Generalized Linear Classifiers

Joakim Nivre

Linear Models in Theory

- The linear model for binary classification:

$$\hat{y} = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ -1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b < 0 \end{cases}$$

- Learning as optimization (loss + regularization):

$$\hat{\mathbf{w}}, \hat{b} = \underset{\mathbf{w},b}{\operatorname{argmin}} \mathcal{L}(\mathbf{w}, b; \mathcal{T}) + \lambda R(\mathbf{w}, b)$$

- Gradient descent for convex optimization:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla f(\mathbf{w}, b; \mathcal{T})$$

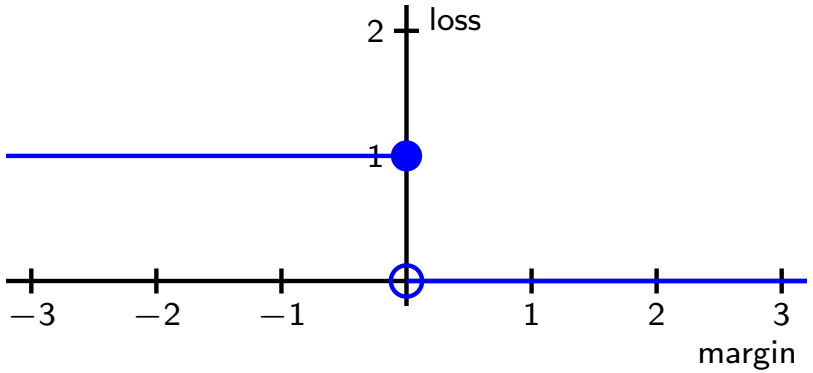
Linear Models in Practice

- Binary classification is sometimes useful
  - Spam filtering, spell checking, ...
- But most NLP problems involve more than two classes
  - Text categorization: news, business, culture, sports, ...
  - Word sense disambiguation: one class per sense
- And many involve structured prediction
  - Part-of-speech tagging: sequence-to-sequence
  - Dependency parsing: sequence-to-tree

Today's Lecture

- Multiclass classification
  - Using binary classifiers (OVA, AVA)
  - Multiclass perceptron (and generalizations)
- Structured prediction
  - Using simple (unstructured) classifiers
  - Structured perceptron (and generalizations)
- But first a little more on learning strategies ...

Minimize Error



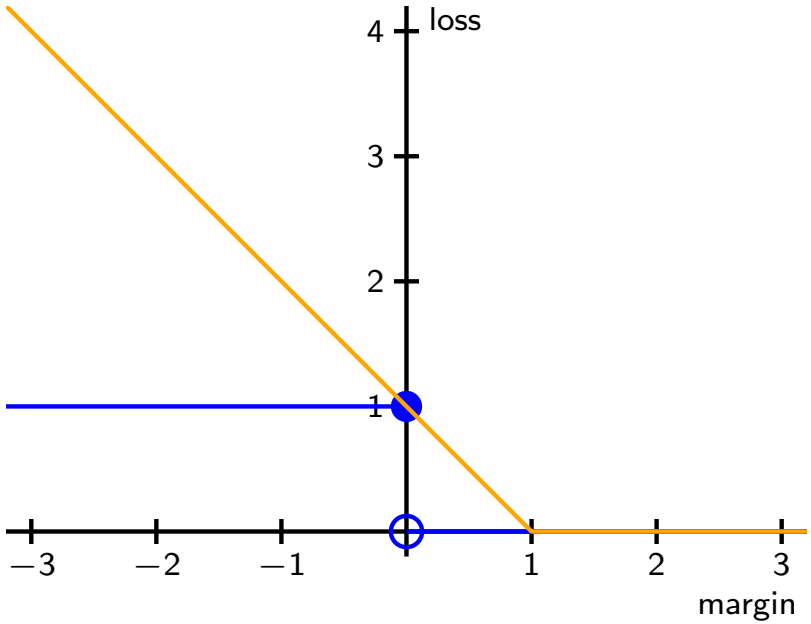
$$\mathcal{L}(\mathbf{w}, b; \mathcal{T}) = \begin{cases} 1 & \text{if } y\hat{y} \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

