Sequential Data Modeling - Conditional Random Fields

A TUTORIAL ON CONDITIONAL RANDOM FIELDS
WITH APPLICATIONS TO MUSIC ANALYSIS

refer to the talk of "Slim Essid" from "Telecom ParisTech"

Contents

The supervised classification problem

Goal: predict *labels* y (aka *classes* or *outputs*) for some *observations* o (aka *data points*, *inputs*).

Examples:

- Predict *genre*, *mood*, *user tag*... for a music excerpt.
- Predict *instrument*, *chord*, *notes* played... for a music segment.

Supervised classification:

- Each observation o is supposed to pertain to a **predefined** class $\mathcal{C}_k$: the k-th (**discrete**) class of a classification problem; $k = 1, \cdots, K$.
- This is represented using a label y for each o; $y \in \mathcal{Y}$, e.g. $\mathcal{Y} = \{0, 1\}$, $\mathcal{Y} = \{1, 2, 3, ..., K\}$.

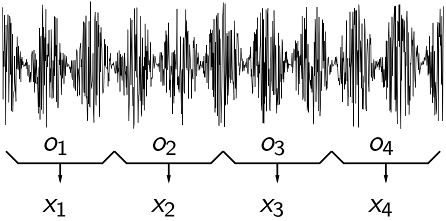
Examples of classes:

Tags: *Jazz, Bebop, Fast, Exciting, Sax, Drums, ...*
Chords: ***C**7, **G**maj7, **F**min7, ...*

Features

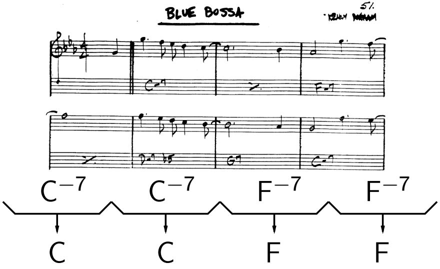
- Classification relies on **features** x: descriptors of some qualities/ attributes of the inputs o. Two types of features:

Continuous features



real-valued: *e.g. MFCC, chroma, tempo...*

Discrete/categorical



symbolic: *e.g. note/chord played, key...*

- Usually assembled as **feature vectors** $\mathbf{x}$.

Notations

- o: an input (observation) to be classified; *e.g.: a music excerpt, a music symbol, an audio frame/segment...*

- $\mathbf{x} = (x_1, \cdots, x_D)^T$: a D-dimensional column vector (usually in $\mathbb{R}^D$); $\mathbf{x}^T$is a row vector.
- $\mathbf{x}_n$ is a **feature vector** among a collection of N examples $\mathbf{x}_1, \cdots, \mathbf{x}_N$.
- x_{jn} is the j-th **feature coefficient** of $\mathbf{x}_n$; $1 \leq j \leq D$.
- $\mathcal{D}$ =$\{\mathbf{x}_1, ..., \mathbf{x}_N\}$: the set of all training feature-vector examples.

Feature functions

Different from features!

Definition

A **feature function** is a real-valued function of both the input space $\mathcal{O}$ (observations) and the output space $\mathcal{Y}$ (target labels), $f_j : \mathcal{O} \times \mathcal{Y} \to \mathbb{R}$, that can be used to compute characteristics of the observations.

- An alternative way to express the characteristics of the observations, in a more **flexible manner**:
  - using **output-specific** features;
  - describing the **context**.

Example: $f_j(o_i, y_i) = \begin{cases} 1 & \text{if } o_i = \mathbf{C},\ o_{i+1} = \mathbf{E} \text{ and } y = \mathbf{C}maj \\ 0 & \text{otherwise} \end{cases}$

Feature functions

- Remarks:
  - Different attributes may thus be considered for different classes.
  - Feature functions are more general than features: one can define
    - $f_j(o, y) \triangleq x_j$;

      or
    - $f_j(o, y) \triangleq \mathbf{x}$.
  - In the following:
    - Feature-function notations will be used only when needed.
    - Otherwise, feature-vectors will be preferred.

Probabilistic classification

Take decisions based on the **MAP** rule:

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y}} p(y|\mathbf{x})$$

in order to minimize the error rate (here the expected 0-1 loss).

MAP: *Maximum A Posteriori* probability

→ this is the **Bayes decision rule** (for the 0-1 loss.)

How to get there?

Generative model based classification

- **Objective**: $\hat{y} = \text{argmax}_y \, p(y|\mathbf{x})$.
- By the **Bayes** rule $p(y|\mathbf{x}) = \frac{p(y,\mathbf{x})}{p(\mathbf{x})} = \frac{p(y)p(\mathbf{x}|y)}{p(\mathbf{x})}$,

$$\hat{y} = \underset{y}{\text{argmax}} \frac{p(y)p(\mathbf{x}|y)}{p(x)} = \underset{y}{\text{argmax}} \, p(y)p(\mathbf{x}|y).$$

- Assuming a fixed prior $p(y)$ (possibly uninformative: $p(y) = \frac{1}{K}$), one is left with:

$$\hat{y} = \underset{y}{\text{argmax}} \, p(\mathbf{x}|y).$$

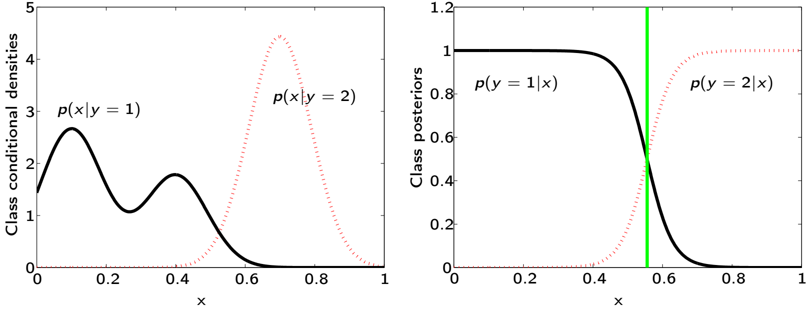
→ Our decision criterion becomes a **maximum-likelihood** criterion.

→ This is a **generative approach** to classification: a probabilistic model of "how to generate $\mathbf{x}$ given a class y" is targeted.

Discriminative model based classification

Directly models $p(y|\mathbf{x})$ without wasting efforts on modeling the observations, which is not needed for the goal $\hat{y} = \text{argmax}_y\, p(y|\mathbf{x})$.

- Pros:
  - The class posteriors $p(y = c|\mathbf{x})$ are potentially simpler than the class-conditional densities.



Generated using pmtk3 (Dunham and Murphy, 2010)

Discriminative model based classification

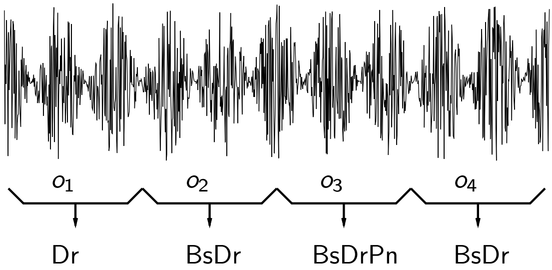
Directly models $p(y|\mathbf{x})$ without wasting efforts on modeling the observations, which is not needed for the goal $\hat{y} = \operatorname{argmax}_y p(y|\mathbf{x})$.

- Pros:
  - The class posteriors $p(y = k|\mathbf{x})$ are potentially simpler than the class-conditional densities.
  - Avoids making unwarranted assumptions about the features which may be highly **dependent** (especially with structured data).
  - Improved robustness to model imperfections, as independence assumptions will be made only among the labels, not the observations.
- Cons:
  - Classes need to be learned jointly and data should be available for all classes.
  - Models do not allow for generating observations.

Predicting structured-output data

- In many **MIR** tasks the outputs are **structured**, *e.g.:*

Musical instrument recognition

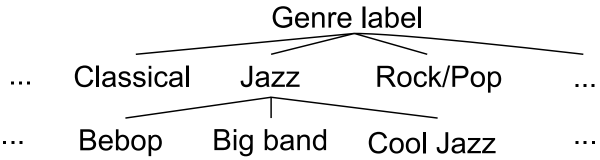


Linear-chain structure

Predicting structured-output data

- In many **MIR** tasks the outputs are **structured**, *e.g.:*

Autotagging tasks:
target tags are correlated (*e.g. bebop*, *Jazz*, *fast tempo*)



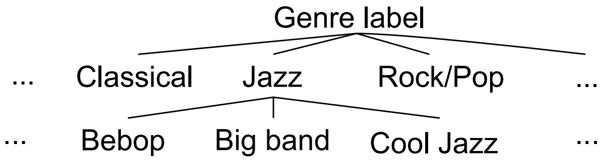
Tree structure

→ Need for predictors able to take advantage of this structure.

Predicting structured-output data

- In many **MIR** tasks the outputs are **structured**, *e.g.:*

Autotagging tasks:
target tags are correlated (*e.g. bebop*, *Jazz*, *fast tempo*)

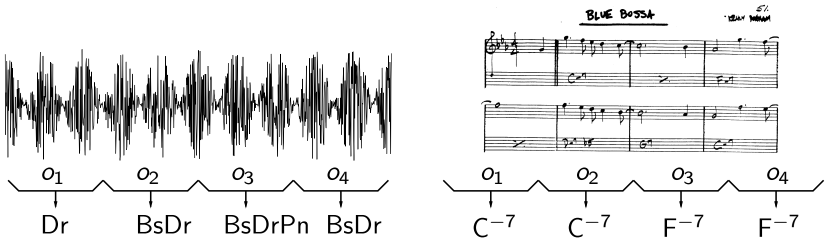


Tree structure

→ Need for predictors able to take advantage of this structure.

Predicting sequential data

- In this tutorial, we focus on **linear-chain data**



- Specialized inference algorithms can then be used (**forward-backward** method), which are easier to apprehend.
- More general methods can be used for more general structure (**belief propagation** and extensions), see *e.g.* (Jensen and Nielsen, 2007).

More notations

- $\underline{\mathbf{x}}$ is a sequence of observations: $\underline{\mathbf{x}} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$.
- $\underline{y}$ is the corresponding sequence of labels: $\underline{y} = (y_1, \cdots, y_n)$.
- We assume we have a training dataset $\mathcal{D}$ of N (i.i.d) such sequences: $\mathcal{D} = \{(\underline{\mathbf{x}}^{(1)}, \underline{y}^{(1)}), \cdots, (\underline{\mathbf{x}}^{(N)}, \underline{y}^{(N)})\}$.
- **Remarks**:
  - Observations are no longer assumed to be i.i.d within each sequence.
  - Sequences $\underline{\mathbf{x}}^{(q)}$ do not necessarily have the same length, when needed n_q will denote the length of $\underline{\mathbf{x}}^{(q)}$.

The CRF model

A discriminative model for structured-output data

CRF model definition

$$
\begin{aligned}
p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta}) &= \frac{1}{Z(\underline{\mathbf{x}},\boldsymbol{\theta})}\exp\sum_{j=1}^{D}\theta_j F_j(\underline{\mathbf{x}},\underline{y}) \\
&= \frac{1}{Z(\underline{\mathbf{x}},\boldsymbol{\theta})}\Psi(\underline{\mathbf{x}},\underline{y};\boldsymbol{\theta});\;\; \boldsymbol{\theta}=\{\theta_1,\cdots,\theta_D\}.
\end{aligned}
$$

- $Z(\underline{\mathbf{x}},\boldsymbol{\theta}) = \sum_{\underline{y}}\exp\sum_j \theta_j F_j(\underline{\mathbf{x}},\underline{y})$ is called a **partition function**.
- $\Psi(\underline{\mathbf{x}},\underline{y};\boldsymbol{\theta}) = \exp\sum_{j=1}^{D}\theta_j F_j(\underline{\mathbf{x}},\underline{y})$ is called a **potential function**.

- **Remark**: feature functions $F_j(\underline{\mathbf{x}},\underline{y})$ depend on the whole sequence of observations $\underline{\mathbf{x}}$ and labels $\underline{y}$.

Applications of CRFs

CRF models have proven to be superior to competitors in a variety of application fields.

- They are the state-of-the-art techniques in many **natural language processing** (NLP) tasks (Taskar et al., 2002; Settles, 2004; Lavergne et al., 2010)

  part-of-speech tagging (POS), named-entity recognition (*NER*)...

- They have been successfully used for various **computer vision** tasks (He et al., 2004; Quattoni et al., 2007; Wang et al., 2006; Morency et al., 2007; Rudovic et al., 2012)

  image labeling, object and gesture recognition, facial expressions...

- Also for **speech analysis** tasks (Gunawardana et al., 2005; Reiter et al., 2007; Morris and Fosler-Lussier, 2008; Hong, 2010)

  speech recognition, speech segmentation, speaker identification...

- To date rarely used for **music analysis**, despite a great potential...

