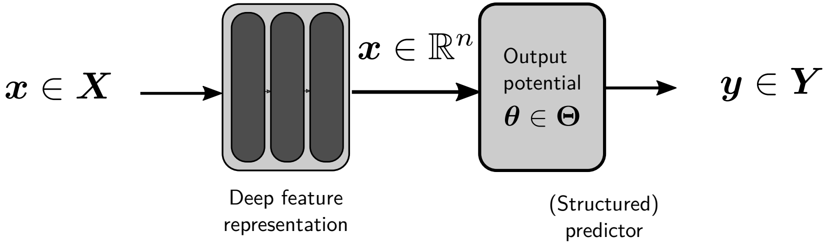
Differentiable dynamic programming for structured prediction and attention

Arthur Mensch, Mathieu Blondel

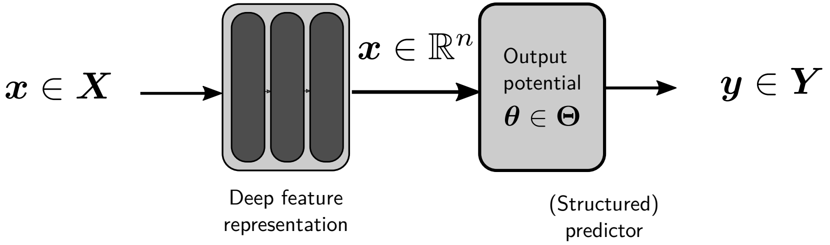
Inria Parietal, Saclay, France
NTT Communication Laboratories, Kyoto, Japan

March 7, 2018

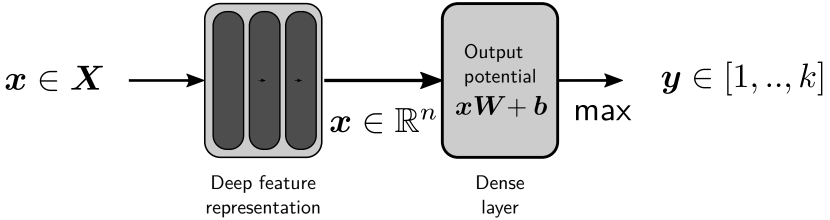
Supervised deep learning



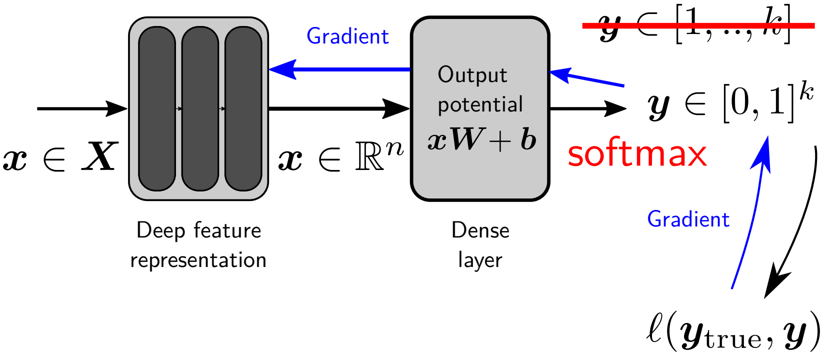
Supervised deep learning



Simplest output structure: one class among many

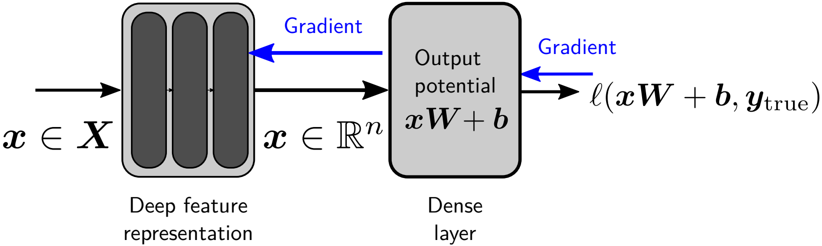


How do we train such a model ?



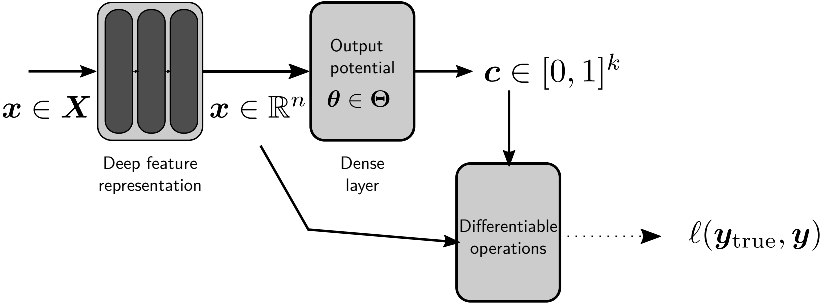
The predictive layer should be differentiable with non zero Jacobian $\longrightarrow$ learn appropriate feature representation.

