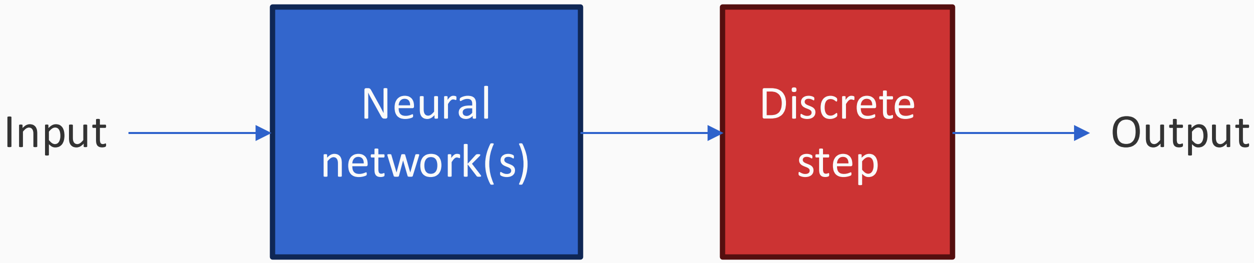
Training with structured outputs

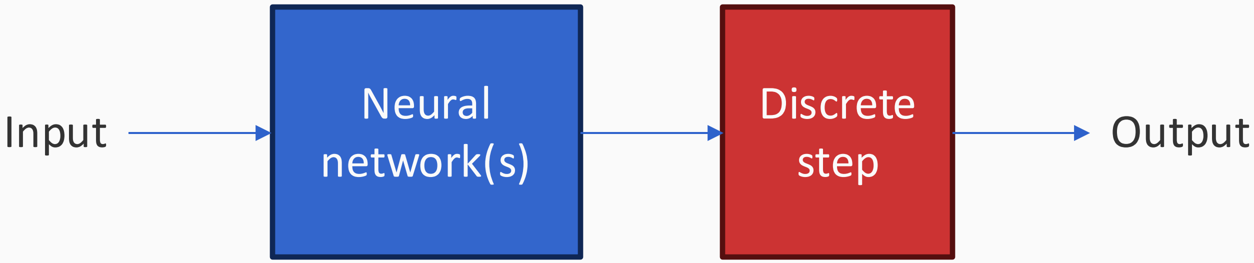
THE UNIVERSITY OF UTAH

Neural networks producing discrete outputs



Importantly: In this case, we have no trainable parameters *after* the discrete step

Neural networks producing discrete outputs



Importantly: In this case, we have no trainable parameters *after* the discrete step

Can we train the network to incorporate feedback from & via the discrete step?

This lecture

- Structural Support Vector Machine
  – How it naturally extends multiclass SVM

- Empirical Risk Minimization
  – Or: how structural SVM and CRF are solving very similar problems

- Training with structured outputs

Where are we?

- Structural Support Vector Machine
  - How it naturally extends multiclass SVM
- Empirical Risk Minimization
  - Or: how structural SVM and CRF are solving very similar problems
- Training with structured outputs

Recall: Binary and Multiclass SVM

- Binary SVM
  - Maximize margin
  - Equivalently,

    Minimize norm of weights such that the closest points to the hyperplane have a score at least 1

- Multiclass SVM
  - Each label has a different weight vector (like one-vs-all)
  - Maximize multiclass margin
  - Equivalently,

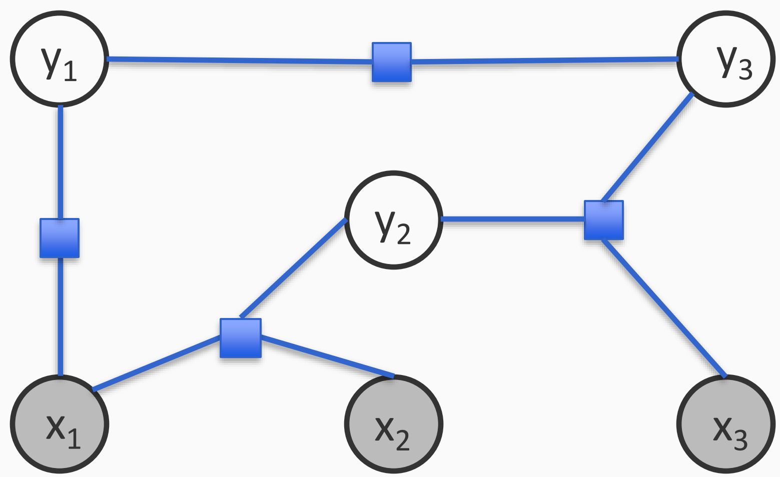
    Minimize total norm of the weights such that the true label is scored at least 1 more than the second best one

Max margin learning: First attempt

Modeling

Suppose we have some definition of a structure (a factor graph)

And scoring functions for each factor (i.e. "part") p as $score(\mathbf{x}, \mathbf{y}_p)$



Max margin learning: First attempt

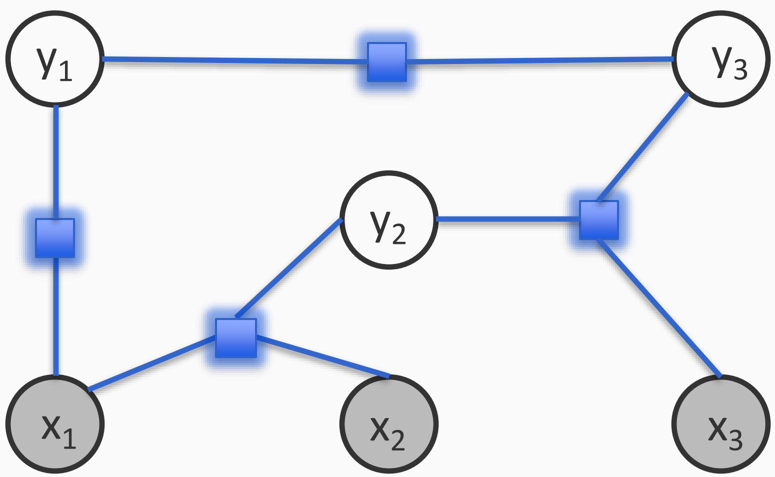
Modeling

Suppose we have some definition of a structure (a factor graph)

And scoring functions for each factor (i.e. "part") p as $score(\mathbf{x}, \mathbf{y}_p)$

Each of these is a neural network, whose architecture we have defined

Our goal is to train them



Max margin learning: First attempt

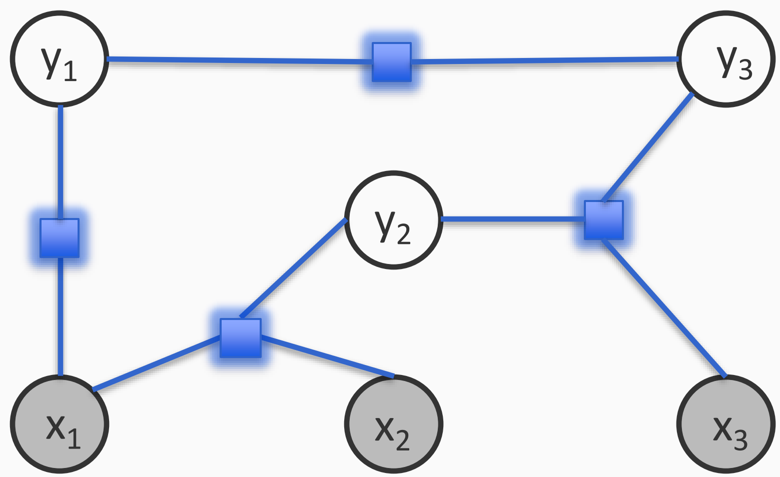
Modeling

Suppose we have some definition of a structure (a factor graph)

And scoring functions for each factor (i.e. "part") p as $score(\mathbf{x}, \mathbf{y}_p)$