The logistic regression model

Approach: model the **posterior** probabilities of the K classes using linear functions of the inputs $\mathbf{x}$, according to:

$$
\begin{aligned}
\log \frac{P(\mathcal{C}_1|\mathbf{x})}{P(\mathcal{C}_K|\mathbf{x})} &= w_{10} + \mathbf{w}_1^T \mathbf{x} \\
\log \frac{P(\mathcal{C}_2|\mathbf{x})}{P(\mathcal{C}_K|\mathbf{x})} &= w_{20} + \mathbf{w}_2^T \mathbf{x} \\
&\vdots \\
\log \frac{P(\mathcal{C}_{K-1}|\mathbf{x})}{P(\mathcal{C}_K|\mathbf{x})} &= w_{(K-1)0} + \mathbf{w}_{K-1}^T \mathbf{x}
\end{aligned}
$$

Defines a **log-linear** model specified in terms of $K-1$ **log-odds:** $\log \frac{P(\mathcal{C}_k|\mathbf{x})}{P(\mathcal{C}_K|\mathbf{x})}$.

The logistic regression model

- From $\log \frac{P(\mathcal{C}_k|\mathbf{x})}{P(\mathcal{C}_K|\mathbf{x})} = w_{k0} + \mathbf{w}_k^T \mathbf{x}$; $k = 1, \cdots, K-1$; it is easy to deduce that:

Multiclass logistic regression model

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{\exp(w_{k0} + \mathbf{w}_k^T \mathbf{x})}{1 + \sum_{l=1}^{K-1} \exp(w_{l0} + \mathbf{w}_l^T \mathbf{x})} \; ; \; k = 1, \cdots, K-1,$$

$$P(\mathcal{C}_K|\mathbf{x}) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(w_{l0} + \mathbf{w}_l^T \mathbf{x})}$$

- **Remarks**
  - The model is a **classification** model (not a regression model!)
  - It is a **discriminative** model as it targets $P(\mathcal{C}_k|\mathbf{x})$ (as opposed to modeling $p(\mathbf{x}|\mathcal{C}_k)$ in **generative** models.)

Binary classification case

- When $K = 2$

$$
\begin{aligned}
P(\mathcal{C}_1|\mathbf{x}) &= p = \frac{1}{1 + \exp -(w_{10} + \mathbf{w}_1^T \mathbf{x})} \\
P(\mathcal{C}_2|\mathbf{x}) &= 1 - p
\end{aligned}
$$

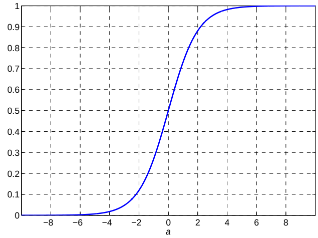
- $p = \dfrac{1}{1 + \exp -a}$; $a = w_{10} + \mathbf{w}_1^T \mathbf{x}$

Logistic sigmoid function

$$
\sigma(a) \triangleq \frac{1}{1 + \exp -a}
$$

The logistic sigmoid function

$$\sigma(a) \triangleq \frac{1}{1+\exp -a}$$



- Properties:

Symmetry: $\sigma(-a) = 1 - \sigma(a)$

Inverse: $a = \log \frac{\sigma}{1-\sigma}$: **logit** function

- The odds $\frac{p}{1-p} \in [0, +\infty]$ hence the **log-odds** $\log \frac{p}{1-p} \in [-\infty, +\infty]$
- Logistic regression models the **log-odds** as **linear functions** of the inputs... ***why is this a good idea?***

$\rightarrow$ *Study the link to* ***maximum entropy models****.*

Maximum Entropy: an introductory example

Inspired by (Berger et al., 1996)

Goal: perform chord transcription using notes played as input (in the symbolic domain).

Method: Use a training dataset to estimate $p(y|o)$: the probability to assign the chord y to the observed note o; to be used for MAP decision.

- The structure of both the ground-truth labels y and the observations o reflect a set of **facts** about the data: rules of harmony.

$\rightarrow$ Our model should capture these facts to perform accurate predictions.

Using facts about the data

- Let's assume we observe the note **C**.

$\rightarrow$ The "matching" chord is among $\{\mathbf{C}maj, \mathbf{C}min, \mathbf{A}\flat maj, \mathbf{A}min, \mathbf{F}maj, \mathbf{F}min\}$.

In terms of statistics

$P(\mathbf{C}maj) + P(\mathbf{C}min) + P(\mathbf{A}\flat maj) + P(\mathbf{A}min) + P(\mathbf{F}maj) + P(\mathbf{F}min) = 1.$

- How to choose $P(\mathbf{C}maj), \cdots, P(\mathbf{F}min)$?
- Safe choice:

In terms of statistics

$$P(\mathbf{C}maj) = P(\mathbf{C}min) = \cdots = P(\mathbf{F}min) = \frac{1}{6}$$

Why "uniform"?

- Intuitively: the most uniform model according to our knowledge, the only **unbiased** assumption

- Ancient wisdom:
  - *Occam's razor* (*William of Ockham, 1287-1347*)*: principle of parsimony: "Nunquam ponenda est pluralitas sine necesitate." [Plurality must never be posited without necessity.]*
  - *Laplace: "when one has no information to distinguish between the probability of two events, the best strategy is to consider them equally likely." (Principle of Insufficient Reason)*

More facts

- *The matching chord is* $\mathbf{C}maj$ *or* $\mathbf{F}maj$ *30% of the time:*

$$
\begin{aligned}
P(\mathbf{C}maj) + P(\mathbf{F}maj) &= 3/10 \\
P(\mathbf{C}maj) + P(\mathbf{C}min) + \cdots + P(\mathbf{F}maj) + P(\mathbf{F}min) &= 1
\end{aligned}
$$

- Again many solutions... and a resonable choice is:

$$
\begin{aligned}
P(\mathbf{C}maj) = P(\mathbf{F}maj) &= 3/20 \\
P(\mathbf{C}min) = P(\mathbf{A}\flat maj) = P(\mathbf{A}min) = P(\mathbf{F}min) &= 7/40
\end{aligned}
$$

- How to generalize this? How to determine the *"most uniform"* model subject to the constraints at hand?

Using feature functions

- Need to express the **facts** about the observations in a **flexible way**, to make sure the model will match them:
  - make use of **statistics** of the observations: e.g. if **C** is played, the matching chord is **C***maj* or **F***maj* with frequency 3/10.
  - allow for using the **context**: e.g. if **C** is followed by **E** then the chord is **C***maj* with frequency 1/2.

→ define **feature functions** to capture these statistics and use them to impose **constraints** to the model.

Example: $f_j(o_i, y_i) = \begin{cases} 1 & \text{if } o_i = \mathbf{C},\ o_{i+1} = \mathbf{E} \text{ and } y = \mathbf{C}maj \\ 0 & \text{otherwise} \end{cases}$

Defining constraints through feature functions

- The training sample can be described in terms of its **empirical** probability distribution $\tilde{p}(o, y)$:

$$\tilde{p}(o, y) \stackrel{\Delta}{=} \frac{1}{N} \times \text{number of times that } (o, y) \text{ occurs in the sample}$$

- $\tilde{\mathbb{E}}(f_j) \stackrel{\Delta}{=} \sum_{o,y} \tilde{p}(o, y) f_j(o, y)$: expected value of f_j w.r.t $\tilde{p}(o, y)$.
- $\mathbb{E}(f_j) \stackrel{\Delta}{=} \sum_{o,y} p(o) p(y|o) f_j(o, y)$: expected v. of f_j w.r.t the **model** $p(o, y)$.

Defining constraints through feature functions

- The observed statistics (facts) are captured by enforcing:

Constraint equation

$$\mathbb{E}(f_j) = \tilde{\mathbb{E}}(f_j),\ i.e.$$

$$\sum_{o,y} p(o)p(y|o)f_j(o,y) = \sum_{o,y} \tilde{p}(o,y)f_j(o,y)$$

Maximum entropy principle

- Now how to implement the idea of **uniform modeling**?
- Among the set $\mathcal{M}$ of probability distributions that satisfy the **constraints**, $\mathbb{E}(f_j) = \tilde{\mathbb{E}}(f_j)$, choose:

Maximum entropy criterion

$$p^*(y|o) = \underset{p(y|o)\in\mathcal{M}}{\operatorname{argmax}}\ H(y|o);$$

$$H(y|o) \triangleq -\sum_{o,y} p(o)p(y|o)\log p(y|o) : \text{ the } \textbf{conditional entropy}$$

- Hint from information theory: the discrete distribution with maximum **entropy** is the **uniform** distribution.

Solving the problem

Primal: $p^*(y|o) = \text{argmax}_{p(y|o)\in\mathcal{M}} H(y|o)$

Constraints: $\mathbb{E}(f_j) = \tilde{\mathbb{E}}(f_j)$ and $\sum_y p(y|o) = 1$

Lagrangian: $L(p, \boldsymbol{\lambda}) \stackrel{\Delta}{=} H(y|o) + \lambda_0 \left(\sum_y p(y|o) - 1\right) + \sum_j \lambda_j \left(\mathbb{E}(f_j) - \tilde{\mathbb{E}}(f_j)\right)$

Equating the derivative of the Lagrangian with 0:

$$p_{\boldsymbol{\lambda}}(y|o) = \frac{1}{Z_\lambda(o)} \exp \sum_j \lambda_j f_j(o, y);$$

$Z_{\boldsymbol{\lambda}}(x) = \sum_y \exp\left(\sum_j \lambda_j f_j(o, y)\right)$

The solution is given by the dual optimal: $\boldsymbol{\lambda}^* = \text{argmax}_{\boldsymbol{\lambda}} L(p, \boldsymbol{\lambda})$.

Compare to the LR model

- Maxent model:

$$p(y=k|o) = \frac{1}{Z_{\boldsymbol{\lambda}}(o)} \exp\left(\sum_j \lambda_{jk} f_j(o,y)\right);$$

$$Z_{\boldsymbol{\lambda}}(o) = \sum_y \exp\left(\sum_j \lambda_{jk} f_j(o,y)\right).$$

- Logistic regression model:

$$\begin{aligned} p(y=k|\mathbf{x}) &= \frac{\exp(w_{k0} + \mathbf{w}_k^T \mathbf{x})}{1 + \sum_{l=1}^{K-1} \exp(w_{l0} + \mathbf{w}_l^T \mathbf{x})} \\ &= \frac{\exp(w'_{k0} + \mathbf{w}_k'^T \mathbf{x})}{\sum_{l=1}^{K} \exp(w'_{l0} + \mathbf{w}_l'^T \mathbf{x})} \\ &= \frac{1}{Z_{\mathbf{w}}(\mathbf{x})} \exp(w'_{k0} + \mathbf{w}_k'^T \mathbf{x}). \end{aligned}$$

Compare to the LR model

- Maxent model:

$$p(y = k|o) = \frac{1}{Z_{\boldsymbol{\lambda}}(o)} \exp \sum_j \lambda_{kj} f_j(o, y)$$

- Logistic regression model:

$$p(y = k|\mathbf{x}) = \frac{1}{Z_{\mathbf{w}}(\mathbf{x})} \exp(w'_{k0} + \mathbf{w}_k'^T \mathbf{x})$$

- Using:
  - feature-function: $f_j(o, y) = x_j$; $f_0(o, y) = 1$ and $\mathbf{x} = (x_1, \cdots, x_j, \cdots, x_D)^T$;
  - $w'_{k0} + \mathbf{w}_k'^T \mathbf{x} = \sum_{j=0}^{D} w'_{kj} f_j(o, y)$;

Compare to the CRF model

- Maxent model:

$$p(y = k|o) = \frac{1}{Z_{\boldsymbol{\lambda}}(o)} \exp \sum_j \lambda_{kj} f_j(o, y)$$

- Logistic regression model:

$$p(y = k|o) = \frac{1}{Z_{\mathbf{w}}(o)} \exp \sum_j w'_{kj} f_j(o, y)$$

- CRF model:

$$p(\underline{y}|\underline{\mathbf{x}}; \boldsymbol{\theta}) = \frac{1}{Z_{\boldsymbol{\theta}}(\underline{\mathbf{x}})} \exp \sum_j \theta_j F_j(\underline{\mathbf{x}}, \underline{y})$$

Conclusion

The solution to the maximum entropy models has the same parametric form as logistic regression and CRF models.

- It is easily shown that the optimal solution is the **maximum-likelihood** solution in the parametric family $p_{\boldsymbol{\lambda}}(y|\mathbf{x}) = \frac{1}{Z_{\boldsymbol{\lambda}}(\mathbf{x})} \exp(\sum_j \lambda_j x_j)$.
- We've only considered discrete inputs, what about **continuous** inputs?
  - It is found that if the class-conditional densities $p(\mathbf{x}|y)$ are members of the **exponential family** of distributions, then the posterior probabilities are again given by **logistic sigmoids** of a linear function.
  - In particular, the model is optimal with **Gaussian densities** (with a shared covariance matrix).