Training directly from potential



Maximize some affinity function between ground truth $\boldsymbol{y}_{true}$ and potential $\boldsymbol{\theta}$.

Prediction mechanism in the middle of a large network



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

Questions

- What if we wish to predict **structured** output, *e.g.*, tag sequences ($\mathcal{Y}$ is more complex than $[1, .., k]$) ?

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

Contributions

Generic framework for differentiable structured prediction:

- Based on relaxation of **dynamic programming** algorithms.
- Regularizing the max operators with strongly convex penalties.
- Allowing to output **sparse** continuous outputs

Applications:

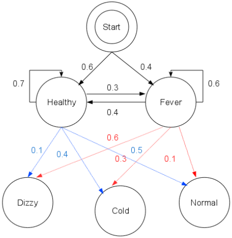
- End-to-end audio to score alignment
- Named entity recognition with sparse predictions
- Block sparse attention mechanisms

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

Differentiable dynamic programming

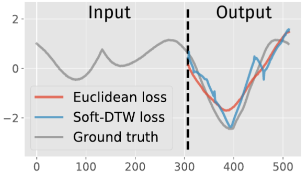
1. Dynamic Programming
   - Structured prediction
2. Differentiable Dynamic Programming
   - Max smoothing
   - Bottom-up construction
   - Backpropagation
3. Applications
   - Audio-to-score alignment
   - Named entity recognition
   - Structured attention — Neural Machine Transation

Dynamic programming in machine learning



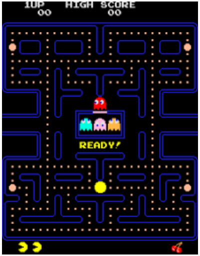
(Fig: Wikipedia)

Belief propagation
Viterbi algorithm



(Fig: Cuturi & Blondel)

Dynamic time warping



Value iteration

Structured prediction

Simplest single class prediction case:

$$\boldsymbol{\theta} = \boldsymbol{x}\boldsymbol{W} + \boldsymbol{b} \quad \text{logits/potentials}$$

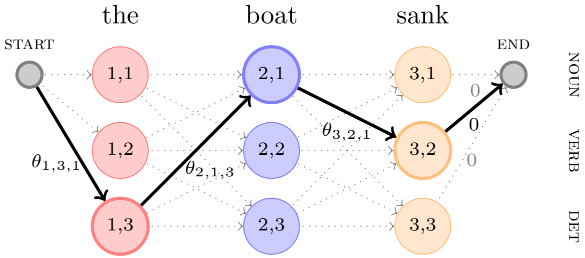
$$\boldsymbol{Y}^\star = \underset{\boldsymbol{Y} \in \mathcal{Y}}{\operatorname{argmax}} \langle \boldsymbol{\theta}, \boldsymbol{Y} \rangle \quad \text{predicted class}$$

where $\mathcal{Y} \in \mathbb{R}^k$ is the set of basis vectors of $\mathbb{R}^k$.

Structured prediction use more complex $\boldsymbol{Y}$, but often solves a similar linear problem:

$$\boldsymbol{Y}^\star = \underset{\boldsymbol{Y} \in \mathcal{Y}}{\operatorname{argmax}} \langle \boldsymbol{\theta}, \boldsymbol{Y} \rangle \quad \text{predicted output}$$

Linear conditional random field (CRF)



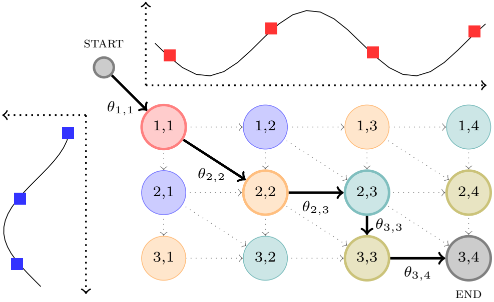
$$\langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle = \theta_{1,3,1} + \theta_{2,1,3} + \theta_{3,2,1}$$

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

$\mathcal{Y} \subset \{0,1\}^{S \times S \times T}$ set of state sequences.

