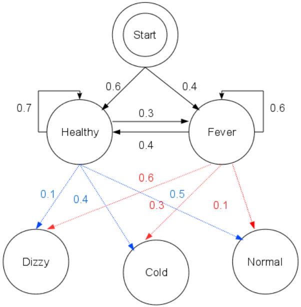
Differentiable dynamic programming for structured prediction and attention

Arthur Mensch

École Normale Supérieure
Département de Mathématiques Appliquées
Paris, France

January 29, 2019

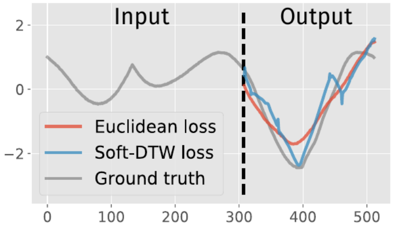
Dynamic programming in machine learning



(Fig: Wikipedia)

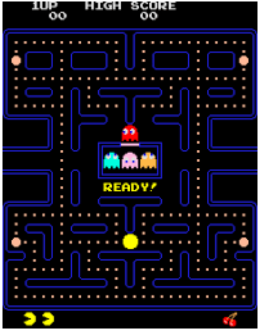
Belief propagation

Viterbi algorithm



(Fig: Cuturi & Blondel)

Dynamic time warping



Value iteration

New layers for deep predictive modelling

Dynamic programming

- Function evaluation on a big space
- From evaluation on smaller space (divide and conquer)

Modern machine learning

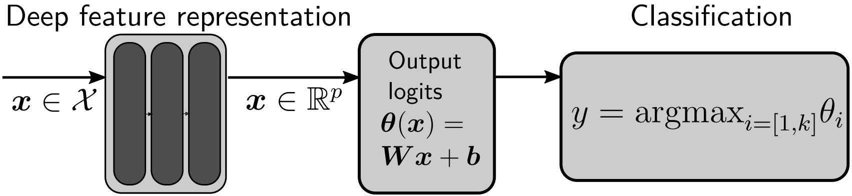
- Modular models, trainable with gradient descent

This talk: Principled differentiable dynamic programming layers

- Application to complex predictive models (e.g. attention mechanisms)
- Sparse output probabilities in structured setting

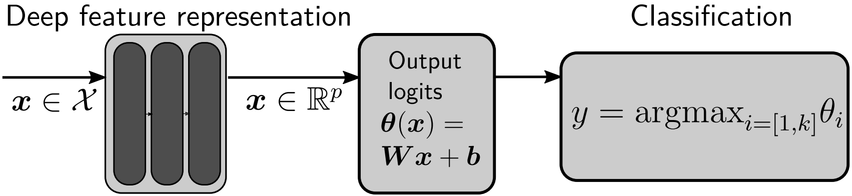
Modular structured prediction: potentials + linear programming

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



Modular structured prediction: potentials + linear programming

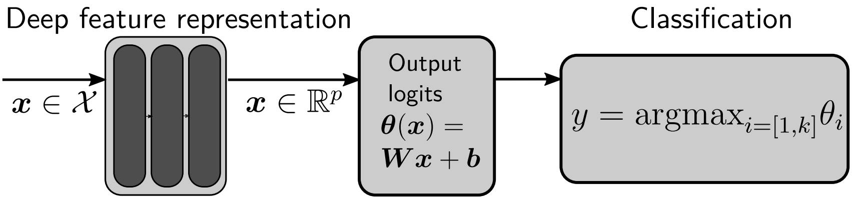
Classification: $\mathcal{Y} = [1, k]$



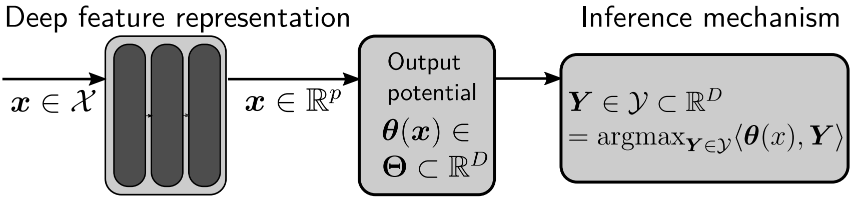
Structured output ? $\mathcal{Y} \subset \mathbb{R}^D$ (edges of a polytope), *e.g.* a tag sequence

Modular structured prediction: potentials + linear programming

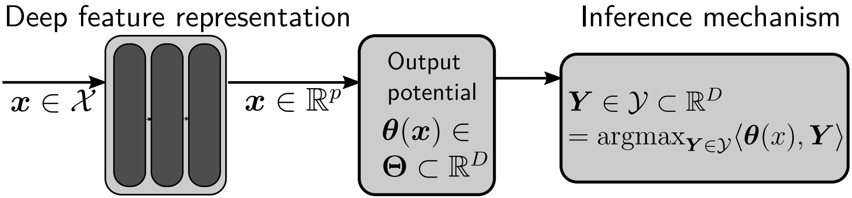
Classification: $\mathcal{Y} = [1, k]$



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

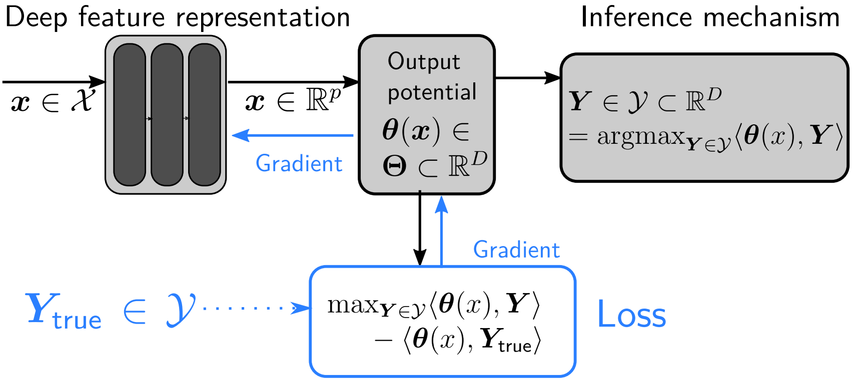


Structure prediction:



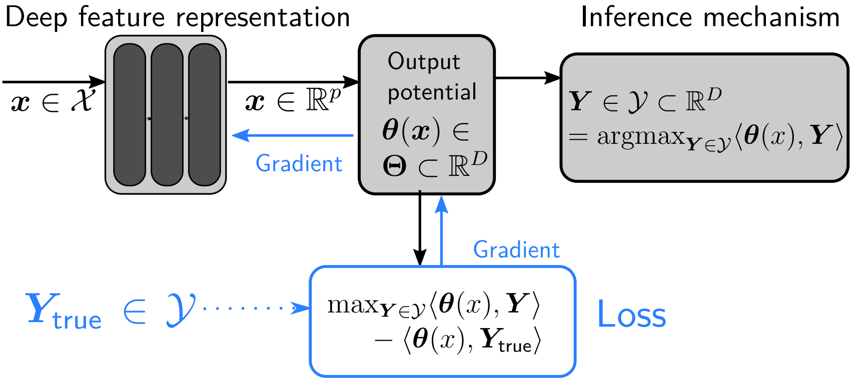
Training

Structure prediction: *Structured perceptron loss*



Training

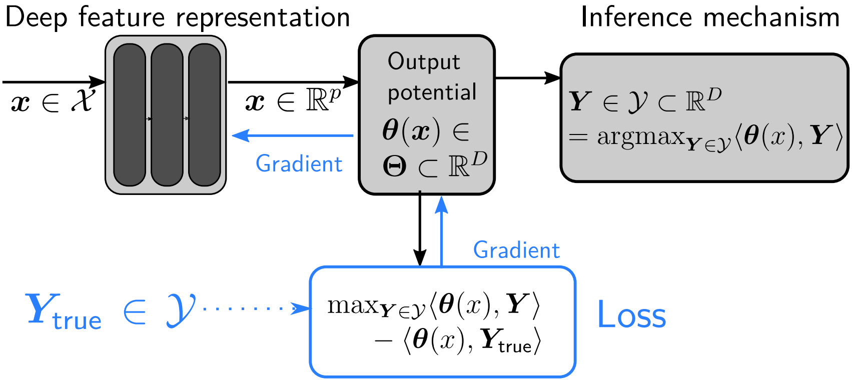
Structure prediction: *Structured perceptron loss*



Backprop through the max operator.

Training

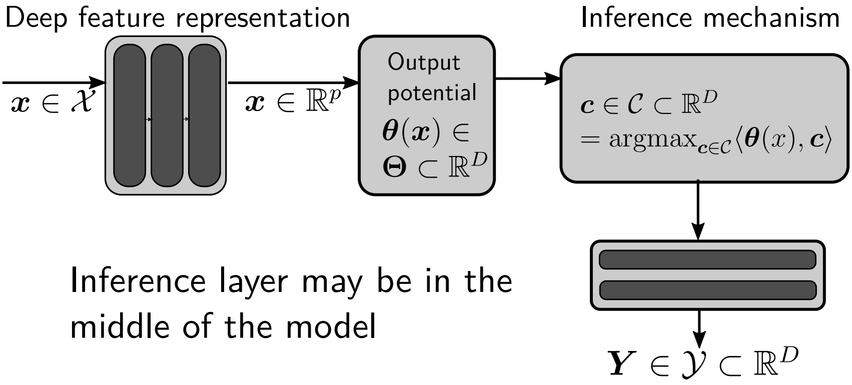
Structure prediction: *Structured perceptron loss*



Backprop through the max operator. Not differentiable everywhere.

Structured prediction as an inner layer

Example: Attention mechanisms,[1] where $\boldsymbol{c}$ are the attention weights.

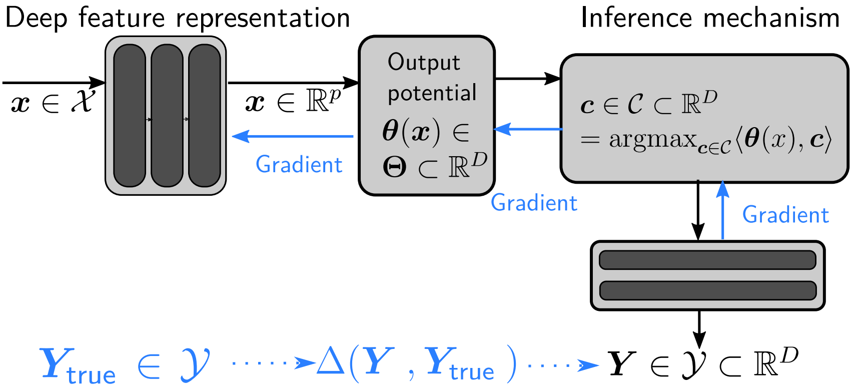


Inference layer may be in the middle of the model

[1] Dzmitry Bahdanau et al. (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *Proc. of ICLR*.