The logistic regression model is quite well justified in a variety of situations.

Fitting the LR models

- Done by **maximum likelihood** estimation; in practice minimizing the **Negative Log-Likelihood** (**NLL**).
- Let $\boldsymbol{\theta}$ denote the set of all parameters: $\boldsymbol{\theta} = \{w_{10}, \mathbf{w}_1, \cdots, w_{(K-1)0}, \mathbf{w}_{K-1}\}$.
- The **log-likelihood** for the N (i.i.d) feature-vector observations is:

$$L(\mathcal{D}; \boldsymbol{\theta}) \stackrel{\Delta}{=} -\sum_{i=1}^{N} \log p(y_i | \mathbf{x}_i; \boldsymbol{\theta})$$

- To simplify, we focus on the **bi-class** case...

NLL for bi-class LR

- Let $y_i = 1$ for $\mathcal{C}_1$ observations and $y_i = 0$ for $\mathcal{C}_2$ observations.
- Let $p(\mathbf{x};\boldsymbol{\theta}) \stackrel{\Delta}{=} p(y_i = 1|\mathbf{x}_i;\boldsymbol{\theta})$; hence $p(y_i = 0|\mathbf{x}_i;\boldsymbol{\theta}) = 1 - p(\mathbf{x};\boldsymbol{\theta})$.
- We can write: $p(y|\mathbf{x};\boldsymbol{\theta}) = p(\mathbf{x};\boldsymbol{\theta})^y(1 - p(\mathbf{x};\boldsymbol{\theta}))^{1-y}$.

Negative Log-Likelihood

$$
\begin{aligned}
L(\mathcal{D};\boldsymbol{\theta}) = L(\tilde{\mathbf{w}}) &= -\sum_{i=1}^{N}\{y_i \log p(\mathbf{x}_i;\tilde{\mathbf{w}}) + (1-y_i)\log(1-p(\mathbf{x}_i;\tilde{\mathbf{w}}))\} \\
&= -\sum_{i=1}^{N}\left\{y_i\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i - \log\left(1+\exp(\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i)\right)\right\}
\end{aligned}
$$

where $\tilde{\mathbf{w}} = (w_0, \mathbf{w})$ and $\tilde{\mathbf{x}}_i = (1, \mathbf{x}_i)$ so that $\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i = w_0 + \mathbf{w}^T\mathbf{x}_i$.

Gradient and Hessian of the NLL

Gradient: $\nabla L(\mathcal{D}; \tilde{\mathbf{w}}) = -\sum_{i=1}^{N} \tilde{\mathbf{x}}_i (y_i - p(\mathbf{x}_i; \tilde{\mathbf{w}}))$

Hessian: $\frac{\partial^2 L(\mathcal{D}; \tilde{\mathbf{w}})}{\partial \tilde{\mathbf{w}} \partial \tilde{\mathbf{w}}^T} = \sum_{i=1}^{N} \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T p(\mathbf{x}_i; \tilde{\mathbf{w}}) \, (1 - p(\mathbf{x}_i; \tilde{\mathbf{w}}))$

$\rightarrow$ so the Hessian is **positive semi-definite**,

$\rightarrow$ the NLL is **convex** and it has a global minimum.

Minimizing the NLL

- By setting the derivatives to zero:

$$\frac{\partial L(\mathcal{D};\tilde{\mathbf{w}})}{\partial w_j} = -\sum_{i=1}^{N} \tilde{x}_{ji}(y_i - p(\mathbf{x}_i;\tilde{\mathbf{w}})) = 0\,;\, 0 \leq j \leq D.$$

Optimization problem

Solve for $\tilde{\mathbf{w}}$ the $D+1$ **non-linear** equations:

$$\sum_{i=1}^{N} y_i \tilde{x}_{ji} = \sum_{i=1}^{N} \tilde{x}_{ji} p(\mathbf{x}_i;\tilde{\mathbf{w}}) \quad ; \quad 0 \leq j \leq D$$

- For $j=0$, since the first coefficient of $\tilde{\mathbf{x}}_i$ is 1, that is $\tilde{x}_{0i} = 1$, we get:

$$\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} p(\mathbf{x}_i;\tilde{\mathbf{w}}).$$

Optimization methods

Objective: Solve $\sum_{i=1}^{N} y_i \tilde{x}_{ji} = \sum_{i=1}^{N} \tilde{x}_{ji} p(\mathbf{x}_i; \tilde{\mathbf{w}})$

Problem: No closed-form solution in general (system of $D+1$ **non-linear** equations).

Solution: use **descent** methods.

Among the many available descent algorithms, two are widely used:

- the **Newton-Raphson** method: fast... but complex (efficient variants exist);
- the **stochastic gradient** descent method: easy to implement, adapted to large scale problems.

Optimization with the Newton-Raphson method

- To minimize $g(\boldsymbol{\theta})$, consider its second-order Taylor series approximation around $\boldsymbol{\theta}_n$:

$$g(\boldsymbol{\theta}) \approx g(\boldsymbol{\theta}_n) + \nabla g(\boldsymbol{\theta}_n)^T(\boldsymbol{\theta} - \boldsymbol{\theta}_n) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_n)^T H(\boldsymbol{\theta}_n)(\boldsymbol{\theta} - \boldsymbol{\theta}_n);$$

$\nabla g(\boldsymbol{\theta}_n)$ and $H(\boldsymbol{\theta}_n)$ are resp. the **gradient** and **Hessian** of $g(\boldsymbol{\theta})$ at $\boldsymbol{\theta}_n$.

- This approximation is a quadratic function which is minimized by solving:

$$\nabla g(\boldsymbol{\theta}_n) + H(\boldsymbol{\theta}_n)(\boldsymbol{\theta} - \boldsymbol{\theta}_n) = 0.$$

Hence the **Newton-Raphson step**

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - H(\boldsymbol{\theta}_n)^{-1}\nabla(\boldsymbol{\theta}).$$

Optimization with the Newton-Raphson method

Discussion

- Typically the algorithm converges (though overshooting may occur), and convergence speed is quadratic.

- D has to be small enough so that it is not too costly to **recompute** and **store** the inverse Hessian matrix at each iteration.
- Otherwise use **Quasi-Newton methods**:
  - **BFGS** (Broyden, Fletcher, Goldfarb and Shanno) method: approximates the inverse Hessian using successive gradient values.
  - **L-BFGS** (limited memory BFGS) method: stores only a few vectors used to approximate the inverse Hessian.

- Alternatively, use **stochastic gradient learning** (*see Appendix*):
  - Makes gradient updates based on **one** training example **at a time.**
  - In practice: simple approach, slow convergence, less accurate than L-BFGS.

ℓ_2-regularization

- To **avoid overfitting** the complexity of the model should be penalized.
- Similarly to **ridge regression** (Hastie et al., 2009), a quadratic regularization term can be added to the NLL:

Regularized logistic regression problem

$$\begin{aligned}\hat{\mathbf{w}} &= \operatorname*{argmin}_{\tilde{\mathbf{w}}} L(\mathcal{D};\tilde{\mathbf{w}}) + \frac{\gamma}{2}||\mathbf{w}||^2 \\ &= \operatorname*{argmin}_{\tilde{\mathbf{w}}} \left\{ -\sum_{i=1}^{N}\left[y_i\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i - \log\left(1+\exp\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i\right)\right] + \frac{\gamma}{2}\sum_{j=1}^{D} w_j^2 \right\}\end{aligned}$$

$\gamma \geq 0$: complexity parameter controlling the amount of shrinkage; usually tuned by **cross-validation**.

ℓ_2-regularization

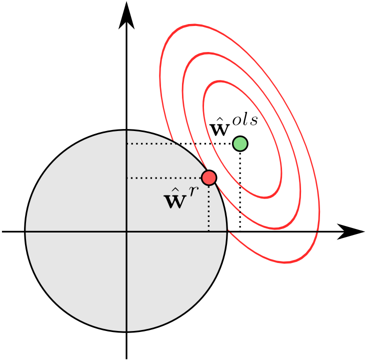
Discussion

Recall that:

$$\hat{\mathbf{w}} = \underset{\tilde{\mathbf{w}}}{\operatorname{argmin}}\, L(\mathcal{D}; \tilde{\mathbf{w}}) + \frac{\gamma}{2}||\mathbf{w}||^2$$

is equivalent to:

$$\begin{cases} \hat{\mathbf{w}} = \operatorname{argmin}_{\tilde{\mathbf{w}}} L(\mathcal{D}; \tilde{\mathbf{w}}) \\ \text{subject to } \; ||\mathbf{w}||^2 \leq t \end{cases}$$



for some t which has a correspondence to γ.

ℓ_2-regularization

Gradient and Hessian

Gradient: $\nabla L_2(\mathcal{D}; \tilde{\mathbf{w}}) = \nabla L_2(\mathcal{D}; \tilde{\mathbf{w}}) + \gamma \mathbf{w}$

Hessian: $H_2(\tilde{\mathbf{w}}) = H(\tilde{\mathbf{w}}) + \gamma \mathbf{I}_{D+1}$

$\rightarrow$ So the Hessian becomes **positive definite**, the NLL is now **strictly convex** and it has a unique global minimum.

$\rightarrow$ The previous optimization methods can be straightforwardly adapted by modifying the expressions of the gradient and Hessian.

ℓ_1-regularization

- Proceed as in the **LASSO** (Hastie et al., 2009), using a ℓ_1-regularization.

ℓ_1-regularized logistic regression problem

$$\begin{aligned}\hat{\mathbf{w}} &= \underset{\tilde{\mathbf{w}}}{\operatorname{argmin}}\, L(\mathcal{D}; \tilde{\mathbf{w}}) + \gamma ||\mathbf{w}||_1 \\ &= \underset{\tilde{\mathbf{w}}}{\operatorname{argmin}}\, L(\mathcal{D}; \tilde{\mathbf{w}}) + \gamma \sum_{j=1}^{D} |w_j|\,;\ \gamma \geq 0.\end{aligned}$$

ℓ_1-regularization

Discussion

- ℓ_1-regularization achieves **feature selection**.

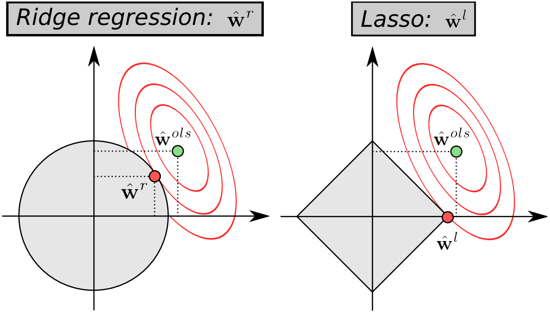


Illustration by Alexandre Gramfort, Telecom ParisTech

ℓ_1-regularization

Discussion

- ℓ_1-regularization achieves **feature selection**.

- **Difficulties**:
  - The regularizer is **not differentiable** at zero yielding **non-smooth** optimization problem.

  $\rightarrow$ specific optimization techniques needed (Yuan et al., 2010).
  - In configurations with groups of highly correlated features:
    - ℓ_1-regularization tends to select randomly one feature in each group;
    - ℓ_2-regularization tends to yield better prediction performance.

  $\rightarrow$ Consider the **elastic net** model (Hastie et al., 2009):

$$L(\mathcal{D}; \tilde{\mathbf{w}}) + \gamma_2 ||\mathbf{w}||_2^2 + \gamma_1 ||\mathbf{w}||_1$$

Kernel logistic regression (KLR)

- Let $\mathcal{K}$: positive definite kernel and $\mathcal{H}_{\mathcal{K}}$: the **RKHS** generated by $\mathcal{K}$.
- Let $\phi \in \mathcal{H}_{\mathcal{K}}$, a feature mapping to $\mathcal{H}_{\mathcal{K}}$.

KLR model

$$p(y_i|\mathbf{x}_i) = \frac{1}{1 + \exp -g(\mathbf{x}_i)} \;; \quad g(\mathbf{x}) = w_0 + \mathbf{w}^T \phi(\mathbf{x})$$

KLR model estimation problem:

$$\min_{\tilde{\mathbf{w}}} L(\tilde{\mathbf{w}}) = -\sum_{i=1}^{N} [y_i g(\mathbf{x}_i) - \log(1 + \exp g(\mathbf{x}_i))]$$

Regularized KLR

Regularized KLR model estimation problem:

$$\min_{\tilde{\mathbf{w}}} L(\tilde{\mathbf{w}}) = -\sum_{i=1}^{N} \left[y_i g(\mathbf{x}_i) - \log(1 + \exp g(\mathbf{x}_i)) \right] + \frac{\gamma}{2} ||g||^2_{\mathcal{H}_{\mathcal{K}}}$$

- By the **representer theorem**: $g(\mathbf{x}) = w_0 + \sum_{i=1}^{N} \alpha_i \mathcal{K}(\mathbf{x}_i, \mathbf{x})$
- The problem is **strictly convex** and can be solved using classic solvers.

