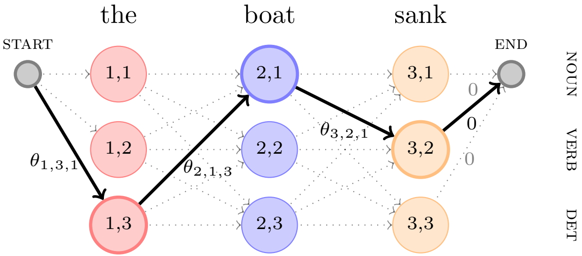
- *PyTorch* package *didyprog* available (fast custom Viterbi and DTW layer)
- Other applications, instantiated algorithms, backprop through $\nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$

Poster #48

Example: Linear conditional random field

$(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ observation, $(y_1, \ldots, y_T) \in [S]^T$ states. $\boldsymbol{Y} \in \mathcal{Y} \in \{0,1\}^{S \times S \times T}$

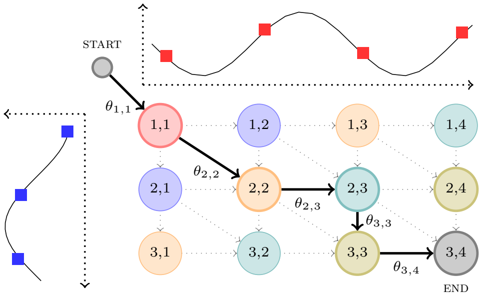
$$\boldsymbol{y} = \underset{\boldsymbol{y} \in \mathcal{Y}}{\operatorname{argmax}} \sum_{t=1}^{T} \theta_t(y_t, y_{t-1}, \boldsymbol{x}_t) = \underset{\boldsymbol{Y} \in \mathcal{Y}}{\operatorname{argmax}} \langle \boldsymbol{\theta}, \boldsymbol{Y} \rangle$$



$$\langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle = \theta_{1,3,1} + \theta_{2,1,3} + \theta_{3,2,1}$$

$\boldsymbol{Y}$ computed with dynamic programming = **Viterbi algorithm.**

Example: Dynamic time warping



$$\langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle = \theta_{1,1} + \theta_{2,2} + \theta_{2,3} + \theta_{3,3} + \theta_{3,4}$$

Elastic matching

- Two time-series $\boldsymbol{A}$, $\boldsymbol{B}$
- Distance matrix: *e.g.*, $\boldsymbol{\theta}_{i,j} = \|a_i - b_i\|_2^2$

Alignment matrices

- $(1,1) \to (N_A, N_B)$
- $\downarrow, \to, \searrow$ moves

Best alignment: $\boldsymbol{Y}(\boldsymbol{A}, \boldsymbol{B}) = \underset{\boldsymbol{Y} \in \mathcal{Y}}{\operatorname{argmax}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle$

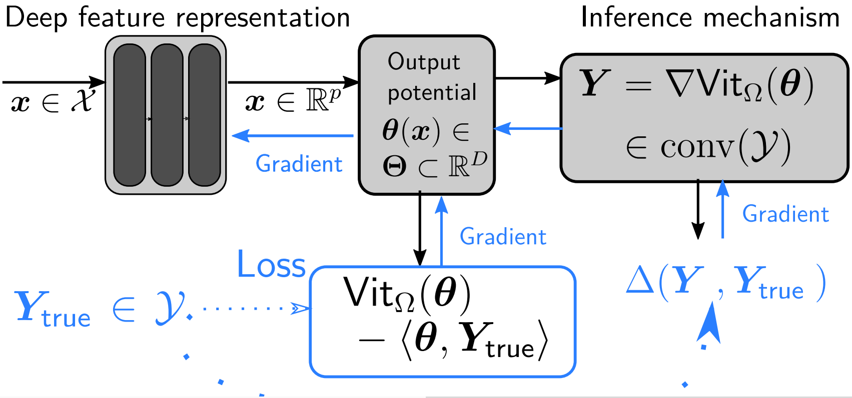
DTW distance: $d(\boldsymbol{A}, \boldsymbol{B}) = \max_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle$

Computable by dynamic programming

- $\mathcal{Y}$ set of alignment matrices
- $\boldsymbol{\theta}$ distance matrix

Named entity recognition

- **Input data:** Sentences x of length T
- **Labels Y:** {Begin/Inside/Outside}{Person/Org./Loc./Misc.}
- **Model:** Char + Word LSTM + **smooth inference mechanism**



Bibliography I

[Bellman, 1958] Bellman, R. (1958).
On a routing problem.
Quarterly of applied mathematics, 16(1):87–90.

[Cuturi and Blondel, 2017] Cuturi, M. and Blondel, M. (2017).
Soft-DTW: a Differentiable Loss Function for Time-Series.
In *Proc. of ICML*, pages 894–903.

[Danskin, 1966] Danskin, J. M. (1966).
The theory of max-min, with applications.
SIAM Journal on Applied Mathematics, 14(4):641–664.

[Kim et al., 2017] Kim, Y., Denton, C., Hoang, L., and Rush, A. M. (2017).
Structured Attention Networks.
In *Proc. of ICLR*.

Bibliography II

[Lampele et al., 2016] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016).

Neural architectures for named entity recognition.

In *Proc. of NAACL*, pages 260–270.

[LeCun et al., 2006] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. (2006).

A tutorial on energy-based learning.

Predicting structured data, 1(0).

[Martins and Astudillo, 2016] Martins, A. F. and Astudillo, R. F. (2016).

From softmax to sparsemax: A sparse model of attention and multi-label classification.

In *Proc. of ICML*, pages 1614–1623.

Bibliography III

[Moreau, 1965] Moreau, J.-J. (1965).
Proximité et dualité dans un espace hilbertien.
Bullet de la Société Mathémathique de France, 93(2):273–299.

[Nesterov, 2005] Nesterov, Y. (2005).
Smooth minimization of non-smooth functions.
Mathematical Programming, 103(1):127–152.