Structured prediction as an inner layer

Example: Attention mechanisms,[1] where $\boldsymbol{c}$ are the attention weights.

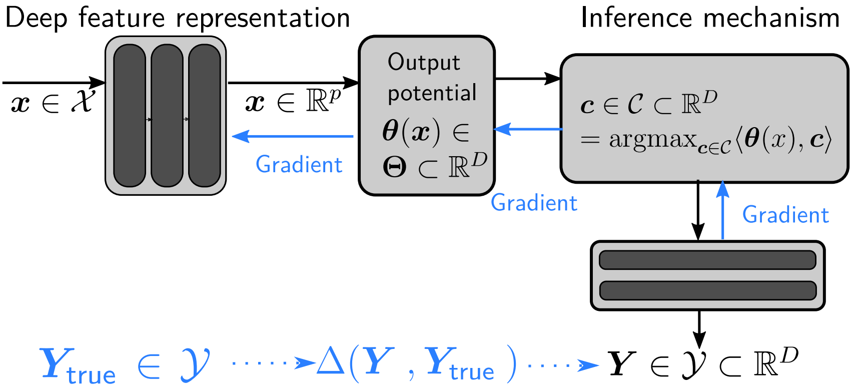


We need to backpropagate through the argmax.

[1] Dzmitry Bahdanau et al. (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *Proc. of ICLR*.

Structured prediction as an inner layer

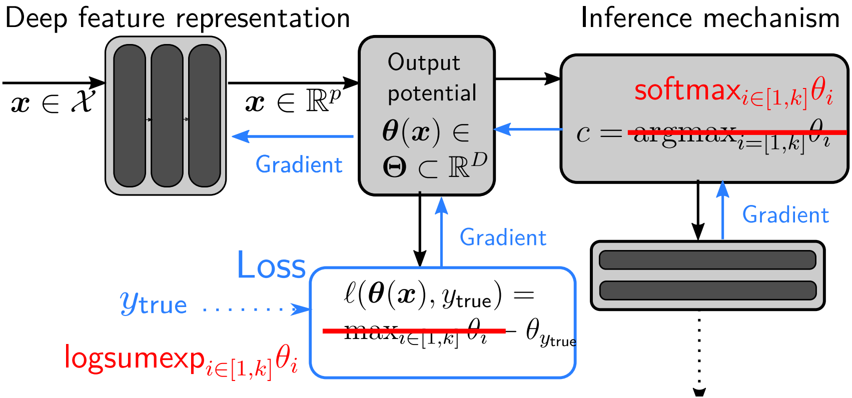
Example: Attention mechanisms,[1] where $\boldsymbol{c}$ are the attention weights.



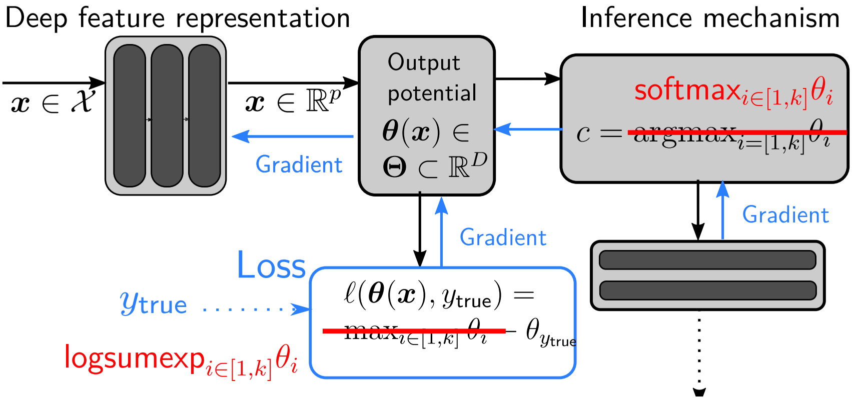
We need to backpropagate through the argmax. Zero gradient.

[1] Dzmitry Bahdanau et al. (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *Proc. of ICLR*.

Gradient from regularization: from max to softmax



Gradient from regularization: from max to softmax



Multinomial loss, softmax attention: **differentiable layers**

Questions and goal

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

Questions and goal

- 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 goal

- 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?

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

Focus on inference mechanisms that relies on **dynamic programming**

- Smooth **max** layers for new structured losses
- Differentiable **argmax** layers for test and 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[2,3,4,5]

[2] Yann LeCun et al. (2006). "A tutorial on energy-based learning". In: *Predicting structured data* 1.0.

[3] Guillaume Lample et al. (2016). "Neural Architectures for Named Entity Recognition". In: *Proc. of NAACL*, pp. 260–270.

[4] Yoon Kim et al. (2017). "Structured Attention Networks". In: *Proc. of ICLR*.

