Introduction to Machine Learning: Linear Learners[1]

Lisbon Machine Learning School, 2016

Stefan Riezler

Computational Linguistics & Scientific Computing
Heidelberg University, Germany
riezler@cl.uni-heidelberg.de

[1] Including (lots of) material from Shay Cohen and Ryan McDonald

Modeling the Frog's Perceptual System

Modeling the Frog's Perceptual System

- [Lettvin et al. 1959] show that the frog's perceptual system constructs reality by four separate operations:
  - contrast detection: presence of sharp boundary?
  - convexity detection: how curved and how big is object?
  - movement detection: is object moving?
  - dimming speed: how fast does object obstruct light?
- The frog's goal: Capture any object of the size of an insect or worm providing it moves like one.

Modeling the Frog's Perceptual System

- [Lettvin et al. 1959] show that the frog's perceptual system constructs reality by four separate operations:
  - contrast detection: presence of sharp boundary?
  - convexity detection: how curved and how big is object?
  - movement detection: is object moving?
  - dimming speed: how fast does object obstruct light?
- The frog's goal: Capture any object of the size of an insect or worm providing it moves like one.
- Can we build a model of this perceptual system and learn to capture the right objects?

Learning from Data

- Assume training data of edible (+) and inedible (-) objects

convex	speed	label	convex	speed	label
small	small	-	small	large	+
small	medium	-	medium	large	+
small	medium	-	medium	large	+
medium	small	-	large	small	+
large	small	-	large	large	+
small	small	-	large	medium	+
small	large	-			
small	medium	-			

Learning from Data

- Assume training data of edible (+) and inedible (-) objects

convex	speed	label	convex	speed	label
small	small	-	small	large	+
small	medium	-	medium	large	+
small	medium	-	medium	large	+
medium	small	-	large	small	+
large	small	-	large	large	+
small	small	-	large	medium	+
small	large	-			
small	medium	-			

- Learning model parameters from data:
  - p(+) = , p(-) =
  - p(convex = small|-) = , p(convex = med|-) = , p(convex = large|-) =
    p(speed = small|-) = , p(speed = med|-) = , p(speed = large|-) =
    p(convex = small|+) = , p(convex = med|+) = , p(convex = large|+) =
    p(speed = small|+) = , p(speed = med|+) = , p(speed = large|+) =

Learning from Data

- Assume training data of edible (+) and inedible (-) objects

convex	speed	label	convex	speed	label
small	small	-	small	large	+
small	medium	-	medium	large	+
small	medium	-	medium	large	+
medium	small	-	large	small	+
large	small	-	large	large	+
small	small	-	large	medium	+
small	large	-			
small	medium	-			

- Learning model parameters from data:
  - p(+) = 8/14, p(-) = 6/14
  - p(convex = small|-) = , p(convex = med|-) = , p(convex = large|-) = p(speed = small|-) = , p(speed = med|-) = , p(speed = large|-) = p(convex = small|+) = , p(convex = med|+) = , p(convex = large|+) = p(speed = small|+) = , p(speed = med|+) = , p(speed = large|+) =

Learning from Data

- Assume training data of edible (+) and inedible (-) objects

convex	speed	label	convex	speed	label
small	small	-	small	large	+
small	medium	-	medium	large	+
small	medium	-	medium	large	+
medium	small	-	large	small	+
large	small	-	large	large	+
small	small	-	large	medium	+
small	large	-			
small	medium	-			

- Learning model parameters from data:
  - $p(+) = 8/14$, $p(-) = 6/14$
  - $p(\text{convex} = \text{small}|-) = 6/8$, $p(\text{convex} = \text{med}|-) = 1/8$, $p(\text{convex} = \text{large}|-) = 1/8$
    $p(\text{speed} = \text{small}|-) = 4/8$, $p(\text{speed} = \text{med}|-) = 3/8$, $p(\text{speed} = \text{large}|-) = 1/8$
    $p(\text{convex} = \text{small}|+) = 1/6$, $p(\text{convex} = \text{med}|+) = 2/6$, $p(\text{convex} = \text{large}|+) = 3/6$
    $p(\text{speed} = \text{small}|+) = 1/6$, $p(\text{speed} = \text{med}|+) = 1/6$, $p(\text{speed} = \text{large}|+) = 4/6$

Learning from Data

- Assume training data of edible (+) and inedible (-) objects

convex	speed	label	convex	speed	label
small	small	-	small	large	+
small	medium	-	medium	large	+
small	medium	-	medium	large	+
medium	small	-	large	small	+
large	small	-	large	large	+
small	small	-	large	medium	+
small	large	-			
small	medium	-			

- Learning model parameters from data:
  - $p(+) = 8/14$, $p(-) = 6/14$
  - $p(\text{convex} = \text{small}|-) = 6/8$, $p(\text{convex} = \text{med}|-) = 1/8$, $p(\text{convex} = \text{large}|-) = 1/8$
    $p(\text{speed} = \text{small}|-) = 4/8$, $p(\text{speed} = \text{med}|-) = 3/8$, $p(\text{speed} = \text{large}|-) = 1/8$
    $p(\text{convex} = \text{small}|+) = 1/6$, $p(\text{convex} = \text{med}|+) = 2/6$, $p(\text{convex} = \text{large}|+) = 3/6$
    $p(\text{speed} = \text{small}|+) = 1/6$, $p(\text{speed} = \text{med}|+) = 1/6$, $p(\text{speed} = \text{large}|+) = 4/6$
- Predict unseen $p(\text{label} = ?, \text{convex} = \text{med}, \text{speed} = \text{med})$
  - $p(-) \cdot p(\text{convex} = \text{med}|-) \cdot p(\text{speed} = \text{med}|-) =$
  - $p(+) \cdot p(\text{convex} = \text{med}|+) \cdot p(\text{speed} = \text{med}|+) =$

Learning from Data

- Assume training data of edible (+) and inedible (-) objects

convex	speed	label	convex	speed	label
small	small	-	small	large	+
small	medium	-	medium	large	+
small	medium	-	medium	large	+
medium	small	-	large	small	+
large	small	-	large	large	+
small	small	-	large	medium	+
small	large	-			
small	medium	-			

- Learning model parameters from data:
  - $p(+) = 8/14$, $p(-) = 6/14$
  - $p(\text{convex} = \text{small}|-) = 6/8$, $p(\text{convex} = \text{med}|-) = 1/8$, $p(\text{convex} = \text{large}|-) = 1/8$
    $p(\text{speed} = \text{small}|-) = 4/8$, $p(\text{speed} = \text{med}|-) = 3/8$, $p(\text{speed} = \text{large}|-) = 1/8$
    $p(\text{convex} = \text{small}|+) = 1/6$, $p(\text{convex} = \text{med}|+) = 2/6$, $p(\text{convex} = \text{large}|+) = 3/6$
    $p(\text{speed} = \text{small}|+) = 1/6$, $p(\text{speed} = \text{med}|+) = 1/6$, $p(\text{speed} = \text{large}|+) = 4/6$
- Predict unseen $p(\text{label} = ?, \text{convex} = \text{med}, \text{speed} = \text{med})$
  - $p(-) \cdot p(\text{convex} = \text{med}|-) \cdot p(\text{speed} = \text{med}|-) = 8/14 \cdot 1/8 \cdot 3/8 = 0.027$
  - $p(+) \cdot p(\text{convex} = \text{med}|+) \cdot p(\text{speed} = \text{med}|+) =$

Learning from Data

- Assume training data of edible (+) and inedible (-) objects

convex	speed	label	convex	speed	label
small	small	-	small	large	+
small	medium	-	medium	large	+
small	medium	-	medium	large	+
medium	small	-	large	small	+
large	small	-	large	large	+
small	small	-	large	medium	+
small	large	-			
small	medium	-			

- Learning model parameters from data:
  - $p(+) = 8/14$, $p(-) = 6/14$
  - $p(\text{convex} = \text{small}|-) = 6/8$, $p(\text{convex} = \text{med}|-) = 1/8$, $p(\text{convex} = \text{large}|-) = 1/8$
    $p(\text{speed} = \text{small}|-) = 4/8$, $p(\text{speed} = \text{med}|-) = 3/8$, $p(\text{speed} = \text{large}|-) = 1/8$
    $p(\text{convex} = \text{small}|+) = 1/6$, $p(\text{convex} = \text{med}|+) = 2/6$, $p(\text{convex} = \text{large}|+) = 3/6$
    $p(\text{speed} = \text{small}|+) = 1/6$, $p(\text{speed} = \text{med}|+) = 1/6$, $p(\text{speed} = \text{large}|+) = 4/6$
- Predict unseen $p(\text{label} = ?, \text{convex} = \text{med}, \text{speed} = \text{med})$
  - $p(-) \cdot p(\text{convex} = \text{med}|-) \cdot p(\text{speed} = \text{med}|-) = 8/14 \cdot 1/8 \cdot 3/8 = 0.027$
  - $p(+) \cdot p(\text{convex} = \text{med}|+) \cdot p(\text{speed} = \text{med}|+) = 6/14 \cdot 2/6 \cdot 1/6 = 0.024$

Learning from Data

- Assume training data of edible (+) and inedible (-) objects

convex	speed	label	convex	speed	label
small	small	-	small	large	+
small	medium	-	medium	large	+
small	medium	-	medium	large	+
medium	small	-	large	small	+
large	small	-	large	large	+
small	small	-	large	medium	+
small	large	-			
small	medium	-			

- Learning model parameters from data:
  - $p(+) = 8/14$, $p(-) = 6/14$
  - $p(\text{convex} = \text{small}|-) = 6/8$, $p(\text{convex} = \text{med}|-) = 1/8$, $p(\text{convex} = \text{large}|-) = 1/8$
    $p(\text{speed} = \text{small}|-) = 4/8$, $p(\text{speed} = \text{med}|-) = 3/8$, $p(\text{speed} = \text{large}|-) = 1/8$
    $p(\text{convex} = \text{small}|+) = 1/6$, $p(\text{convex} = \text{med}|+) = 2/6$, $p(\text{convex} = \text{large}|+) = 3/6$
    $p(\text{speed} = \text{small}|+) = 1/6$, $p(\text{speed} = \text{med}|+) = 1/6$, $p(\text{speed} = \text{large}|+) = 4/6$
- Predict unseen $p(\text{label} = ?, \text{convex} = \text{med}, \text{speed} = \text{med})$
  - $p(-) \cdot p(\text{convex} = \text{med}|-) \cdot p(\text{speed} = \text{med}|-) = 8/14 \cdot 1/8 \cdot 3/8 = 0.027$
  - $p(+) \cdot p(\text{convex} = \text{med}|+) \cdot p(\text{speed} = \text{med}|+) = 6/14 \cdot 2/6 \cdot 1/6 = 0.024$
  - Inedible: $p(\text{convex} = \text{med}, \text{speed} = \text{med}, \text{label} = -) > p(\text{convex} = \text{med}, \text{speed} = \text{med}, \text{label} = +)$!

Machine Learning is a Frog's World

- Machine learning problems can be seen as problems of function estimation where
  - our models are based on a combined feature representation of inputs and outputs
    - *similar to the frog whose world is constructed by four-dimensional feature vector based on detection operations*

Machine Learning is a Frog's World

- Machine learning problems can be seen as problems of function estimation where
  - our models are based on a combined feature representation of inputs and outputs
    - *similar to the frog whose world is constructed by four-dimensional feature vector based on detection operations*
  - learning of parameter weights is done by optimizing fit of model to training data
    - *frog uses binary classification into edible/inedible objects as supervision signals for learning*

Machine Learning is a Frog's World

- Machine learning problems can be seen as problems of function estimation where
  - our models are based on a combined feature representation of inputs and outputs
    - *similar to the frog whose world is constructed by four-dimensional feature vector based on detection operations*
  - learning of parameter weights is done by optimizing fit of model to training data
    - *frog uses binary classification into edible/inedible objects as supervision signals for learning*
  - The model used in the frog's perception example is called *Naive Bayes*: It measures compatibility of inputs to outputs by a linear model and optimizes parameters by linear optimization

Lecture Outline: Linear Learners for NLP

- Preliminaries
  - Data: input/output, assumptions
  - Feature representations
  - Linear models
- Linear learners
  - Naive Bayes
  - Generative versus discriminative
  - Logistic Regression
  - Perceptron
  - Large-Margin Learners (SVMs)
- Regularization
- Online learning
- Non-linear models

Inputs and Outputs

- Input: $\boldsymbol{x} \in \mathcal{X}$
  - e.g., document or sentence with some words $\boldsymbol{x} = w_1 \dots w_n$
- Output: $\boldsymbol{y} \in \mathcal{Y}$
  - e.g., document class, translation, parse tree
- Input/Output pair: $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$
  - e.g., a document $\boldsymbol{x}$ and its class label $\boldsymbol{y}$,
  - a source sentence $\boldsymbol{x}$ and its translation $\boldsymbol{y}$,
  - a sentence $\boldsymbol{x}$ and its parse tree $\boldsymbol{y}$

Feature Representations

- We assume a mapping from input $\boldsymbol{x}$ to a high dimensional feature vector
  - $\boldsymbol{\phi}(\boldsymbol{x}) : \mathcal{X} \rightarrow \mathbb{R}^m$
- For many cases, more convenient to have mapping from input-output pairs $(\boldsymbol{x}, \boldsymbol{y})$
  - $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^m$
- Under certain assumptions, these are equivalent
- Most papers in NLP use $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$

Feature Representations

- We assume a mapping from input $\boldsymbol{x}$ to a high dimensional feature vector
  - $\boldsymbol{\phi}(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}^m$
- For many cases, more convenient to have mapping from input-output pairs $(\boldsymbol{x}, \boldsymbol{y})$
  - $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^m$
- Under certain assumptions, these are equivalent
- Most papers in NLP use $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$
- (Was?) not so common in NLP: $\boldsymbol{\phi} \in \mathbb{R}^m$ (but see word embeddings)
- More common: $\boldsymbol{\phi}_i \in \{1, \ldots, F_i\}$, $F_i \in \mathbb{N}^+$ (categorical)
- Very common: $\boldsymbol{\phi} \in \{0, 1\}^m$ (binary)

Feature Representations

- We assume a mapping from input $\boldsymbol{x}$ to a high dimensional feature vector
  - $\boldsymbol{\phi}(\boldsymbol{x}) : \mathcal{X} \rightarrow \mathbb{R}^m$
- For many cases, more convenient to have mapping from input-output pairs $(\boldsymbol{x}, \boldsymbol{y})$
  - $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^m$
- Under certain assumptions, these are equivalent
- Most papers in NLP use $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$
- (Was?) not so common in NLP: $\boldsymbol{\phi} \in \mathbb{R}^m$ (but see word embeddings)
- More common: $\boldsymbol{\phi}_i \in \{1, \ldots, F_i\}$, $F_i \in \mathbb{N}^+$ (categorical)
- Very common: $\boldsymbol{\phi} \in \{0, 1\}^m$ (binary)
- For any vector $\mathbf{v} \in \mathbb{R}^m$, let $\mathbf{v}_j$ be the j^{th} value

Examples

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

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

We expect this feature to have a positive weight, "interest" is a positive indicator for the label "financial"

Examples

$\phi_j(\boldsymbol{x}, \boldsymbol{y}) =$ % of words in $\boldsymbol{x}$ containing punctuation and $\boldsymbol{y} =$ "scientific"

Punctuation symbols - positive indicator or negative indicator for scientific articles?

Examples

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

$$\phi_j(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{x} = \text{“bank”} \text{ and } \boldsymbol{y} = \text{ Verb} \\ 0 & \text{otherwise} \end{cases}$$

What weight would it get?