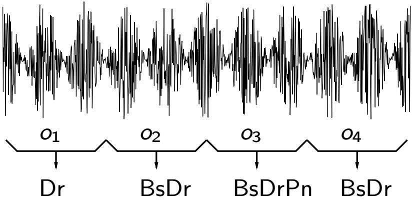
KLR vs SVM

- It can be shown that KLR and SVM are quite related (*see Appendix*).
- Very similar prediction performance and optimal margin properties.
- Same refinements are possible: SMO, MKL...

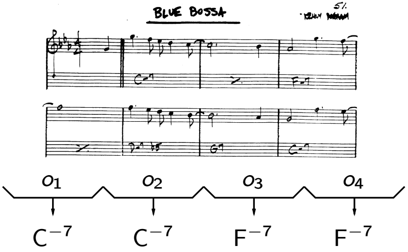
+ Provides well-calibrated **class probabilities.**
+ Naturally generalizes to multi-class problems.

- No support vectors! $\rightarrow$ ***Import Vector Machines*** (Zhu and Hastie, 2002).

Structured-output data



Musical instrument classification



Chord transcription

Recalling the notations

- $\underline{\mathbf{x}}$ is a sequence of observations: $\underline{\mathbf{x}} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$.
- $\underline{y}$ is the corresponding sequence of labels: $\underline{y} = (y_1, \cdots, y_n)$.
- We assume we have a training dataset $\mathcal{D}$ of N (i.i.d) such sequences: $\mathcal{D} = \{(\underline{\mathbf{x}}^{(1)}, \underline{y}^{(1)}), \cdots, (\underline{\mathbf{x}}^{(N)}, \underline{y}^{(N)})\}$.
- **Remarks**:
  - Observations are no longer assumed to be i.i.d within each sequence.
  - Sequences $\underline{\mathbf{x}}^{(q)}$ do not necessarily have the same length, when needed n_q will denote the length of $\underline{\mathbf{x}}^{(q)}$.

The CRF model

CRF model definition

$$
\begin{aligned}
p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta}) &= \frac{1}{Z(\underline{\mathbf{x}},\boldsymbol{\theta})} \exp \sum_{j=1}^{D} \theta_j F_j(\underline{\mathbf{x}},\underline{y}) \\
&= \frac{1}{Z(\underline{\mathbf{x}},\boldsymbol{\theta})} \Psi(\underline{\mathbf{x}},\underline{y};\boldsymbol{\theta}); \quad \boldsymbol{\theta} = \{\theta_1, \cdots, \theta_D\}.
\end{aligned}
$$

- $Z(\underline{\mathbf{x}},\boldsymbol{\theta}) = \sum_{\underline{y}} \exp \sum_j \theta_j F_j(\underline{\mathbf{x}},\underline{y})$ is called a **partition function**.
- $\Psi(\underline{\mathbf{x}},\underline{y};\boldsymbol{\theta}) = \exp \sum_{j=1}^{D} \theta_j F_j(\underline{\mathbf{x}},\underline{y})$ is called a **potential function**.
- **Remarks**:
  - CRFs appear to be an extension of logistic regression to structured data.
  - Feature functions $F_j(\underline{\mathbf{x}},\underline{y})$ depend on the whole sequence of observations $\underline{\mathbf{x}}$ and labels $\underline{y}$.

Defining label constraints

- Without any further assumptions on the structure of $\underline{y}$ the model is hardly usable:

  one needs to enumerate all possible sequences $\underline{y}$ for:

  - $Z(\underline{\mathbf{x}}, \boldsymbol{\theta}) = \sum_{\underline{y}} \exp \sum_j \theta_j F_j(\underline{\mathbf{x}}, \underline{y})$;
  - $\underline{\hat{y}} = \text{argmax}_{\underline{y}}\, p(\underline{y}|\underline{\mathbf{x}}; \boldsymbol{\theta})$.

  with $|\mathcal{Y}|^n$ possible assignments !

Defining label constraints

Using feature functions

- Consider feature functions $F_j(\underline{\mathbf{x}}, \underline{y})$ such that:

$$F_j(\underline{\mathbf{x}}, \underline{y}) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, \underline{\mathbf{x}}, i) \ ; \quad \text{where } n \text{ is the length of } \underline{\mathbf{x}}.$$

$\rightarrow$ defines **linear-chain** CRFs: at each position i, $1 \leq i \leq n$,

- each f_j depends on the **whole observation sequence**,
- but only on the **current** and **previous labels**.

Defining label constraints

Valid feature functions

$$F_j(\underline{\mathbf{x}}, \underline{y}) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, \underline{\mathbf{x}}, i)$$

Examples of such feature functions (for discrete observations):

- *The current observation is* **G**, *the current label is* **C***min7 and the previous is* **G7***;*
- *The past 4 observations..., the current label is...*
- *The next observation is...*
- *The current label is...*

Defining label constraints

Observation and transition feature functions

- For convenience, one can define two types of feature functions:
  - **Observation** (aka **state**) feature functions: $b_j(y_i, \underline{\mathbf{x}}, i)$;
  - **Transition** feature functions: $t_j(y_{i-1}, y_i, \underline{\mathbf{x}}, i)$.
- Hence:

$$p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta}) = \frac{1}{Z(\underline{\mathbf{x}}, \boldsymbol{\theta})} \exp\left\{ \sum_{i=1}^{n} \sum_{j=1}^{D_o} \theta_j b_j(y_i, \underline{\mathbf{x}}, i) + \sum_{i=1}^{n} \sum_{j=1}^{D_t} \theta_j t_j(y_{i-1}, y_i, \underline{\mathbf{x}}, i) \right\}$$

Connection to HMM

The Hidden Markov Model

$$p_{hmm}(\underline{y}, \underline{\mathbf{x}}) \triangleq \prod_{i=1}^{n} p(y_i|y_{i-1})p(\mathbf{x}_i|y_i) \quad ; \quad \text{where } p(y_1|y_0) \triangleq p(y_1).$$

One can write:

$$\begin{aligned} p_{hmm}(\underline{y}, \underline{\mathbf{x}}) &= \exp\left\{\sum_{i=1}^{n} \log p(y_i|y_{i-1}) + \sum_{i=1}^{n} \log p(\mathbf{x}_i|y_i)\right\} \\ &= \exp\left\{\sum_{i=1}^{n} \sum_{l,q\in\mathcal{Y}} \lambda_{lq} \mathbb{I}(y_i = l)\mathbb{I}(y_{i-1} = q)\right. \\ &\qquad \left. + \sum_{i=1}^{n} \sum_{l\in\mathcal{Y},\mathbf{o}\in\mathcal{X}} \mu_{\mathbf{o}l} \mathbb{I}(y = l)\mathbb{I}(\mathbf{x}_i = \mathbf{o})\right\}; \end{aligned}$$

where $\lambda_{lq} = \log p(y_i = l|y_{i-1} = q)$ and $\mu_{\mathbf{o}l} = \log p(\mathbf{x}_i = \mathbf{o}|y_i = l)$.

Connection to HMM

- Using the feature functions:
  - $b_j(y_i, \underline{\mathbf{x}}, i) = \mathbb{I}(y = l)\mathbb{I}(\mathbf{x}_i = \mathbf{o})$, where each j indexes a different "$l, \mathbf{o}$ configuration";
  - $t_j(y_{i-1}, y_i, \mathbf{x}, i) = \mathbb{I}(y_i = l)\mathbb{I}(y_{i-1} = q)$, where j indexes a different "l, q configuration";

- also using $p(\underline{y}|\underline{\mathbf{x}}) = \frac{p(\underline{y},\underline{\mathbf{x}})}{\sum_{\underline{y}'} p(\underline{y}',\underline{\mathbf{x}})}$ and letting $Z(\underline{\mathbf{x}}) = \sum_{\underline{y}'} p(\underline{y}', \underline{\mathbf{x}})$, one gets:

$$p_{hmm}(\underline{y}|\underline{\mathbf{x}}) = \frac{1}{Z(\underline{\mathbf{x}})} \exp\left\{ \sum_{i=1}^{n} \sum_{j} \theta_j b_j(y_i, \underline{\mathbf{x}}, i) + \sum_{i=1}^{n} \sum_{j} \theta_j t_j(y_{i-1}, y_i, \underline{\mathbf{x}}, i) \right\};$$

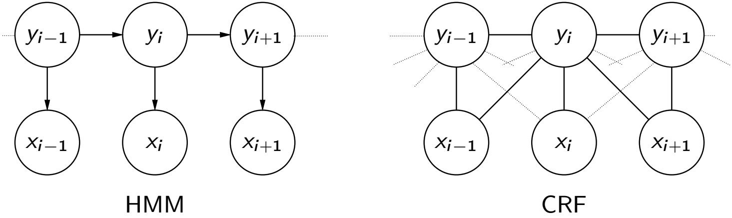
$\rightarrow$ HMMs are a particular type of linear-chain CRFs.

Connection to HMM

Discussion

CRFs have a number of advantages over HMMs, as a consequence of two major differences:

- CRFs are **discriminative models**.
- CRFs are **undirected models**.



Connection to HMM

Advantage of the discriminative nature of CRF

HMM: observation $\mathbf{x}_i$ is independent of all other variables given its parent state y_i.

CRF: no assumptions on the dependencies among the observations: only $p(\underline{y}|\underline{\mathbf{x}})$ is modeled.

$\rightarrow$ CRFs can safely:

- exploit **overlapping features**;
- account for **long-term dependencies**, considering the whole sequence of observations $\underline{\mathbf{x}}$ at each location i ($i \mapsto b_j(y_i, \underline{\mathbf{x}}, i)$);
- use **transition feature-functions** $t_j(y_{i-1}, y_i, \underline{\mathbf{x}}, i)$.

Using linear-chain CRFs

Problems to be solved:

- **Inference**: given a model $\boldsymbol{\theta}$, how to compute:
  - $\hat{\underline{y}} = \text{argmax}_{\underline{y}}\, p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta})$?
  - $Z(\underline{\mathbf{x}},\boldsymbol{\theta}) = \sum_{\underline{y}} \exp \sum_j F_j(\underline{\mathbf{x}},\underline{y})$ to deduce $p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta}) = \frac{1}{Z(\underline{\mathbf{x}},\boldsymbol{\theta})} \exp \sum_{j=1}^{D} \theta_j F_j(\underline{\mathbf{x}},\underline{y})$?

- **Parameter estimation**: given a training dataset $\mathcal{D} = \{(\underline{\mathbf{x}}^{(1)},\underline{y}^{(1)}),\cdots,(\underline{\mathbf{x}}^{(N)},\underline{y}^{(N)})\}$, how to estimate the optimal $\boldsymbol{\theta}$?

Decoding the optimal sequence

- **Problem**: solve $\hat{\underline{y}} = \operatorname{argmax}_{\underline{y} \in \mathcal{Y}^n} p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta})$, with $|\mathcal{Y}|^n$ possible assignments!
- **Solution**: use the **Viterbi** algorithm.

  Exploit the linear-chain structure:

$$
\begin{aligned}
\hat{\underline{y}} = \operatorname*{argmax}_{\underline{y}} p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta}) &= \operatorname*{argmax}_{\underline{y}} \frac{1}{Z(\underline{\mathbf{x}},\boldsymbol{\theta})} \exp \sum_{j=1}^{D} \theta_j F_j(\underline{\mathbf{x}},\underline{y}) \\
&= \operatorname*{argmax}_{\underline{y}} \sum_{j=1}^{D} \theta_j F_j(\underline{\mathbf{x}},\underline{y}) \\
&= \operatorname*{argmax}_{\underline{y}} \sum_{i=1}^{n} \sum_{j=1}^{D} \theta_j f_j(y_{i-1}, y_i, \underline{\mathbf{x}}, i)
\end{aligned}
$$

Decoding the optimal sequence

Let: $g_i(y_{i-1}, y_i) \triangleq \sum_{j=1}^{D} \theta_j f_j(y_{i-1}, y_i, \underline{\mathbf{x}}, i)$; then:

$$\hat{y} = \underset{\underline{y}}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{j=1}^{D} \theta_j f_j(y_{i-1}, y_i, \underline{\mathbf{x}}, i) = \underset{\underline{y}}{\operatorname{argmax}} \sum_{i=1}^{n} g_i(y_{i-1}, y_i).$$

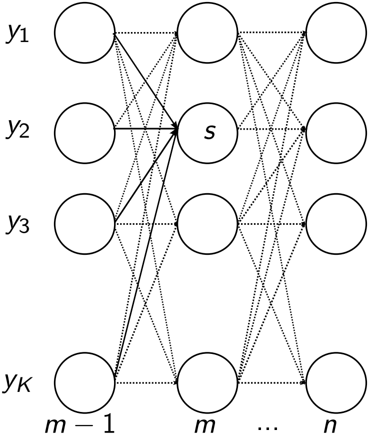
Let $\delta_m(s)$ be the optimal "intermediate score" such that at time step m the label value is s:

$$\delta_m(s) \triangleq \max_{\{y_1, \cdots, y_{m-1}\}} \left[\sum_{i=1}^{m-1} g_i(y_{i-1}, y_i) + g_m(y_{m-1}, s) \right]$$

Decoding the optimal sequence

Trellis representation

$$\delta_m(s) \triangleq \max_{\{y_1,\cdots,y_{m-1}\}} \left[\sum_{i=1}^{m-1} g_i(y_{i-1}, y_i) + g_m(y_{m-1}, s) \right]$$



Decoding the optimal sequence

- The intermediate scores $\delta_m(s)$ can be efficiently computed using[1]:

Viterbi recursion

$$\delta_m(s) = \max_{y_{m-1} \in \mathcal{Y}} \left[\delta_{m-1}(y_{m-1}) + g_m(y_{m-1}, s)\right]; \; 1 \leq m \leq n$$

- As we proceed we need to keep track of the selected predecessor of s, at each time step m.
- We use $\psi_m(s)$ for this purpose.