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

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

Dynamic programming as best path in a DAG

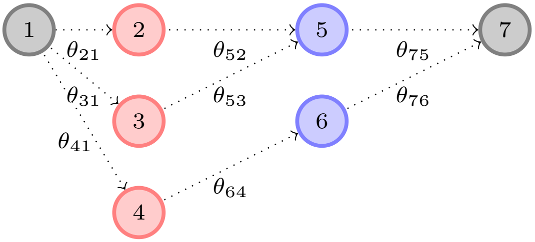
Directed acyclic graph

- $G = (\mathcal{N}, \mathcal{E})$, 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)

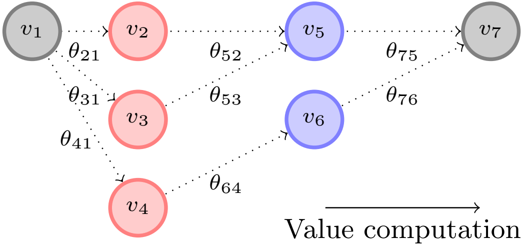
- **Max value from 1 to i**

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

- 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)

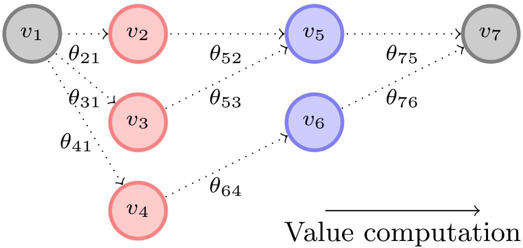
- **Max value from 1 to i**

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

- One pass over the graph

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

= **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: $\partial_{\boldsymbol{\theta}} \mathrm{DP}(\boldsymbol{\theta}) = \mathrm{argmax}_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle$

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

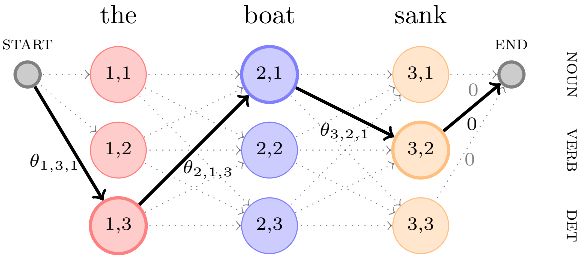
Dynamic programming layers

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

Example: Linear conditional random field

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

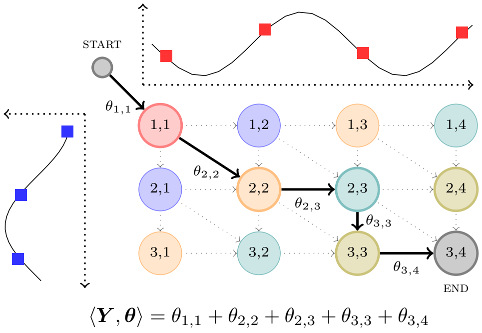
$$\boldsymbol{y} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \sum_{t=1}^{T} \theta_t(y_t, y_{t-1}, \boldsymbol{x}_t) = \operatorname*{argmax}_{\boldsymbol{Y} \in \mathcal{Y}} \langle \boldsymbol{\theta}, \boldsymbol{Y} \rangle$$



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

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

Example: Dynamic time warping



Elastic matching

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

Alignment matrices

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

Regularizing dynamic programming

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

Regularizing dynamic programming

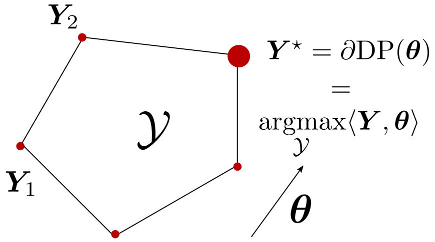
Obstacles to end-to-end training

- 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

Regularizing dynamic programming

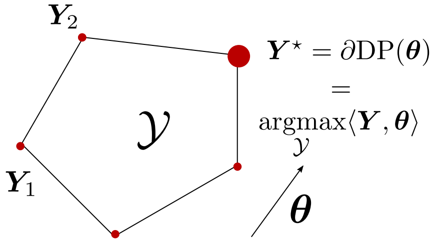
Obstacles to end-to-end training

- 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

Examples of regularization

Shannon entropy: $\Omega(x) = x \log(x) \longrightarrow$ *Softmax* operator

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

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

Examples of regularization

Shannon entropy: $\Omega(x) = x \log(x) \longrightarrow$ *Softmax* operator

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

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

ℓ_2^2 **norm:** $\Omega(x) = x^2 \longrightarrow$ *Sparsemax* (Martins and Astudillo, 2016)

$\nabla \max_\Omega(\boldsymbol{x}) = \operatorname{argmin}_{p \in \triangle^d} \|\boldsymbol{x} - \boldsymbol{p}\|_2^2$ Sparse: eucl. projection on simplex

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

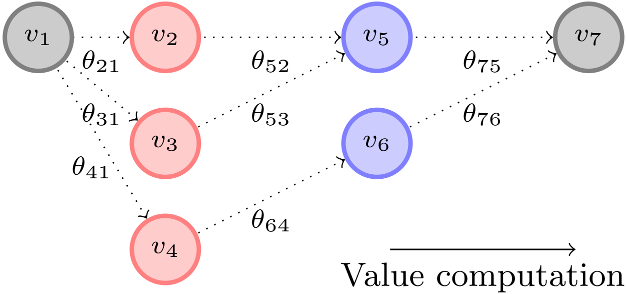
1. **Smooth max:** $\max_\Omega(\boldsymbol{x}) = \max_{\boldsymbol{y} \in \Delta^d} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \sum_{i=1}^d \Omega(y_i)$
2. **Bellman recursion:** $v_i = \max_{j \in \mathcal{P}_i} \theta_{i,j} + v_j, \quad \mathrm{DP}(\boldsymbol{\Theta}) \triangleq v_N$