Examples

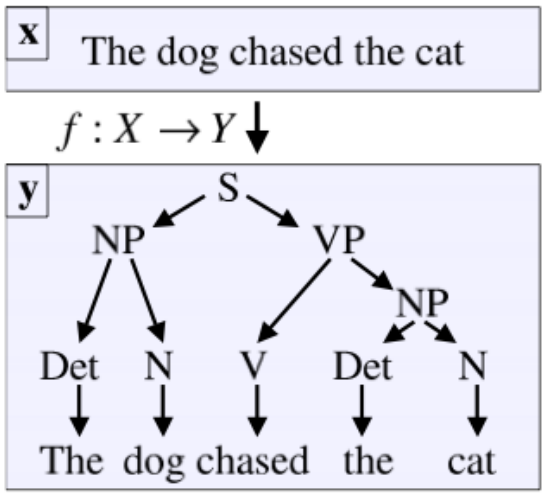
- $\boldsymbol{x}$ is a source sentence and $\boldsymbol{y}$ is translation

$$\phi_j(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if “y a-t-il” present in } \boldsymbol{x} \\ & \quad \text{and “are there” present in } \boldsymbol{y} \\ 0 & \text{otherwise} \end{cases}$$

$$\phi_k(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if “y a-t-il” present in } \boldsymbol{x} \\ & \quad \text{and “are there any” present in } \boldsymbol{y} \\ 0 & \text{otherwise} \end{cases}$$

Which phrase indicator should be preferred?

Examples



$$\Psi(\mathbf{x},\mathbf{y}) = \begin{pmatrix} 1 & S \rightarrow NP\ VP \\ 0 & S \rightarrow NP \\ 2 & NP \rightarrow Det\ N \\ 1 & VP \rightarrow V\ NP \\ \vdots & \\ 0 & Det \rightarrow dog \\ 2 & Det \rightarrow the \\ 1 & N \rightarrow dog \\ 1 & V \rightarrow chased \\ 1 & N \rightarrow cat \end{pmatrix}$$

Note: Label $\boldsymbol{y}$ includes sentence $\boldsymbol{x}$

Linear Models

- **Linear model**: Defines a discriminant function that is based on linear combination of features and weights

$$
\begin{aligned}
f(\boldsymbol{x};\boldsymbol{\omega}) &= \underset{\boldsymbol{y}\in\mathcal{Y}}{\arg\max}\ \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y}) \\
&= \underset{\boldsymbol{y}\in\mathcal{Y}}{\arg\max}\ \sum_{j=0}^{m}\boldsymbol{\omega}_j\times\boldsymbol{\phi}_j(\boldsymbol{x},\boldsymbol{y})
\end{aligned}
$$

Linear Models

- **Linear model**: Defines a discriminant function that is based on linear combination of features and weights

$$\begin{aligned} f(\boldsymbol{x};\boldsymbol{\omega}) &= \arg\max_{\boldsymbol{y}\in\mathcal{Y}} \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y}) \\ &= \arg\max_{\boldsymbol{y}\in\mathcal{Y}} \sum_{j=0}^{m} \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x},\boldsymbol{y}) \end{aligned}$$

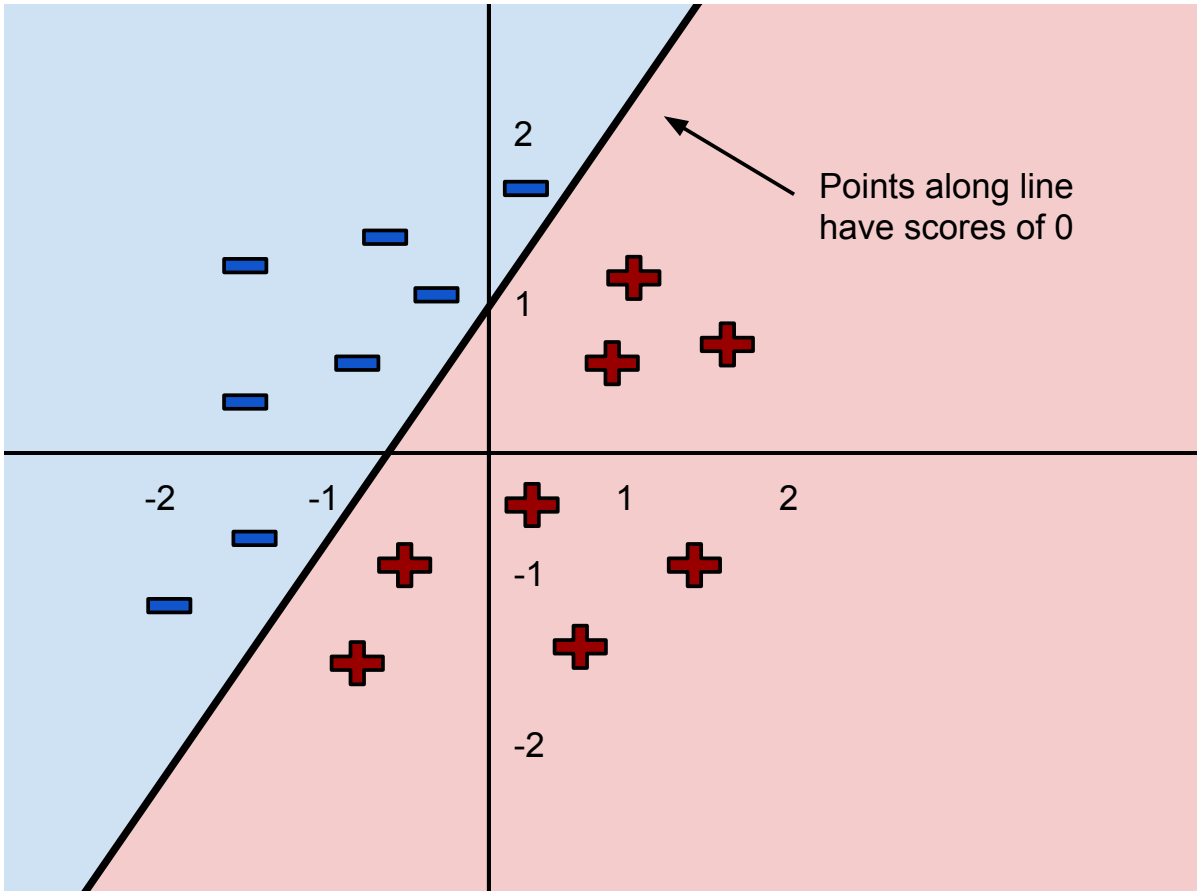
- Let $\boldsymbol{\omega} \in \mathbb{R}^m$ be a high dimensional weight vector
- Assume that $\boldsymbol{\omega}$ is known
  - **Multiclass Classification**: $\mathcal{Y} = \{0, 1, \ldots, N\}$

$$\boldsymbol{y} = \arg\max_{\boldsymbol{y}\in\mathcal{Y}} \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y})$$

  - **Binary Classification** just a special case of multiclass

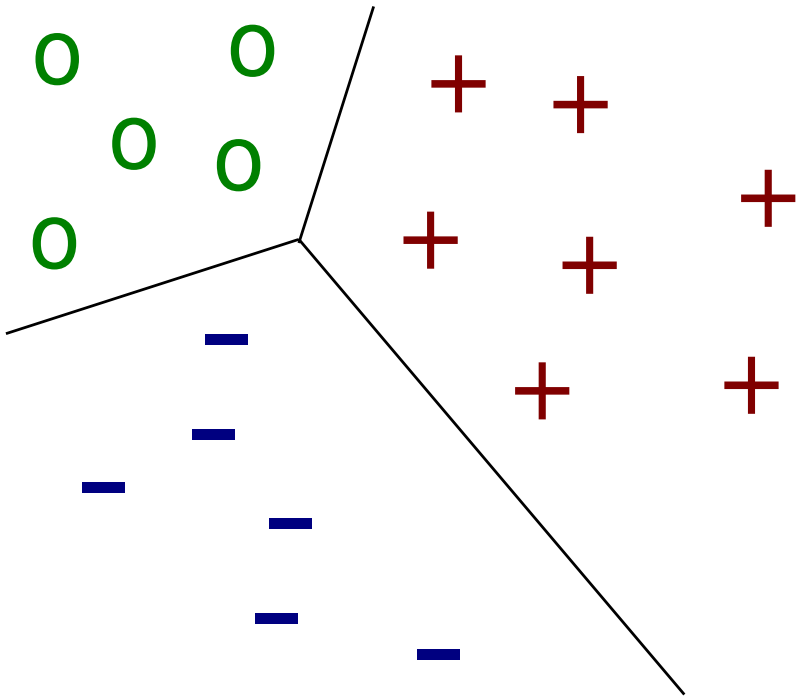
Binary Linear Model

ω defines a hyperplane (line in 2 dimensions) that divides all points:



Multiclass Linear Model

Defines regions of space. Visualization difficult.

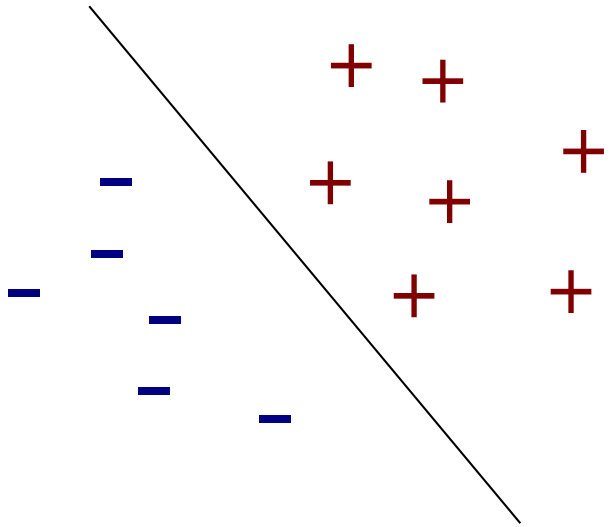


- $+$ are all points $(\boldsymbol{x}, \boldsymbol{y})$ where $+ = \arg\max_{\boldsymbol{y}} \ \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$

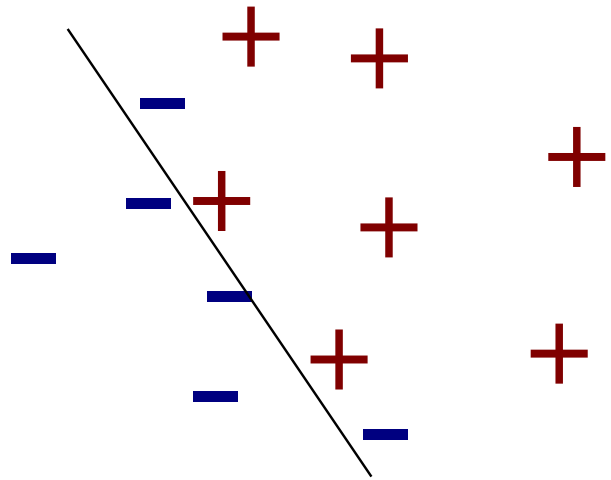
Separability

- A set of points is separable, if there exists a ω such that classification is perfect

Separable



Not Separable



Supervised Learning

We now have a way to make decisions... if we have a weight vector $\boldsymbol{\omega}$. But where do we get this $\boldsymbol{\omega}$?

- Input:
  - i.i.d. (independent and identically distributed) training examples $\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{|\mathcal{T}|}$
  - feature representation $\boldsymbol{\phi}$

Supervised Learning

We now have a way to make decisions... if we have a weight vector $\boldsymbol{\omega}$. But where do we get this $\boldsymbol{\omega}$?

- Input:
  - i.i.d. (independent and identically distributed) training examples $\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{|\mathcal{T}|}$
  - feature representation $\boldsymbol{\phi}$
- Output: $\boldsymbol{\omega}$ that maximizes an objective function on the training set
  - $\boldsymbol{\omega} = \arg\max \mathcal{L}(\mathcal{T}; \boldsymbol{\omega})$
  - Equivalently minimize: $\boldsymbol{\omega} = \arg\min -\mathcal{L}(\mathcal{T}; \boldsymbol{\omega})$

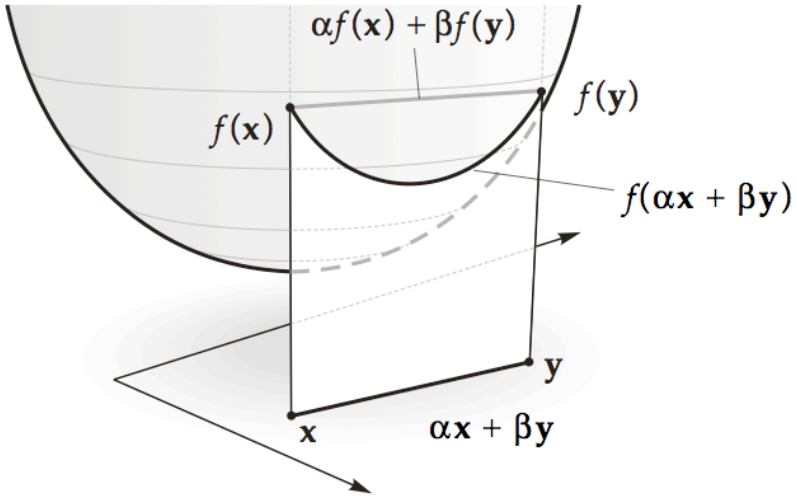
Objective Functions

- Ideally we can decompose $\mathcal{L}$ by training pairs $(\boldsymbol{x}, \boldsymbol{y})$
  - $\mathcal{L}(\mathcal{T}; \boldsymbol{\omega}) \propto \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{T}} \mathit{loss}((\boldsymbol{x}, \boldsymbol{y}); \boldsymbol{\omega})$
  - *loss* is a function that measures some value correlated with errors of parameters $\boldsymbol{\omega}$ on instance $(\boldsymbol{x}, \boldsymbol{y})$

Objective Functions

- Ideally we can decompose $\mathcal{L}$ by training pairs $(\boldsymbol{x}, \boldsymbol{y})$
  - $\mathcal{L}(\mathcal{T}; \boldsymbol{\omega}) \propto \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{T}} \mathit{loss}((\boldsymbol{x}, \boldsymbol{y}); \boldsymbol{\omega})$
  - *loss* is a function that measures some value correlated with errors of parameters $\boldsymbol{\omega}$ on instance $(\boldsymbol{x}, \boldsymbol{y})$
- Example:
  - $\boldsymbol{y} \in \{1, -1\}$, $f(\boldsymbol{x}; \boldsymbol{\omega})$ is the prediction we make for x using $\boldsymbol{\omega}$
  - 0-1 loss function: $\mathit{loss}((\boldsymbol{x}, \boldsymbol{y}); \boldsymbol{\omega}) = \begin{cases} 0 & \text{if } f(\boldsymbol{x}; \boldsymbol{\omega}) = y, \\ 1 & \text{else} \end{cases}$

Convex Optimization



- A function is convex if its graph lies on or below the line segment connecting any two points on the graph

$$f(\alpha \boldsymbol{x} + \beta \boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + \beta f(\boldsymbol{y}) \text{ for all } \alpha, \beta \geq 0,\ \alpha + \beta = 1 \quad (1)$$

- For linear models we have equality in (1)

Convex Optimization

- Optimization problem is defined as problem of finding a point that minimizes our objective function (maximization is minimization of $-f(x)$)
- Limit attention to convex (or even linear) functions
- Find point at which gradient of f is 0
  - Slope of the gradient is non-decreasing as one moves away from minimum, 0 at minimum
  - In order to find minimum, follow opposite direction of gradient
  - For convex functions, this will lead to the single global minimum