The perceptron (implicitly) minimizes 0-1 loss

Minimize Error

The perceptron (or 0-1 loss) does not care about margin

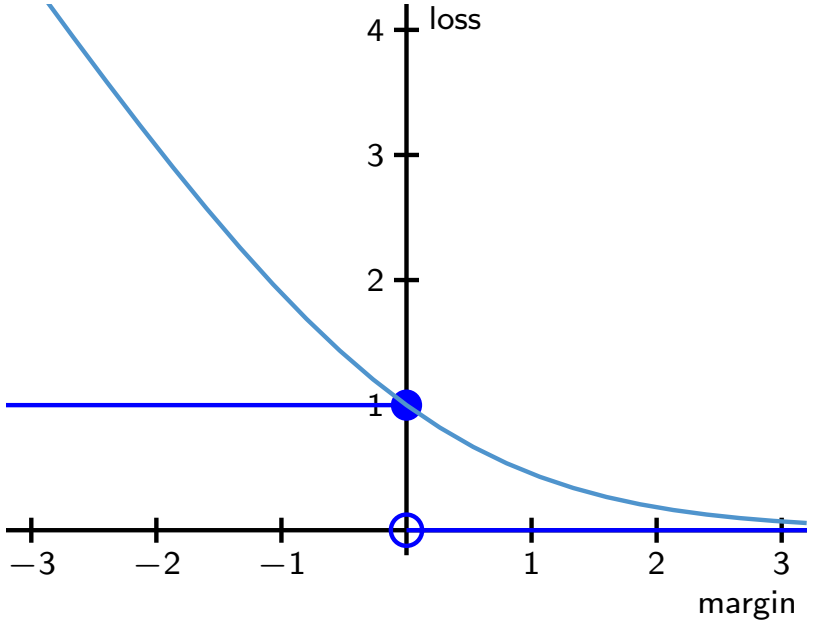
Maximize Margin



$$\mathcal{L}(\mathbf{w}, b; \mathcal{T}) = \max(0, 1 - y\hat{y})$$

Hinge loss goes to 0 with a margin of (at least) 1
Typical of (min-error) max-margin methods: SVM, MIRA, ...

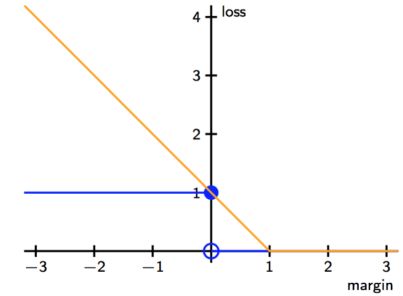
Maximize Likelihood

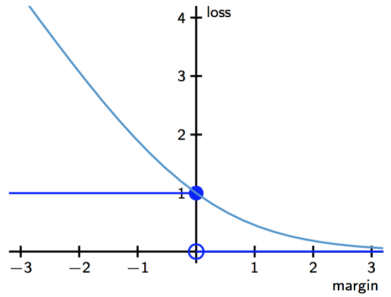


$$\mathcal{L}(\mathbf{w}, b; \mathcal{T}) = \frac{1}{\log 2} \log(1 + \exp(-y\hat{y}))$$

Log loss improves beyond a margin of 1
Minimizing log loss means maximizing likelihood