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

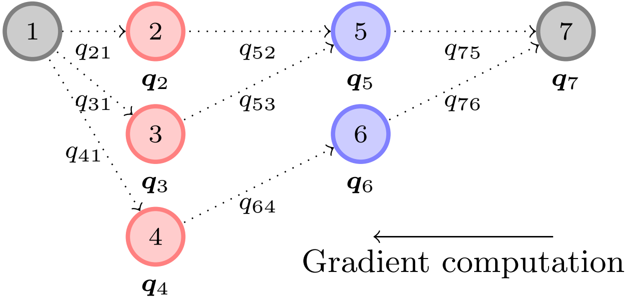
From max to smoothed max:

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

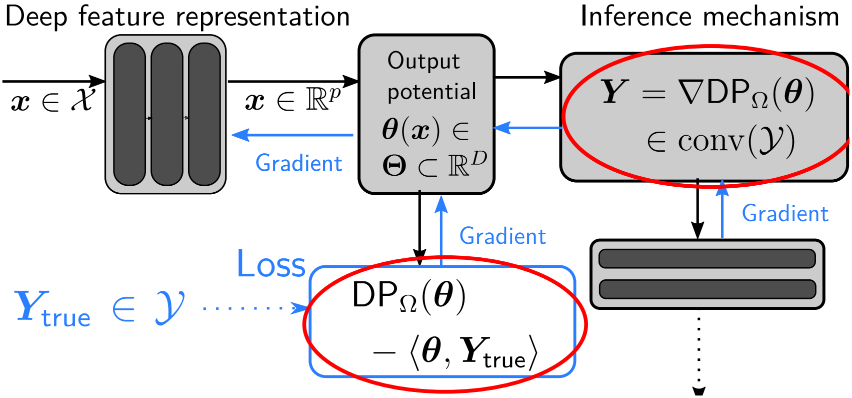
Computed with backpropagation

Requirements: Gradients of Bellman equations

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



Differentiable dynamic programming layers



Differentiable DP properties

Usable for loss design: $\boldsymbol{\theta} \to \mathrm{DP}_\Omega(\boldsymbol{\theta})$ is convex, bounds $\mathrm{DP}(\boldsymbol{\theta})$

Differentiable DP properties

Usable for loss design: $\boldsymbol{\theta} \to \mathrm{DP}_\Omega(\boldsymbol{\theta})$ is convex, bounds $\mathrm{DP}(\boldsymbol{\theta})$

Is local regularization equivalent to global regularization ?

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

Theorem:

- $\mathrm{DP}_\Omega(\boldsymbol{\theta}) = \mathrm{LP}_\Omega(\boldsymbol{\theta})$ **if and only if** $\Omega(p) = -\gamma \sum_i p_i \log p_i$
- $\mathrm{DP}_\Omega(\boldsymbol{\theta})$ is the CRF (Lafferty et al., 2001) log-partition

$$\mathrm{DP}_\Omega(\boldsymbol{\theta}) = \log\Big(\sum_{\boldsymbol{Y}\in\mathcal{Y}} \exp(\langle \boldsymbol{\theta}, \boldsymbol{Y}\rangle)\Big)$$

New motivation for Shannon reg. But ℓ_2^2 has other interesting properties.

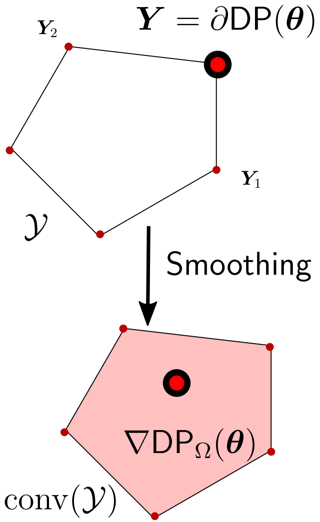
Relaxed gradient properties

Probabilistic interpretation

Backprop defines a distribution $\mathcal{D}_\Omega$ on the set of paths $\mathcal{Y}$

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

$\Rightarrow$ Probability of path $\boldsymbol{Y}$: $p_{\boldsymbol{\theta},\Omega}(\boldsymbol{Y})$



Relaxed gradient properties

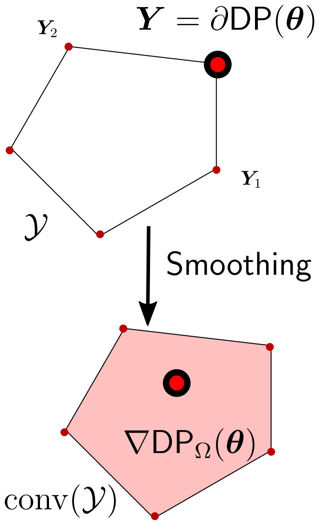
Probabilistic interpretation

Backprop defines a distribution $\mathcal{D}_\Omega$ on the set of paths $\mathcal{Y}$

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

$\Rightarrow$ Probability of path $\boldsymbol{Y}$: $p_{\boldsymbol{\theta},\Omega}(\boldsymbol{Y})$

- **Shannon:** 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**



Backpropagating through $\nabla \mathrm{DP}_\Omega(\boldsymbol{\Theta})$

Regularized best-path layer: $\boldsymbol{\theta} \in \mathbb{R}^{N \times N} \to \nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$

Jacobian ? $\nabla \nabla \mathrm{DP}_\Omega(\boldsymbol{\Theta}) = \nabla^2 \mathrm{DP}_\Omega(\boldsymbol{\Theta}) = \text{Hessian}$

Hessian vector-product

$$\nabla(\nabla \mathrm{DP}_\Omega(\boldsymbol{\Theta}))\boldsymbol{Z} = \nabla^2 \mathrm{DP}_\Omega(\boldsymbol{\Theta})\boldsymbol{Z}, \qquad \boldsymbol{Z} \in \mathbb{R}^{n \times n} \quad \text{direction}$$