Naive Bayes

Naive Bayes

- Probabilistic decision model:

$$\arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}) \propto \arg\max_{\boldsymbol{y}} P(\boldsymbol{y})P(\boldsymbol{x}|\boldsymbol{y})$$

  - Uses Bayes Rule:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{P(\boldsymbol{y})P(\boldsymbol{x}|\boldsymbol{y})}{P(\boldsymbol{x})} \text{ for fixed } \boldsymbol{x}$$

- Generative model since $P(\boldsymbol{y})P(\boldsymbol{x}|\boldsymbol{y}) = P(\boldsymbol{x}, \boldsymbol{y})$ is a joint probability
  - Because we model a distribution that can randomly generate outputs and inputs, not just outputs

Naivety of Naive Bayes

- We need to decide on the structure of $P(\boldsymbol{x}, \boldsymbol{y})$
- $P(\boldsymbol{x}|\boldsymbol{y}) = P(\boldsymbol{\phi}(\boldsymbol{x})|\boldsymbol{y}) = P(\boldsymbol{\phi}_1(\boldsymbol{x}), \ldots, \boldsymbol{\phi}_m(\boldsymbol{x})|\boldsymbol{y})$

Naive Bayes Assumption

(conditional independence)

$$P(\boldsymbol{\phi}_1(\boldsymbol{x}), \ldots, \boldsymbol{\phi}_m(\boldsymbol{x})|\boldsymbol{y}) = \prod_i P(\boldsymbol{\phi}_i(\boldsymbol{x})|\boldsymbol{y})$$

$$P(\boldsymbol{x}, \boldsymbol{y}) = P(\boldsymbol{y})P(\boldsymbol{\phi}_1(\boldsymbol{x}), \ldots, \boldsymbol{\phi}_m(\boldsymbol{x})|\boldsymbol{y}) = P(\boldsymbol{y}) \prod_{i=1}^{m} P(\boldsymbol{\phi}_i(\boldsymbol{x})|\boldsymbol{y})$$

Naive Bayes – Learning

- Input: $\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{|\mathcal{T}|}$
- Let $\boldsymbol{\phi}_i(\boldsymbol{x}) \in \{1, \ldots, F_i\}$
- Parameters $\mathcal{P} = \{P(\boldsymbol{y}), P(\boldsymbol{\phi}_i(\boldsymbol{x})|\boldsymbol{y})\}$
  - Both $P(\boldsymbol{y})$ and $P(\boldsymbol{\phi}_i(\boldsymbol{x})|\boldsymbol{y})$ are multinomials

Maximum Likelihood Estimation

- What's left? Defining an objective $\mathcal{L}(\mathcal{T})$
- $\mathcal{P}$ plays the role of $\boldsymbol{\omega}$
- What objective to use?
- Objective: Maximum Likelihood Estimation (MLE)

$$\mathcal{L}(\mathcal{T}) = \prod_{t=1}^{|\mathcal{T}|} P(\boldsymbol{x}_t, \boldsymbol{y}_t) = \prod_{t=1}^{|\mathcal{T}|} \left(P(\boldsymbol{y}_t) \prod_{i=1}^{m} P(\phi_i(\boldsymbol{x}_t) | \boldsymbol{y}_t) \right)$$

Naive Bayes – Learning

MLE has closed form solution

$$\mathcal{P} = \arg\max_{\mathcal{P}} \prod_{t=1}^{|\mathcal{T}|} \left(P(\boldsymbol{y}_t) \prod_{i=1}^{m} P(\boldsymbol{\phi}_i(\boldsymbol{x}_t)|\boldsymbol{y}_t) \right)$$

$$P(\boldsymbol{y}) = \frac{\sum_{t=1}^{|\mathcal{T}|} [\![\boldsymbol{y}_t = \boldsymbol{y}]\!]}{|\mathcal{T}|}$$

$$P(\boldsymbol{\phi}_i(\boldsymbol{x})|\boldsymbol{y}) = \frac{\sum_{t=1}^{|\mathcal{T}|} [\![\boldsymbol{\phi}_i(\boldsymbol{x}_t) = \boldsymbol{\phi}_i(\boldsymbol{x}) \text{ and } \boldsymbol{y}_t = \boldsymbol{y}]\!]}{\sum_{t=1}^{|\mathcal{T}|} [\![\boldsymbol{y}_t = \boldsymbol{y}]\!]}$$

$$\text{where } [\![p]\!] = \begin{cases} 1 & \text{if } p \text{ is true,} \\ 0 & \text{otherwise.} \end{cases}$$

Thus, these are just normalized counts over events in $\mathcal{T}$

Naive Bayes Document Classification

- doc 1: $y_1 =$ sports, "hockey is fast"
- doc 2: $y_2 =$ politics, "politicians talk fast"
- doc 3: $y_3 =$ politics, "washington is sleazy"

- $\phi_0(x) = 1$ if doc has word 'hockey', 0 else
- $\phi_1(x) = 1$ if doc has word 'is', 0 else
- $\phi_2(x) = 1$ if doc has word 'fast', 0 else
- $\phi_3(x) = 1$ if doc has word 'politicians', 0 else
- $\phi_4(x) = 1$ if doc has word 'talk', 0 else
- $\phi_5(x) = 1$ if doc has word 'washington', 0 else
- $\phi_6(x) = 1$ if doc has word 'sleazy', 0 else

Your turn? What is $P(\text{sports})$? What is $P(\phi_0(x) = 1|\text{politics})$?

Naive Bayes Document Classification

- doc 1: $y_1 =$ sports, "hockey is fast"
- doc 2: $y_2 =$ politics, "politicians talk fast"
- doc 3: $y_3 =$ politics, "washington is sleazy"

- $\phi_0(x) = 1$ if doc has word 'hockey', 0 else
- $\phi_1(x) = 1$ if doc has word 'is', 0 else
- $\phi_2(x) = 1$ if doc has word 'fast', 0 else
- $\phi_3(x) = 1$ if doc has word 'politicians', 0 else
- $\phi_4(x) = 1$ if doc has word 'talk', 0 else
- $\phi_5(x) = 1$ if doc has word 'washington', 0 else
- $\phi_6(x) = 1$ if doc has word 'sleazy', 0 else

Your turn? What is $P(\text{sports})$? $1/3$ What is $P(\phi_0(x) = 1|\text{politics})$?

Naive Bayes Document Classification

- doc 1: $y_1 =$ sports, "hockey is fast"
- doc 2: $y_2 =$ politics, "politicians talk fast"
- doc 3: $y_3 =$ politics, "washington is sleazy"

- $\phi_0(x) = 1$ if doc has word 'hockey', 0 else
- $\phi_1(x) = 1$ if doc has word 'is', 0 else
- $\phi_2(x) = 1$ if doc has word 'fast', 0 else
- $\phi_3(x) = 1$ if doc has word 'politicians', 0 else
- $\phi_4(x) = 1$ if doc has word 'talk', 0 else
- $\phi_5(x) = 1$ if doc has word 'washington', 0 else
- $\phi_6(x) = 1$ if doc has word 'sleazy', 0 else

Your turn? What is $P(\text{sports})$? $1/3$ What is $P(\phi_0(x) = 1|\text{politics})$? 0

Deriving MLE

$$\mathcal{P} = \arg\max_{\mathcal{P}} \prod_{t=1}^{|\mathcal{T}|} \left(P(\boldsymbol{y}_t) \prod_{i=1}^{m} P(\boldsymbol{\phi}_i(\boldsymbol{x}_t)|\boldsymbol{y}_t) \right)$$

Deriving MLE

$$
\begin{aligned}
\mathcal{P} &= \arg\max_{\mathcal{P}} \prod_{t=1}^{|\mathcal{T}|} \left(P(\boldsymbol{y}_t) \prod_{i=1}^{m} P(\boldsymbol{\phi}_i(\boldsymbol{x}_t)|\boldsymbol{y}_t) \right) \\
&= \arg\max_{\mathcal{P}} \sum_{t=1}^{|\mathcal{T}|} \left(\log P(\boldsymbol{y}_t) + \sum_{i=1}^{m} \log P(\boldsymbol{\phi}_i(\boldsymbol{x}_t)|\boldsymbol{y}_t) \right) \\
&= \arg\max_{P(\boldsymbol{y})} \sum_{t=1}^{|\mathcal{T}|} \log P(\boldsymbol{y}_t) + \arg\max_{P(\boldsymbol{\phi}_i(\boldsymbol{x})|\boldsymbol{y})} \sum_{t=1}^{|\mathcal{T}|} \sum_{i=1}^{m} \log P(\boldsymbol{\phi}_i(\boldsymbol{x}_t)|\boldsymbol{y}_t)
\end{aligned}
$$

such that $\sum_{\boldsymbol{y}} P(\boldsymbol{y}) = 1$, $\sum_{j=1}^{F_i} P(\boldsymbol{\phi}_i(\boldsymbol{x}) = j|\boldsymbol{y}) = 1$, $P(\cdot) \geq 0$

Deriving MLE

$$\mathcal{P} = \arg\max_{P(\boldsymbol{y})} \sum_{t=1}^{|\mathcal{T}|} \log P(\boldsymbol{y}_t) + \arg\max_{P(\phi_i(\boldsymbol{x})|\boldsymbol{y})} \sum_{t=1}^{|\mathcal{T}|} \sum_{i=1}^{m} \log P(\phi_i(\boldsymbol{x}_t)|\boldsymbol{y}_t)$$

Both optimizations are of the form

$$\arg\max_P \sum_v \text{count}(v) \log P(v), \text{ s.t., } \sum_v P(v) = 1,\ P(v) \geq 0$$

where v is event in $\mathcal{T}$, e.g. $(\boldsymbol{y}_t = \boldsymbol{y})$ or
$(\phi_i(\boldsymbol{x}_t) = \phi_i(\boldsymbol{x})$ and $\boldsymbol{y}_t = \boldsymbol{y})$

Deriving MLE

$$\arg\max_P \sum_v \text{count}(v) \log P(v)$$
$$\text{s.t., } \sum_v P(v) = 1,\ P(v) \geq 0$$

Introduce Lagrangian multiplier λ, optimization becomes

$$\arg\max_{P,\lambda} \ \sum_v \text{count}(v) \log P(v) - \lambda \left(\sum_v P(v) - 1\right)$$

- Derivative:
- Set to zero:
- Final solution:

Deriving MLE

$$\arg\max_P \sum_v \text{count}(v) \log P(v)$$
$$\text{s.t., } \sum_v P(v) = 1,\ P(v) \geq 0$$

Introduce Lagrangian multiplier λ, optimization becomes

$$\arg\max_{P,\lambda} \sum_v \text{count}(v) \log P(v) - \lambda \left(\sum_v P(v) - 1\right)$$

- Derivative w.r.t $P(v)$ is $\frac{\text{count(v)}}{P(v)} - \lambda$
- Setting this to zero $P(v) = \frac{\text{count(v)}}{\lambda}$
- Combine with $\sum_v P(v) = 1.\ P(v) \geq 0$, then $P(v) = \frac{\text{count(v)}}{\sum_{v'} \text{count}(v')}$

Deriving MLE

Reinstantiate events v in $\mathcal{T}$:

$$P(\boldsymbol{y}) = \frac{\sum_{t=1}^{|\mathcal{T}|} [\![\boldsymbol{y}_t = \boldsymbol{y}]\!]}{|\mathcal{T}|}$$

$$P(\phi_i(\boldsymbol{x})|\boldsymbol{y}) = \frac{\sum_{t=1}^{|\mathcal{T}|} [\![\phi_i(\boldsymbol{x}_t) = \phi_i(\boldsymbol{x}) \text{ and } \boldsymbol{y}_t = \boldsymbol{y}]\!]}{\sum_{t=1}^{|\mathcal{T}|} [\![\boldsymbol{y}_t = \boldsymbol{y}]\!]}$$

Naive Bayes is a linear model

- Let $\omega_y = \log P(y), \forall y \in \mathcal{Y}$
- Let $\omega_{\phi_i(x),y} = \log P(\phi_i(x)|y), \forall y \in \mathcal{Y}, \phi_i(x) \in \{1, \ldots, F_i\}$
- Let ω be set of all ω_* and $\omega_{*,*}$

$$\arg\max_y P(y|\phi(x)) \quad \propto \quad \arg\max_y P(\phi(x), y) = \arg\max_y P(y) \prod_{i=1}^{m} P(\phi_i(x)|y) =$$

where $\psi_* \in \{0, 1\}$, $\psi_{i,j}(x) = [\![\phi_i(x) = j]\!]$, $\psi_{y'}(y) = [\![y = y']\!]$

Naive Bayes is a linear model

- Let $\omega_y = \log P(y)$, $\forall y \in \mathcal{Y}$
- Let $\omega_{\phi_i(x),y} = \log P(\phi_i(x)|y)$, $\forall y \in \mathcal{Y}, \phi_i(x) \in \{1, \ldots, F_i\}$
- Let ω be set of all ω_* and $\omega_{*,*}$

$$\begin{aligned}
\arg\max_y \; P(y|\phi(x)) \quad &\propto \quad \arg\max_y \; P(\phi(x), y) = \arg\max_y \; P(y) \prod_{i=1}^{m} P(\phi_i(x)|y) \\
&= \quad \arg\max_y \; \log P(y) + \sum_{i=1}^{m} \log P(\phi_i(x)|y) \\
&= \quad \arg\max_y \; \omega_y + \sum_{i=1}^{m} \omega_{\phi_i(x),y} \\
&= \quad \arg\max_y \; \sum_{y'} \omega_y \psi_{y'}(y) + \sum_{i=1}^{m} \sum_{j=1}^{F_i} \omega_{\phi_i(x),y} \psi_{i,j}(x)
\end{aligned}$$

where $\psi_* \in \{0, 1\}$, $\psi_{i,j}(x) = [\![\phi_i(x) = j]\!]$, $\psi_{y'}(y) = [\![y = y']\!]$

Smoothing

- doc 1: $y_1 =$ sports, "hockey is fast"
- doc 2: $y_2 =$ politics, "politicians talk fast"
- doc 3: $y_3 =$ politics, "washington is sleazy"

- New doc: "washington hockey is fast"
- Both 'sports' and 'politics' have probabilities of 0

- Smoothing aims to assign a small amount of probability to unseen events
- E.g., Additive/Laplacian smoothing

$$P(v) = \frac{\text{count}(v)}{\sum_{v'} \text{count}(v')} \implies P(v) = \frac{\text{count}(v) + \alpha}{\sum_{v'} (\text{count}(v') + \alpha)}$$

Discriminative versus Generative

- Generative models attempt to model inputs and outputs
  - e.g., Naive Bayes = MLE of joint distribution $P(\boldsymbol{x}, \boldsymbol{y})$
  - Statistical model must explain generation of input
- Occam's Razor: "Among competing hypotheses, the one with the fewest assumptions should be selected"
- Discriminative models
  - Use $\mathcal{L}$ that directly optimizes $P(\boldsymbol{y}|\boldsymbol{x})$ (or something related)
  - Logistic Regression – MLE of $P(\boldsymbol{y}|\boldsymbol{x})$
  - Perceptron and SVMs – minimize classification error
- Generative and discriminative models use $P(\boldsymbol{y}|\boldsymbol{x})$ for prediction
- Differ only on what distribution they use to set $\boldsymbol{\omega}$

Logistic Regression

Logistic Regression

Define a conditional probability:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{e^{\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y})}}{Z_{\boldsymbol{x}}}, \qquad \text{where } Z_{\boldsymbol{x}} = \sum_{\boldsymbol{y}'\in\mathcal{Y}} e^{\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y}')}$$