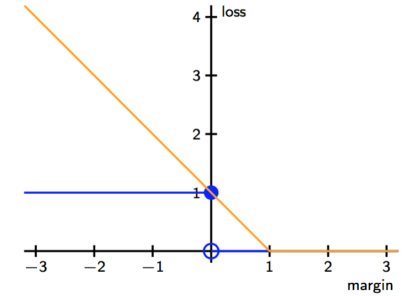
Min Error $\neq$ Max Likelihood

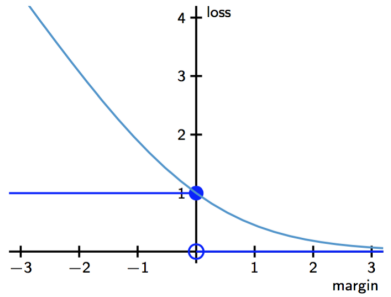




- Consider a training set $\mathcal{T}$ with 100 instances
  - 99 negative instances: $\langle\langle 2, 1\rangle, -1\rangle$
  - 1 positive instance: $\langle\langle 2, 3\rangle, 1\rangle$
- Consider the weight vector $\mathbf{w} = \langle -1, 1\rangle$
  - $\langle -1, 1\rangle \cdot \langle 2, 1\rangle = -1$
  - $\langle -1, 1\rangle \cdot \langle 2, 3\rangle = 1$
- Loss functions:
  - 0/1 loss $= 0$
  - Hinge loss $= 0$
  - Log loss $= 0.452 \times 100 = 45.2$

Min Error $\neq$ Max Likelihood





- Consider a training set $\mathcal{T}$ with 100 instances
  - 99 negative instances: $\langle\langle 2, 1\rangle, -1\rangle$
  - 1 positive instance: $\langle\langle 2, 3\rangle, 1\rangle$
- Consider the weight vector $\mathbf{w} = \langle -2, 1\rangle$
  - $\langle -2, 1\rangle \cdot \langle 2, 1\rangle = -3$
  - $\langle -2, 1\rangle \cdot \langle 2, 3\rangle = -1$
- Loss functions:
  - 0/1 loss $= 1$
  - Hinge loss $= 2$
  - Log loss $= 0.07 \times 99 + 1.895 \times 1 = 8.82$

Remember

- Different loss functions represent different learning strategies
- Regularization is important to prevent overfitting
- Training loss is not what we really care about

Multiclass Classification

- Can we do multiclass classification with binary classifiers?

Multiclass Classification

- Can we do multiclass classification with binary classifiers?
- Yes, but we need more than one classifier
  - One-Versus-All (OVA): one classifier for every class y_j
  - All-Versus-All (AVA): one classifier for every pair y_j, y_k

One-Versus-All

- Given multiclass training data:

$$\mathcal{T} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N} \quad y^{(i)} \in \{y_1, \ldots, y_n\}$$

- Create training set for each class y_j:

$$\mathcal{T}_j = \{(\boldsymbol{x}^{(i)}, z^{(i)})\}_{i=1}^{N} \quad z^{(i)} = \begin{cases} 1 & \text{if } y^{(i)} = y_j \\ -1 & \text{otherwise} \end{cases}$$

- Train one classifier (weight vector) $\mathbf{w}_{y_j}$ for each class y_j
- Decision rule:

$$f(\boldsymbol{x}) = \operatorname*{argmax}_y \mathbf{w}_y \cdot \boldsymbol{x}$$

All-Versus-All

- Given multiclass training data:

$$\mathcal{T} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N} \quad y^{(i)} \in \{y_1, \ldots, y_n\}$$

- Create training set for each pair y_j, y_k:

$$\mathcal{T}_{jk} = \{(\boldsymbol{x}^{(i)}, z^{(i)})\}_{i=1}^{N_{j,k}} \quad z^{(i)} = \begin{cases} 1 & \text{if } y^{(i)} = y_j \\ -1 & \text{if } y^{(i)} = y_k \end{cases}$$

- Train one classifier (weight vector) $\mathbf{w}_{jk}$ for each pair y_j, y_k
- Score for y_j combines all classifiers involving y_j

OVA or AVA?