[1] *See Appendix for more details.*

Decoding the optimal sequence

Viterbi algorithm

Initialization:

$$\begin{aligned}\delta_1(s) &= g_1(y_0, s);\ \forall s \in \mathcal{Y};\ y_0 = \text{start} \\ \psi_1(s) &= \text{start}\end{aligned}$$

Recursion: $\forall s \in \mathcal{Y};\ 1 \le m \le n$

$$\begin{aligned}\delta_m(s) &= \max_{y \in \mathcal{Y}} \left[\delta_{m-1}(y) + g_m(y, s)\right] \\ \psi_m(s) &= \operatorname*{argmax}_{y \in \mathcal{Y}} \left[\delta_{m-1}(y) + g_m(y, s)\right]\end{aligned}$$

Termination:

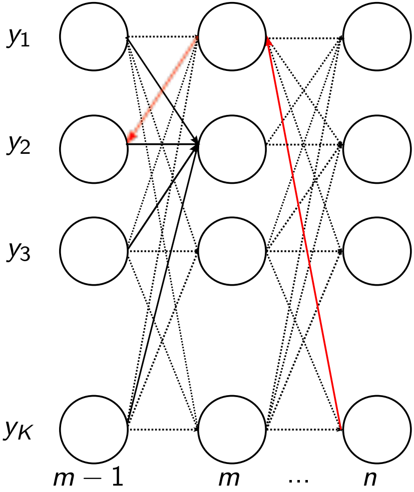
$$\begin{aligned}\delta_n(y_n^*) &= \max_{y \in \mathcal{Y}} \delta_n(y) = \max_{\underline{y}} \sum_{i=1}^{n} g_i(y_{i-1}, y_i). \\ y_n^* &= \operatorname*{argmax}_{y \in \mathcal{Y}} \delta_n(y)\end{aligned}$$

Path backtracking:

$$y_m^* = \psi_{m+1}(y_{m+1}^*);\ m = n-1, n-2, \cdots, 1.$$

Decoding the optimal sequence
Backtracking

$$y_m^* = \psi_{m+1}(y_{m+1}^*); \quad m = n-1, n-2, \cdots, 1.$$



Complexity of Viterbi decoding

Remarks on the computational cost:

- $O(K^2 n)$ in the worst case; $K = |\mathcal{Y}|$.
- In practice: $O(\mathcal{T} K n)$, where $\mathcal{T}$: average number of possible "transitions" between labels y.
- Can be reduced using **beam search**: exploring a subset of possible labels at each time position (the "most promising" ones) (Ortmanns et al., 1996).

Using linear-chain CRFs

Problems to be solved:

- **Inference**: given a model $\boldsymbol{\theta}$, how to compute:
  - $\underline{\hat{y}} = \text{argmax}_{\underline{y}}\, p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta})$ ✓
  - $Z(\underline{\mathbf{x}},\boldsymbol{\theta}) = \sum_{\underline{y}} \exp \sum_j F_j(\underline{\mathbf{x}},\underline{y})$ to deduce $p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta}) = \frac{1}{Z(\underline{\mathbf{x}},\boldsymbol{\theta})} \exp \sum_{j=1}^{D} \theta_j F_j(\underline{\mathbf{x}},\underline{y})$?

Computing the partition function $Z(\underline{\mathbf{x}}, \boldsymbol{\theta})$

The sum-product problem

Recall the CRF model:

$$p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta}) = \frac{1}{Z(\underline{\mathbf{x}},\boldsymbol{\theta})} \prod_{i=1}^{n} M_i(y_{i-1}, y_i, \underline{\mathbf{x}})\,;$$

$$M_i(y_{i-1}, y_i, \underline{\mathbf{x}}) = \exp\left(\sum_{j=1}^{D} \theta_j f_j(y_{i-1}, y_i, \underline{\mathbf{x}}, i)\right);$$

$Z(\underline{\mathbf{x}}, \boldsymbol{\theta}) = \sum_{\underline{y}\in\mathcal{Y}^n} \prod_{i=1}^{n} M_i(y_{i-1}, y_i, \underline{\mathbf{x}})$: intractable as is...

$\rightarrow$ use the **forward-backward** method: reduces **complexity** from $O(K^n)$ to $O(nK^2)$.

The forward-backward method

- Defining $\alpha_m(y_m) = \sum_{y_{m-1}} M_m(y_{m-1}, y_m)\alpha_{m-1}(y_{m-1})$; $2 \leq m \leq n$, it is easily shown[2] that:

At the end of the sequence

$$Z(\underline{x}, \theta) = \sum_{y_n \in \mathcal{Y}} \alpha_n(y_n).$$

- Alternatively, defining $\beta_m(y_m) = \sum_{y_{m+1}} M_{m+1}(y_m, y_{m+1})\beta_{m+1}(y_{m+1})$; $1 \leq m \leq n-1$ and $\beta_n(y_n) = 1$, one gets:

At the beginning of the sequence

$$Z(\underline{x}, \theta) = \sum_{y_1 \in \mathcal{Y}} M_1(y_0, y_1)\beta_1(y_1).$$

[2] *See Appendix for more details*

Marginal probability

$$p(y_{m-1}, y_m|\underline{\mathbf{x}}) = \sum_{\underline{y} \setminus \{y_{m-1}, y_m\}} p(\underline{y}|\mathbf{x})$$

Marginal probability by forward-backward

$$p(y_{m-1}, y_m|\underline{\mathbf{x}}) = \frac{1}{Z(\underline{\mathbf{x}})} \alpha_{m-1}(y_{m-1}) M_m(y_{m-1}, y_m, \underline{\mathbf{x}}) \beta_m(y_m).$$

More details in the appendix.

Negative log-likelihood (NLL)

- Given training data $\mathcal{D} = \{(\underline{\mathbf{x}}^{(1)}, \underline{y}^{(1)}), \cdots, (\underline{\mathbf{x}}^{(N)}, \underline{y}^{(N)})\}$, the NLL is:

$$
\begin{aligned}
L(\mathcal{D};\boldsymbol{\theta}) &\triangleq -\sum_{q=1}^{N} \log p(\underline{y}^{(q)} | \underline{\mathbf{x}}^{(q)}; \boldsymbol{\theta}) \\
&= \sum_{q=1}^{N} \left\{ \log Z(\underline{\mathbf{x}}^{(q)}; \boldsymbol{\theta}) - \sum_{i=1}^{n_q} \sum_{j=1}^{D} \theta_j f_j(y_{i-1}^{(q)}, y_i^{(q)}, \underline{\mathbf{x}}^{(q)}, i) \right\} \\
&= \sum_{q=1}^{N} \left\{ \log Z(\underline{\mathbf{x}}^{(q)}; \boldsymbol{\theta}) - \sum_{j=1}^{D} \theta_j F_j(\underline{\mathbf{x}}^{(q)}, \underline{y}^{(q)}) \right\}.
\end{aligned}
$$

- $L(\mathcal{D};\boldsymbol{\theta})$ is **convex** $\rightarrow$ gradient-descent will converge to global minimum.

NLL gradient

Gradient:
$$\frac{\partial L(\mathcal{D};\boldsymbol{\theta})}{\partial \theta_k} = \sum_{q=1}^{N} \left\{ \frac{\partial}{\partial \theta_k} \log Z(\underline{\mathbf{x}}^{(q)};\boldsymbol{\theta}) - F_k(\underline{\mathbf{x}}^{(q)}, \underline{y}^{(q)}) \right\}.$$

$$\begin{aligned}
\frac{\partial}{\partial \theta_k} \log Z(\underline{\mathbf{x}};\boldsymbol{\theta}) &= \frac{1}{Z(\underline{\mathbf{x}};\boldsymbol{\theta})} \sum_{\underline{y}\in\mathcal{Y}^n} \frac{\partial}{\partial \theta_k} \left[\exp \sum_{j=1}^{D} \theta_j F_j(\underline{\mathbf{x}}, \underline{y}) \right] \\
&= \frac{1}{Z(\underline{\mathbf{x}};\boldsymbol{\theta})} \sum_{\underline{y}\in\mathcal{Y}^n} F_k(\underline{\mathbf{x}}, \underline{y}) \exp \sum_{j=1}^{D} \theta_j F_j(\underline{\mathbf{x}}, \underline{y}) \\
&= \sum_{\underline{y}\in\mathcal{Y}^n} F_k(\underline{\mathbf{x}}, \underline{y}) \frac{\exp \sum_j \theta_j F_j(\underline{\mathbf{x}}, \underline{y})}{Z(\underline{\mathbf{x}};\boldsymbol{\theta})} \\
&= \sum_{\underline{y}\in\mathcal{Y}^n} F_k(\underline{\mathbf{x}}, \underline{y}) p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta}) \\
&= \mathbb{E}_{p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta})} \left[F_k(\underline{\mathbf{x}}, \underline{y}) \right].
\end{aligned}$$

NLL gradient

Gradient:
$$\frac{\partial L(\mathcal{D};\boldsymbol{\theta})}{\partial \theta_k} = \sum_{q=1}^{N} \left\{ \frac{\partial}{\partial \theta_k} \log Z(\underline{\mathbf{x}}^{(q)};\boldsymbol{\theta}) - F_k(\underline{\mathbf{x}}^{(q)}, \underline{y}^{(q)}) \right\}.$$

$\frac{\partial}{\partial \theta_k} \log Z(\underline{\mathbf{x}};\boldsymbol{\theta}) = \mathbb{E}_{p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta})}\left[F_k(\underline{\mathbf{x}}, \underline{y})\right]$: conditional expectation given $\underline{\mathbf{x}}$.

$$\frac{\partial L(\mathcal{D};\boldsymbol{\theta})}{\partial \theta_k} = \sum_{q=1}^{N} \left\{ \mathbb{E}_{p(\underline{y}|\underline{\mathbf{x}}^{(q)};\boldsymbol{\theta})}\left[F_k(\underline{\mathbf{x}}^{(q)}, \underline{y})\right] - F_k(\underline{\mathbf{x}}^{(q)}, \underline{y}^{(q)}) \right\}.$$

Optimality condition

- Setting the derivatives to 0, *i.e.* $\frac{\partial L(\mathcal{D};\boldsymbol{\theta})}{\partial \theta_k} = 0$, yields:

$$\sum_{q=1}^{N} \mathbb{E}_{p(\underline{y}|\underline{\mathbf{x}}^{(q)};\boldsymbol{\theta})}\left[F_k(\underline{\mathbf{x}}^{(q)}, \underline{y})\right] = \sum_{q=1}^{N} F_k(\underline{\mathbf{x}}^{(q)}, \underline{y}^{(q)});\ 1 \leq k \leq D$$

- No closed-form solution: numerical optimization is again needed.

- Need to compute $\mathbb{E}_{p(\underline{y}|\underline{\mathbf{x}}^{(q)};\boldsymbol{\theta})}\left[F_k(\underline{\mathbf{x}}^{(q)}, \underline{y})\right]$ efficiently.

Optimality condition

- Setting the derivatives to 0, *i.e.* $\frac{\partial L(\mathcal{D};\boldsymbol{\theta})}{\partial \theta_k} = 0$, yields:

$$\frac{1}{N}\sum_{q=1}^{N} \mathbb{E}_{p(\underline{y}|\underline{\mathbf{x}}^{(q)};\boldsymbol{\theta})}\left[F_k(\underline{\mathbf{x}}^{(q)},\underline{y})\right] \;=\; \frac{1}{N}\sum_{q=1}^{N} F_k(\underline{\mathbf{x}}^{(q)},\underline{y}^{(q)});\; 1 \leq k \leq D$$

- Average expectation under the model = empirical mean.
- No closed-form solution: numerical optimization is again needed.
- Need to compute $\mathbb{E}_{p(\underline{y}|\underline{\mathbf{x}}^{(q)};\boldsymbol{\theta})}\left[F_k(\underline{\mathbf{x}}^{(q)},\underline{y})\right]$ efficiently.