Note: still a linear model

$$\begin{aligned}
\arg\max_{\boldsymbol{y}} \; P(\boldsymbol{y}|\boldsymbol{x}) &= \arg\max_{\boldsymbol{y}} \; \frac{e^{\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y})}}{Z_{\boldsymbol{x}}} \\
&= \arg\max_{\boldsymbol{y}} \; e^{\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y})} \\
&= \arg\max_{\boldsymbol{y}} \; \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y})
\end{aligned}$$

Logistic Regression

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{e^{\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y})}}{Z_{\boldsymbol{x}}}$$

- Q: How do we learn weights $\boldsymbol{\omega}$
- A: Set weights to maximize log-likelihood of training data:

$$\begin{aligned}\boldsymbol{\omega} &= \arg\max_{\boldsymbol{\omega}} \mathcal{L}(\mathcal{T};\boldsymbol{\omega}) \\ &= \arg\max_{\boldsymbol{\omega}} \prod_{t=1}^{|\mathcal{T}|} P(\boldsymbol{y}_t|\boldsymbol{x}_t) = \arg\max_{\boldsymbol{\omega}} \sum_{t=1}^{|\mathcal{T}|} \log P(\boldsymbol{y}_t|\boldsymbol{x}_t)\end{aligned}$$

- In a nutshell we set the weights $\boldsymbol{\omega}$ so that we assign as much probability to the correct label $\boldsymbol{y}$ for each $\boldsymbol{x}$ in the training set

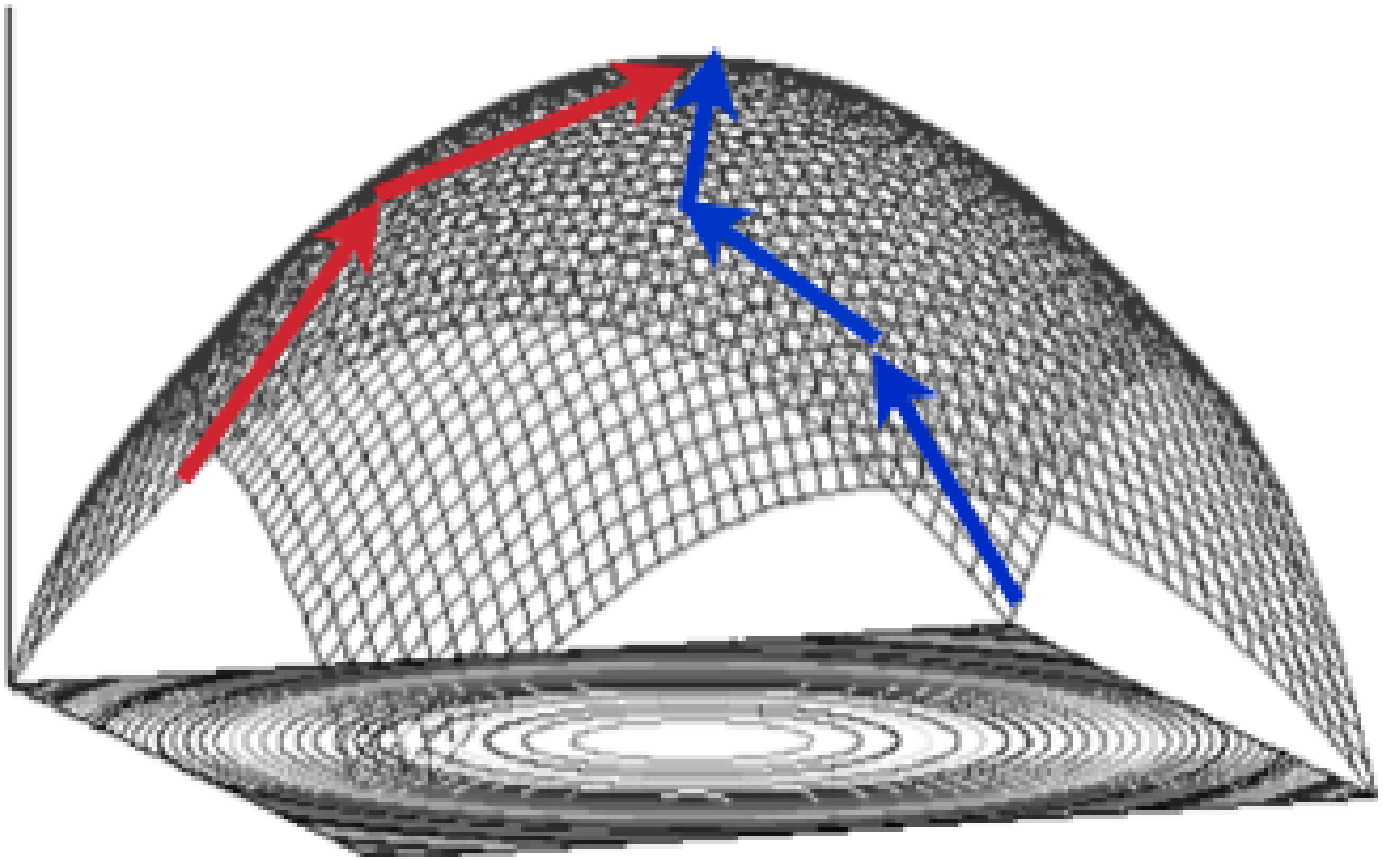
Logistic Regression

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{e^{\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y})}}{Z_{\boldsymbol{x}}}, \qquad \text{where } Z_{\boldsymbol{x}} = \sum_{\boldsymbol{y}'\in\mathcal{Y}} e^{\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y}')}$$

$$\boldsymbol{\omega} = \arg\max_{\boldsymbol{\omega}} \sum_{t=1}^{|\mathcal{T}|} \log P(\boldsymbol{y}_t|\boldsymbol{x}_t) \ (*)$$

- The objective function (*) is concave
- Therefore there is a global maximum
- No closed form solution, but lots of numerical techniques
  - Gradient methods (gradient ascent, conjugate gradient, iterative scaling)
  - Newton methods (limited-memory quasi-newton)

Gradient Ascent



Gradient Ascent

- Let $\mathcal{L}(\mathcal{T};\boldsymbol{\omega}) = \sum_{t=1}^{|\mathcal{T}|} \log\left(e^{\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t)}/Z_{\boldsymbol{x}}\right)$
- Want to find $\arg\max_{\boldsymbol{\omega}} \mathcal{L}(\mathcal{T};\boldsymbol{\omega})$
  - Set $\boldsymbol{\omega}^0 = O^m$
  - Iterate until convergence

$$\boldsymbol{\omega}^i = \boldsymbol{\omega}^{i-1} + \alpha\nabla\mathcal{L}(\mathcal{T};\boldsymbol{\omega}^{i-1})$$

- $\alpha > 0$ and set so that $\mathcal{L}(\mathcal{T};\boldsymbol{\omega}^i) > \mathcal{L}(\mathcal{T};\boldsymbol{\omega}^{i-1})$
- $\nabla\mathcal{L}(\mathcal{T};\boldsymbol{\omega})$ is gradient of $\mathcal{L}$ w.r.t. $\boldsymbol{\omega}$
  - A gradient is all partial derivatives over variables w_i
  - i.e., $\nabla\mathcal{L}(\mathcal{T};\boldsymbol{\omega}) = (\frac{\partial}{\partial\boldsymbol{\omega}_0}\mathcal{L}(\mathcal{T};\boldsymbol{\omega}), \frac{\partial}{\partial\boldsymbol{\omega}_1}\mathcal{L}(\mathcal{T};\boldsymbol{\omega}), \ldots, \frac{\partial}{\partial\boldsymbol{\omega}_m}\mathcal{L}(\mathcal{T};\boldsymbol{\omega}))$
- Gradient ascent will always find $\boldsymbol{\omega}$ to maximize $\mathcal{L}$

Gradient Descent

- Let $\mathcal{L}(\mathcal{T};\boldsymbol{\omega}) = -\sum_{t=1}^{|\mathcal{T}|} \log\left(e^{\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t)}/Z_{\boldsymbol{x}}\right)$
- Want to find $\arg\min_{\boldsymbol{\omega}} \mathcal{L}(\mathcal{T};\boldsymbol{\omega})$
  - Set $\boldsymbol{\omega}^0 = O^m$
  - Iterate until convergence

$$\boldsymbol{\omega}^i = \boldsymbol{\omega}^{i-1} - \alpha\nabla\mathcal{L}(\mathcal{T};\boldsymbol{\omega}^{i-1})$$

- $\alpha > 0$ and set so that $\mathcal{L}(\mathcal{T};\boldsymbol{\omega}^i) < \mathcal{L}(\mathcal{T};\boldsymbol{\omega}^{i-1})$
- $\nabla\mathcal{L}(\mathcal{T};\boldsymbol{\omega})$ is gradient of $\mathcal{L}$ w.r.t. $\boldsymbol{\omega}$
  - A gradient is all partial derivatives over variables w_i
  - i.e., $\nabla\mathcal{L}(\mathcal{T};\boldsymbol{\omega}) = (\frac{\partial}{\partial\omega_0}\mathcal{L}(\mathcal{T};\boldsymbol{\omega}), \frac{\partial}{\partial\omega_1}\mathcal{L}(\mathcal{T};\boldsymbol{\omega}), \ldots, \frac{\partial}{\partial\omega_m}\mathcal{L}(\mathcal{T};\boldsymbol{\omega}))$
- Gradient descent will always find $\boldsymbol{\omega}$ to minimize $\mathcal{L}$

The partial derivatives

- Need to find all partial derivatives $\frac{\partial}{\partial \boldsymbol{\omega}_i} \mathcal{L}(\mathcal{T}; \boldsymbol{\omega})$

$$
\begin{aligned}
\mathcal{L}(\mathcal{T}; \boldsymbol{\omega}) &= \sum_t \log P(\boldsymbol{y}_t | \boldsymbol{x}_t) \\
&= \sum_t \log \frac{e^{\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t)}}{\sum_{\boldsymbol{y}' \in \mathcal{Y}} e^{\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')}} \\
&= \sum_t \log \frac{e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}_t)}}{Z_{\boldsymbol{x}_t}}
\end{aligned}
$$

Partial derivatives - some reminders

1. $\frac{\partial}{\partial x} \log F = \frac{1}{F} \frac{\partial}{\partial x} F$
   - We always assume log is the natural logarithm $\log_e$
2. $\frac{\partial}{\partial x} e^F = e^F \frac{\partial}{\partial x} F$
3. $\frac{\partial}{\partial x} \sum_t F_t = \sum_t \frac{\partial}{\partial x} F_t$
4. $\frac{\partial}{\partial x} \frac{F}{G} = \frac{G \frac{\partial}{\partial x} F - F \frac{\partial}{\partial x} G}{G^2}$

The partial derivatives

$$\frac{\partial}{\partial \boldsymbol{\omega}_i} \mathcal{L}(\mathcal{T}; \boldsymbol{\omega}) =$$

The partial derivatives (1)

$$
\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\omega}_i}\mathcal{L}(\mathcal{T};\boldsymbol{\omega}) &= \frac{\partial}{\partial \boldsymbol{\omega}_i}\sum_t \log \frac{e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t,\boldsymbol{y}_t)}}{Z_{\boldsymbol{x}_t}} \\
&= \sum_t \frac{\partial}{\partial \boldsymbol{\omega}_i} \log \frac{e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t,\boldsymbol{y}_t)}}{Z_{\boldsymbol{x}_t}} \\
&= \sum_t \left(\frac{Z_{\boldsymbol{x}_t}}{e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t,\boldsymbol{y}_t)}}\right)\left(\frac{\partial}{\partial \boldsymbol{\omega}_i}\frac{e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t,\boldsymbol{y}_t)}}{Z_{\boldsymbol{x}_t}}\right)
\end{aligned}
$$

The partial derivatives

Now, $\frac{\partial}{\partial \boldsymbol{\omega}_i} \frac{e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}_t)}}{Z_{\boldsymbol{x}_t}} =$

The partial derivatives (2)

Now,

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\omega}_i} \frac{e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}_t)}}{Z_{\boldsymbol{x}_t}} &= \frac{Z_{\boldsymbol{x}_t} \frac{\partial}{\partial \boldsymbol{\omega}_i} e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}_t)} - e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}_t)} \frac{\partial}{\partial \boldsymbol{\omega}_i} Z_{\boldsymbol{x}_t}}{Z_{\boldsymbol{x}_t}^2} \\
&= \frac{Z_{\boldsymbol{x}_t} e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}_t)} \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}_t) - e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}_t)} \frac{\partial}{\partial \boldsymbol{\omega}_i} Z_{\boldsymbol{x}_t}}{Z_{\boldsymbol{x}_t}^2} \\
&= \frac{e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}_t)}}{Z_{\boldsymbol{x}_t}^2} (Z_{\boldsymbol{x}_t} \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}_t) - \frac{\partial}{\partial \boldsymbol{\omega}_i} Z_{\boldsymbol{x}_t}) \\
&= \frac{e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}_t)}}{Z_{\boldsymbol{x}_t}^2} (Z_{\boldsymbol{x}_t} \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}_t) \\
&\qquad - \sum_{\boldsymbol{y}' \in \mathcal{Y}} e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}')} \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}'))
\end{aligned}$$

because

$$\frac{\partial}{\partial \boldsymbol{\omega}_i} Z_{\boldsymbol{x}_t} = \frac{\partial}{\partial \boldsymbol{\omega}_i} \sum_{\boldsymbol{y}' \in \mathcal{Y}} e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}')} = \sum_{\boldsymbol{y}' \in \mathcal{Y}} e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}')} \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}')$$

The partial derivatives

The partial derivatives (3)

From (2),

$$\frac{\partial}{\partial \boldsymbol{\omega}_i} \frac{e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}_t)}}{Z_{\boldsymbol{x}_t}} = \frac{e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}_t)}}{Z_{\boldsymbol{x}_t}^2} (Z_{\boldsymbol{x}_t} \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}_t) - \sum_{\boldsymbol{y}' \in \mathcal{Y}} e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}')} \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}'))$$

Sub this in (1),

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\omega}_i} \mathcal{L}(\mathcal{T}; \boldsymbol{\omega}) &= \sum_t \left(\frac{Z_{\boldsymbol{x}_t}}{e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}_t)}}\right)\left(\frac{\partial}{\partial \boldsymbol{\omega}_i} \frac{e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}_t)}}{Z_{\boldsymbol{x}_t}}\right) \\
&= \sum_t \frac{1}{Z_{x_t}} \left(Z_{\boldsymbol{x}_t} \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}_t) - \sum_{\boldsymbol{y}' \in \mathcal{Y}} e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}')} \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}')\right) \\
&= \sum_t \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}_t) - \sum_t \sum_{\boldsymbol{y}' \in \mathcal{Y}} \frac{e^{\sum_j \boldsymbol{\omega}_j \times \boldsymbol{\phi}_j(\boldsymbol{x}_t, \boldsymbol{y}')}}{Z_{x_t}} \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}') \\
&= \sum_t \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}_t) - \sum_t \sum_{\boldsymbol{y}' \in \mathcal{Y}} P(\boldsymbol{y}' | \boldsymbol{x}_t) \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}')
\end{aligned}$$

FINALLY!!!

- After all that,

$$\frac{\partial}{\partial \omega_i}\mathcal{L}(\mathcal{T};\boldsymbol{\omega}) = \sum_t \phi_i(\boldsymbol{x}_t, \boldsymbol{y}_t) - \sum_t \sum_{\boldsymbol{y}' \in \mathcal{Y}} P(\boldsymbol{y}'|\boldsymbol{x}_t)\phi_i(\boldsymbol{x}_t, \boldsymbol{y}')$$

- And the gradient is:

$$\nabla\mathcal{L}(\mathcal{T};\boldsymbol{\omega}) = \left(\frac{\partial}{\partial \omega_0}\mathcal{L}(\mathcal{T};\boldsymbol{\omega}), \frac{\partial}{\partial \omega_1}\mathcal{L}(\mathcal{T};\boldsymbol{\omega}), \ldots, \frac{\partial}{\partial \omega_m}\mathcal{L}(\mathcal{T};\boldsymbol{\omega})\right)$$

- So we can now use gradient ascent to find $\boldsymbol{\omega}$!!