- Both methods come with guarantees (see textbook)
- Both methods can work well in practice
- OVA is more efficient both at training and classification time
- Given n classes:
  - OVA only needs to train and run n classifiers
  - AVA requires $\frac{n(n-1)}{2}$ classifiers

Generalized Linear Models

- In binary classification, we use feature vectors over inputs:

$$\mathbf{f}(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}^m$$

- For multiple classes, we need to represent input-output pairs:

$$\mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^m$$

- This can be generalized to structured outputs (more later)

Examples

- $\boldsymbol{x}$ is a document and $\boldsymbol{y}$ is a label

$$\mathbf{f}_j(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ contains the word "interest"} \\ & \text{and } \boldsymbol{y} = \text{"financial"} \\ 0 & \text{otherwise} \end{cases}$$

$\mathbf{f}_j(\boldsymbol{x}, \boldsymbol{y}) =$ % of words in $\boldsymbol{x}$ with punctuation and $\boldsymbol{y} =$ "scientific"

- $\boldsymbol{x}$ is a word and $\boldsymbol{y}$ is a part-of-speech tag

$$\mathbf{f}_j(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{x} = \text{"bank" and } \boldsymbol{y} = \text{Verb} \\ 0 & \text{otherwise} \end{cases}$$

Examples

- $\boldsymbol{x}$ is a name, $\boldsymbol{y}$ is a label classifying the name

$$\mathbf{f}_0(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ contains "George"} \\ & \text{and } \boldsymbol{y} = \text{"Person"} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_4(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ contains "George"} \\ & \text{and } \boldsymbol{y} = \text{"Object"} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_1(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ contains "Washington"} \\ & \text{and } \boldsymbol{y} = \text{"Person"} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_5(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ contains "Washington"} \\ & \text{and } \boldsymbol{y} = \text{"Object"} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_2(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ contains "Bridge"} \\ & \text{and } \boldsymbol{y} = \text{"Person"} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_6(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ contains "Bridge"} \\ & \text{and } \boldsymbol{y} = \text{"Object"} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_3(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ contains "General"} \\ & \text{and } \boldsymbol{y} = \text{"Person"} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_7(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ contains "General"} \\ & \text{and } \boldsymbol{y} = \text{"Object"} \\ 0 & \text{otherwise} \end{cases}$$

- $\boldsymbol{x}$=General George Washington, $\boldsymbol{y}$=Person $\rightarrow \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = [1\ 1\ 0\ 1\ 0\ 0\ 0\ 0]$
- $\boldsymbol{x}$=George Washington Bridge, $\boldsymbol{y}$=Object $\rightarrow \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = [0\ 0\ 0\ 0\ 1\ 1\ 1\ 0]$
- $\boldsymbol{x}$=George Washington George, $\boldsymbol{y}$=Object $\rightarrow \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = [0\ 0\ 0\ 0\ 1\ 1\ 0\ 0]$

Block Feature Vectors

- $\boldsymbol{x}$=General George Washington, $\boldsymbol{y}$=Person $\rightarrow \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) =$ [1 1 0 1 0 0 0 0]
- $\boldsymbol{x}$=George Washington Bridge, $\boldsymbol{y}$=Object $\rightarrow \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) =$ [0 0 0 0 1 1 1 0]
- $\boldsymbol{x}$=George Washington George, $\boldsymbol{y}$=Object $\rightarrow \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) =$ [0 0 0 0 1 1 0 0]

- One equal-size block of the feature vector for each label
- Input features duplicated in each block
- Non-zero values allowed only in one block

Multiclass Linear Classification

- Let $\mathbf{w} \in \mathbb{R}^m$ be a weight vector
- If we assume that $\mathbf{w}$ is known, then we define our classifier as

$$\begin{aligned} \hat{\boldsymbol{y}} &= \operatorname*{argmax}_{\boldsymbol{y}} \ \mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) \\ &= \operatorname*{argmax}_{\boldsymbol{y}} \ \sum_{j=0}^{m} \mathbf{w}_j \times \mathbf{f}_j(\boldsymbol{x}, \boldsymbol{y}) \end{aligned}$$

Bias Terms

- Often linear classifiers presented as

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}} \sum_{j=0}^{m} \mathbf{w}_j \times \mathbf{f}_j(\boldsymbol{x}, \boldsymbol{y}) + b_{\boldsymbol{y}}$$