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

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



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

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

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

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

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

Dynamic Programming

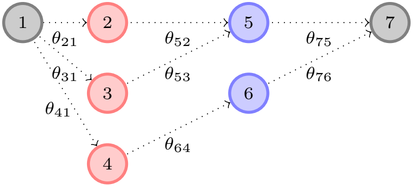
Directed Acyclic Graph

- $G = (\mathcal{N}, \mathcal{E})$, with one root and one leaf
- Nodes numbered in topological 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

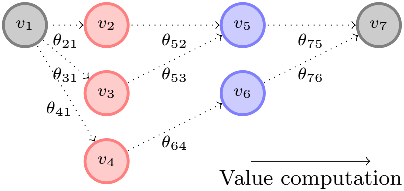
Split the combinatorial problem into subproblems

- **Maximum 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]

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

Argmax path computation

The argmax is computable using backpropagation

Danskin theorem

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

Argmax path computation

The argmax is computable using backpropagation

Danskin theorem

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

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

Dynamic programming layers

- Value layer: $\boldsymbol{\theta} \to \mathrm{DP}(\Omega) = \max_{\boldsymbol{Y}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle$
- Best-path layer: $\boldsymbol{\theta} \to \partial \mathrm{DP}(\Omega) \sim \operatorname{argmax}_{\boldsymbol{Y}} \langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle$

Both layers are useful:

- Value layer used when maximizing target/potential affinity
- Best-path layer outputs a prediction $\boldsymbol{Y}^\star$

The need for regularization

Blocker for end-to-end training

- $\boldsymbol{\theta} \rightarrow \mathrm{DP}(\boldsymbol{\theta})$ is not differentiable everywhere
- $\boldsymbol{\theta} \rightarrow \partial\mathrm{DP}(\boldsymbol{\theta})$ is piecewise constant / ill defined

The need for regularization

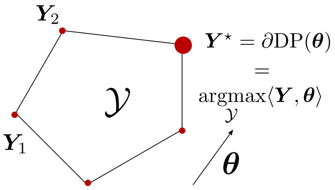
Blocker for end-to-end training

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

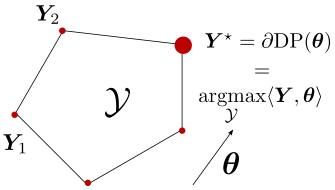
The need for regularization

Blocker for end-to-end training

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

Smooth the maximum operator

Differentiable dynamic programming

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
- For some Ω: **Non-zero Hessian** — allows backpropagation

Regularization examples

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

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

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

Regularization examples

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

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

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

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

$\nabla \max_\Omega(\boldsymbol{x}) = \boldsymbol{P}_{\Delta^d}(\boldsymbol{x}/\gamma) = \boldsymbol{y}^\star$ Sparse: ℓ_2 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:** $\max_\Omega(\boldsymbol{x}) = \max_{\boldsymbol{y}\in\Delta^d} \langle \boldsymbol{x}, \boldsymbol{y}\rangle - \sum_{i=1}^d \Omega(y_i)$

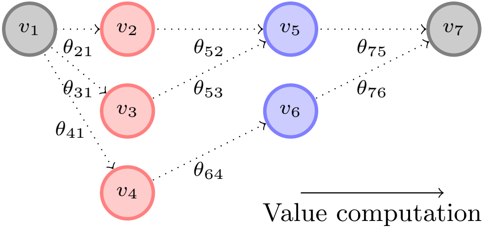
2. **Bellman recursion:** $v_i = \max_{j\in\mathcal{P}_i} \theta_{i,j} + v_j, \quad \mathrm{DP}(\boldsymbol{\Theta}) \triangleq v_N$

Bottom-up construction

For all $i \in [N]$:

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

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



Well defined for all Ω.

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

Best path without smoothing: $\boldsymbol{Y}^\star(\boldsymbol{\theta}) = \partial \mathrm{DP}(\boldsymbol{\theta})$

Regularized best-path:

$$\boldsymbol{Y}_\Omega^\star(\boldsymbol{\theta}) \triangleq \nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$$

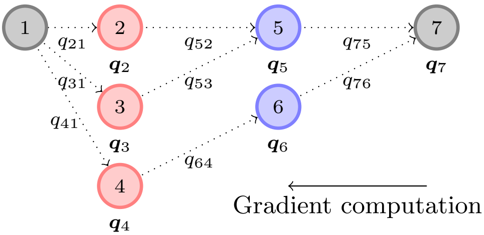
Smooth best-path: $\nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$