Each of these is a neural network, whose architecture we have defined

Our goal is to train them



$$score(\mathbf{x}, \mathbf{y}) = \sum_p score(\mathbf{x}, \mathbf{y}_p)$$

Max margin learning: First attempt

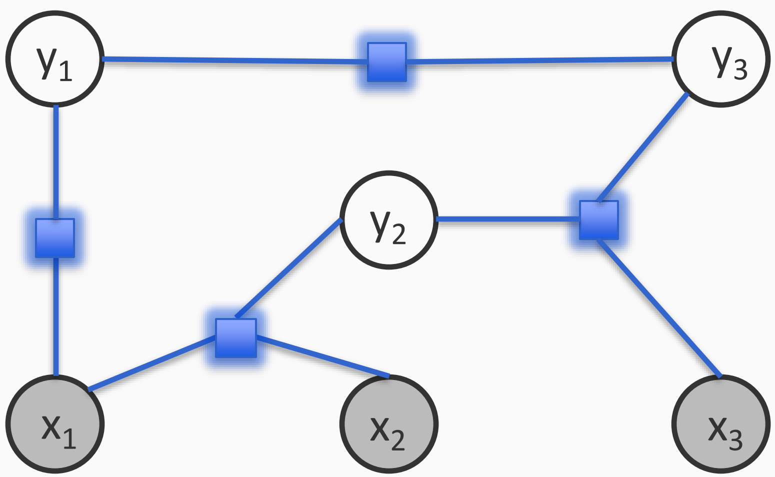
Suppose we have some definition of a structure (a factor graph)

Modeling

And scoring functions for each factor (i.e. "part") p as $score(\mathbf{x}, \mathbf{y}_p)$

Each of these is a neural network, whose architecture we have defined

Our goal is to train them



Data

We also have a data set $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$

Max margin learning: First attempt

Suppose we have some definition of a structure (a factor graph)

Modeling

And scoring functions for each factor (i.e. "part") p as $score(\mathbf{x}, \mathbf{y}_p)$

Data

We also have a data set $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$

What we want from training (following the multiclass idea)

For each training example $(\mathbf{x}_i, \mathbf{y}_i)$:

- The annotated structure $\mathbf{y}_i$ gets the highest score among all structures
- Or to be safe, $\mathbf{y}_i$ gets a score that is at least one more than all other structures

Max margin learning: First attempt

Suppose we have some definition of a structure (a factor graph)

Modeling

And scoring functions for each factor (i.e. "part") p as $score(\mathbf{x}, \mathbf{y}_p)$

Data

We also have a data set $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$

What we want from training (following the multiclass idea)

For each training example $(\mathbf{x}_i, \mathbf{y}_i)$:

- The annotated structure $\mathbf{y}_i$ gets the highest score among all structures
- Or to be safe, $\mathbf{y}_i$ gets a score that is at least one more than all other structures

$$\forall \mathbf{y} \neq \mathbf{y}_i, \qquad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + 1$$

Max margin learning: First attempt

Suppose we have some definition of a structure (a factor graph)

Modeling

And scoring functions for each factor (i.e. “part”) p as $score(\mathbf{x}, \mathbf{y}_p)$

Data

We also have a data set $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$

What we want from training (following the multiclass idea)

For each training example $(\mathbf{x}_i, \mathbf{y}_i)$:

- The annotated structure $\mathbf{y}_i$ gets the highest score among all structures
- Or to be safe, $\mathbf{y}_i$ gets a score that is at least one more than all other structures

$$\forall \mathbf{y} \neq \mathbf{y}_i, \qquad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + 1$$

Max margin learning: First attempt

Suppose we have some definition of a structure (a factor graph)

Modeling

And scoring functions for each factor (i.e. "part") p as $score(\mathbf{x}, \mathbf{y}_p)$

Data

We also have a data set $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$

What we want from training (following the multiclass idea)

For each training example $(\mathbf{x}_i, \mathbf{y}_i)$:

- The annotated structure $\mathbf{y}_i$ gets the highest score among all structures
- Or to be safe, $\mathbf{y}_i$ gets a score that is at least one more than all other structures

$$\forall \mathbf{y} \neq \mathbf{y}_i, \qquad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + 1$$

Max margin learning: First attempt

Suppose we have some definition of a structure (a factor graph)

Modeling

And scoring functions for each factor (i.e. "part") p as $score(\mathbf{x}, \mathbf{y}_p)$

Data

We also have a data set $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$

What we want from training (following the multiclass idea)

For each training example $(\mathbf{x}_i, \mathbf{y}_i)$:

- The annotated structure $\mathbf{y}_i$ gets the highest score among all structures
- Or to be safe, $\mathbf{y}_i$ gets a score that is at least one more than all other structures

$$\forall \mathbf{y} \neq \mathbf{y}_i, \qquad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + 1$$

Max margin learning: First attempt

Maximize margin

$\mathrm{s.t.}$ Score for gold structure $\geq$ Score for other structure $+ 1$ For every training example

Max margin learning: First attempt

Maximize margin

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + 1 \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y} \neq \mathbf{y}_i$$

Score for gold

Score for other

Input with gold structure

Some other structure

Max margin learning: First attempt

Maximize margin, e.g. by minimizing norm of **w**

$$\min_{w} Regularizer(w)$$

$$\text{s.t.}\ \ score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + 1 \ \ \forall(\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y} \neq \mathbf{y}_i$$

Score for gold

Score for other

Input with gold structure

Some other structure

Max margin learning: First attempt

Maximize margin, e.g. by minimizing norm of **w**

$$\min_w Regularizer(w)$$

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + 1 \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y} \neq \mathbf{y}_i$$

Score for gold

Score for other

Input with gold structure

Some other structure

Problem?

Max margin learning: First attempt

Maximize margin, e.g. by minimizing norm of **w**

$$\min_{w} Regularizer(w)$$

$$\text{s.t.} \;\; score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + 1 \;\; \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y} \neq \mathbf{y}_i$$

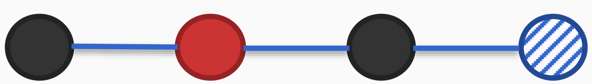
Score for gold

Score for other

Input with gold structure

Some other structure

Problem



Gold structure

Max margin learning: First attempt

Maximize margin, e.g. by minimizing norm of **w**

$$\min_{w} Regularizer(w)$$

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + 1 \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y} \neq \mathbf{y}_i$$

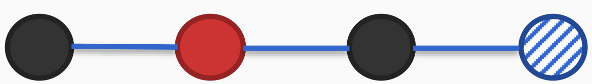
Score for gold

Score for other

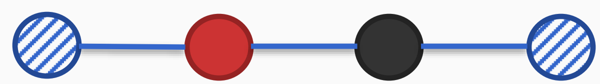
Input with gold structure

Some other structure

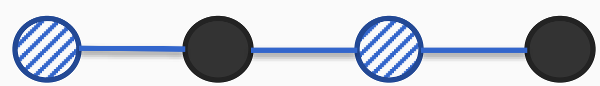
Problem



Gold structure



Other structure A: Only one mistake



Other structure B: Fully incorrect

Max margin learning: First attempt

Maximize margin, e.g. by minimizing norm of **w**

$$\min_{w} Regularizer(w)$$

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + 1 \quad \forall(\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y} \neq \mathbf{y}_i$$

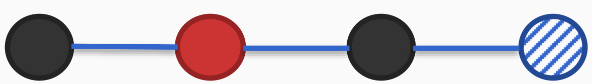
Score for gold

Score for other

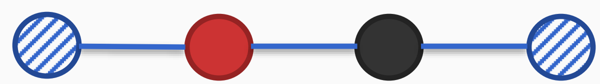
Input with gold structure

Some other structure

Problem



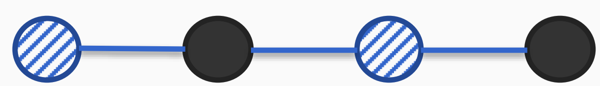
Gold structure



Other structure A: Only one mistake

Structure B has is more wrong, but this formulation will be happy if *both* A & B are scored one less than gold!

No partial credit!