Efficient gradient computation

$$\begin{aligned}
\mathbb{E}_{p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta})}\left[F_k(\underline{\mathbf{x}},\underline{y})\right] &= \sum_{\underline{y}\in\mathcal{Y}^n} F_k(\underline{\mathbf{x}},\underline{y})p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta}) \\
&= \sum_{i=1}^{n}\sum_{\underline{y}\in\mathcal{Y}^n} f_k(y_{i-1},y_i,\underline{\mathbf{x}})p(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta}) \\
&= \sum_{i=1}^{n}\sum_{y_{i-1},y_i\in\mathcal{Y}^2} f_k(y_{i-1},y_i,\underline{\mathbf{x}})p(y_{i-1},y_i|\underline{\mathbf{x}};\boldsymbol{\theta})
\end{aligned}$$

$p(y_{i-1},y_i|\underline{\mathbf{x}};\boldsymbol{\theta})$ is the **marginal probability** which thanks to the **forward-backward** algorithm is obtained by:

$$p(y_{i-1},y_i|\underline{\mathbf{x}}) = \frac{1}{Z(\underline{\mathbf{x}})}\boldsymbol{\alpha}_{i-1}(y_{i-1})M_i(y_{i-1},y_i,\underline{\mathbf{x}})\boldsymbol{\beta}_i(y_i).$$

Optimization

Now that we are able to compute the **gradient**, we can use a descent method to solve for $\boldsymbol{\theta}$.

Many **algorithms** are available (see Sokolovska, 2010; Lavergne et al., 2010):

- Generalized iterative scaling (Lafferty et al., 2001): original algorithm, slow convergence, suboptimal.
- Conjugate gradient (Wallach, 2002): faster convergence, better quality.
- **L-BFGS** (McCallum, 2002): fast convergence, scalable; a good option, most used.
- Stochastic gradient: suboptimal, simple, online, **large-scale applications**.

Regularization

Using ℓ_2-norm

- Redefine the objective function as: $L(\mathcal{D};\boldsymbol{\theta}) = L(\mathcal{D};\boldsymbol{\theta}) + \frac{||\boldsymbol{\theta}||_2^2}{2\sigma^2}$;
  σ^2: a free parameter penalizing large weights (as in **ridge regression**).

- The gradient coefficients become:
  $\frac{\partial L(\mathcal{D};\boldsymbol{\theta})}{\partial \theta_k} = \sum_{q=1}^{N} \left\{ \mathbb{E}_{p(\underline{y}|\underline{\mathbf{x}}^{(q)};\boldsymbol{\theta})} \left[F_k(\underline{\mathbf{x}}^{(q)}, \underline{y}) \right] - F_k(\underline{\mathbf{x}}^{(q)}, \underline{y}^{(q)}) \right\} + \frac{\theta_k}{\sigma^2}$.
- **Advantages**:
  - The objective becomes **strictly convex**.
  - Shrinkage of $\boldsymbol{\theta}$ coefficients is achieved avoiding overfitting and numerical problems.

- σ^2 needs to be tuned (usually by cross-validation).

Regularization

Using ℓ_1-norm to perform feature selection

- Redefine the objective function as:
  $L(\mathcal{D};\boldsymbol{\theta}) = L(\mathcal{D};\boldsymbol{\theta}) + \rho||\boldsymbol{\theta}||_1 = L(\mathcal{D};\boldsymbol{\theta}) + \rho\sum_{j=1}^{D}|\theta_j|$ (as in the LASSO).

- **Advantage:** performs **feature selection**

  in some NLP apps: up to 95% of the features can be discarded without affecting performance! (see Sokolovska, 2010).

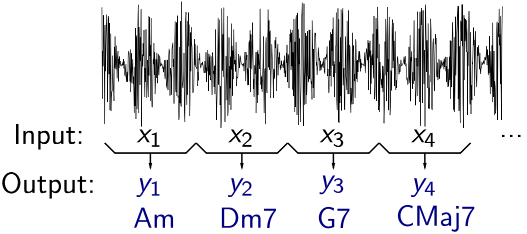
- **Difficulties**:
  - The regularizer is **not differentiable** at zero: specific optimization techniques needed (Sokolovska, 2010).
  - In configurations with groups of highly correlated features, tend to select randomly one feature in each group.

Motivation

Problem: the CRF model does not support **hidden states**.

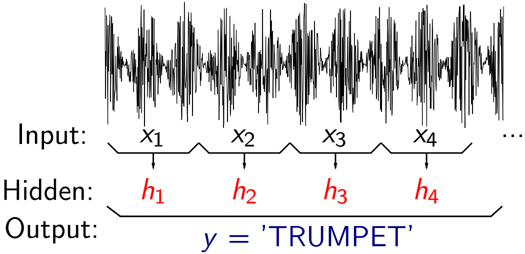
CRF

$$\mathcal{D} = \{(\underline{\mathbf{x}}^{(i)}, \underline{y}^{(i)})\}_i$$



Hidden-state CRF

$$\mathcal{D} = \{(\underline{\mathbf{x}}^{(i)}, y^{(i)})\}_i$$

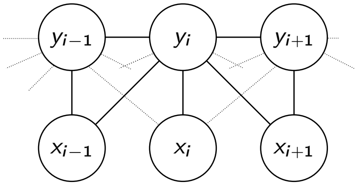


Motivation

Problem: the CRF model does not support **hidden states**.

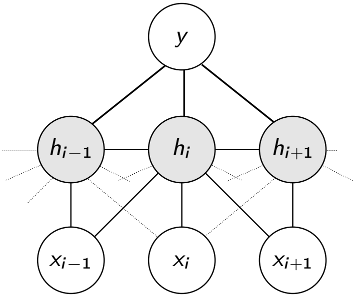
CRF

$$\mathcal{D} = \{(\underline{\mathbf{x}}^{(i)}, \underline{y}^{(i)})\}_i$$



Hidden-state CRF

$$\mathcal{D} = \{(\underline{\mathbf{x}}^{(i)}, y^{(i)})\}_i$$



The HCRF model

(Quattoni et al., 2007)

- Each sequence of observations $\underline{\mathbf{x}} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$ is associated with:
  - a unique label y;
  - a sequence of **latent variables** $\underline{h} = (h_1, \cdots, h_n)$, where $h_i \in \mathcal{H}$.

HCRF model definition

$$p(y, \underline{h} | \underline{\mathbf{x}}; \boldsymbol{\theta}) = \frac{1}{Z(\underline{\mathbf{x}}, \boldsymbol{\theta})} \exp \sum_{j=1}^{D} \theta_j F_j(\underline{\mathbf{x}}, y, \underline{h})$$

$Z(\underline{\mathbf{x}}, \boldsymbol{\theta}) = \sum_{y,\underline{h}} \exp \sum_{j=1}^{D} \theta_j F_j(\underline{\mathbf{x}}, y, \underline{h})$; $\boldsymbol{\theta} = \{\theta_1, \cdots, \theta_D\}$.

Inference in HCRF

- Using the HCRF model: $p(y, \underline{h}|\underline{\mathbf{x}}; \boldsymbol{\theta}) = \frac{1}{Z(\underline{\mathbf{x}}, \boldsymbol{\theta})} \exp \sum_{j=1}^{D} \theta_j F_j(\underline{\mathbf{x}}, y, \underline{h})$; entails being able to compute:
  - $\hat{y} = \text{argmax}_{y \in \mathcal{Y}}\, p(y|\underline{\mathbf{x}}; \boldsymbol{\theta}^*)$, to classify new test cases;
  - the partition function $Z(\underline{\mathbf{x}}, \boldsymbol{\theta})$, to evaluate posterior probabilities.

- Let $Z'(y, \underline{\mathbf{x}}, \boldsymbol{\theta}) \triangleq \sum_{\underline{h} \in \mathcal{H}^n} \exp \sum_{j=1}^{D} \theta_j F_j(\underline{\mathbf{x}}, y, \underline{h})$: **marginalization** wrt $\underline{h}$.

- We have:
  - $p(y|\underline{\mathbf{x}}; \boldsymbol{\theta}) = \sum_{\underline{h} \in \mathcal{H}^n} p(y, \underline{h}|\underline{\mathbf{x}}; \boldsymbol{\theta}) = \frac{Z'(y, \underline{\mathbf{x}}, \boldsymbol{\theta})}{\sum_y Z'(y, \underline{\mathbf{x}}, \boldsymbol{\theta})}$;
  - $Z(\underline{\mathbf{x}}, \boldsymbol{\theta}) = \sum_y Z'(y, \underline{\mathbf{x}}, \boldsymbol{\theta})$.
- $Z'(y, \underline{\mathbf{x}}, \boldsymbol{\theta})$ can be easily computed using forward/backward recursions (as done in CRF).

Negative log-likelihood

$$
\begin{aligned}
L(\mathcal{D};\boldsymbol{\theta}) &\triangleq -\sum_{q=1}^{N} \log p(y^{(q)}|\underline{\mathbf{x}}^{(q)};\boldsymbol{\theta}) \\
&= \sum_{q=1}^{N} \left\{ \log \left(\sum_{y} Z'(y,\underline{\mathbf{x}}^{(q)},\boldsymbol{\theta}) \right) - \log Z'(y^{(q)},\underline{\mathbf{x}}^{(q)},\boldsymbol{\theta}) \right\};
\end{aligned}
$$

$$
Z'(y,\underline{\mathbf{x}},\boldsymbol{\theta}) \triangleq \sum_{\underline{h}\in\mathcal{H}^n} \exp \sum_{j=1}^{D} \theta_j F_j(\underline{\mathbf{x}},y,\underline{h})
$$

$L(D;\boldsymbol{\theta})$ is no longer convex $\rightarrow$ convergence to a **local** minimum.

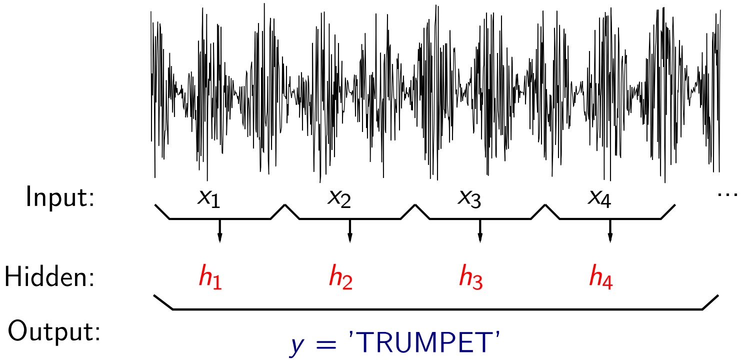
NLL gradient

$$
\begin{aligned}
\frac{\partial L(\mathcal{D};\boldsymbol{\theta})}{\partial \theta_k} &= \sum_{q=1}^{N} \Bigg\{ \sum_{y,\underline{h}} F_k(\underline{\mathbf{x}}^{(q)}, y, \underline{h}) p(y, \underline{h}|\underline{\mathbf{x}};\boldsymbol{\theta}) \\
&\qquad - \sum_{\underline{h}} F_k(\underline{\mathbf{x}}^{(q)}, y^{(q)}, \underline{h}) p(\underline{h}|y^{(q)}, \underline{\mathbf{x}}^{(q)};\boldsymbol{\theta}) \Bigg\}
\end{aligned}
$$

which can be again computed using the **forward-backward** method.

A gradient descent method (L-BFGS) can be again used to solve for $\boldsymbol{\theta}$.

Application to musical instrument classification



Feature functions used

Following (Quattoni et al., 2007)

$$\Psi(\underline{\mathbf{x}}, y, \underline{h}, \boldsymbol{\theta}) = \sum_{i=1}^{N} < \boldsymbol{\theta}(h_i), \mathbf{x}_i > + \sum_{i=1}^{N} \theta(y, h_i) + \sum_{i=1}^{N} \theta(y, h_{i-1}, h_i)$$

- $< \boldsymbol{\theta}(h_i), \mathbf{x}_i >$: compatibility between observation $\mathbf{x}_i$ and hidden state $h_i \in \mathcal{H}$;
- $\theta(y, h_i)$: compatibility between hidden state h_i and label y;
- $\theta(y, h_{i-1}, h_i)$: compatibility between transition $h_{i-1} \leftrightarrow h_i$ and label y.