Best path without smoothing: $\boldsymbol{Y}^\star(\boldsymbol{\theta}) = \partial \mathrm{DP}(\boldsymbol{\theta})$

Regularized best-path:

$$\boldsymbol{Y}^\star_\Omega(\boldsymbol{\theta}) \triangleq \nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$$

Computed with backpropagation



Requirements: Gradients of Bellman equations

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

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

Summary: differentiable dynamic programming

Highest-score layer, forward-pass

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

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

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

Best-path layer: backward pass

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

Summary: differentiable dynamic programming

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

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

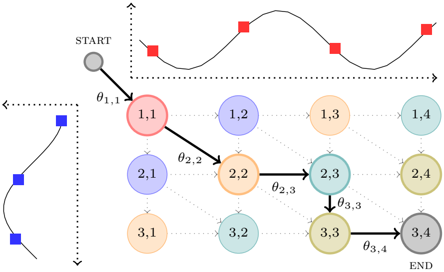
Best-path layer: backward pass

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

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

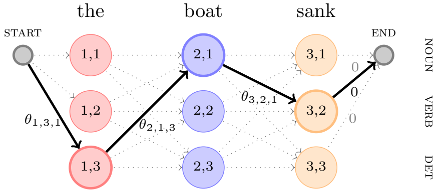
Specialization

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



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

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



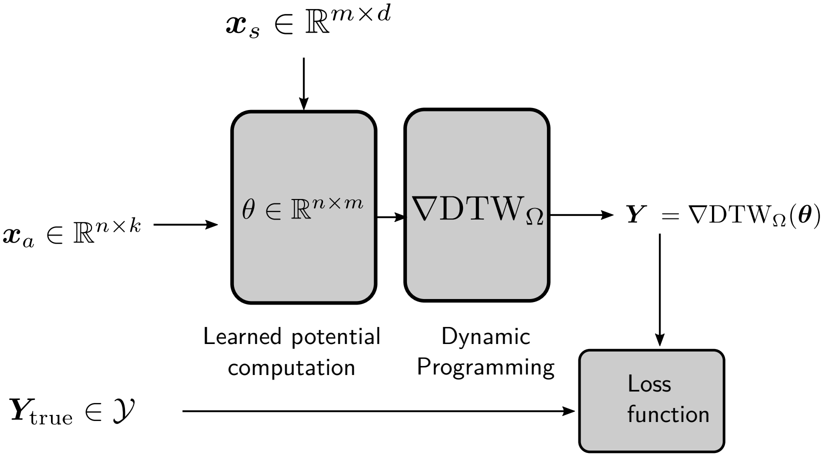
$$\langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle = \theta_{1,3,1} + \theta_{2,1,3} + \theta_{3,2,1}$$

Differentiable dynamic programming

1. Dynamic Programming
   - Structured prediction
2. Differentiable Dynamic Programming
   - Max smoothing
   - Bottom-up construction
   - Backpropagation
3. Applications
   - Audio-to-score alignment
   - Named entity recognition
   - Structured attention — Neural Machine Transation

Audio-to-score alignment

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



Experiment

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

Learn the distance matrix:

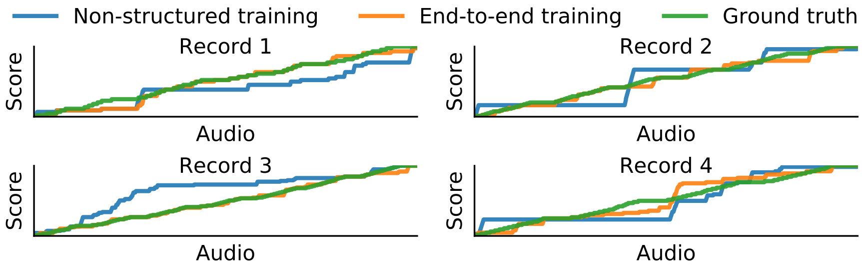
- Baseline: audio frame to key multinomial classification
- Our model: end-to-end training with final soft-DTW layer

Validation:

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

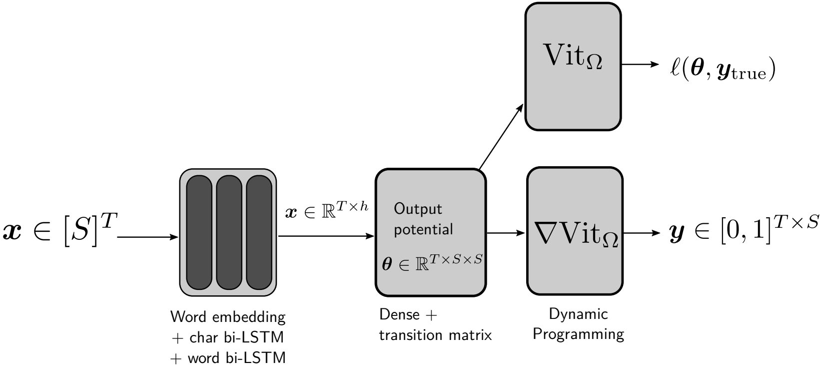
Results

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



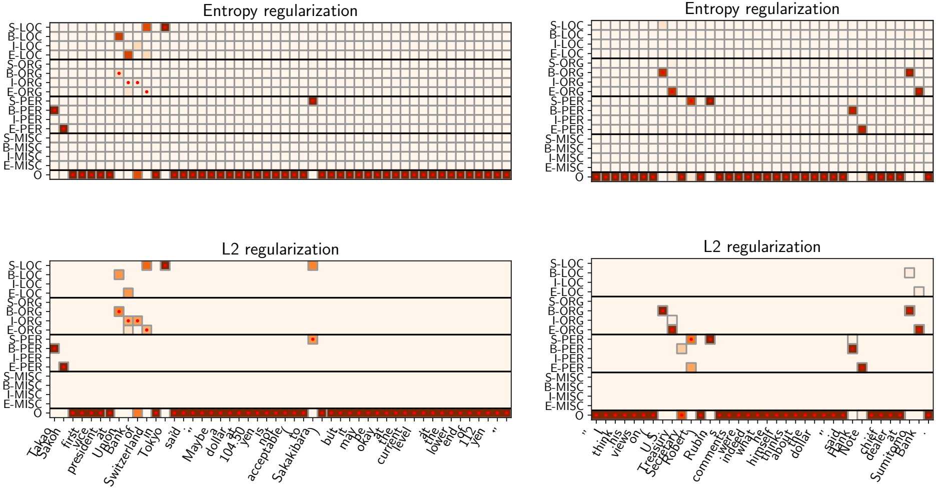
Named entity recognition

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



- Extension of [Lample et al., 2016]
- Various losses based on $\nabla\mathrm{Vit}_\Omega$, Vit_Ω layer
- Sparse tag probability output

Sparse predictions

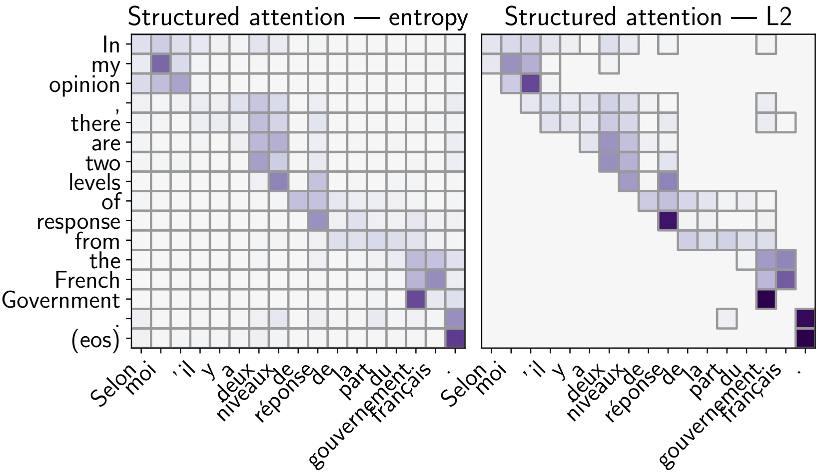


Comparison

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

Structured attention — Neural Machine Transation

- Compute the vector $\boldsymbol{c}$ by marginalizing a graphical model (Vit_Ω), with sparse marginal computation $\Omega = \ell_2^2$.
- vs simple softmax in original version



Conclusion

General framework to integrate dynamic programming algorithms in arbitrary networks

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

Experiments

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

Similar BLEU scores

Attention model	WMT14 1M fr$\rightarrow$en	WMT14 en$\rightarrow$fr
Softmax	**27.96**	**28.08**
Entropy regularization	**27.96**	27.98
ℓ_2^2 reg.	27.21	27.28

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.

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

[Lample 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.

Bibliography II

[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.

[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.