Other structure B: Fully incorrect

Max margin learning: First attempt

Maximize margin, e.g. by minimizing norm of **w**

$$\min_{w} Regularizer(w)$$

$$\text{s.t.}\quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + 1 \quad \forall(\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y} \neq \mathbf{y}_i$$

Score for gold

Score for other

Input with gold structure

Some other structure

$$\min_{w} Regularizer(w)$$

$$\text{s.t.}\quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall(\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Max margin learning: First attempt

$$\min_{w} Regularizer(w)$$

Maximize margin, e.g. by minimizing norm of **w**

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + 1 \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y} \neq \mathbf{y}_i$$

Score for gold

Score for other

Input with gold structure

Some other structure

$$\min_{w} Regularizer(w)$$

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Hamming distance between structures: Counts the number of differences between them

Max margin learning: First attempt

Maximize margin, e.g. by minimizing norm of **w**

$$\min_{w} Regularizer(w)$$

$$\text{s.t.}\quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + 1 \quad \forall(\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y} \neq \mathbf{y}_i$$

Score for gold

Score for other

Input with gold structure

Some other structure

$$\min_{w} Regularizer(w)$$

$$\text{s.t.}\quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall(\mathbf{x}_i, \mathbf{y}_i) \in D, \boxed{\forall \mathbf{y}}$$

because the Hamming distance of $\mathbf{y}$ and itself is zero

Max margin learning: Second attempt

$$\min_{w} Regularizer(w)$$

$$\text{s.t.}\ \ score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Intuition

- It is okay for a structure that is close (in Hamming sense) to the true one to get a score that is close to the true structure
- Structures that are very different from the true structure should get much lower scores

Max margin learning: Second attempt

$$\min_{w} Regularizer(w)$$

Maximize margin, e.g. by minimizing norm of **w**

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Intuition

- It is okay for a structure that is close (in Hamming sense) to the true one to get a score that is close to the true structure
- Structures that are very different from the true structure should get much lower scores

Max margin learning: Second attempt

$$\min_{w} Regularizer(w)$$

Maximize margin, e.g. by minimizing norm of **w**

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Input with gold structure

Intuition

- It is okay for a structure that is close (in Hamming sense) to the true one to get a score that is close to the true structure
- Structures that are very different from the true structure should get much lower scores

Max margin learning: Second attempt

$$\min_{w} Regularizer(w)$$

Maximize margin, e.g. by minimizing norm of **w**

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Score for gold

Input with gold structure

Intuition

- It is okay for a structure that is close (in Hamming sense) to the true one to get a score that is close to the true structure
- Structures that are very different from the true structure should get much lower scores

Max margin learning: Second attempt

$$\min_{w} Regularizer(w)$$

Maximize margin, e.g. by minimizing norm of **w**

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Score for gold

Score for other

Input with gold structure

Intuition

- It is okay for a structure that is close (in Hamming sense) to the true one to get a score that is close to the true structure
- Structures that are very different from the true structure should get much lower scores

Max margin learning: Second attempt

$$\min_{w} Regularizer(w)$$

Maximize margin, e.g. by minimizing norm of $\mathbf{w}$

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Score for gold

Score for other

Input with gold structure

Hamming distance between structures. Defined to be zero if $\mathbf{y} = \mathbf{y}_i$

Intuition

- It is okay for a structure that is close (in Hamming sense) to the true one to get a score that is close to the true structure
- Structures that are very different from the true structure should get much lower scores

Max margin learning: Second attempt

$$\min_{w} Regularizer(w)$$

Maximize margin, e.g. by minimizing norm of $\mathbf{w}$

$$\text{s.t.}\quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Score for gold

Score for other

Input with gold structure

Another structure, could be $\mathbf{y}_i$

Hamming distance between structures. Defined to be zero if $\mathbf{y} = \mathbf{y}_i$

Intuition
- It is okay for a structure that is close (in Hamming sense) to the true one to get a score that is close to the true structure
- Structures that are very different from the true structure should get much lower scores

Max margin learning: Second attempt

$$\min_{w} Regularizer(w)$$

Maximize margin, e.g. by minimizing norm of **w**

$$\text{s.t.}\quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Score for gold

Score for other

Hamming distance between structures. Defined to be zero if $\mathbf{y} = \mathbf{y}_i$

Input with gold structure

Another structure, could be $\mathbf{y}_i$

Problem?

Max margin learning: Second attempt

$$\min_{w} Regularizer(w)$$

Maximize margin, e.g. by minimizing norm of **w**

$$\text{s.t.}\quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Score for gold

Score for other

Input with gold structure

Another structure, could be $\mathbf{y}_i$

Hamming distance between structures. Defined to be zero if $\mathbf{y} = \mathbf{y}_i$

Problem?

What if these constraints are not satisfied for any parameters for a given dataset?

Max margin learning: Third attempt

$$\min_{w} Regularizer(w)$$

Maximize margin, e.g. by minimizing norm of **w**

$$\text{s.t.}\quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Score for gold

Score for other

Hamming distance between structures. Defined to be zero if $\mathbf{y} = \mathbf{y}_i$

Input with gold structure

Another structure, could be $\mathbf{y}_i$

$$\min_{w} Regularizer(w)$$

$$\text{s.t.}\quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Max margin learning: Third attempt

$$\min_w Regularizer(w)$$

Maximize margin, e.g. by minimizing norm of **w**

$$\text{s.t.}\quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Score for gold

Score for other

Input with gold structure

Another structure, could be $\mathbf{y}_i$

Hamming distance between structures. Defined to be zero if $\mathbf{y} = \mathbf{y}_i$

$$\min_w Regularizer(w)$$

$$\text{s.t.}\quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Slack variable for each example, **must be positive**

Max margin learning: Third attempt

$$\min_{w} Regularizer(w)$$

Maximize margin, e.g. by minimizing norm of **w**

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Score for gold

Score for other

Input with gold structure

Another structure, could be $\mathbf{y}_i$

Hamming distance between structures. Defined to be zero if $\mathbf{y} = \mathbf{y}_i$

$$\min_{w,\xi} Regularizer(w) \boxed{+ C \sum \xi_i}$$

Also minimize total slack

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Slack variable for each example, **must be positive**

Max margin learning: Third attempt

$$\min_{w,\xi} Regularizer(w) + C\sum \xi_i$$

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Max margin learning: Third attempt

$$\min_{w,\xi} Regularizer(w) + C\sum \xi_i$$

$$\text{s.t.}\ \ score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Input with gold structure

Another structure

For every labeled example, and every competing structure

Max margin learning: Third attempt

$$\min_{w,\xi} Regularizer(w) + C\sum \xi_i$$

$$\text{s.t.} \;\; score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Score for gold

Score for other

Hamming distance

Input with gold structure

Another structure

For every labeled example, and every competing structure,
the score for the ground truth should be greater than the score for
the competing structure by the Hamming distance between them

Max margin learning: Third attempt

$$\min_{w,\xi} Regularizer(w) + C\sum \xi_i$$

$$\text{s.t.}\ \ score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i \quad \forall(\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

Score for gold

Score for other

Hamming distance

Slack variable for each example

Input with gold structure

Another structure

Slack variables allow some examples to be misclassified.

Max margin learning: Third attempt

$$\min_{w,\xi} Regularizer(w) + C\sum \xi_i$$

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

$$\forall i, \xi_i \geq 0$$

Score for gold

Score for other

Hamming distance

Input with gold structure

Another structure

All slacks must be positive

Slack variable for each example

Slack variables allow some examples to be misclassified.

Max margin learning: Third attempt

Improve generalization & minimize slack C: the tradeoff parameter

$$\min_{w,\xi} Regularizer(w) + C\sum \xi_i$$

$$\text{s.t. } score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

$$\forall i, \xi_i \geq 0$$

Score for gold

Score for other

Hamming distance

Input with gold structure

Another structure

All slacks must be positive

Slack variable for each example

Slack variables allow some examples to be misclassified.

Minimizing the slack forces this to happen as few times as possible

Max margin learning: Third attempt

Improve generalization & minimize slack C: the tradeoff parameter

$$\min_{w,\xi} Regularizer(w) + C\sum\xi_i$$

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i \quad \forall(\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

$$\forall i, \xi_i \geq 0$$

Score for gold

Score for other

Hamming distance

Input with gold structure

Another structure

All slacks must be positive

Slack variable for each example

Slack variables allow some examples to be misclassified.

Minimizing the slack forces this to happen as few times as possible

Max margin learning: Third attempt

Improve generalization & minimize slack C: the tradeoff parameter

$$\min_{w,\xi} Regularizer(w) + C\sum\xi_i$$

$$\text{s.t.}\ \ score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i \quad \forall(\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

$$\forall i, \xi_i \geq 0$$

Score for gold

Score for other

Hamming distance

Input with gold structure

Another structure

All slacks must be positive

Slack variable for each example

Slack variables allow some examples to be misclassified.

Minimizing the slack forces this to happen as few times as possible

Questions?