- Where b is a bias or offset term
- But this can be folded into $\mathbf{f}$

$\boldsymbol{x}$=General George Washington, $\boldsymbol{y}$=Person $\rightarrow \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = [1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0]$

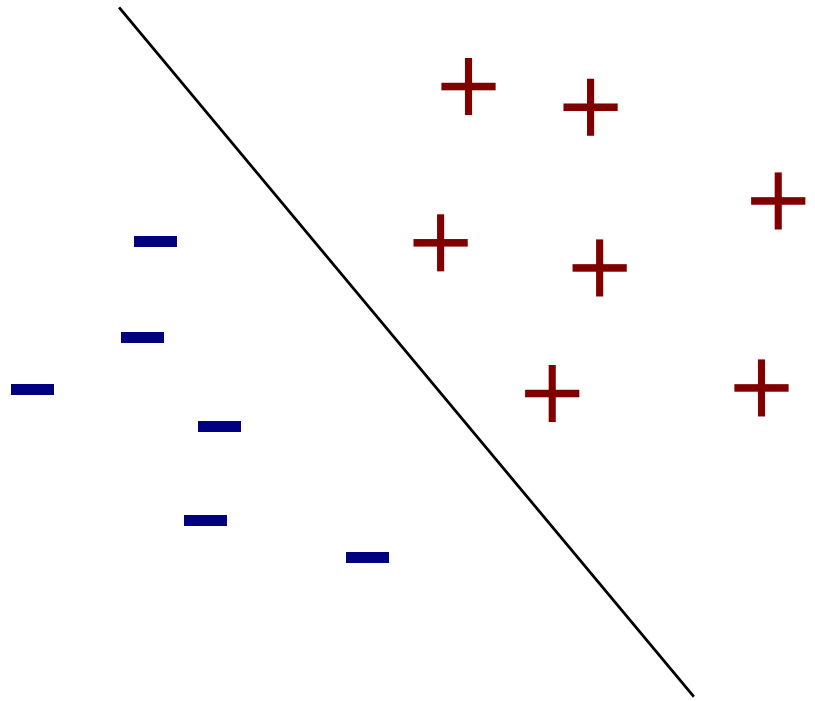
$\boldsymbol{x}$=General George Washington, $\boldsymbol{y}$=Object $\rightarrow \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = [0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1]$

$$\mathbf{f}_4(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \boldsymbol{y} = \text{“Person”} \\ 0 & \text{otherwise} \end{cases} \qquad \mathbf{f}_9(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \boldsymbol{y} = \text{“Object”} \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{w}_4$ and $\mathbf{w}_9$ are now the bias terms for the labels