Computable in $\mathcal{O}(|\mathcal{E}|)$: reverse-on-forward differentiation

Differentiable dynamic programming layers

Highest-score layer, forward-pass

$$\boldsymbol{\theta} \in \mathbb{R}^{N \times N} \to \mathrm{DP}_\Omega(\boldsymbol{\theta})$$

Highest score layer, backward pass
Best path layer, forward-pass

$$\boldsymbol{\theta} \in \mathbb{R}^{N \times N} \to \nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$$

Differentiable dynamic programming layers

Highest-score layer, forward-pass

$$\boldsymbol{\theta} \in \mathbb{R}^{N \times N} \to \mathrm{DP}_\Omega(\boldsymbol{\theta})$$

Highest score layer, backward pass
Best path layer, forward-pass

$$\boldsymbol{\theta} \in \mathbb{R}^{N \times N} \to \nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$$

Best-path layer: backward pass

$$\boldsymbol{\theta}, Z \in \mathbb{R}^{N \times N} \times \mathbb{R}^{N \times N} \to \nabla^2 \mathrm{DP}_\Omega(\boldsymbol{\theta}) Z$$

Differentiable dynamic programming layers

Highest-score layer, forward-pass

$$\boldsymbol{\theta} \in \mathbb{R}^{N \times N} \to \mathrm{DP}_\Omega(\boldsymbol{\theta})$$

Highest score layer, backward pass
Best path layer, forward-pass

$$\boldsymbol{\theta} \in \mathbb{R}^{N \times N} \to \nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$$

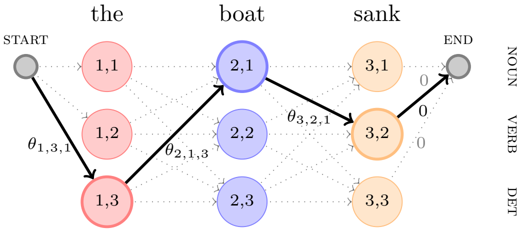
Best-path layer: backward pass

$$\boldsymbol{\theta}, \boldsymbol{Z} \in \mathbb{R}^{N \times N} \times \mathbb{R}^{N \times N} \to \nabla^2 \mathrm{DP}_\Omega(\boldsymbol{\theta}) \boldsymbol{Z}$$

- Sparse/dense output with ℓ_2/entropy regularization
- Total computational cost: $\mathcal{O}(|\mathcal{E}|)$

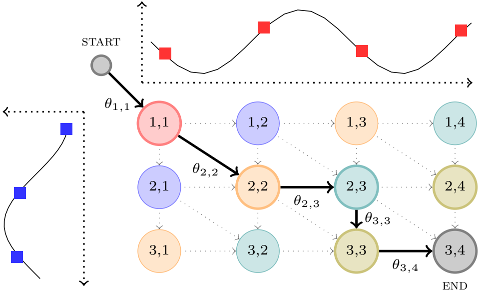
Applications

Viterbi



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

Dynamic time warping



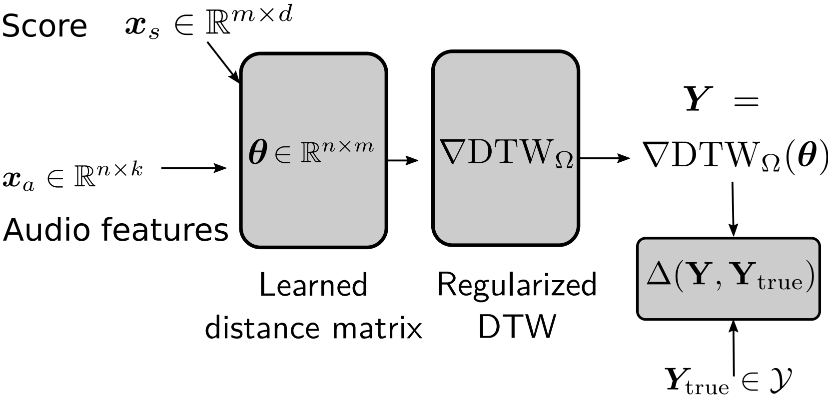
$$\langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle = \theta_{1,1} + \theta_{2,2} + \theta_{2,3} + \theta_{3,3} + \theta_{3,4}$$

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

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

Audio-to-score alignment

- **Input data:** audio sequence $\boldsymbol{x}_a \in \mathbb{R}^{n \times k}$, one-hot key sequence $\boldsymbol{x}_s \in \mathbb{R}^{m \times d}$
- **Labels:** Alignment $\boldsymbol{Y}_{\text{true}} \in \mathcal{Y} \subset \mathbb{R}^{n \times m}$



Metric learning experiment

Learn the distance matrix:

- Baseline: multinomial classification, audio-frame to score key
- Our model: end-to-end training of a linear model with final soft-DTW layer

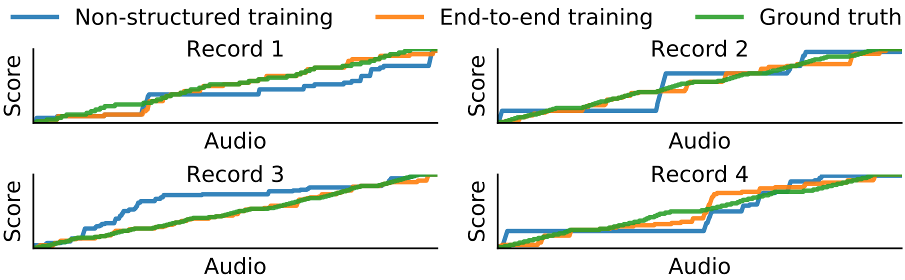
Data: Supervised dataset: 10 annotated Bach quatuors (**Bach10**)