Max margin learning (a.k.a structural SVM)

Improve generalization & minimize slack C: the tradeoff parameter

$$\min_{w,\xi} Regularizer(w) + C\sum \xi_i$$

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

$$\forall i, \xi_i \geq 0$$

Score for gold

Score for other

All slacks must be positive

Hamming distance

Slack variable for each example

Input with gold structure

Another structure

Max margin learning (a.k.a structural SVM)

Improve generalization & minimize slack C: the tradeoff parameter

$$\min_{w,\xi} Regularizer(w) + C\sum \xi_i$$

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i \quad \forall(\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

$$\forall i, \xi_i \geq 0$$

Score for gold

Score for other

All slacks must be positive

Hamming distance

Slack variable for each example

Input with gold structure

Another structure

Equivalent formulation

$$\min_{w} Regularizer(w) + C\sum_i \max_{\mathbf{y}}\left(score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score(\mathbf{x}_i, \mathbf{y}_i)\right)$$

Max margin learning (a.k.a structural SVM)

Improve generalization & minimize slack C: the tradeoff parameter

$$\min_{w,\xi} Regularizer(w) + C \sum \xi_i$$

$$\text{s.t.} \quad score(\mathbf{x}_i, \mathbf{y}_i) \geq score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in D, \forall \mathbf{y}$$

$$\forall i, \xi_i \geq 0$$

Score for gold

Score for other

All slacks must be positive

Hamming distance

Slack variable for each example

Input with gold structure

Another structure

Equivalent formulation

$$\min_{w} Regularizer(w) + C \sum_i \max_{\mathbf{y}} \left(score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score(\mathbf{x}_i, \mathbf{y}_i) \right)$$

Questions?

Max margin learning (a.k.a structural SVM)

$$\min_w Regularizer(w) + C\sum_i \max_{\mathbf{y}}\left(score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score(\mathbf{x}_i, \mathbf{y}_i)\right)$$

Max margin learning (a.k.a structural SVM)

$$\min_{w} Regularizer(w) + C \sum_{i} \max_{\mathbf{y}} \left(score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score(\mathbf{x}_i, \mathbf{y}_i) \right)$$

Score of the ground truth

Max margin learning (a.k.a structural SVM)

$$\min_{w} Regularizer(w) + C \sum_{i} \max_{\mathbf{y}} \left(\underbrace{score(\mathbf{x}_i, \mathbf{y})}_{\text{Score of the structure } \mathbf{y}} + \Delta(\mathbf{y}, \mathbf{y}_i) - \underbrace{score(\mathbf{x}_i, \mathbf{y}_i)}_{\text{Score of the ground truth}} \right)$$

Max margin learning (a.k.a structural SVM)

$$\min_w Regularizer(w) + C \sum_i \max_{\mathbf{y}} \left(score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score(\mathbf{x}_i, \mathbf{y}_i) \right)$$

Score of the structure $\mathbf{y}$

Score of the ground truth

The Hamming distance between the two sets of decisions

Max margin learning (a.k.a structural SVM)

$$\min_{w} Regularizer(w) + C \sum_{i} \max_{\mathbf{y}} \Big(\underbrace{score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score(\mathbf{x}_i, \mathbf{y}_i)} \Big)$$

The additional score assigned to the structure $\mathbf{y}$ over the ground truth

Max margin learning (a.k.a structural SVM)

$$\min_{w} Regularizer(w) + C \sum_{i} \max_{\mathbf{y}} \left(score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score(\mathbf{x}_i, \mathbf{y}_i) \right)$$

Gives the other structure additional points in this optimization if it is really different from the ground truth

Max margin learning (a.k.a structural SVM)

$$\min_{w} Regularizer(w) + C \sum_{i} \max_{\mathbf{y}}\left(score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score(\mathbf{x}_i, \mathbf{y}_i)\right)$$

Find the structure that has the highest augmented score. This is a bad structure whose score needs to be minimized

Max margin learning (a.k.a structural SVM)

$$\min_{w} Regularizer(w) + C \sum_{i} \max_{\mathbf{y}}\left(score(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score(\mathbf{x}_i, \mathbf{y}_i)\right)$$

Find the structure that has the highest augmented score. This is a bad structure whose score needs to be minimized

Questions?

Comments

- Other slightly different formulations exist
  - Generally same principle
- Multiclass is a special case of structure
  - Structural SVM strictly generalizes multiclass SVM

Exercise: Work it out

- Can be seen as minimizing structured version of hinge loss
  - Remember **empirical risk minimization**?
- Learning as optimization
  - We have framed the optimization problem
    - That is, we don't have a learning algorithm yet

This lecture

- Structural Support Vector Machine
  – How it naturally extends multiclass SVM

- Empirical Risk Minimization
  – Or: how structural SVM and CRF are solving very similar problems

- Training with structured outputs

Broader picture: Learning as loss minimization

- Collect some annotated data. More is generally better
- Pick a hypothesis class (also called model)
  - Decide how the score decomposes over the parts of the output
- Choose a loss function
  - Decide on how to penalize incorrect decisions
- Learning = minimize empirical risk + regularizer
  - Typically an optimization procedure needed here

This must look familiar. We have seen this before for binary classification!

Structured classifiers: Different learning objectives

- Structural SVM

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathrm{C}\sum_i \max_{\mathbf{y}}\left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i)\right)$$

Structured classifiers: Different learning objectives

- Structural SVM

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathrm{C}\sum_i \max_{\mathbf{y}}\left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i)\right)$$

- Conditional Random Field (via the maximum a posteriori criterion)

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathrm{C}\sum_i -\log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})$$

Structured classifiers: Different learning objectives

- Structural SVM

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathrm{C}\sum_i \max_{\mathbf{y}}\left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i)\right)$$

- Conditional Random Field (via the maximum a posteriori criterion)

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathrm{C}\sum_i -\log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})$$

Where P is defined as

$$P(\,\mathbf{y}_\mathrm{i} \mid \mathbf{x}_i, \mathbf{w}\,) = \frac{\exp(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i))}{Z(\mathbf{x}_i, \mathbf{w})}$$

Structured classifiers: Different learning objectives

- Structural SVM

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max_{\mathbf{y}}\left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i)\right)$$

Regularizer

- Conditional Random Field (via the maximum a posteriori criterion)

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i -\log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})$$

Where P is defined as

$$P(\mathbf{y}_\mathrm{i} \mid \mathbf{x}_i, \mathbf{w}) = \frac{\exp(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i))}{Z(\mathbf{x}_i, \mathbf{w})}$$

Structured classifiers: Different learning objectives

- Structural SVM

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max_{\mathbf{y}}\left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i)\right)$$

Regularizer

How badly does **w** do on the training data

- Conditional Random Field (via the maximum a posteriori criterion)

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i -\log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})$$

Where P is defined as

$$P(\mathbf{y}_\mathrm{i} \mid \mathbf{x}_i, \mathbf{w}) = \frac{\exp(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i))}{Z(\mathbf{x}_i, \mathbf{w})}$$

Structured classifiers: Different learning objectives

- Structural SVM

Structured hinge loss

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathrm{C}\sum_i \boxed{\max_{\mathbf{y}}\left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i)\right)}$$

Regularizer

How badly does **w** do on the training data

- Conditional Random Field (via the maximum a posteriori criterion)

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathrm{C}\sum_i -\log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})$$

Where P is defined as

$$P(\mathbf{y}_\mathrm{i} \mid \mathbf{x}_i, \mathbf{w}) = \frac{\exp(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i))}{Z(\mathbf{x}_i, \mathbf{w})}$$

Structured classifiers: Different learning objectives

- Structural SVM

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max_{\mathbf{y}}\left(score_{\mathbf{w}}(\mathbf{x}_i,\mathbf{y}) + \Delta(\mathbf{y},\mathbf{y}_i) - score_{\mathbf{w}}(\mathbf{x}_i,\mathbf{y}_i)\right)$$

Regularizer

How badly does **w** do on the training data

- Conditional Random Field (via the maximum a posteriori criterion)

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \boxed{-\log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})}$$

Log loss

Where P is defined as

$$P(\mathbf{y}_\mathrm{i} \mid \mathbf{x}_i, \mathbf{w}) = \frac{\exp(score_{\mathbf{w}}(\mathbf{x}_i,\mathbf{y}_i))}{Z(\mathbf{x}_i,\mathbf{w})}$$

Structured classifiers: Different learning objectives

- Structural SVM

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathrm{C}\sum_i \max_{\mathbf{y}}\left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i)\right)$$

- Conditional Random Field (via the maximum a posteriori criterion)

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathrm{C}\sum_i -\log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})$$

- Structured Perceptron

$$\min_{\mathbf{w}} \sum_i \max_{\mathbf{y}}\left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i)\right)$$