Logistic Regression Summary

- Define conditional probability

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{e^{\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{y})}}{Z_{\boldsymbol{x}}}$$

- Set weights to maximize log-likelihood of training data:

$$\boldsymbol{\omega} = \arg\max_{\boldsymbol{\omega}} \sum_{t} \log P(\boldsymbol{y}_t|\boldsymbol{x}_t)$$

- Can find the gradient and run gradient ascent (or any gradient-based optimization algorithm)

$$\frac{\partial}{\partial \boldsymbol{\omega}_i}\mathcal{L}(\mathcal{T};\boldsymbol{\omega}) = \sum_{t} \boldsymbol{\phi}_i(\boldsymbol{x}_t,\boldsymbol{y}_t) - \sum_{t}\sum_{\boldsymbol{y}'\in\mathcal{Y}} P(\boldsymbol{y}'|\boldsymbol{x}_t)\boldsymbol{\phi}_i(\boldsymbol{x}_t,\boldsymbol{y}')$$

Logistic Regression = Maximum Entropy

- Well-known equivalence
- Max Ent: maximize entropy subject to constraints on features: $P = \arg\max_P H(P)$ under constraints
  - Empirical feature counts must equal expected counts
- Quick intuition
  - Partial derivative in logistic regression

$$\frac{\partial}{\partial \boldsymbol{\omega}_i} \mathcal{L}(\mathcal{T}; \boldsymbol{\omega}) = \sum_t \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}_t) - \sum_t \sum_{\boldsymbol{y}' \in \mathcal{Y}} P(\boldsymbol{y}'|\boldsymbol{x}_t) \boldsymbol{\phi}_i(\boldsymbol{x}_t, \boldsymbol{y}')$$

  - First term is empirical feature counts and second term is expected counts
  - Derivative set to zero maximizes function
  - Therefore when both counts are equivalent, we optimize the logistic regression objective!

Perceptron

Perceptron

- Choose a $\boldsymbol{\omega}$ that minimizes error

$$\mathcal{L}(\mathcal{T};\boldsymbol{\omega}) = \sum_{t=1}^{|\mathcal{T}|} 1 - [\![\boldsymbol{y}_t = \arg\max_{\boldsymbol{y}} \ \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y})]\!]$$

$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega}} \sum_{t=1}^{|\mathcal{T}|} 1 - [\![\boldsymbol{y}_t = \arg\max_{\boldsymbol{y}} \ \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y})]\!]$$

$$[\![p]\!] = \begin{cases} 1 & p \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

- This is a 0-1 loss function
  - We will see later how to formalize the perceptron error function as hinge-loss

Perceptron Learning Algorithm

Training data: $\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{|\mathcal{T}|}$

1. $\boldsymbol{\omega}^{(0)} = 0;\ i = 0$
2. for $n : 1..N$
3. for $t : 1..T$
4. Let $\boldsymbol{y}' = \arg\max_{\boldsymbol{y}'} \boldsymbol{\omega}^{(i)} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$
5. if $\boldsymbol{y}' \neq \boldsymbol{y}_t$
6. $\boldsymbol{\omega}^{(i+1)} = \boldsymbol{\omega}^{(i)} + \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$
7. $i = i + 1$
8. return $\boldsymbol{\omega}^{i}$

Perceptron: Separability and Margin

- Given an training instance $(\boldsymbol{x}_t, \boldsymbol{y}_t)$, define:
  - $\bar{\mathcal{Y}}_t = \mathcal{Y} - \{\boldsymbol{y}_t\}$
  - i.e., $\bar{\mathcal{Y}}_t$ is the set of incorrect labels for $\boldsymbol{x}_t$
- A training set $\mathcal{T}$ is separable with margin $\gamma > 0$ if there exists a vector $\mathbf{u}$ with $\|\mathbf{u}\| = 1$ such that:

$$\mathbf{u} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \mathbf{u} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq \gamma \tag{2}$$

  for all $\boldsymbol{y}' \in \bar{\mathcal{Y}}_t$ and $||\mathbf{u}|| = \sqrt{\sum_j \mathbf{u}_j^2}$
- **Assumption**: the training set is separable with margin γ

Perceptron: Main Theorem

- **Theorem**: For any training set separable with a margin of γ, the following holds for the perceptron algorithm:

$$\text{mistakes made during training} \leq \frac{R^2}{\gamma^2}$$

where $R \geq ||\boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')||$ for all $(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$ and $\boldsymbol{y}' \in \bar{\mathcal{Y}}_t$

- Thus, after a finite number of training iterations, the error on the training set will converge to zero
- **Let's prove it!** (proof taken from [Collins 2002])

Perceptron Learning Algorithm

Training data: $\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{|\mathcal{T}|}$

1. $\boldsymbol{\omega}^{(0)} = 0$; $i = 0$
2. for $n : 1..N$
3. for $t : 1..T$
4. Let $\boldsymbol{y}' = \arg\max_{\boldsymbol{y}'} \boldsymbol{\omega}^{(i)} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$
5. if $\boldsymbol{y}' \neq \boldsymbol{y}_t$
6. $\boldsymbol{\omega}^{(i+1)} = \boldsymbol{\omega}^{(i)} + \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$
7. $i = i + 1$
8. return $\boldsymbol{\omega}^{i}$

- Lower bound:

$\boldsymbol{\omega}^{(k-1)}$ are weights before k^{th} error

Suppose k^{th} error made at $(\boldsymbol{x}_t, \boldsymbol{y}_t)$

$\boldsymbol{y}' = \arg\max_{\boldsymbol{y}'} \boldsymbol{\omega}^{(k-1)} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$

$\boldsymbol{y}' \neq \boldsymbol{y}_t$

$\boldsymbol{\omega}^{(k)} = \boldsymbol{\omega}^{(k-1)} + \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$

Perceptron Learning Algorithm

Training data: $\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{|\mathcal{T}|}$

1. $\boldsymbol{\omega}^{(0)} = 0$; $i = 0$
2. for $n : 1..N$
3. for $t : 1..T$
4. Let $\boldsymbol{y}' = \arg\max_{\boldsymbol{y}'} \boldsymbol{\omega}^{(i)} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$
5. if $\boldsymbol{y}' \neq \boldsymbol{y}_t$
6. $\boldsymbol{\omega}^{(i+1)} = \boldsymbol{\omega}^{(i)} + \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$
7. $i = i + 1$
8. return $\boldsymbol{\omega}^{i}$

- Lower bound:

$\boldsymbol{\omega}^{(k-1)}$ are weights before k^{th} error

Suppose k^{th} error made at $(\boldsymbol{x}_t, \boldsymbol{y}_t)$

$\boldsymbol{y}' = \arg\max_{\boldsymbol{y}'} \boldsymbol{\omega}^{(k-1)} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$

$\boldsymbol{y}' \neq \boldsymbol{y}_t$

$\boldsymbol{\omega}^{(k)} =$
$\boldsymbol{\omega}^{(k-1)} + \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$

$\mathbf{u} \cdot \boldsymbol{\omega}^{(k)} = \mathbf{u} \cdot \boldsymbol{\omega}^{(k-1)} + \mathbf{u} \cdot (\boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')) \geq \mathbf{u} \cdot \boldsymbol{\omega}^{(k-1)} + \gamma$, by (2)
Since $\boldsymbol{\omega}^{(0)} = 0$ and $\mathbf{u} \cdot \boldsymbol{\omega}^{(0)} = 0$, for all k: $\mathbf{u} \cdot \boldsymbol{\omega}^{(k)} \geq k\gamma$, by induction on k
Since $\mathbf{u} \cdot \boldsymbol{\omega}^{(k)} \leq ||\mathbf{u}|| \times ||\boldsymbol{\omega}^{(k)}||$, by the law of cosines, and $||\mathbf{u}|| = 1$, then $||\boldsymbol{\omega}^{(k)}|| \geq k\gamma$

Perceptron Learning Algorithm

Training data: $\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{|\mathcal{T}|}$

1. $\boldsymbol{\omega}^{(0)} = 0;\ i = 0$
2. for $n : 1..N$
3. for $t : 1..T$
4. Let $\boldsymbol{y}' = \arg\max_{\boldsymbol{y}'} \boldsymbol{\omega}^{(i)} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$
5. if $\boldsymbol{y}' \neq \boldsymbol{y}_t$
6. $\boldsymbol{\omega}^{(i+1)} = \boldsymbol{\omega}^{(i)} + \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$
7. $i = i + 1$
8. return $\boldsymbol{\omega}^{i}$

- Lower bound:

$\boldsymbol{\omega}^{(k-1)}$ are weights before k^{th} error

Suppose k^{th} error made at $(\boldsymbol{x}_t, \boldsymbol{y}_t)$

$\boldsymbol{y}' = \arg\max_{\boldsymbol{y}'} \boldsymbol{\omega}^{(k-1)} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$

$\boldsymbol{y}' \neq \boldsymbol{y}_t$

$\boldsymbol{\omega}^{(k)} =$
$\boldsymbol{\omega}^{(k-1)} + \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$

$\mathbf{u} \cdot \boldsymbol{\omega}^{(k)} = \mathbf{u} \cdot \boldsymbol{\omega}^{(k-1)} + \mathbf{u} \cdot (\boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')) \geq \mathbf{u} \cdot \boldsymbol{\omega}^{(k-1)} + \gamma$, by (2)
Since $\boldsymbol{\omega}^{(0)} = 0$ and $\mathbf{u} \cdot \boldsymbol{\omega}^{(0)} = 0$, for all k: $\mathbf{u} \cdot \boldsymbol{\omega}^{(k)} \geq k\gamma$, by induction on k
Since $\mathbf{u} \cdot \boldsymbol{\omega}^{(k)} \leq ||\mathbf{u}|| \times ||\boldsymbol{\omega}^{(k)}||$, by the law of cosines, and $||\mathbf{u}|| = 1$, then $||\boldsymbol{\omega}^{(k)}|| \geq k\gamma$

- Upper bound:

$$
\begin{aligned}
||\boldsymbol{\omega}^{(k)}||^2 &= ||\boldsymbol{\omega}^{(k-1)}||^2 + ||\boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')||^2 + 2\boldsymbol{\omega}^{(k-1)} \cdot (\boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')) \\
||\boldsymbol{\omega}^{(k)}||^2 &\leq ||\boldsymbol{\omega}^{(k-1)}||^2 + R^2, \text{ since } R \geq ||\boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')|| \\
&\qquad \text{and } \boldsymbol{\omega}^{(k-1)} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\omega}^{(k-1)} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \leq 0 \\
&\leq kR^2 \text{ for all } k, \text{ by induction on } k
\end{aligned}
$$

Perceptron Learning Algorithm

- We have just shown that $||\boldsymbol{\omega}^{(k)}|| \geq k\gamma$ and $||\boldsymbol{\omega}^{(k)}||^2 \leq kR^2$
- Therefore,
$$k^2\gamma^2 \leq ||\boldsymbol{\omega}^{(k)}||^2 \leq kR^2$$
- and solving for k
$$k \leq \frac{R^2}{\gamma^2}$$
- Therefore the number of errors is bounded!

Perceptron Summary

- Learns parameters of a linear model by minimizing error
- Guaranteed to find a $\boldsymbol{\omega}$ in a finite amount of time
- Perceptron is an example of an Online Learning Algorithm
  - $\boldsymbol{\omega}$ is updated based on a single training instance in isolation

$$\boldsymbol{\omega}^{(i+1)} = \boldsymbol{\omega}^{(i)} + \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$$

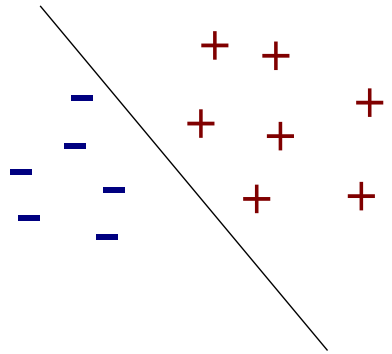
Averaged Perceptron

Training data: $\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{|\mathcal{T}|}$

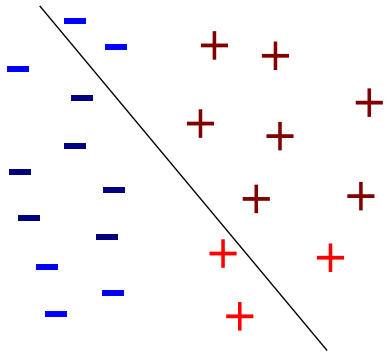
1. $\boldsymbol{\omega}^{(0)} = 0;\ i = 0$
2. for $n : 1..N$
3. for $t : 1..T$
4. Let $\boldsymbol{y}' = \arg\max_{\boldsymbol{y}'} \boldsymbol{\omega}^{(i)} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$
5. if $\boldsymbol{y}' \neq \boldsymbol{y}_t$
6. $\boldsymbol{\omega}^{(i+1)} = \boldsymbol{\omega}^{(i)} + \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')$
7. else
6. $\boldsymbol{\omega}^{(i+1)} = \boldsymbol{\omega}^{(i)}$
7. $i = i + 1$
8. return $\left(\sum_i \boldsymbol{\omega}^{(i)}\right) / (N \times T)$

Margin

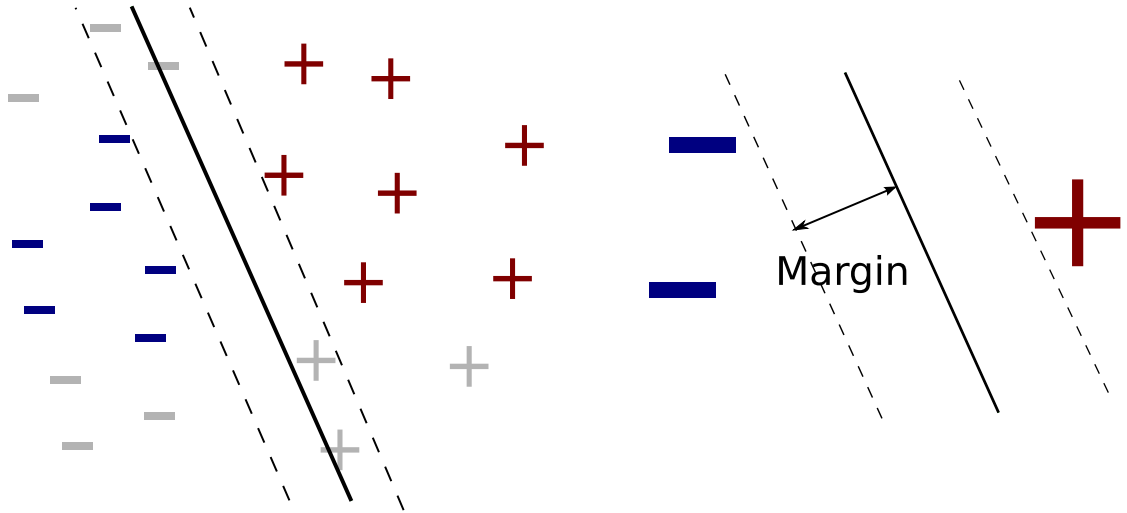
Training



Testing



Denote the value of the margin by γ



Maximizing Margin

- For a training set $\mathcal{T}$
- Margin of a weight vector $\boldsymbol{\omega}$ is smallest γ such that

$$\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq \gamma$$

- for every training instance $(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$, $\boldsymbol{y}' \in \bar{\mathcal{Y}}_t$

Maximizing Margin

- Intuitively maximizing margin makes sense
- More importantly, generalization error to unseen test data is proportional to the inverse of the margin

$$\epsilon \propto \frac{R^2}{\gamma^2 \times |\mathcal{T}|}$$

- **Perceptron**: we have shown that:
  - If a training set is separable by some margin, the perceptron will find a $\boldsymbol{\omega}$ that separates the data
  - However, the perceptron does not pick $\boldsymbol{\omega}$ to maximize the margin!