Validation:

- Leave-one-out prediction
- At test time: Hard DTW on the learned distance matrix
- RMSE between predicted onsets

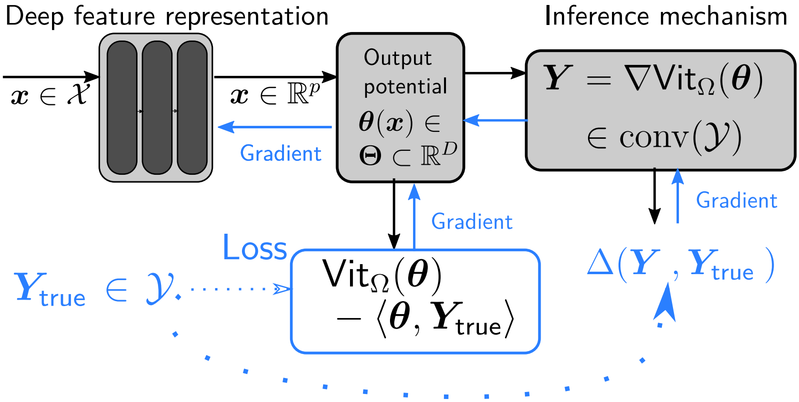
Results

RMSE	Test set	Train set
End-to-end training	$\mathbf{1.26 \pm 0.64}$	$\mathbf{0.17 \pm 0.01}$
Non-structure training	3.70 ± 2.85	1.80 ± 0.14
Random	14.64 ± 2.63	14.64 ± 0.29



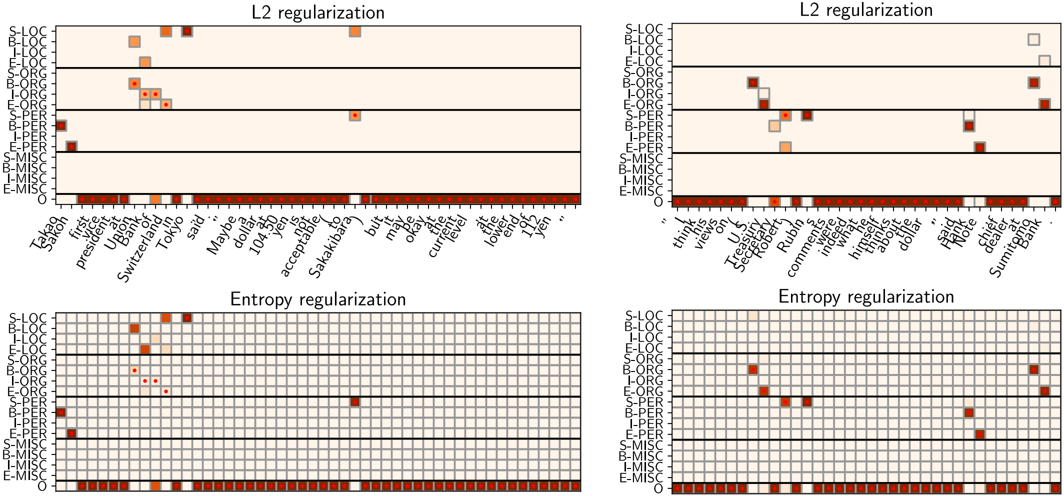
Named entity recognition

- **Input data:** Sentences $\boldsymbol{x}$ of length T
- **Labels** $\boldsymbol{Y}$**:** {Begin/Inside/Outside}{Person/Org./Loc./Misc.}



K-best set predictions in named entity recognition

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



Quantitative comparison of losses

- **Potential-convex loss:** $\ell_\Omega(\boldsymbol{\theta}, \boldsymbol{Y}) = \mathrm{DP}_\Omega(\boldsymbol{\theta}) - \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle$
- **Cost-sensitive loss:** $\ell_\Omega(\boldsymbol{\theta}, \boldsymbol{Y}) = \Delta(\nabla \mathrm{DP}_\Omega(\boldsymbol{\theta}), \boldsymbol{Y}\rangle$.

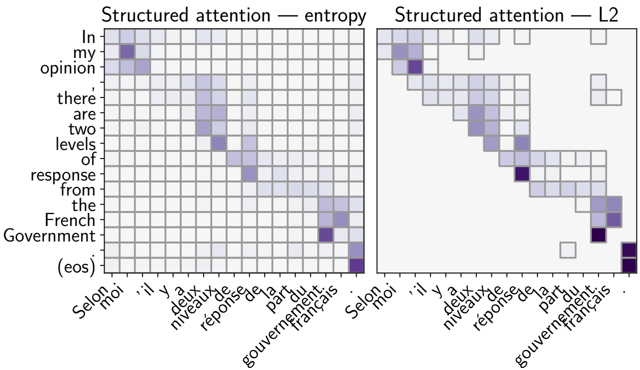
Ω	Loss	English	Spanish	German	Dutch
Negentropy	Convex loss	90.80	**86.68**	77.35	**87.56**
	Cost-sensitive	90.47	86.20	**77.56**	87.37
ℓ_2^2	Convex loss	**90.86**	85.51	76.01	86.58
	Cost-sensitive	89.49	84.07	76.91	85.90
Lample et al., 2016[6]		*90.96*	*85.75*	*78.76*	*81.74*

- ℓ_2^2 reg. achieves comparable accuracy with more interpretable predictions
- Training directly from potential-derived losses is slightly better

[6] Guillaume Lample et al. (2016). "Neural Architectures for Named Entity Recognition". In: *Proc. of NAACL*, pp. 260–270.