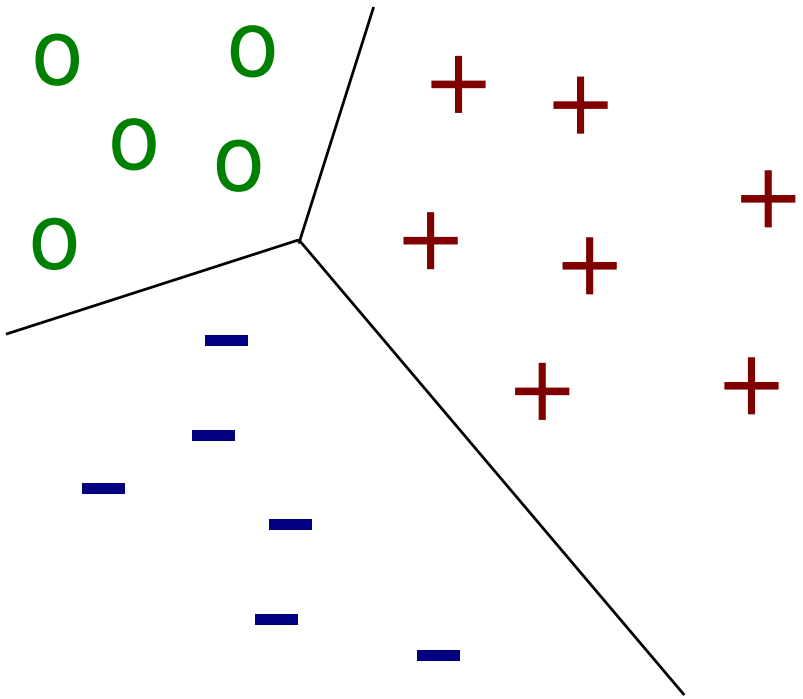
Binary Linear Classifier

Divides all points:



Multiclass Linear Classifier

Defines regions of space:



- i.e., $+$ are all points $(\boldsymbol{x}, \boldsymbol{y})$ where $+ = \text{argmax}_{\boldsymbol{y}}\ \mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$

Supervised Learning

- Input: Training examples $\mathcal{T} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}_t^{(i)})\}_{i=1}^{N}$
- Feature representation $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^m$
- Output: A vector $\mathbf{w}$ that optimizes some important function of the training set:
  - minimize error (Perceptron, SVMs, Boosting)
  - maximize likelihood of data (Logistic Regression, Naive Bayes)
- **NB:** Same as binary case except for feature representation

Perceptron Learning Algorithm

$\mathbf{w} \leftarrow 0$
for a fixed number of iterations **do**
  for all $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{T}$ **do**
    $\hat{\boldsymbol{y}} = \text{argmax}_{\boldsymbol{y}}\, \mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$
    if $\hat{\boldsymbol{y}} \neq \boldsymbol{y}$
      $\mathbf{w} = \mathbf{w} + \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) - \mathbf{f}(\boldsymbol{x}, \hat{\boldsymbol{y}})$
    end if
  end for
end for

Perceptron Learning Algorithm

$\mathbf{w} \leftarrow 0$
for a fixed number of iterations **do**
  for all $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{T}$ **do**
    $\hat{\boldsymbol{y}} = \text{argmax}_{\boldsymbol{y}}\, \mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$ [binary: $\hat{y} = \text{sign}(\mathbf{w} \cdot \boldsymbol{x})$]
    if $\hat{\boldsymbol{y}} \neq \boldsymbol{y}$
      $\mathbf{w} = \mathbf{w} + \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) - \mathbf{f}(\boldsymbol{x}, \hat{\boldsymbol{y}})$
    end if
  end for
end for

Perceptron Learning Algorithm

1: $\mathbf{w} \leftarrow 0$
2: **for** a fixed number of iterations **do**
3: **for all** $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{T}$ **do**
4: $\hat{\boldsymbol{y}} = \text{argmax}_{\boldsymbol{y}}\, \mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$
5: **if** $\hat{\boldsymbol{y}} \neq \boldsymbol{y}$ [binary: **if** $y\hat{y} < 0$]
6: $\mathbf{w} = \mathbf{w} + \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) - \mathbf{f}(\boldsymbol{x}, \hat{\boldsymbol{y}})$
7: **end if**
8: **end for**
9: **end for**

Perceptron Learning Algorithm

1: $\mathbf{w} \leftarrow 0$
2: **for** a fixed number of iterations **do**
3: **for all** $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{T}$ **do**
4: $\hat{\boldsymbol{y}} = \text{argmax}_{\boldsymbol{y}}\, \mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}, \boldsymbol{y})$
5: **if** $\hat{\boldsymbol{y}} \neq \boldsymbol{y}$
6: $\mathbf{w} = \mathbf{w} + \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) - \mathbf{f}(\boldsymbol{x}, \hat{\boldsymbol{y}})$ [binary: $\mathbf{w} = \mathbf{w} + \boldsymbol{x}$]
7: **end if**
8: **end for**
9: **end for**

More Generalized Classifiers

- The multiclass version with block vectors can be generalized
- Loss functions need to be adapted to new setup
- Example hinge loss:

Binary: $\max(0, 1 - y\hat{y})$

Multiclass: $\max(0, 1 - (\mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) - \max_{\boldsymbol{y}' \neq \boldsymbol{y}} \mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}')))$

Structured Prediction

- Sometimes $\mathcal{Y}$ does not consist of simple atomic classes
- Examples:
  - **Parsing**: for a sentence $\boldsymbol{x}$, $\mathcal{Y}$ is the set of possible parse trees
  - **Sequence tagging**: for a sentence $\boldsymbol{x}$, $\mathcal{Y}$ is the set of possible tag sequences, e.g., part-of-speech tags, named-entity tags
  - **Machine translation**: for a source sentence $\boldsymbol{x}$, $\mathcal{Y}$ is the set of possible target language sentences
- Can't we just use our multiclass learning algorithms?
  - The size of $\mathcal{Y}$ is exponential in the length of the input $\boldsymbol{x}$
  - It is non-trivial to apply our learning algorithms in such cases

Perceptron

1: $\mathbf{w} \leftarrow 0$
2: **for** a fixed number of iterations **do**
3: **for all** $(x, y) \in \mathcal{T}$ **do**
4: $\hat{y} = \text{argmax}_y \, \mathbf{w} \cdot \mathbf{f}(x, y)$
5: **if** $\hat{y} \neq y$
6: $\mathbf{w} = \mathbf{w} + \mathbf{f}(x, y) - \mathbf{f}(x, \hat{y})$
7: **end if**
8: **end for**
9: **end for**

Solving the argmax requires searching an exponential output space!

Factored Feature Representations

- Solution: Factor feature representations relative to the output
  - Context-free parsing:

$$\mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{A \rightarrow BC \in \boldsymbol{y}} \mathbf{f}(\boldsymbol{x}, A \rightarrow BC)$$

  - Sequence analysis – Markov assumptions:

$$\mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{|\boldsymbol{y}|} \mathbf{f}(\boldsymbol{x}, y_{i-1}, y_i)$$

- These kinds of factorizations allow us to run algorithms like CKY and Viterbi to compute the argmax function

Example – Sequence Labeling

- Many NLP problems can be cast in this light
  - Part-of-speech tagging
  - Named-entity extraction
  - Semantic role labeling
- Input: $\boldsymbol{x} = x_0 x_1 \ldots x_n$
- Output: $\boldsymbol{y} = y_0 y_1 \ldots y_n$
  - Each $y_i \in \mathcal{Y}_{\text{atom}}$ – which is small
  - Each $\boldsymbol{y} \in \mathcal{Y} = \mathcal{Y}_{\text{atom}}^n$ – which is large
- Example: part-of-speech tagging – $\mathcal{Y}_{\text{atom}}$ is set of tags

$\boldsymbol{x}$	$=$	John	saw	Mary	with	the	telescope
$\boldsymbol{y}$	$=$	PROPN	VERB	PROPN	ADP	DET	NOUN

Sequence Labeling – Output Interaction

$\boldsymbol{x}$	$=$	John	saw	Mary	with	the	telescope
$\boldsymbol{y}$	$=$	PROPN	VERB	PROPN	ADP	DET	NOUN

- Why not just make a sequence of multi-class predictions?

Sequence Labeling – Output Interaction

$\boldsymbol{x}$	$=$	John	saw	Mary	with	the	telescope
$\boldsymbol{y}$	$=$	PROPN	VERB	PROPN	ADP	DET	NOUN

- Why not just make a sequence of multi-class predictions?
- Because there are interactions between neighbouring tags
  - What tag does "saw" have?
  - What if I told you the previous tag was DET?
  - What if it was PROPN?