Structured classifiers: Different learning objectives

- Structural SVM

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathrm{C}\sum_i \max_{\mathbf{y}}\left(score_{\mathbf{w}}(\mathbf{x}_i,\mathbf{y}) + \Delta(\mathbf{y},\mathbf{y}_i) - score_{\mathbf{w}}(\mathbf{x}_i,\mathbf{y}_i)\right)$$

- Conditional Random Field (via the maximum a posteriori criterion)

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathrm{C}\sum_i -\log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})$$

How badly does **w** do on the training data

- Structured Perceptron

Structured Perceptron loss

$$\min_{\mathbf{w}} \sum_i \boxed{\max_{\mathbf{y}}\left(score_{\mathbf{w}}(\mathbf{x}_i,\mathbf{y}) - score_{\mathbf{w}}(\mathbf{x}_i,\mathbf{y}_i)\right)}$$

This lecture

- Structural Support Vector Machine
  - How it naturally extends multiclass SVM

- Empirical Risk Minimization
  - Or: how structural SVM and CRF are solving very similar problems

- Training with structured outputs

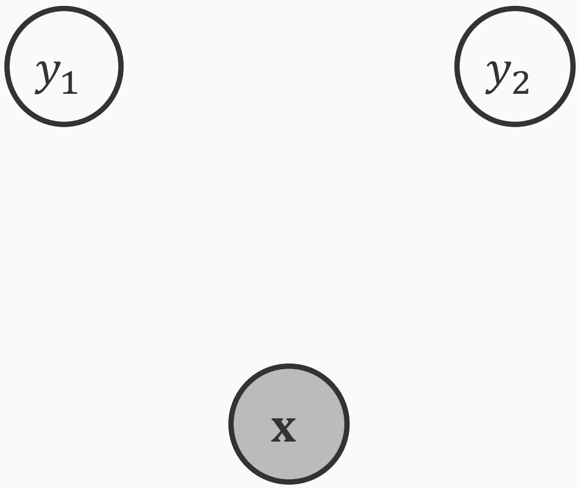
How do we learn in these settings?

Short answer: gradient based optimization

But how do we compute gradients when there is inference in the mix?

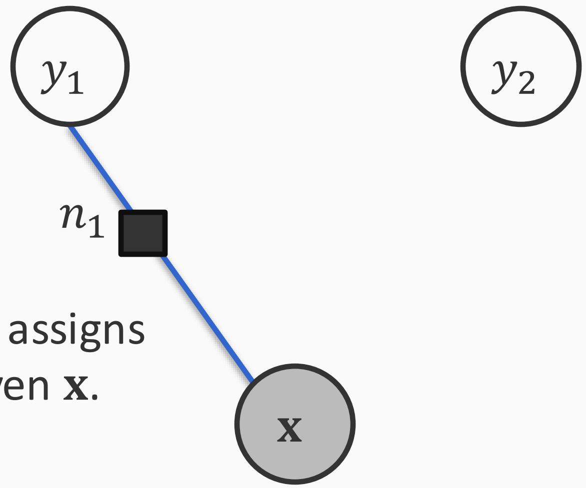
An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.



An example

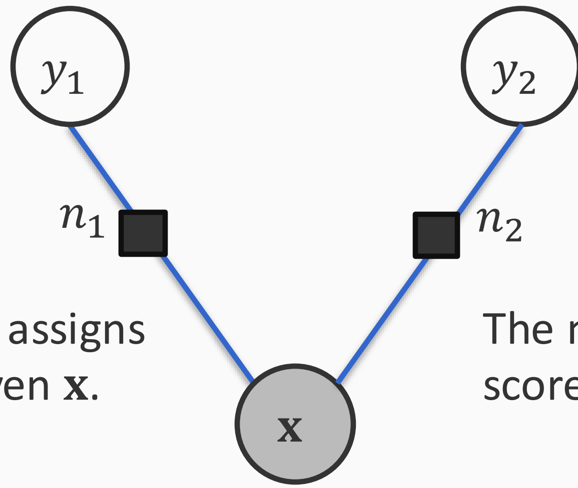
We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.



The network n_1 assigns scores for y_1 given $\mathbf{x}$.

An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.



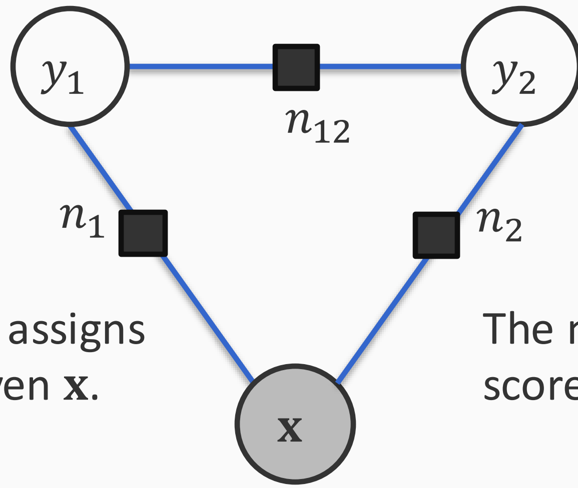
The network n_1 assigns scores for y_1 given $\mathbf{x}$.

The network n_2 assigns scores for y_2 given $\mathbf{x}$.

An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.

The network n_{12} assigns scores for y_1 and y_2 coexisting.



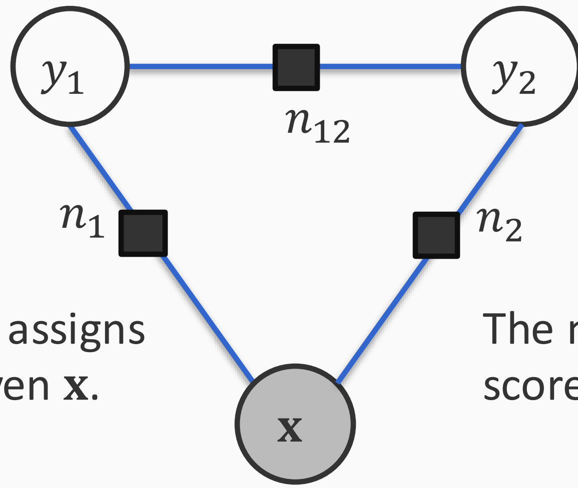
The network n_1 assigns scores for y_1 given $\mathbf{x}$.

The network n_2 assigns scores for y_2 given $\mathbf{x}$.

An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.

The network n_{12} assigns scores for y_1 and y_2 coexisting.



The network n_1 assigns scores for y_1 given $\mathbf{x}$.

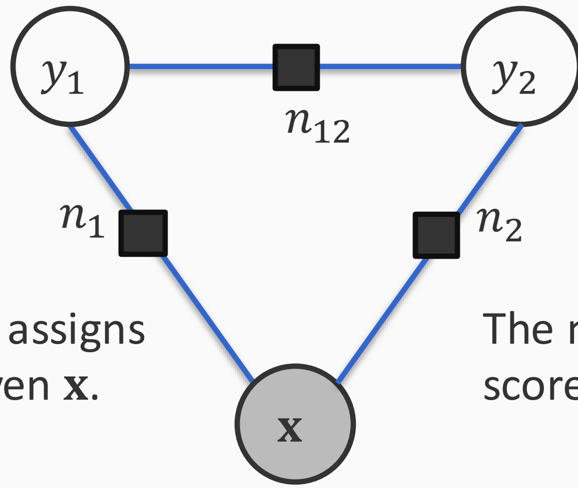
The network n_2 assigns scores for y_2 given $\mathbf{x}$.

Each network may have its own parameters

An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.

The network n_{12} assigns scores for y_1 and y_2 coexisting.



The network n_1 assigns scores for y_1 given $\mathbf{x}$.

The network n_2 assigns scores for y_2 given $\mathbf{x}$.

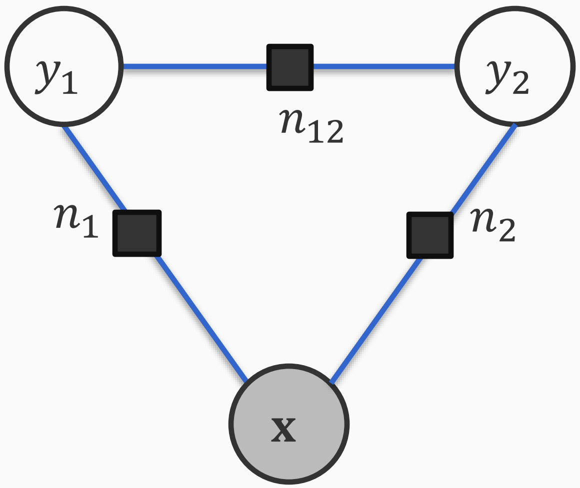
Each network may have its own parameters

Define $score_w(x, y_1, y_2) = n_1(x, y_1) + n_2(x, y_2) + n_{12}(y_1, y_2)$

An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.