Support Vector Machines (SVMs)

Maximizing Margin

Let $\gamma > 0$

$$\max_{||\boldsymbol{\omega}||=1} \quad \gamma$$

such that:

$$\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq \gamma$$

$$\forall (\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$$

$$\text{and } \boldsymbol{y}' \in \bar{\mathcal{Y}}_t$$

- Note: algorithm still minimizes error if data is separable
- $||\boldsymbol{\omega}||$ is bound since scaling trivially produces larger margin

$$\beta(\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}')) \geq \beta\gamma, \text{ for some } \beta \geq 1$$

Max Margin = Min Norm

Let $\gamma > 0$

Max Margin:

$$\max_{||\boldsymbol{\omega}||=1} \quad \gamma$$

such that:

$$\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq \gamma$$

$$\forall (\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$$

$$\text{and } \boldsymbol{y}' \in \bar{\mathcal{Y}}_t$$

Max Margin = Min Norm

Let $\gamma > 0$

Max Margin:

$$\max_{||\boldsymbol{\omega}||=1} \quad \gamma$$

such that:

$$\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq \gamma$$

$$\forall (\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$$

$$\text{and } \boldsymbol{y}' \in \bar{\mathcal{Y}}_t$$

Change variables: $\mathbf{u} = \frac{\boldsymbol{\omega}}{\gamma}$?

$||\boldsymbol{\omega}|| = 1$ iff $||\mathbf{u}|| = 1/\gamma$,
then $\gamma = 1/||\mathbf{u}||$

Max Margin = Min Norm

Let $\gamma > 0$

Max Margin:

$$\max_{||\boldsymbol{\omega}||=1} \quad \gamma$$

such that:

$$\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq \gamma$$

$$\forall (\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$$

$$\text{and } \boldsymbol{y}' \in \bar{\mathcal{Y}}_t$$

Change variables: $\mathbf{u} = \dfrac{\boldsymbol{\omega}}{\gamma}$?

$||\boldsymbol{\omega}|| = 1$ iff $||\mathbf{u}|| = 1/\gamma$,
then $\gamma = 1/||\mathbf{u}||$

Min Norm (step 1):

$$\max_{\mathbf{u}} \quad \frac{1}{||\mathbf{u}||} = \min_{\mathbf{u}} ||\mathbf{u}||$$

such that:

$$\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq \gamma$$

$$\forall (\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$$

$$\text{and } \boldsymbol{y}' \in \bar{\mathcal{Y}}_t$$

Max Margin = Min Norm

Let $\gamma > 0$

Max Margin:

$$\max_{||\boldsymbol{\omega}||=1} \quad \gamma$$

such that:

$$\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq \gamma$$

$$\forall (\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$$

$$\text{and } \boldsymbol{y}' \in \bar{\mathcal{Y}}_t$$

Change variables: $\mathbf{u} = \dfrac{\boldsymbol{\omega}}{\gamma}$?

$||\boldsymbol{\omega}|| = 1$ iff $||\mathbf{u}|| = 1/\gamma$,
then $\gamma = 1/||\mathbf{u}||$

Min Norm (step 2):

$$\min_{\mathbf{u}} ||\mathbf{u}||$$

such that:

$$\gamma \mathbf{u} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \gamma \mathbf{u} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq \gamma$$

$$\forall (\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$$

$$\text{and } \boldsymbol{y}' \in \bar{\mathcal{Y}}_t$$

Max Margin = Min Norm

Let $\gamma > 0$

Max Margin:

$$\max_{||\boldsymbol{\omega}||=1} \quad \gamma$$

such that:

$$\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq \gamma$$

$$\forall (\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$$

$$\text{and } \boldsymbol{y}' \in \bar{\mathcal{Y}}_t$$

Change variables: $\mathbf{u} = \frac{\boldsymbol{\omega}}{\gamma}$?

$||\boldsymbol{\omega}|| = 1$ iff $||\mathbf{u}|| = 1/\gamma$,
then $\gamma = 1/||\mathbf{u}||$

Min Norm (step 2):

$$\min_{\mathbf{u}} ||\mathbf{u}||$$

such that:

$$\mathbf{u} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \mathbf{u} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq 1$$

$$\forall (\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$$

$$\text{and } \boldsymbol{y}' \in \bar{\mathcal{Y}}_t$$

Max Margin = Min Norm

Let $\gamma > 0$

Max Margin:

$$\max_{||\boldsymbol{\omega}||=1} \quad \gamma$$

such that:

$$\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq \gamma$$

$$\forall (\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$$

$$\text{and } \boldsymbol{y}' \in \bar{\mathcal{Y}}_t$$

Change variables: $\mathbf{u} = \frac{\boldsymbol{\omega}}{\gamma}$?

$||\boldsymbol{\omega}|| = 1$ iff $||\mathbf{u}|| = 1/\gamma$,
then $\gamma = 1/||\mathbf{u}||$

Min Norm (step 3):

$$\min_{\mathbf{u}} \quad \frac{1}{2}||\mathbf{u}||^2$$

such that:

$$\mathbf{u} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \mathbf{u} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq 1$$

$$\forall (\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$$

$$\text{and } \boldsymbol{y}' \in \bar{\mathcal{Y}}_t$$

Max Margin = Min Norm

Let $\gamma > 0$

Max Margin:

$$\max_{||\boldsymbol{\omega}||=1} \quad \gamma$$

such that:

$$\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq \gamma$$

$$\forall (\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$$

$$\text{and } \boldsymbol{y}' \in \bar{\mathcal{Y}}_t$$

Min Norm:

$$\min_{\mathbf{u}} \quad \frac{1}{2}||\mathbf{u}||^2$$

such that:

$$\mathbf{u} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \mathbf{u} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') \geq 1$$

$$\forall (\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$$

$$\text{and } \boldsymbol{y}' \in \bar{\mathcal{Y}}_t$$

- Intuition: Instead of fixing $||\boldsymbol{\omega}||$ we fix the margin $\gamma = 1$

Support Vector Machines

What if data is not separable? (Original problem: will not satisfy the constraints!)

$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega},\xi} \ \frac{1}{2}||\boldsymbol{\omega}||^2 + C\sum_{t=1}^{|\mathcal{T}|}\xi_t$$

such that:

$$\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t) - \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}') \geq 1-\xi_t \text{ and } \xi_t \geq 0$$

$$\forall(\boldsymbol{x}_t,\boldsymbol{y}_t)\in\mathcal{T} \text{ and } \boldsymbol{y}'\in\bar{\mathcal{Y}}_t$$

ξ_t: slack variable representing amount of constraint violation
If data is separable, optimal solution has $\xi_i = 0, \forall i$
C balances focus on margin ($C < \frac{1}{2}$) and on error ($C > \frac{1}{2}$)

Support Vector Machines

$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega},\xi} \ \frac{1}{2}||\boldsymbol{\omega}||^2 + C\sum_{t=1}^{|\mathcal{T}|}\xi_t$$

such that:

$$\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t) - \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}') \geq 1-\xi_t$$

where $\xi_t \geq 0$ and $\forall(\boldsymbol{x}_t,\boldsymbol{y}_t)\in\mathcal{T}$ and $\boldsymbol{y}'\in\bar{\mathcal{Y}}_t$

- Computing the dual form results in a quadratic programming problem – a well-known convex optimization problem

  [Boyd and Vandenberghe 2004]
- Can we have representation of this objective that allows more direct optimization?

Support Vector Machines

$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega},\xi} \ \frac{1}{2}||\boldsymbol{\omega}||^2 + C\sum_{t=1}^{|\mathcal{T}|}\xi_t$$

such that:

$$\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t) - \max_{\boldsymbol{y}'\neq\boldsymbol{y}_t} \ \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}') \geq 1-\xi_t$$

Support Vector Machines

$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega},\xi} \ \frac{1}{2}||\boldsymbol{\omega}||^2 + C\sum_{t=1}^{|\mathcal{T}|}\xi_t$$

such that:

$$\xi_t \geq 1 + \underbrace{\max_{\boldsymbol{y}' \neq \boldsymbol{y}_t} \ \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}') - \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t)}_{\textit{negated margin for example}}$$

Support Vector Machines

$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega},\xi} \ \frac{\lambda}{2}||\boldsymbol{\omega}||^2 + \sum_{t=1}^{|\mathcal{T}|} \xi_t \qquad \lambda = \frac{1}{C}$$

such that:

$$\xi_t \geq 1 + \underbrace{\max_{\boldsymbol{y}' \neq \boldsymbol{y}_t} \ \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t)}_{\textit{negated margin for example}}$$

Support Vector Machines

$$\xi_t \geq 1 + \underbrace{\max_{\boldsymbol{y}' \neq \boldsymbol{y}_t} \; \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t)}_{\textit{negated margin for example}}$$

- If $\|\boldsymbol{\omega}\|$ classifies $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ with margin 1, penalty $\xi_t = 0$
- Otherwise: $\xi_t = 1 + \max_{\boldsymbol{y}' \neq \boldsymbol{y}_t} \; \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t)$
- That means that in the end ξ_t will be:

$$\xi_t = \max\{0, 1 + \max_{\boldsymbol{y}' \neq \boldsymbol{y}_t} \; \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t)\}$$

Support Vector Machines

$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega},\xi} \frac{\lambda}{2}||\boldsymbol{\omega}||^2 + \sum_{t=1}^{|\mathcal{T}|} \xi_t \text{ s.t. } \xi_t \geq 1 + \max_{\boldsymbol{y}' \neq \boldsymbol{y}_t} \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t)$$

Hinge loss

$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega}} \mathcal{L}(\mathcal{T}; \boldsymbol{\omega}) = \arg\min_{\boldsymbol{\omega}} \sum_{t=1}^{|\mathcal{T}|} \mathit{loss}((\boldsymbol{x}_t, \boldsymbol{y}_t); \boldsymbol{\omega}) + \frac{\lambda}{2}||\boldsymbol{\omega}||^2$$

$$= \arg\min_{\boldsymbol{\omega}} \left(\sum_{t=1}^{|\mathcal{T}|} \max\left(0, 1 + \max_{\boldsymbol{y}' \neq \boldsymbol{y}_t} \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}') - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t)\right) \right) + \frac{\lambda}{2}||\boldsymbol{\omega}||^2$$

- We will see later how to do efficient optimization of hinge loss

Summary

What we have covered

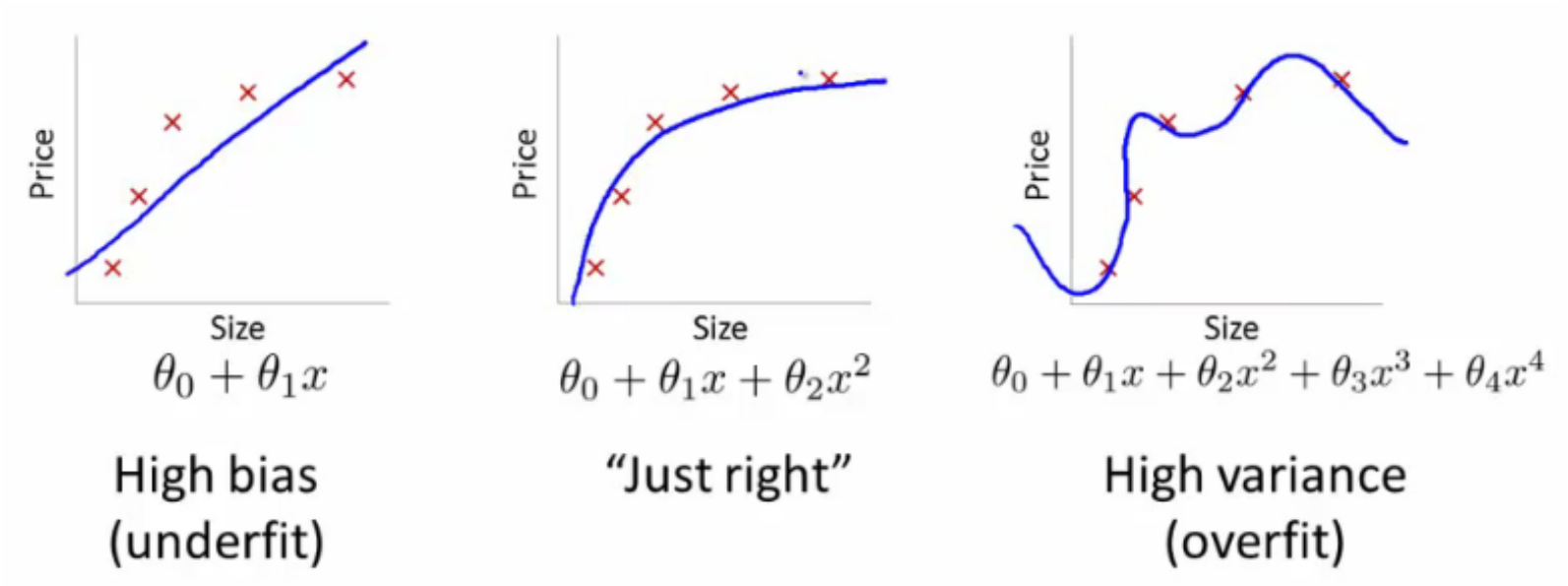
- Linear Learners
  - Naive Bayes
  - Logistic Regression
  - Perceptron
  - Support Vector Machines

What is next

- Regularization
- Online learning
- Non-linear models

Regularization

Fit of a Model



- Two sources of error:
  - Bias error, measures how well the hypothesis class fits the space we are trying to model
  - Variance error, measures sensitivity to training set selection
  - Want to balance these two things

Overfitting

- Early in lecture we made assumption data was i.i.d.
- Rarely is this true
  - E.g., syntactic analyzers typically trained on 40,000 sentences from early 1990s WSJ news text
- Even more common: $\mathcal{T}$ is very small
- This leads to overfitting
- E.g.: 'fake' is never a verb in WSJ treebank (only adjective)
  - High weight on "$\phi(\boldsymbol{x}, \boldsymbol{y}) = 1$ if $\boldsymbol{x}$=fake and $\boldsymbol{y}$=adjective"
  - Of course: leads to high log-likelihood / low error
- Other features might be more indicative
- Adjacent word identities: 'He wants to X his death' $\rightarrow$ X=verb

Regularization

- In practice, we regularize models to prevent overfitting

$$\arg\max_{\boldsymbol{\omega}} \; \mathcal{L}(\mathcal{T}; \boldsymbol{\omega}) - \lambda \mathcal{R}(\boldsymbol{\omega})$$

- Where $\mathcal{R}(\boldsymbol{\omega})$ is the regularization function
- λ controls how much to regularize
- Common functions
  - L2: $\mathcal{R}(\boldsymbol{\omega}) \propto \|\boldsymbol{\omega}\|_2 = \|\boldsymbol{\omega}\| = \sqrt{\sum_i \boldsymbol{\omega}_i^2}$ – smaller weights desired
  - L0: $\mathcal{R}(\boldsymbol{\omega}) \propto \|\boldsymbol{\omega}\|_0 = \sum_i [\![\boldsymbol{\omega}_i > 0]\!]$ – zero weights desired
    - Non-convex
    - Approximate with L1: $\mathcal{R}(\boldsymbol{\omega}) \propto \|\boldsymbol{\omega}\|_1 = \sum_i |\boldsymbol{\omega}_i|$