Evaluation

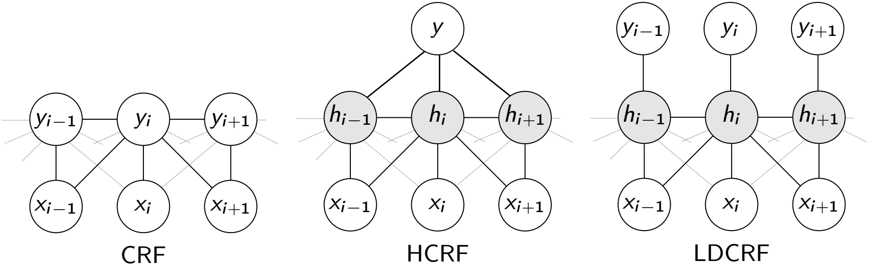
- Classifying 1-second long segments of solo excerpts of **Cello**, **Guitar**, **Piano**, **Bassoon** and **Oboe.**
- Data:
  - training set: 2505 segments (*i.e. 42');*
  - testing set: 2505 segments.
- Classifiers:
  - ℓ_2-regularized **HCRF** with 3 hidden states;
  - **Linear SVM**.
- Features: 47 cepstral, perceptual and temporal features.
- Results

Classifier	SVM	HCRF
Average accuracy	75%	76%

Other extensions

- **LDCRF**: Latent-Dynamic Conditional Random Field (Sung and Jurafsky, 2009)

  modeling both hidden-states and structured-outputs.



Other extensions

- **LDCRF**: Latent-Dynamic Conditional Random Field (Sung and Jurafsky, 2009)

  modeling both hidden-states and structured-outputs.

- **Kernel-CRF** (Laferty et al., 2004; Altun et al., 2004)

  introducing implicit features to account for (non-linear) interactions between original features.

- **Semi-Markov CRF** (Sarawagi and Cohen, 2005)

  modeling segment-level labels.

- **CCRF**: Continuous CRF (Qin and Liu, 2008)

  modeling continuous labels in a regression setting.

Take-home messages I

- CRFs are powerful structured-data prediction models (more flexible than HMMs and other more general Bayesian networks) as they are:
  - **discriminative models**: focus on modeling the target labels;

  → can handle a high number of feature functions, including transition features, and account for long-range dependencies.

  - **undirected models**: no need to normalize potentials locally.

  → allow for incorporating prior knowledge about constraints and label dependencies in an intuitive way.

- Easily **extendable** with key mechanisms: regularization, sparsity, latent variables, kernels...

CRF software packages

Package	Language	Main features	Reference
CRF++	C++	Linear-chain CRF, NLP, L-BFGS optimization	(Taku-ku, 2003)
crfChain	Matlab, C mex	Linear-chain CRF, categorical features, L-BFGS optimization	Schmidt (2008)
CRFsuite	C++, Python	Linear-chain CRF, NLP, various regularization and optimization methods (L-BFGS), designed for fast training	(Okazaki, 2007)
HCRF library	C++, Matlab, Python	CRF, HCRF, LDCRF, continuous inputs, L-BFGS optimization	(Morency, 2010)
Mallet	Java	CRF, maxent, HMM, NLP, text feature extraction routines, various optimization methods (L-BFGS)	(McCallum, 2002)
Wapiti	C99	Linear-chain CRF, NLP, large label and feature sets, various regularization and optimization methods (L-BFGS, SGD), multi-threaded	(Lavergne et al., 2010)

CRF tutorials

- Charles Sutton and Andrew McCallum. **An Introduction to Conditional Random Fields for Relational Learning**. In *Introduction to Statistical Relational Learning*. Edited by Lise Getoor and Ben Taskar. MIT Press, 2006.
- Charles Elkan. **Log-linear Models and Conditional Random Fields**. *Notes for a tutorial at CIKM'08*, October 2008.
- Roman Klinger and Katrin Tomanek. **Classical Probabilistic Models and Conditional Random Fields**. *Algorithm Engineering Report TR07-2-2013*, December 2007.
- Hanna M. Wallach. **Conditional Random Fields: An Introduction**. Technical Report MS-CIS-04-21, Department of Computer and Information Science, University of Pennsylvania, 2004.
- Roland Memisevic. **An Introduction to Structured Discriminative Learning**. Technical Report, University of Toronto, 2006.
- Rahul Gupta. **Conditional Random Fields**. Unpublished report, IIT Bombay, 2006.

Bibliography I

Y. Altun, A. Smola, and T. Hofmann. Exponential families for conditional random fields. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, volume 0, 2004. URL http://dl.acm.org/citation.cfm?id=1036844.

A. L. Berger, S. A. D. Pietra, and Vincent J. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1), 1996.

J. Corey and K. Fujinaga. A cross-validated study of modelling strategies for automatic chord recognition in audio. *International Conference on Music Information Retrieval (ISMIR)*, 2007. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.5760&rep=rep1&type=pdf.

M. E. P. Davies and M. D. Plumbley. Context-dependent beat tracking of musical audio. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(3):1009–1020, 2007.

Z. Duan, L. Lu, A. Msra, and H. District. Collective annotation of music from multiple semantic categories. In *Proceedings of the 9th International Conference of Music Information Retrieval*, pages 237–242, Philadelphia, Pennsylvania, USA, 2008.

M. Dunham and K. Murphy. pmtk3: probabilistic modeling toolkit for Matlab/Octave, version 3, 2010. URL https://code.google.com/p/pmtk3/.

S. Durand. *Estimation de la position des temps et des premiers temps d'un morceau de musique*. Master thesis, Universite Pierre et Marie Curie, 2013.

D. Ellis. Beat Tracking by Dynamic Programming. *Journal of New Music Research*, 36(1):51–60, 2007. ISSN 09298215. URL http://www.informaworld.com/openurl?genre=article&doi=10.1080/09298210701653344&magic=crossref.

A. Gunawardana, M. Mahajan, A. Acero, and J. Platt. Hidden conditional random fields for phone classification. In *Interspeech*, 2005. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.113.8190&rep=rep1&type=pdf.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009. ISBN 0387848576. URL http://books.google.com/books?id=tVIjmNS3Ob8C&pgis=1.

Bibliography II

X. He, R. Zemel, and M. Carreira-Perpinan. Multiscale conditional random fields for image labeling. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR*, volume 2, pages 695–702. IEEE, 2004. ISBN 0-7695-2158-4. doi: 10.1109/CVPR.2004.1315232. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1315232.

W.-T. Hong. Speaker identification using Hidden Conditional Random Field-based speaker models. In *2010 International Conference on Machine Learning and Cybernetics*, pages 2811–2816. IEEE, July 2010. ISBN 978-1-4244-6526-2. doi: 10.1109/ICMLC.2010.5580793. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5580793&contentType=Conference+Publications&searchField=Search_All&queryText=Hidden+Conditional+Random+Fields.

V. Imbrasaite, T. Baltrusaitis, and P. Robinson. Emotion tracking in music using continuous conditional random fields and relative feature representation. In *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. IEEE, July 2013. ISBN 978-1-4799-1604-7. doi: 10.1109/ICMEW.2013.6618357. URL http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6618357.

F. V. Jensen and T. D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer, 2007. ISBN 0387682813. URL http://books.google.fr/books/about/Bayesian_Networks_and_Decision_Graphs.html?id=goSLmQq4UBcC&pgis=1.

C. Joder, S. Essid, and G. Richard. A Conditional Random Field Viewpoint of Symbolic Audio-to-Score Matching. In *ACM Multimedia 2010*, Florence, Italy, 2010.

C. Joder, S. Essid, and G. Richard. A Conditional Random Field Framework for Robust and Scalable Audio-to-Score Matching. *IEEE Transactions on Audio, Speech and Language Processing*, 19(8):2385–2397, 2011.

C. Joder, S. Essid, and G. Richard. Learning Optimal Features for Polyphonic Audio-to-Score Alignment. *IEEE Transactions on Audio Speech and Language Processing*, 2013. doi: 10.1109/TASL.2013.2266794.

A. P. Klapuri, A. J. Eronen, and J. T. Astola. Analysis of the meter of acoustic musical signals. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(1):342–355, 2006.

Bibliography III

J. Laferty, Y. Liu, and X. Zhu. Kernel Conditional Random Fields: Representation, Clique Selection, and Semi-Supervised Learning. In *International Conference on Machine Learning*, 2004. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Kernel+Conditional+Random+Fields+:+Representation+,+Clique+Selection+,+and+Semi-Supervised+Learning#5.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Machine learning international workshop then conference*, pages 282–289. Citeseer, 2001. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.9849&rep=rep1&type=pdf.

T. Lavergne, O. Cappe, and F. Yvon. Practical very large scale CRFs. In *48th Annual Meeting of the Association for Computational Linguistics (ACL)*, number July, pages 504–513, Uppsala, Sweeden, 2010. URL http://www.aclweb.org/anthology/P10-1052.

C.-X. Li. *Exploiting label correlations for multi-label classification*. Master thesis, UC San Diego, 2011.

A. K. McCallum. Mallet: A machine learning for language toolkit, 2002. URL http://mallet.cs.umass.edu.

L.-P. Morency. HCRF: Hidden- state Conditional Random Field Library, 2010. URL http://sourceforge.net/projects/hcrf/.

L.-P. Morency, A. Quattoni, and T. Darrell. Latent-Dynamic Discriminative Models for Continuous Gesture Recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, June 2007. ISBN 1-4244-1179-3. doi: 10.1109/CVPR.2007.383299. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4270324.

J. Morris and E. Fosler-Lussier. Conditional Random Fields for Integrating Local Discriminative Classifiers. *TASLP*, 16(3):617–628, 2008. ISSN 1558-7916. doi: 10.1109/TASL.2008.916057. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4451149.

N. Okazaki. CRFsuite: a fast implementation of Conditional Random Fields (CRFs), 2007. URL http://www.chokkan.org/software/crfsuite/.

Bibliography IV

S. Ortmanns, H. Ney, and A. Eiden. Language-model look-ahead for large vocabulary speech recognition. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP '96*, volume 4, pages 2095–2098. IEEE, 1996. ISBN 0-7803-3555-4. doi: 10.1109/ICSLP.1996.607215. URL http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=607215.

T. Qin and T. Liu. Global ranking using continuous conditional random fields. In *Advances in Neural and Information Processing Systems*, 2008. URL http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2008_0518.pdf.

A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *IEEE transactions on pattern analysis and machine intelligence*, 29(10):1848–53, Oct. 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1124. URL http://www.ncbi.nlm.nih.gov/pubmed/17699927.

S. Reiter, B. Schuller, and G. Rigoll. Hidden Conditional Random Fields for Meeting Segmentation. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 639–642. IEEE, July 2007. ISBN 1-4244-1016-9. doi: 10.1109/ICME.2007.4284731. URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=4284731&contentType=Conference+Publications&searchField=Search_All&queryText=Hidden+Conditional+Random+Fields.

O. Rudovic, V. Pavlovic, and M. Pantic. Kernel conditional ordinal random fields for temporal segmentation of facial action units. In *Computer Vision - ECCV*, pages 1–10, 2012. URL http://link.springer.com/chapter/10.1007/978-3-642-33868-7_26.

S. Sarawagi and W. Cohen. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, 2005. URL http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2005_427.pdf.

E. Schmidt and Y. Kim. Modeling musical emotion dynamics with conditional random fields. In *International Conference on Music Information Retrieval (ISMIR)*, number Ismir, pages 777–782, Miami, FL, USA, 2011. URL http://music.ece.drexel.edu/files/Navigation/Publications/Schmidt2011c.pdf.

M. Schmidt. crfChain, 2008. URL http://www.di.ens.fr/~mschmidt/Software/crfChain.html.

Bibliography V

B. Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. *International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, 2004. URL http://dl.acm.org/citation.cfm?id=1567618.

N. Sokolovska. *Contributions to the estimation of probabilistic discriminative models: semi-supervised learning and feature selection*. PhD thesis, Telecom ParisTech, 2010.

Y.-H. Sung and D. Jurafsky. Hidden Conditional Random Fields for phone recognition. *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 107–112, Dec. 2009. doi: 10.1109/ASRU.2009.5373329. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5373329.

Taku-ku. CRF++, 2003. URL http://crfpp.googlecode.com/svn/trunk/doc/index.html.

B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth conference on Uncertainty in Artificial Intelligence (UAI02)*, 2002. URL http://dl.acm.org/citation.cfm?id=2073934.

H. M. Wallach. *Efficient Training of Conditional Random Fields*. PhD thesis, University of Edinburgh, 2002.

S. B. Wang, A. Quattoni, L.-P. Morency, and D. Demirdjian. Hidden Conditional Random Fields for Gesture Recognition. pages 1521–1527. IEEE Computer Society, 2006. ISBN 0-7695-2597-0. URL http://portal.acm.org/citation.cfm?id=1153171.1153627.

G. Yuan, K. Chang, C. Hsieh, and C. Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *The Journal of Machine Learning Research*, 2010. URL http://dl.acm.org/citation.cfm?id=1953034.