Structured attention — Neural machine transation

- Compute an attention vector $\boldsymbol{c}$: 2 state linear-chain CRF
- $\boldsymbol{c} = \mathbb{E}[\boldsymbol{z}]$, $z_i = 1$ if attention, $z_i = 0$ if no-attention
- Use Vit_Ω, with sparse marginal computation $\Omega = \ell_2^2$.

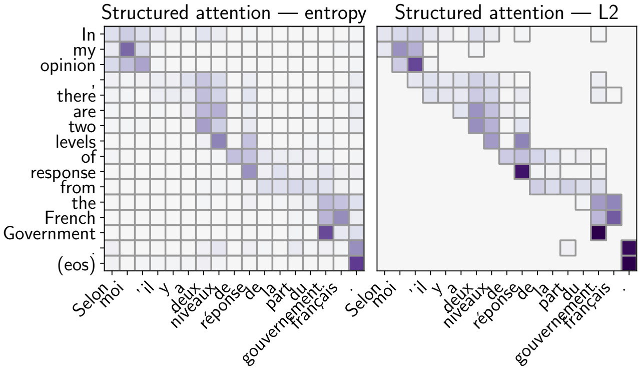


Similar BLEU scores WMT14 1M

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

Structured attention — Neural machine transation

- Compute an attention vector $\boldsymbol{c}$: 2 state linear-chain CRF
- $\boldsymbol{c} = \mathbb{E}[\boldsymbol{z}]$, $z_i = 1$ if attention, $z_i = 0$ if no-attention
- Use Vit_Ω, with sparse marginal computation $\Omega = \ell_2^2$.



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

Block sparse attention

Conclusion

General framework to use DP algorithms in arbitrary networks

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

Experiments: ℓ_2/entropy have similar performance

- ℓ_2^2: More interpretable outputs / k-best sets with sparsity

PyTorch package *didyprog* available (fast custom Viterbi and DTW layer)

Conclusion

General framework to use DP algorithms in arbitrary networks

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

Experiments: ℓ_2/entropy have similar performance

- ℓ_2^2: More interpretable outputs / k-best sets with sparsity

PyTorch package *didyprog* available (fast custom Viterbi and DTW layer)

Arthur Mensch and Mathieu Blondel (2018). "Differentiable Dynamic Programming for Structured Prediction and Attention". In: *Proceedings of the International Conference on Machine Learning*

Conclusion

General framework to use DP algorithms in arbitrary networks

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

Experiments: ℓ_2/entropy have similar performance

- ℓ_2^2: More interpretable outputs / k-best sets with sparsity

PyTorch package *didyprog* available (fast custom Viterbi and DTW layer)

Arthur Mensch and Mathieu Blondel (2018). "Differentiable Dynamic Programming for Structured Prediction and Attention". In: *Proceedings of the International Conference on Machine Learning*

Related work at Google: Framwork formalizes differentiable beam search (Goyal et al., 2017), similar effor in reinforcement learning (Haarnoja et al., 2018)

Bibliography I

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *Proc. of ICLR*.

Bellman, Richard (1958). "On a routing problem". In: *Quarterly of applied mathematics* 16.1, pp. 87–90.

Cuturi, Marco and Mathieu Blondel (2017). "Soft-DTW: a Differentiable Loss Function for Time-Series". In: *Proc. of ICML*, pp. 894–903.

Danskin, John M (1966). "The theory of max-min, with applications". In: *SIAM Journal on Applied Mathematics* 14.4, pp. 641–664.

Goyal, Kartik et al. (July 2017). "A Continuous Relaxation of Beam Search for End-to-End Training of Neural Sequence Models". In: *arXiv:1708.00111 [cs]*. arXiv: `1708.00111 [cs]`.

Bibliography II

Haarnoja, Tuomas et al. (2018). "Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". en. In: *Proceedings of the International Conference on Machine Learning*, p. 10.

Kim, Yoon et al. (2017). "Structured Attention Networks". In: *Proc. of ICLR*.

Lafferty, John, Andrew McCallum, and Fernando CN Pereira (2001). "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". In: *Proc. of ICML*, pp. 282–289.

Lample, Guillaume et al. (2016). "Neural Architectures for Named Entity Recognition". In: *Proc. of NAACL*, pp. 260–270.

LeCun, Yann et al. (2006). "A tutorial on energy-based learning". In: *Predicting structured data* 1.0.

Bibliography III

Martins, André F.T. and Ramón Fernandez Astudillo (2016). "From softmax to sparsemax: A sparse model of attention and multi-label classification". In: *Proc. of ICML*, pp. 1614–1623.

Mensch, Arthur and Mathieu Blondel (2018). "Differentiable Dynamic Programming for Structured Prediction and Attention". In: *Proceedings of the International Conference on Machine Learning*.

Moreau, Jean-Jacques (1965). "Proximité et dualité dans un espace hilbertien". In: *Bullet de la Société Mathémathique de France* 93.2, pp. 273–299.

Nesterov, Yurii (2005). "Smooth minimization of non-smooth functions". In: *Mathematical Programming* 103.1, pp. 127–152.