Logistic Regression with L2 Regularization

- Perhaps most common learner in NLP

$$\mathcal{L}(\mathcal{T};\boldsymbol{\omega}) - \lambda\mathcal{R}(\boldsymbol{\omega}) = \sum_{t=1}^{|\mathcal{T}|} \log\left(e^{\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t)}/Z_{\boldsymbol{x}}\right) - \frac{\lambda}{2}\|\boldsymbol{\omega}\|^2$$

- What are the new partial derivatives?

$$\frac{\partial}{\partial w_i}\mathcal{L}(\mathcal{T};\boldsymbol{\omega}) - \frac{\partial}{\partial w_i}\lambda\mathcal{R}(\boldsymbol{\omega})$$

- We know $\frac{\partial}{\partial w_i}\mathcal{L}(\mathcal{T};\boldsymbol{\omega})$
- Just need $\frac{\partial}{\partial w_i}\frac{\lambda}{2}\|\boldsymbol{\omega}\|^2 = \frac{\partial}{\partial w_i}\frac{\lambda}{2}\left(\sqrt{\sum_i \boldsymbol{\omega}_i^2}\right)^2 = \frac{\partial}{\partial w_i}\frac{\lambda}{2}\sum_i \boldsymbol{\omega}_i^2 = \lambda\boldsymbol{\omega}_i$

Support Vector Machines

- SVM in hinge-loss formulation: L2 regularization corresponds to margin maximization!

$$
\begin{aligned}
\boldsymbol{\omega} &= \arg\min_{\boldsymbol{\omega}} \; \mathcal{L}(\mathcal{T};\boldsymbol{\omega}) + \lambda\mathcal{R}(\boldsymbol{\omega}) \\
&= \arg\min_{\boldsymbol{\omega}} \; \sum_{t=1}^{|\mathcal{T}|} \mathit{loss}((\boldsymbol{x}_t,\boldsymbol{y}_t);\boldsymbol{\omega}) + \lambda\mathcal{R}(\boldsymbol{\omega}) \\
&= \arg\min_{\boldsymbol{\omega}} \; \sum_{t=1}^{|\mathcal{T}|} \max\left(0, 1 + \max_{\boldsymbol{y}\neq\boldsymbol{y}_t} \; \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}) - \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t)\right) + \lambda\mathcal{R}(\boldsymbol{\omega}) \\
&= \arg\min_{\boldsymbol{\omega}} \; \sum_{t=1}^{|\mathcal{T}|} \max\left(0, 1 + \max_{\boldsymbol{y}\neq\boldsymbol{y}_t} \; \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}) - \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t)\right) + \frac{\lambda}{2}\|\boldsymbol{\omega}\|^2
\end{aligned}
$$

SVMs vs. Logistic Regression

$$
\begin{aligned}
\boldsymbol{\omega} &= \arg\min_{\boldsymbol{\omega}} \; \mathcal{L}(\mathcal{T}; \boldsymbol{\omega}) + \lambda \mathcal{R}(\boldsymbol{\omega}) \\
&= \arg\min_{\boldsymbol{\omega}} \; \sum_{t=1}^{|\mathcal{T}|} \mathit{loss}((\boldsymbol{x}_t, \boldsymbol{y}_t); \boldsymbol{\omega}) + \lambda \mathcal{R}(\boldsymbol{\omega})
\end{aligned}
$$

SVMs vs. Logistic Regression

$$\begin{aligned}\boldsymbol{\omega} &= \arg\min_{\boldsymbol{\omega}} \; \mathcal{L}(\mathcal{T};\boldsymbol{\omega}) + \lambda\mathcal{R}(\boldsymbol{\omega}) \\ &= \arg\min_{\boldsymbol{\omega}} \; \sum_{t=1}^{|\mathcal{T}|} \mathit{loss}((\boldsymbol{x}_t,\boldsymbol{y}_t);\boldsymbol{\omega}) + \lambda\mathcal{R}(\boldsymbol{\omega})\end{aligned}$$

SVMs/hinge-loss: $\max\left(0, 1 + \max_{\boldsymbol{y}\neq\boldsymbol{y}_t}\left(\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}) - \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t)\right)\right)$

$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega}} \; \sum_{t=1}^{|\mathcal{T}|} \max\left(0, 1 + \max_{\boldsymbol{y}\neq\boldsymbol{y}_t} \; \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}) - \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t)\right) + \frac{\lambda}{2}\|\boldsymbol{\omega}\|^2$$

SVMs vs. Logistic Regression

$$\begin{aligned}\boldsymbol{\omega} &= \arg\min_{\boldsymbol{\omega}} \; \mathcal{L}(\mathcal{T};\boldsymbol{\omega}) + \lambda\mathcal{R}(\boldsymbol{\omega}) \\ &= \arg\min_{\boldsymbol{\omega}} \; \sum_{t=1}^{|\mathcal{T}|} \mathit{loss}((\boldsymbol{x}_t,\boldsymbol{y}_t);\boldsymbol{\omega}) + \lambda\mathcal{R}(\boldsymbol{\omega})\end{aligned}$$

SVMs/hinge-loss: $\max\left(0, 1 + \max_{\boldsymbol{y}\neq\boldsymbol{y}_t}\left(\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}) - \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t)\right)\right)$

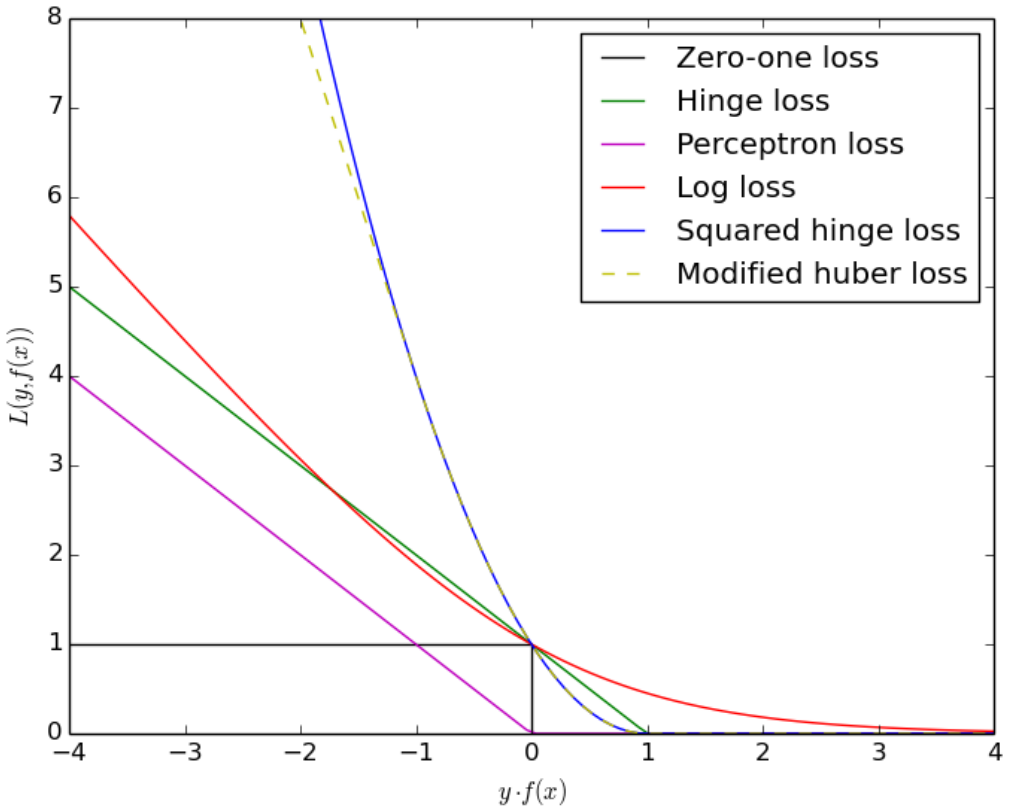
$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega}} \; \sum_{t=1}^{|\mathcal{T}|} \max\left(0, 1 + \max_{\boldsymbol{y}\neq\boldsymbol{y}_t} \; \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}) - \boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t)\right) + \frac{\lambda}{2}\|\boldsymbol{\omega}\|^2$$

Logistic Regression/log-loss: $-\log\left(e^{\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t)}/Z_{\boldsymbol{x}}\right)$

$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega}} \; \sum_{t=1}^{|\mathcal{T}|} -\log\left(e^{\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_t,\boldsymbol{y}_t)}/Z_{\boldsymbol{x}}\right) + \frac{\lambda}{2}\|\boldsymbol{\omega}\|^2$$

Generalized Linear Learners

$$\boldsymbol{\omega} = \arg\min_{\boldsymbol{\omega}} \; \mathcal{L}(\mathcal{T}; \boldsymbol{\omega}) + \lambda \mathcal{R}(\boldsymbol{\omega}) = \arg\min_{\boldsymbol{\omega}} \; \sum_{t=1}^{|\mathcal{T}|} \mathit{loss}((\boldsymbol{x}_t, \boldsymbol{y}_t); \boldsymbol{\omega}) + \lambda \mathcal{R}(\boldsymbol{\omega})$$



Which Learner to Use?

- Trial and error
- Training time available
- Choice of features is often more important

Online Learning

Online vs. Batch Learning

Batch($\mathcal{T}$);

- for 1 ... N
  - $\boldsymbol{\omega} \leftarrow$ update($\mathcal{T}$; $\boldsymbol{\omega}$)
- return $\boldsymbol{\omega}$

E.g., SVMs, logistic regression, Naive Bayes

Online($\mathcal{T}$);

- for 1 ... N
  - for $(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$
    - $\boldsymbol{\omega} \leftarrow$ update($(\boldsymbol{x}_t, \boldsymbol{y}_t)$; $\boldsymbol{\omega}$)
  - end for
- end for
- return $\boldsymbol{\omega}$

E.g., Perceptron

$$\boldsymbol{\omega} = \boldsymbol{\omega} + \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y})$$

Batch Gradient Descent

- Let $\mathcal{L}(\mathcal{T};\boldsymbol{\omega}) = \sum_{t=1}^{|\mathcal{T}|} loss((\boldsymbol{x}_t, \boldsymbol{y}_t);\boldsymbol{\omega})$
  - Set $\boldsymbol{\omega}^0 = O^m$
  - Iterate until convergence

$$\begin{aligned} \boldsymbol{\omega}^i &= \boldsymbol{\omega}^{i-1} - \alpha \nabla \mathcal{L}(\mathcal{T};\boldsymbol{\omega}^{i-1}) \\ &= \boldsymbol{\omega}^{i-1} - \sum_{t=1}^{|\mathcal{T}|} \alpha \nabla loss((\boldsymbol{x}_t, \boldsymbol{y}_t);\boldsymbol{\omega}^{i-1}) \end{aligned}$$

- $\alpha > 0$ and set so that $\mathcal{L}(\mathcal{T};\boldsymbol{\omega}^i) < \mathcal{L}(\mathcal{T};\boldsymbol{\omega}^{i-1})$

Stochastic Gradient Descent

- Stochastic Gradient Descent (SGD)
  - Approximate batch gradient $\nabla \mathcal{L}(\mathcal{T}; \boldsymbol{\omega})$ with stochastic gradient $\nabla \mathit{loss}((\boldsymbol{x}_t, \boldsymbol{y}_t); \boldsymbol{\omega})$
- Let $\mathcal{L}(\mathcal{T}; \boldsymbol{\omega}) = \sum_{t=1}^{|\mathcal{T}|} \mathit{loss}((\boldsymbol{x}_t, \boldsymbol{y}_t); \boldsymbol{\omega})$
  - Set $\boldsymbol{\omega}^0 = O^m$
  - iterate until convergence
    - sample $(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$ // "stochastic"
    - $\boldsymbol{\omega}^i = \boldsymbol{\omega}^{i-1} - \alpha \nabla \mathit{loss}((\boldsymbol{x}_t, \boldsymbol{y}_t); \boldsymbol{\omega}^{i-1})$
  - return $\boldsymbol{\omega}$

SGD over Finite Training Data

- NLP practice: Cycling over finite data set
  - Set $\boldsymbol{\omega}^0 = O^m$
  - for $1 \ldots N$
    - for $(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{T}$
    - $\boldsymbol{\omega}^i = \boldsymbol{\omega}^{i-1} - \alpha \nabla loss((\boldsymbol{x}_t, \boldsymbol{y}_t); \boldsymbol{\omega}^{i-1})$
  - return $\boldsymbol{\omega}$

Online Logistic Regression

- Stochastic Gradient Descent (SGD)
- $loss((\boldsymbol{x}_t, \boldsymbol{y}_t); \boldsymbol{\omega}) =$ log-loss
- $\nabla loss((\boldsymbol{x}_t, \boldsymbol{y}_t); \boldsymbol{\omega}) = \nabla \left(-\log \left(e^{\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t)} / Z_{\boldsymbol{x}_t} \right) \right)$
- From logistic regression section:

$$\nabla \left(-\log \left(e^{\boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t)} / Z_{\boldsymbol{x}_t} \right) \right) = - \left(\boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \sum_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}) \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) \right)$$

- Plus regularization term (if part of model)

Online SVMs

- Stochastic Gradient Descent (SGD)
- $\mathit{loss}((\boldsymbol{x}_t, \boldsymbol{y}_t); \boldsymbol{\omega}) =$ hinge-loss

$$\nabla \mathit{loss}((\boldsymbol{x}_t, \boldsymbol{y}_t); \boldsymbol{\omega}) = \nabla \left(\max\left(0, 1 + \max_{\boldsymbol{y} \neq \boldsymbol{y}_t} \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t)\right)\right)$$

- Subgradient is:

$$\nabla \left(\max\left(0, 1 + \max_{\boldsymbol{y} \neq \boldsymbol{y}_t} \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t)\right)\right)$$

$$= \begin{cases} 0, & \text{if } \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \max_{\boldsymbol{y}} \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) \geq 1 \\ \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t), & \text{otherwise, where } \boldsymbol{y} = \max_{\boldsymbol{y}} \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) \end{cases}$$

- Plus regularization term (required for SVMs)

Perceptron and Hinge-Loss

SVM subgradient update looks like perceptron update

$$\boldsymbol{\omega}^i = \boldsymbol{\omega}^{i-1} - \alpha \begin{cases} 0, & \text{if } \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \max_{\boldsymbol{y}} \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) \geq 1 \\ \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t), & \text{otherwise, where } \boldsymbol{y} = \max_{\boldsymbol{y}} \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) \end{cases}$$

Perceptron

$$\boldsymbol{\omega}^i = \boldsymbol{\omega}^{i-1} - \alpha \begin{cases} 0, & \text{if } \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \max_{\boldsymbol{y}} \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) \geq 0 \\ \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t), & \text{otherwise, where } \boldsymbol{y} = \max_{\boldsymbol{y}} \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) \end{cases}$$

where $\alpha = 1$, note $\boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t)$ not $\boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y})$ since '$-$' (descent)

Perceptron = SGD with no-margin hinge-loss

$$\max\left(0, 1 + \max_{\boldsymbol{y} \neq \boldsymbol{y}_t} \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) - \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t)\right)$$

Online vs. Batch Learning

- Online algorithms
  - Each update step relies only on the derivative for a single randomly chosen example
    - Computational cost of one step is $1/\mathcal{T}$ compared to batch
    - Easier to implement
  - Larger variance since each gradient is different
    - Variance slows down convergence
    - Requires fine-tuning of decaying learning rate
- Batch algorithms
  - Higher cost of averaging gradients over $\mathcal{T}$ for each update
    - Implementation more complex
    - Less fine-tuning, e.g., allows constant learning rates
    - Faster convergence

Variance-Reduced Online Learning

- SGD update extended by velocity vector $\boldsymbol{v}$ weighted by momentum coefficient $0 \leq \mu < 1$ [Polyak 1964]:
  - $$\boldsymbol{\omega}^{i+1} = \boldsymbol{\omega}^i - \alpha \nabla loss((\boldsymbol{x}_t, \boldsymbol{y}_t); \boldsymbol{\omega}^i) + \mu \boldsymbol{v}^i$$

    where

    $$\boldsymbol{v}^i = \boldsymbol{\omega}^i - \boldsymbol{\omega}^{i-1}$$

    - Momentum accelerates learning if gradients are aligned along same direction, and restricts changes when successive gradient are opposite of each other
    - General direction of gradient reinforced, perpendicular directions filtered out
- Best of both worlds: Efficient and effective!