J. Zhu and T. Hastie. Support vector machines, kernel logistic regression and boosting. *Multiple Classifier Systems*, 2364:16–26, 2002. URL http://link.springer.com/chapter/10.1007/3-540-45428-4_2.

LR model learning with stochastic gradient descent (SGL)

- **Idea**: make gradient updates based on **one** training example **at a time**
- **Use**: $\frac{\partial L(\mathcal{D};\tilde{\mathbf{w}})}{\partial w_j} = (y_i - p(\mathbf{x}_i;\tilde{\mathbf{w}}))\, x_{ji}$

Algorithm

- Initialise $\tilde{\mathbf{w}}$
- Repeat (until convergence)
  - Randomly permute training examples $\mathbf{x}_i$
  - For $i = 1 : N$

    $w_j \leftarrow w_j + t\,(y_{\sigma_i} - p_{\sigma_i})\, x_{j\sigma_i}\ ;\ j = 1, \cdots, D$

- t : *step size*, to be tuned
- **Complexity of SGL:** $O(NFD)$ per *epoch*; with F the average number of non-zero feature coefficients per example; an *epoch* is a "complete" update using all training examples.

Support Vector Machines

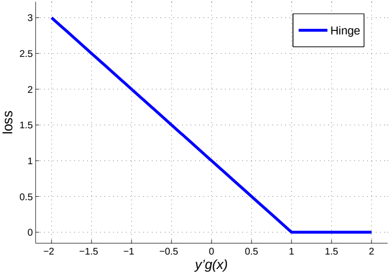
Recalling SVM as a regularized function fitting problem

- The SVM solution, $g(\mathbf{x}) = w_0 + \mathbf{w}^T \phi(\mathbf{x})$, can be found by solving:

$$\min_{\tilde{\mathbf{w}}} \sum_{i=1}^{N} \left[1 - y_i' g(\mathbf{x}_i)\right]_+ + \frac{\gamma}{2} ||g||^2_{\mathcal{H}_\mathcal{K}} \; ; \; y_i' \in \{-1, 1\}$$

Hinge loss

$$[1 - y_i' g(\mathbf{x}_i)]_+ = \max(0, 1 - y_i' g(\mathbf{x}_i))$$



KLR vs SVM

- Let $y_i' = \begin{cases} 1 & \text{if } y_i = 1 \\ -1 & \text{if } y_i = 0 \end{cases}$
- The negative log-likelihood of the KLR model can then be written as $L(\mathcal{D}; \tilde{\mathbf{w}}) = \sum_{i=1}^{N} \log\left(1 + \exp -y_i' g(\mathbf{x}_i)\right)$.
- Both KLR and SVM solve:

$$\min_{\tilde{\mathbf{w}}} \sum_{i=1}^{N} l(y_i' g(\mathbf{x}_i)) + \frac{\lambda}{2} ||g||^2_{\mathcal{H}_\mathcal{K}};$$

KLR

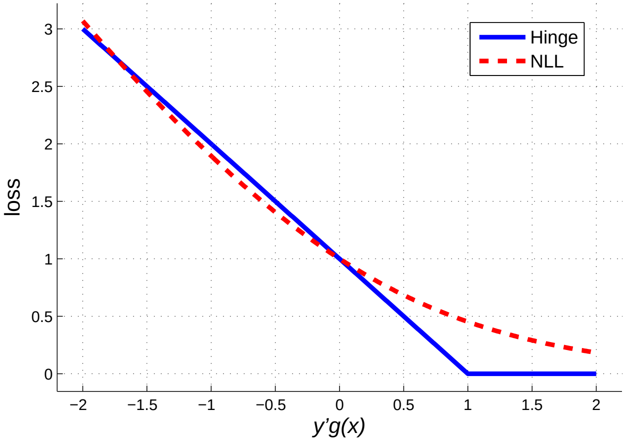
$$l(y_i' g(\mathbf{x}_i)) = \log\left(1 + \exp -y_i' f(\mathbf{x}_i)\right)$$

SVM

$$l(y_i' g(\mathbf{x}_i)) = \left[1 - y_i' f(\mathbf{x}_i)\right]_+$$

KLR vs SVM

Hinge vs negative binomial log-likelihood



Decoding the optimal sequence

$$\text{Let: } g_i(y_{i-1}, y_i) \triangleq \sum_{j=1}^{D} \theta_j f_j(y_{i-1}, y_i, \underline{\mathbf{x}}, i); \text{ then:}$$

$$\hat{y} = \underset{\underline{y}}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{j=1}^{D} \theta_j f_j(y_{i-1}, y_i, \underline{\mathbf{x}}, i) = \underset{\underline{y}}{\operatorname{argmax}} \sum_{i=1}^{n} g_i(y_{i-1}, y_i).$$

$$\begin{aligned} \text{Let } \delta_m(s) &\triangleq \max_{\{y_1, \cdots, y_{m-1}\}} \left[\sum_{i=1}^{m-1} g_i(y_{i-1}, y_i) + g_m(y_{m-1}, s) \right] \\ &= \max_{\{y_1, \cdots, y_{m-1}\}} \sum_{i=1}^{m-1} g_i(y_{i-1}, y_i) + \max_{y_{m-1}} g_m(y_{m-1}, s). \end{aligned}$$

Viterbi decoding

$$\delta(s) = \max_{\{y_1,\cdots,y_{m-1}\}} \sum_{i=1}^{m-1} g_i(y_{i-1}, y_i) + \max_{y_{m-1}} g_m(y_{m-1}, s)$$

$$\begin{aligned}\text{So: } \delta_{m-1}(y_{m-1}) &= \max_{\{y_1,\cdots,y_{m-2}\}} \left[\sum_{i=1}^{m-2} g_i(y_{i-1}, y_i) + g_{m-1}(y_{m-2}, y_{m-1})\right] \\ &= \max_{\{y_1,\cdots,y_{m-2}\}} \sum_{i=1}^{m-1} g_i(y_{i-1}, y_i).\end{aligned}$$

$$\delta_m(s) = \max_{y_{m-1}\in\mathcal{Y}} [\delta_{m-1}(y_{m-1}) + g_m(y_{m-1}, s)].$$

Viterbi decoding

The algorithm

Initialization:

$$\begin{aligned}\delta_1(s) &= g_1(y_0, s);\ \forall s \in \mathcal{Y};\ y_0 = \text{start} \\ \psi_1(s) &= \text{start}\end{aligned}$$

Recursion: $\forall s \in \mathcal{Y};\ 1 \leq m \leq n$

$$\begin{aligned}\delta_m(s) &= \max_{y \in \mathcal{Y}} [\delta_{m-1}(y) + g_m(y, s)] \\ \psi_m(s) &= \operatorname*{argmax}_{y \in \mathcal{Y}} [\delta_{m-1}(y) + g_m(y, s)]\end{aligned}$$

Termination:

$$\begin{aligned}\delta_n(y_n^*) &= \max_{y \in \mathcal{Y}} \delta_n(y) \\ y_n^* &= \operatorname*{argmax}_{y \in \mathcal{Y}} \delta_n(y)\end{aligned}$$

Path backtracking:

$$y_m^* = \psi_{m+1}(y_{m+1}^*);\ m = n-1, n-2, \cdots, 1.$$

The forward recursion

Define $\boldsymbol{\alpha}$ scores as:

$$
\begin{aligned}
\boldsymbol{\alpha}_1(y_1) &= M_1(y_0, y_1) \\
\boldsymbol{\alpha}_2(y_2) &= \sum_{y_1 \in \mathcal{Y}} M_2(y_1, y_2)\boldsymbol{\alpha}_1(y_1) \\
\boldsymbol{\alpha}_3(y_3) &= \sum_{y_2 \in \mathcal{Y}} M_3(y_2, y_3)\boldsymbol{\alpha}_2(y_2) \\
&\vdots \\
\boldsymbol{\alpha}_m(y_m) &= \sum_{y_{m-1}} M_m(y_{m-1}, y_m)\boldsymbol{\alpha}_{m-1}(y_{m-1})\,;\ 2 \leq m \leq n
\end{aligned}
$$

The forward recursion

Define $\boldsymbol{\alpha}$ scores as:

$$
\begin{aligned}
\boldsymbol{\alpha}_1(y_1) &= M_1(y_0, y_1) \\
\boldsymbol{\alpha}_2(y_2) &= \sum_{y_1 \in \mathcal{Y}} M_2(y_1, y_2)\boldsymbol{\alpha}_1(y_1) \\
\boldsymbol{\alpha}_3(y_3) &= \sum_{y_2 \in \mathcal{Y}} M_3(y_2, y_3)\boldsymbol{\alpha}_2(y_2) = \sum_{y_1, y_2} M_3(y_2, y_3) M_2(y_1, y_2) M_1(y_0, y_1) \\
&\vdots \\
\boldsymbol{\alpha}_m(y_m) &= \sum_{y_{m-1}} M_m(y_{m-1}, y_m)\boldsymbol{\alpha}_{m-1}(y_{m-1})\,;\; 2 \leq m \leq n
\end{aligned}
$$

At the end of the sequence

$$
\sum_{y_n \in \mathcal{Y}} \boldsymbol{\alpha}_n(y_n) = \sum_{\underline{y} \in \mathcal{Y}^n} \prod_{i=1}^{n} M_i(y_{i-1}, y_i, \underline{\mathbf{x}}) = Z(\underline{\mathbf{x}}, \boldsymbol{\theta}).
$$

The forward recursion

Define $\boldsymbol{\alpha}$ scores as:

$$\boldsymbol{\alpha}_m(y_m) = \sum_{y_{m-1}} M_m(y_{m-1}, y_m)\boldsymbol{\alpha}_{m-1}(y_{m-1})\,;\; 2 \leq m \leq n.$$

At the end of the sequence

$$\sum_{y_n \in \mathcal{Y}} \boldsymbol{\alpha}_n(y_n) = \sum_{\underline{y} \in \mathcal{Y}^n} \prod_{i=1}^{n} M_i(y_{i-1}, y_i, \underline{\mathrm{x}}) = Z(\underline{\mathrm{x}}, \boldsymbol{\theta}).$$

Complexity: reduced from $O(K^n)$ to $O(nK^2)$.

The backward recursion

$$
\begin{aligned}
\beta_m(y_m) &= \sum_{y_{m+1}\in\mathcal{Y}} M_{m+1}(y_m, y_{m+1})\beta_{m+1}(y_{m+1})\,;\ 1 \leq m \leq n-1 \\
\beta_n(y_n) &= 1
\end{aligned}
$$

At the beginning of the sequence

$$
Z(\underline{\mathbf{x}}, \boldsymbol{\theta}) = \sum_{y_1\in\mathcal{Y}} M_1(y_0, y_1)\beta_1(y_1).
$$

Marginal probability

$p(y_{m-1}, y_m|\underline{\mathbf{x}}) = \sum_{\underline{y}\backslash\{y_{m-1},y_m\}} p(\underline{y}|\mathbf{x})$;

$\underline{y}\backslash\{y_{m-1}, y_m\} \triangleq \{y_1, \cdots, y_{m-2}, y_{m+1}, \cdots, y_n\}$.

$$
\begin{aligned}
p(y_{m-1}, y_m|\underline{\mathbf{x}}) &= \frac{1}{Z(\underline{\mathbf{x}})} \sum_{\underline{y}\backslash\{y_{m-1},y_m\}} \prod_{i=1}^{n} M_i(y_{i-1}, y_i, \underline{\mathbf{x}}) \\
&= \frac{1}{Z(\underline{\mathbf{x}})} \sum_{\underline{y}\backslash\{y_{m-1},y_m\}} \prod_{i=1}^{m-1} M_i(y_{i-1}, y_i, \underline{\mathbf{x}}) \times M_m(y_{m-1}, y_m, \underline{\mathbf{x}}) \\
&\qquad \times \prod_{i=m+1}^{n} M_i(y_{i-1}, y_i, \underline{\mathbf{x}})
\end{aligned}
$$

Marginal probability

$$p(y_{m-1}, y_m|\underline{\mathbf{x}}) = \frac{1}{Z(\underline{\mathbf{x}})} M_m(y_{m-1}, y_m, \underline{\mathbf{x}}) \times \sum_{\{y_1,\dots,y_{m-2}\}} \prod_{i=1}^{m-1} M_i(y_{i-1}, y_i, \underline{\mathbf{x}})$$

$$\times \sum_{\{y_{m+1},\dots,y_n\}} \prod_{i=m+1}^{n} M_i(y_{i-1}, y_i, \underline{\mathbf{x}})$$

$$p(y_{m-1}, y_m|\underline{\mathbf{x}}) = \frac{1}{Z(\underline{\mathbf{x}})} \alpha_{m-1}(y_{m-1}) M_m(y_{m-1}, y_m, \underline{\mathbf{x}}) \beta_m(y_m).$$