Define $score_w(x, y_1, y_2) = n_1(x, y_1) + n_2(x, y_2) + n_{12}(y_1, y_2)$

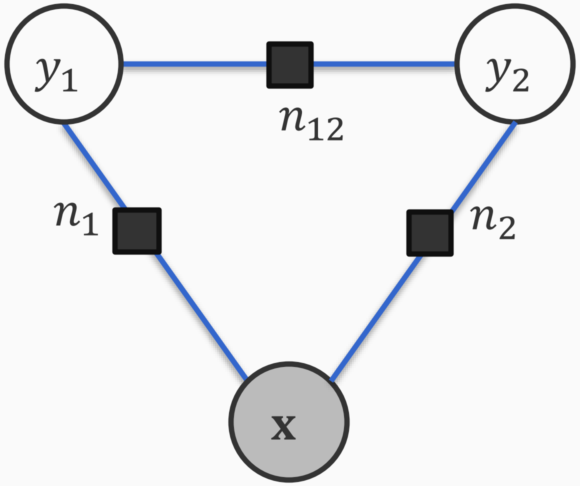


An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.

Define $score_w(x, y_1, y_2) = n_1(x, y_1) + n_2(x, y_2) + n_{12}(y_1, y_2)$

y_1	$n_1(x, y_1)$
A	-10
B	41
C	3

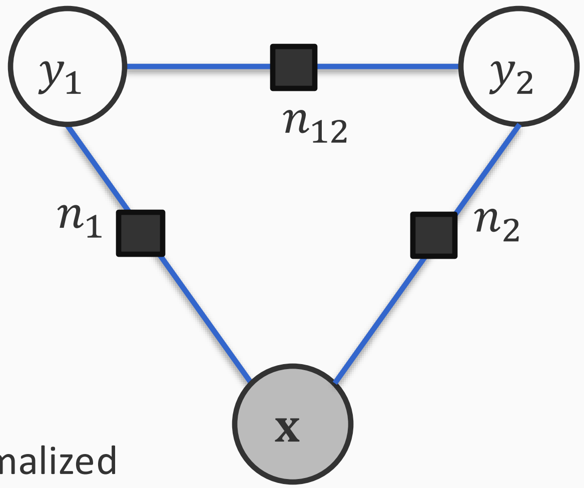


An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.

Define $score_w(x, y_1, y_2) = n_1(x, y_1) + n_2(x, y_2) + n_{12}(y_1, y_2)$

y_1	$n_1(x, y_1)$
A	-10
B	41
C	3



These are unnormalized probabilities.

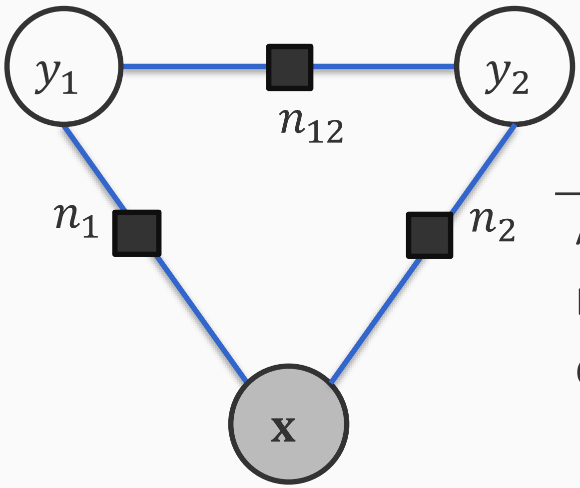
Clearly the network n_1 prefers the label B for y_2

An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.

Define $score_w(x, y_1, y_2) = n_1(x, y_1) + n_2(x, y_2) + n_{12}(y_1, y_2)$

y_1	$n_1(x, y_1)$
A	-10
B	41
C	3



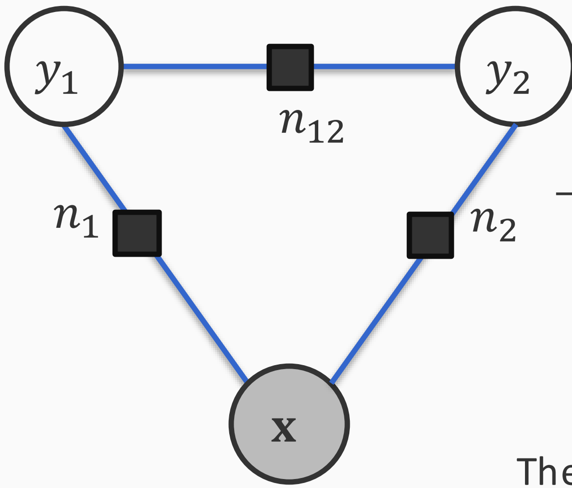
y_2	$n_2(x, y_2)$
A	-4
B	10
C	-11

An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.

Define $score_w(x, y_1, y_2) = n_1(x, y_1) + n_2(x, y_2) + n_{12}(y_1, y_2)$

y_1	$n_1(x, y_1)$
A	-10
B	41
C	3



y_2	$n_2(x, y_2)$
A	-4
B	10
C	-11

These are unnormalized probabilities.

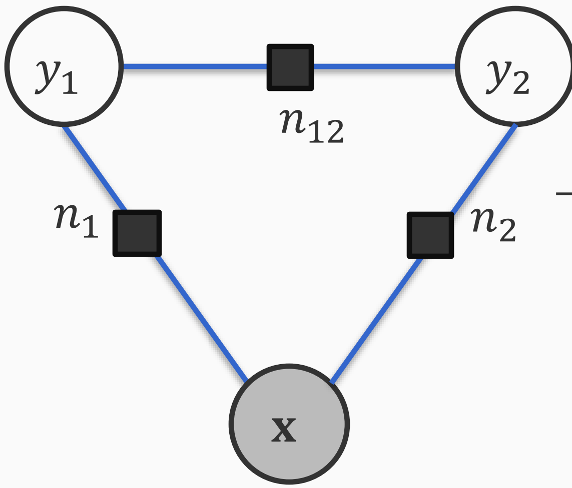
Clearly the network n_2 prefers the label B for y_2

An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.

Define $score_w(x, y_1, y_2) = n_1(x, y_1) + n_2(x, y_2) + n_{12}(y_1, y_2)$

y_1	$n_1(x, y_1)$
A	-10
B	41
C	3



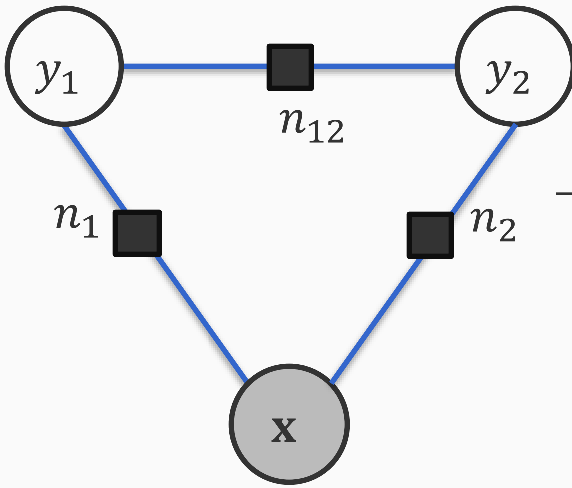
y_2	$n_2(x, y_2)$
A	-4
B	10
C	-11

$n_{12}(y_1, y_2)$	A	B	C
A	-100	0	0
B	0	-100	0
C	0	0	-100

An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.