Sequence Labeling – Markov Factorization

$\boldsymbol{x}$	$=$	John	saw	Mary	with	the	telescope
$\boldsymbol{y}$	$=$	PROPN	VERB	PROPN	ADP	DET	NOUN

- Markov factorization – factor by adjacent labels
- First-order (like HMMs)

$$\mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{|\boldsymbol{y}|} \mathbf{f}(\boldsymbol{x}, y_{i-1}, y_i)$$

- Kth-order

$$\mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=k}^{|\boldsymbol{y}|} \mathbf{f}(\boldsymbol{x}, y_{i-k}, \ldots, y_{i-1}, y_i)$$

Sequence Labeling – Features

$\boldsymbol{x}$	$=$	John	saw	Mary	with	the	telescope
$\boldsymbol{y}$	$=$	PROPN	VERB	PROPN	ADP	DET	NOUN

- First-order

$$\mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{|\boldsymbol{y}|} \mathbf{f}(\boldsymbol{x}, y_{i-1}, y_i)$$

- $\mathbf{f}(\boldsymbol{x}, y_{i-1}, y_i)$ is any feature of the input & two adjacent labels

$$\mathbf{f}_j(\boldsymbol{x}, y_{i-1}, y_i) = \begin{cases} 1 & \text{if } x_i = \text{“saw”} \\ & \text{and } y_{i-1} = \text{PROPN and } y_i = \text{VERB} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_{j'}(\boldsymbol{x}, y_{i-1}, y_i) = \begin{cases} 1 & \text{if } x_i = \text{“saw”} \\ & \text{and } y_{i-1} = \text{DET and } y_i = \text{VERB} \\ 0 & \text{otherwise} \end{cases}$$

- $\mathbf{w}_j$ should get high weight and $\mathbf{w}_{j'}$ should get low weight

Sequence Labeling - Inference

- How does factorization effect inference?

$$
\begin{aligned}
\boldsymbol{y} &= \operatorname*{argmax}_{\boldsymbol{y}} \; \mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) \\
&= \operatorname*{argmax}_{\boldsymbol{y}} \; \mathbf{w} \cdot \sum_{i=1}^{|\boldsymbol{y}|} \mathbf{f}(\boldsymbol{x}, y_{i-1}, y_i) \\
&= \operatorname*{argmax}_{\boldsymbol{y}} \; \sum_{i=1}^{|\boldsymbol{y}|} \mathbf{w} \cdot \mathbf{f}(\boldsymbol{x}, y_{i-1}, y_i) \\
&= \operatorname*{argmax}_{\boldsymbol{y}} \; \sum_{i=1}^{|\boldsymbol{y}|} \sum_{j=1}^{m} \mathbf{w}_j \cdot \mathbf{f}_j(\boldsymbol{x}, y_{i-1}, y_i)
\end{aligned}
$$

- We can use the Viterbi algorithm!

Structured Learning

- Efficient inference for factored representations
  - Viterbi for sequence labeling
  - CKY for constituency parsing
  - Spanning tree algorithms for dependency parsing
- But what about learning?

Structured Perceptron

1: $\mathbf{w} \leftarrow 0$
2: **for** a fixed number of iterations **do**
3: **for all** $(x, y) \in \mathcal{T}$ **do**
4: $\hat{y} = \text{argmax}_y \, \mathbf{w} \cdot \mathbf{f}(x, y)$
5: **if** $\hat{y} \neq y$
6: $\mathbf{w} = \mathbf{w} + \mathbf{f}(x, y) - \mathbf{f}(x, \hat{y})$
7: **end if**
8: **end for**
9: **end for**

Solving the argmax is tractable with factored representations!

More Structured Classifiers

- Just as for multiclass classification, we can reuse the same trick for other learning strategies and loss functions
- Minimum error:
  - Structured SVM and MIRA
- Maximum likelihood:
  - Conditional Random Fields (CRF) – structured log regression
  - Hidden Markov Models (HMM) – structured Naive Bayes

Summing Up

- Linear models
  - Simple yet powerful learning framework
  - Generalize to multiclass and structured prediction
- Limitations:
  - Every classifier defines a hyperplane (linearity)
  - Complex features have to be engineered
- Next lecture (with Yan): May 3