Define $score_w(x, y_1, y_2) = n_1(x, y_1) + n_2(x, y_2) + n_{12}(y_1, y_2)$



y_1	$n_1(x, y_1)$
A	-10
B	41
C	3

y_2	$n_2(x, y_2)$
A	-4
B	10
C	-11

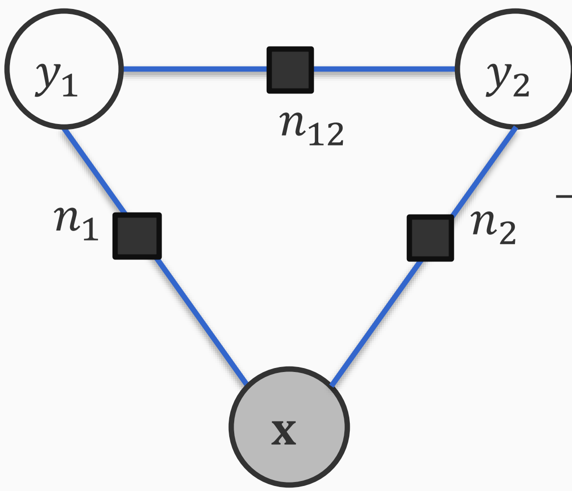
$n_{12}(y_1, y_2)$	A	B	C
A	-100	0	0
B	0	-100	0
C	0	0	-100

The network n_{12} strongly disprefers the two labels from being the same

An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.

Define $score_w(x, y_1, y_2) = n_1(x, y_1) + n_2(x, y_2) + n_{12}(y_1, y_2)$



y_1	$n_1(x, y_1)$
A	-10
B	41
C	3

y_2	$n_2(x, y_2)$
A	-4
B	10
C	-11

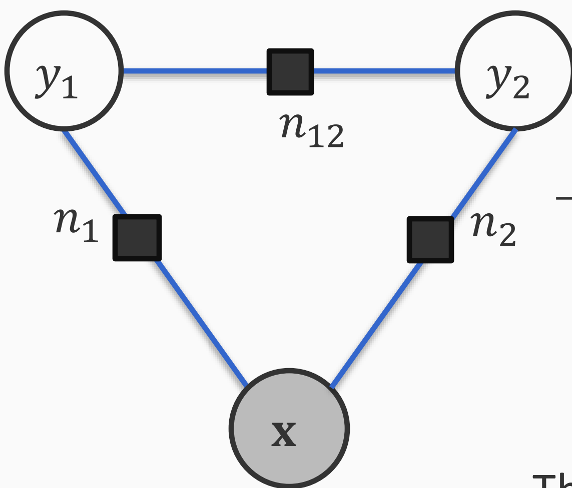
$n_{12}(y_1, y_2)$	A	B	C
A	-100	0	0
B	0	-100	0
C	0	0	-100

y_1	y_2	$score_{\mathbf{w}}$
A	A	-114
A	B	0
A	C	-21
B	A	37
B	B	-49
B	C	30
C	A	-1
C	B	13
C	C	-108

An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.

Define $score_w(x, y_1, y_2) = n_1(x, y_1) + n_2(x, y_2) + n_{12}(y_1, y_2)$



y_1	$n_1(x, y_1)$
A	-10
B	41
C	3

y_2	$n_2(x, y_2)$
A	-4
B	10
C	-11

$n_{12}(y_1, y_2)$	A	B	C
A	-100	0	0
B	0	-100	0
C	0	0	-100

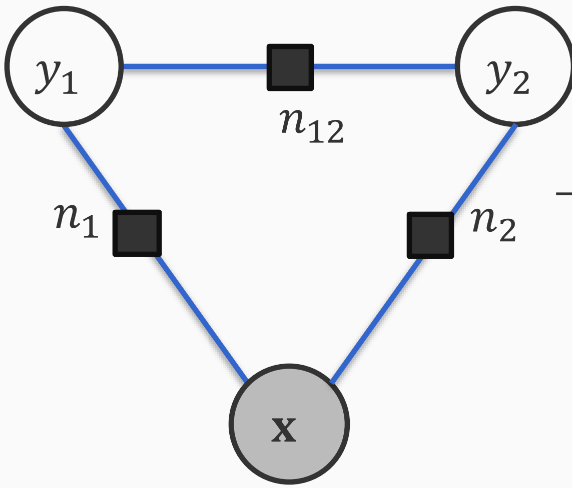
y_1	y_2	$score_{\mathbf{w}}$
A	A	-114
A	B	0
A	C	-21
B	A	37
B	B	-49
B	C	30
C	A	-1
C	B	13
C	C	-108

The highest scoring consensus decision

An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.

Define $score_w(x, y_1, y_2) = n_1(x, y_1) + n_2(x, y_2) + n_{12}(y_1, y_2)$



y_1	$n_1(x, y_1)$
A	-10
B	41
C	3

y_2	$n_2(x, y_2)$
A	-4
B	10
C	-11

$n_{12}(y_1, y_2)$	A	B	C
A	-100	0	0
B	0	-100	0
C	0	0	-100

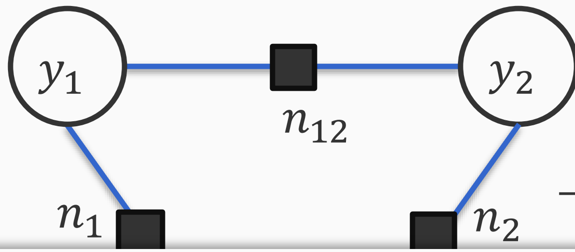
y_1	y_2	$score_{\mathbf{w}}$
A	A	-114
A	B	0
A	C	-21
B	A	37
B	B	-49
B	C	30
C	A	-1
C	B	13
C	C	-108

Suppose this is the true label

An example

We have an input x and need to predict two labels y_1 and y_2.
Suppose each label can be one of {A, B, C}.

Define $score_w(x, y_1, y_2) = n_1(x, y_1) + n_2(x, y_2) + n_{12}(y_1, y_2)$



y_1	$n_1(x, y_1)$
A	-10
B	
C	

y_2	$n_2(x, y_2)$
A	-4

A	-100	0	0
B	0	-100	0
C	0	0	-100

y_1	y_2	$score_{\mathbf{w}}$
A	A	-114
A	B	0
A	C	-21
B	A	37
B	B	-49
B	C	30
C	A	-1
C	B	13
C	C	-108

Suppose this is the true label

The goal of learning
To update the underlying scoring functions so that the score of the true assignment increases and the score of the prediction goes down

How do we learn in these settings?

Short answer: gradient based optimization

But how do we compute gradients when there is inference in the mix?

Consider, e.g. the structured perceptron objective:

$$\min_{\mathbf{w}} \sum_i \max_{\mathbf{y}} \left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i) \right)$$

How do we learn in these settings?

Short answer: gradient based optimization

But how do we compute gradients when there is inference in the mix?

Consider, e.g. the structured perceptron objective:

$$\min_{\mathbf{w}} \sum_i \max_{\mathbf{y}} \left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i) \right)$$

These are both the same neural network, with different inputs

How do we learn in these settings?

Short answer: gradient based optimization

But how do we compute gradients when there is inference in the mix?

Consider, e.g. the structured perceptron objective:

$$\min_{\mathbf{w}} \sum_i \max_{\mathbf{y}} \left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i) \right)$$

The true labeled structure (a collection of decisions), whose score we want to maximize

How do we learn in these settings?

Short answer: gradient based optimization

But how do we compute gradients when there is inference in the mix?

Consider, e.g. the structured perceptron objective:

$$\min_{\mathbf{w}} \sum_i \max_{\mathbf{y}} \left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i) \right)$$

A competing label assignment, whose score we want to minimize if it is not the same as $\mathbf{y}_i$

How do we learn in these settings?

Short answer: gradient based optimization

But how do we compute gradients when there is inference in the mix?

Consider, e.g. the structured perceptron objective:

$$\min_{\mathbf{w}} \sum_i \max_{\mathbf{y}} \left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i) \right)$$

If the highest scoring assignment is not the same as the ground truth, the value of the loss will be non-zero

A competing label assignment, whose score we want to minimize if it is not the same as $\mathbf{y}_i$

Structured Perceptron algorithm

Given a training set $D = \{(x_i, y_i)\}$

Initialize the model parameters $\mathbf{w}$

1. For epoch = 1 … T:
    1. Shuffle data
    2. For each training example $(\mathbf{x}_i, \mathbf{y}_i) \in D$:
        1. Compute the gradient of the structured perceptron loss
        2. Take a gradient step

2. Return **w**

Structured Perceptron algorithm

Use any optimizer and all the standard optimization tricks here

- Initialization strategies
- Mini-batches instead of single examples
- Your favorite optimizer (e.g. Adam), learning rates, choices of the number of epochs, etc, dropout

Given a training set $D = \{(x_i, y_i)\}$

Initialize the model parameters $\mathbf{w}$

1. For epoch = 1 … T:
   1. Shuffle data
   2. For each training example $(\mathbf{x}_i, \mathbf{y}_i) \in D$:
      1. Compute the gradient of the structured perceptron loss
      2. Take a gradient step
2. Return **w**

Structured Perceptron algorithm

Given a training set $D = \{(x_i, y_i)\}$

Initialize the model parameters $\mathbf{w}$

1. For epoch = 1 … T:
    1. Shuffle data
    2. For each training example $(\mathbf{x}_i, \mathbf{y}_i) \in D$:
        1. Compute the gradient of the structured perceptron loss
        2. Take a gradient step
2. Return **w**

Let us focus on the one step that is different

How do we compute the loss of the objective?

The structured perceptron loss

$$l(x_i, y_i, w) = \max_{\mathbf{y}}\left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i)\right)$$

This step searches over all possible discrete assignments to the output labels. How do we compute the loss?

The structured perceptron loss

$$l(x_i, y_i, w) = \max_{\mathbf{y}} \left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i) \right)$$

This step searches over all possible discrete assignments to the output labels. How do we compute the loss?

Answer: Subgradients

The structured perceptron loss

$$l(x_i, y_i, w) = \max_{\mathbf{y}}\left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i)\right)$$

This step searches over all possible discrete assignments to the output labels. How do we compute the loss?

Answer: Subgradients

Subgradient of a $\max$ …: first solve the maximization and then compute gradient of the argmax

Structured Perceptron algorithm

Given a training set $D = \{(x_i, y_i)\}$

Initialize the model parameters $\mathbf{w}$

1. For epoch = 1 … T:
    1. Shuffle data
    2. For each training example $(\mathbf{x}_i, \mathbf{y}_i) \in D$:
        1. Let $\mathbf{y}' = \max_{\mathbf{y}} score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y})$
        2. If $\mathbf{y}' \neq \mathbf{y}_i$

           Update $\mathbf{w} \leftarrow \mathbf{w} - \gamma_t\left(\nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y') - \nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y_i)\right)$
2. Return **w**

Structured Perceptron algorithm

Given a training set $D = \{(x_i, y_i)\}$

Initialize the model parameters $\mathbf{w}$

1. For epoch = 1 … T:
    1. Shuffle data
    2. For each training example $(\mathbf{x}_i, \mathbf{y}_i) \in D$:
        1. Let $\mathbf{y}' = \max_{\mathbf{y}} score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y})$ Inference within the training loop
        2. If $\mathbf{y}' \neq \mathbf{y}_i$

            Update $\mathbf{w} \leftarrow \mathbf{w} - \gamma_t\big(\nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y') - \nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y_i)\big)$
2. Return **w**

Structured Perceptron algorithm

Given a training set $D = \{(x_i, y_i)\}$

Initialize the model parameters $\mathbf{w}$

1. For epoch = 1 … T:
   1. Shuffle data
   2. For each training example $(\mathbf{x}_i, \mathbf{y}_i) \in D$:
      1. Let $\mathbf{y}' = \max_{\mathbf{y}} score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y})$
      2. If $\mathbf{y}' \neq \mathbf{y}_i$

         Update $\mathbf{w} \leftarrow \mathbf{w} - \gamma_t\big(\nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y') - \nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y_i)\big)$
2. Return **w**

Inference within the training loop
We can use any of the inference strategies here. E.g. beam search

Structured Perceptron algorithm

Given a training set $D = \{(x_i, y_i)\}$

Initialize the model parameters $\mathbf{w}$

1. For epoch = 1 … T:
    1. Shuffle data
    2. For each training example $(\mathbf{x}_i, \mathbf{y}_i) \in D$:
        1. Let $\mathbf{y}' = \max_{\mathbf{y}} score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y})$
        2. If $\mathbf{y}' \neq \mathbf{y}_i$

            Update $\mathbf{w} \leftarrow \mathbf{w} - \gamma_t\big(\nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y') - \nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y_i)\big)$
2. Return **w**

Update only on an error.
Structured Perceptron is an mistake-driven algorithm.
If there is a mistake, promote **y** and demote **y'**

Structured Perceptron algorithm

Given a training set $D = \{(x_i, y_i)\}$

Initialize the model parameters $\mathbf{w}$

1. For epoch = 1 … T:
    1. Shuffle data
    2. For each training example $(\mathbf{x}_i, \mathbf{y}_i) \in D$:
        1. Let $\mathbf{y}' = \max_{\mathbf{y}} score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y})$
        2. If $\mathbf{y}' \neq \mathbf{y}_i$

           Update $\mathbf{w} \leftarrow \mathbf{w} - \gamma_t\big(\nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y') - \nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y_i)\big)$
2. Return **w**

Note that the gradients will be distributed over the underlying factors that make up $score_{\mathbf{w}}$

Similar strategies for the structural SVM and CRF objectives

Structural SVM

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathrm{C}\sum_i \max_{\mathbf{y}}\left(score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i) - score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i)\right)$$

Conditional Random Field (via the maximum a posteriori criterion)

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathrm{C}\sum_i -\log P(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{w})$$

Example: The max margin objective

Given a training set $D = \{(x_i, y_i)\}$

Initialize the model parameters $\mathbf{w}$

1. For epoch = 1 … T:
   1. Shuffle data
   2. For each training example $(\mathbf{x}_i, \mathbf{y}_i) \in D$:
      1. Let $\mathbf{y}' = \max_{\mathbf{y}} score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i)$
2. Return **w**

Example: The max margin objective

Given a training set $D = \{(x_i, y_i)\}$

Initialize the model parameters $\mathbf{w}$

1. For epoch = 1 … T:
    1. Shuffle data
    2. For each training example $(\mathbf{x}_i, \mathbf{y}_i) \in D$:
        1. Let $\mathbf{y}' = \max_{\mathbf{y}} score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i)$

"Loss-augmented inference" within the training loop

2. Return **w**

Example: The max margin objective

Given a training set $D = \{(x_i, y_i)\}$

Initialize the model parameters $\mathbf{w}$

1. For epoch = 1 … T:
    1. Shuffle data
    2. For each training example $(\mathbf{x}_i, \mathbf{y}_i) \in D$:
        1. Let $\mathbf{y}' = \max_{\mathbf{y}} score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i)$
        2. If $\mathbf{y}' \neq \mathbf{y}_i$:

            Update $\mathbf{w} \leftarrow (1 - \gamma_t)\mathbf{w}$
        3. Else:

            Update $\mathbf{w} \leftarrow (1 - \gamma_t)\mathbf{w} - C\gamma_t\left(\nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y') - \nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y_i)\right)$
2. Return **w**

Example: The max margin objective

Given a training set $D = \{(x_i, y_i)\}$

Initialize the model parameters $\mathbf{w}$

1. For epoch = 1 … T:
    1. Shuffle data
    2. For each training example $(\mathbf{x}_i, \mathbf{y}_i) \in D$:
        1. Let $\mathbf{y}' = \max_{\mathbf{y}} score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i)$
        2. If $\mathbf{y}' \neq \mathbf{y}_i$:
           Update $\mathbf{w} \leftarrow (1 - \gamma_t)\mathbf{w}$
           If there is no error in the inference, then make the weights smaller
        3. Else:
           Update $\mathbf{w} \leftarrow (1 - \gamma_t)\mathbf{w} - C\gamma_t\left(\nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y') - \nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y_i)\right)$
2. Return **w**

Example: The max margin objective

Given a training set $D = \{(x_i, y_i)\}$

Initialize the model parameters $\mathbf{w}$

1. For epoch = 1 … T:
    1. Shuffle data
    2. For each training example $(\mathbf{x}_i, \mathbf{y}_i) \in D$:
        1. Let $\mathbf{y}' = \max_{\mathbf{y}} score_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i)$
        2. If $\mathbf{y}' \neq \mathbf{y}_i$:

           Update $\mathbf{w} \leftarrow (1 - \gamma_t)\mathbf{w}$
        3. Else:

           Update $\mathbf{w} \leftarrow (1 - \gamma_t)\mathbf{w} - C\gamma_t\big(\nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y') - \nabla_{\mathbf{w}} score_{\mathbf{w}}(x_i, y_i)\big)$
2. Return $\mathbf{w}$

If there is an error, shrink the weights, and promote the ground truth and demote the prediction

Summary

- Different structured training objectives are really different loss functions
- The structured versions of hinge, log and Perceptron losses all involve inference
  - Hinge, Perceptron: Solve a maximization problem
  - Log: Solve an expectation problem
- Learning as stochastic optimization, even for structures
  - But, computing the loss (and the gradient) can be expensive