Online-to-Batch Conversion

- Classical online learning:
  - data are given as an infinite sequence of input examples
  - model makes prediction on next example in sequence
  - online error is averaged over predictions after each update
- Standard NLP applications:
  - Finite set of training data, prediction on new batch of test data
  - Online learning applied by cycling over finite data
  - Online-to-batch conversion: Which model to use at test time?
    - Last model? Random model? Best model on heldout set?

Online-to-Batch Conversion by Averaging

- Averaged Perceptron
  - $\bar{\boldsymbol{\omega}} = \left(\sum_i \boldsymbol{\omega}^{(i)}\right) / (N \times T)$
  - Use weight vector averaged over online updates for prediction
- How does the perceptron mistake bound carry over to batch?
  - Let M_K be number of mistakes made during online learning, then with probability of at least $1 - \delta$:

$$\mathbb{E}[loss((\boldsymbol{x}, \boldsymbol{y}); \bar{\boldsymbol{\omega}}] \leq M_k + \sqrt{\frac{2}{k} \ln \frac{1}{\delta}}$$

  - $=$ generalization bound based on online performance [Cesa-Bianchi et al. 2004]
  - can be applied to all online learners with convex losses

Quick Summary

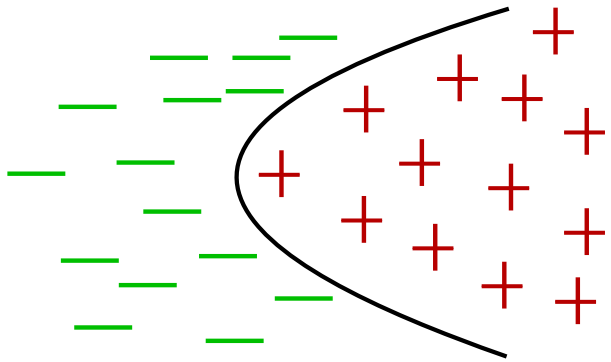
Linear Learners

- Naive Bayes, Perceptron, Logistic Regression and SVMs
- Generative vs. Discriminative
- Objective functions and loss functions
  - Log-loss, min error and hinge loss
  - Generalized linear learners
- Regularization
- Online vs. Batch learning

Non-Linear Models

Non-Linear Models

- Some data sets require more than a linear decision boundary to be correctly modeled
- Decision boundary is no longer a hyperplane in the feature space
- A lot of models out there
  - K-Nearest Neighbours
  - Decision Trees
  - Neural Networks
  - **Kernels**



Kernels

- A kernel is a similarity function between two points that is symmetric and positive semi-definite, which we denote by:

$$K(\boldsymbol{x}_t, \boldsymbol{x}_r) \in \mathbb{R}$$

- Let M be a $n \times n$ matrix such that ...

$$M_{t,r} = K(\boldsymbol{x}_t, \boldsymbol{x}_r)$$

- ... for any n points. Called the Gram matrix.
- Symmetric:

$$K(\boldsymbol{x}_t, \boldsymbol{x}_r) = K(\boldsymbol{x}_r, \boldsymbol{x}_t)$$

- Positive definite: positivity on diagonal

$$K(\boldsymbol{x}, \boldsymbol{x}) \geq 0 \text{ forall } \boldsymbol{x} \text{ with equality only for } \boldsymbol{x} = 0$$

Kernels

- **Mercer's Theorem**: for any kernel K, there exists an ϕ, in some $\mathbb{R}^d$, such that:

$$K(\boldsymbol{x}_t, \boldsymbol{x}_r) = \boldsymbol{\phi}(\boldsymbol{x}_t) \cdot \boldsymbol{\phi}(\boldsymbol{x}_r)$$

- Since our features are over pairs $(\boldsymbol{x}, \boldsymbol{y})$, we will write kernels over pairs

$$K((\boldsymbol{x}_t, \boldsymbol{y}_t), (\boldsymbol{x}_r, \boldsymbol{y}_r)) = \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) \cdot \boldsymbol{\phi}(\boldsymbol{x}_r, \boldsymbol{y}_r)$$

Kernel Trick: General Overview

- Define a kernel, and do not explicitly use dot product between vectors, only kernel calculations
- In some high-dimensional space, this corresponds to dot product
- In that space, the decision boundary is linear, but in the original space, we now have a non-linear decision boundary
- Let's do it for the Perceptron!

Kernel Trick – Perceptron Algorithm

Training data: $\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{|\mathcal{T}|}$

1. $\boldsymbol{\omega}^{(0)} = 0$; $i = 0$
2. for $n : 1..N$
3. for $t : 1..T$
4. Let $\boldsymbol{y} = \arg\max_{\boldsymbol{y}} \boldsymbol{\omega}^{(i)} \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y})$
5. if $\boldsymbol{y} \neq \boldsymbol{y}_t$
6. $\boldsymbol{\omega}^{(i+1)} = \boldsymbol{\omega}^{(i)} + \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y})$
7. $i = i + 1$
8. return $\boldsymbol{\omega}^i$

- Each feature function $\boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t)$ is added and $\boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y})$ is subtracted to $\boldsymbol{\omega}$ say $\alpha_{\boldsymbol{y},t}$ times
  - $\alpha_{\boldsymbol{y},t}$ is the # of times during learning label $\boldsymbol{y}$ is predicted for example t
- Thus,

$$\boldsymbol{\omega} = \sum_{t,\boldsymbol{y}} \alpha_{\boldsymbol{y},t} [\boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y})]$$

Kernel Trick – Perceptron Algorithm

- We can re-write the argmax function as:
  $\boldsymbol{y}* = \arg\max_{\boldsymbol{y}^*} \boldsymbol{\omega}^{(i)} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}^*)$

  $=$

  $=$

  $=$

- We can then re-write the perceptron algorithm strictly with kernels

Kernel Trick – Perceptron Algorithm

- We can re-write the argmax function as:

$$
\begin{aligned}
\boldsymbol{y}* &= \arg\max_{\boldsymbol{y}^*} \boldsymbol{\omega}^{(i)} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}^*) \\
&= \arg\max_{\boldsymbol{y}^*} \sum_{t,\boldsymbol{y}} \alpha_{\boldsymbol{y},t} [\boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y})] \cdot \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}^*) \\
&= \arg\max_{\boldsymbol{y}^*} \sum_{t,\boldsymbol{y}} \alpha_{\boldsymbol{y},t} [\boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}_t) \cdot \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}^*) - \boldsymbol{\phi}(\boldsymbol{x}_t, \boldsymbol{y}) \cdot \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}^*)] \\
&= \arg\max_{\boldsymbol{y}^*} \sum_{t,\boldsymbol{y}} \alpha_{\boldsymbol{y},t} [K((\boldsymbol{x}_t, \boldsymbol{y}_t), (\boldsymbol{x}_t, \boldsymbol{y}^*)) - K((\boldsymbol{x}_t, \boldsymbol{y}), (\boldsymbol{x}, \boldsymbol{y}^*))]
\end{aligned}
$$

- We can then re-write the perceptron algorithm strictly with kernels

Kernel Trick – Perceptron Algorithm

Training data: $\mathcal{T} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{|\mathcal{T}|}$

1. $\forall \boldsymbol{y}, t$ set $\alpha_{\boldsymbol{y},t} = 0$
2. for $n : 1..N$
3. for $t : 1..T$
4. Let $\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}^*} \sum_{t,\boldsymbol{y}} \alpha_{\boldsymbol{y},t}[K((\boldsymbol{x}_t, \boldsymbol{y}_t), (\boldsymbol{x}_t, \boldsymbol{y}^*)) - K((\boldsymbol{x}_t, \boldsymbol{y}), (\boldsymbol{x}_t, \boldsymbol{y}^*))]$
5. if $\boldsymbol{y}^* \neq \boldsymbol{y}_t$
6. $\alpha_{\boldsymbol{y}^*,t} = \alpha_{\boldsymbol{y}^*,t} + 1$

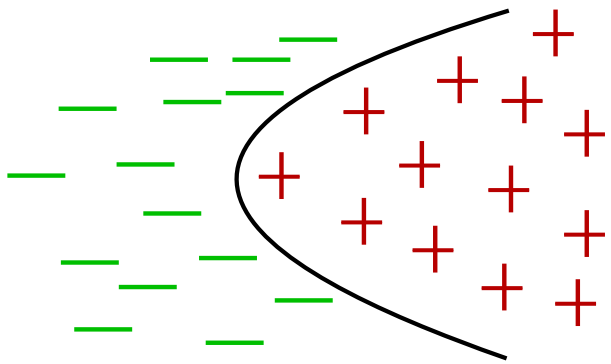
- Given a new instance $\boldsymbol{x}$

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}^*} \sum_{t,\boldsymbol{y}} \alpha_{\boldsymbol{y},t}[K((\boldsymbol{x}_t, \boldsymbol{y}_t), (\boldsymbol{x}, \boldsymbol{y}^*)) - K((\boldsymbol{x}_t, \boldsymbol{y}), (\boldsymbol{x}, \boldsymbol{y}^*))]$$

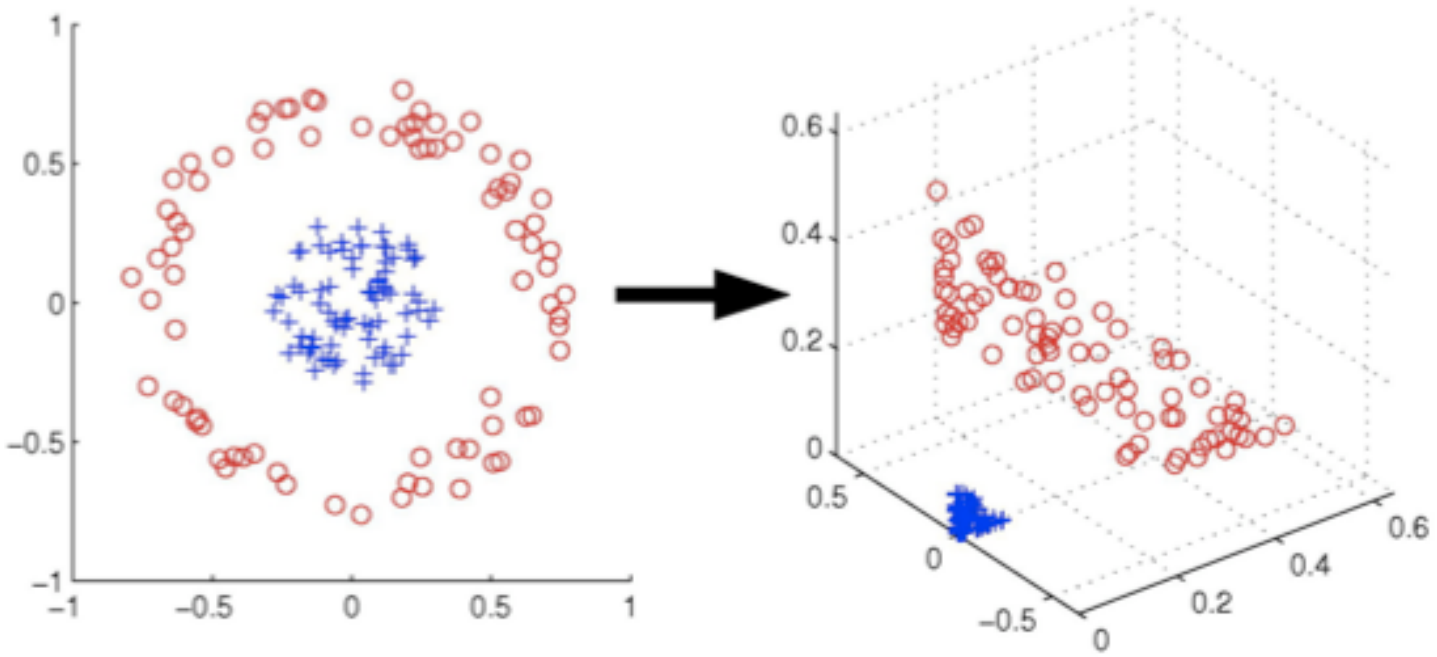
- But it seems like we have just complicated things???

Kernels = Tractable Non-Linearity

- A linear model in a higher dimensional feature space is a non-linear model in the original space
- Computing a non-linear kernel is often better computationally than calculating the corresponding dot product in the high dimensional feature space
- Thus, kernels allow us to efficiently learn non-linear models by convex optimization



Linear Learners in High Dimension



$$\begin{array}{ccc} \Re^2 & \longrightarrow & \Re^3 \\ (x_1, x_2) & \longmapsto & (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \end{array}$$

Example: Polynomial Kernel

- $\phi(\boldsymbol{x}) \in \mathbb{R}^M$, $d \geq 2$
- $K(\boldsymbol{x}_t, \boldsymbol{x}_s) = (\phi(\boldsymbol{x}_t) \cdot \phi(\boldsymbol{x}_s) + 1)^d$
  - $O(M)$ to calculate for any d!!
- But in the original feature space (primal space)
  - Consider $d = 2$, $M = 2$, and $\phi(\boldsymbol{x}_t) = [x_{t,1}, x_{t,2}]$

$$
\begin{aligned}
(\phi(\boldsymbol{x}_t) \cdot \phi(\boldsymbol{x}_s) + 1)^2 &= ([x_{t,1}, x_{t,2}] \cdot [x_{s,1}, x_{s,2}] + 1)^2 \\
&= (x_{t,1}x_{s,1} + x_{t,2}x_{s,2} + 1)^2 \\
&= (x_{t,1}x_{s,1})^2 + (x_{t,2}x_{s,2})^2 + 2(x_{t,1}x_{s,1}) + 2(x_{t,2}x_{s,2}) \\
&\quad +2(x_{t,1}x_{t,2}x_{s,1}x_{s,2}) + (1)^2
\end{aligned}
$$

which equals:

$$
\underbrace{[(x_{t,1})^2, (x_{t,2})^2, \sqrt{2}x_{t,1}, \sqrt{2}x_{t,2}, \sqrt{2}x_{t,1}x_{t,2}, 1]}_{\textit{feature vector in high-dimensional space}} \cdot \underbrace{[(x_{s,1})^2, (x_{s,2})^2, \sqrt{2}x_{s,1}, \sqrt{2}x_{s,2}, \sqrt{2}x_{s,1}x_{s,2}, 1]}_{\textit{feature vector in high-dimensional space}}
$$

Popular Kernels

- Polynomial kernel

$$K(\boldsymbol{x}_t, \boldsymbol{x}_s) = (\boldsymbol{\phi}(\boldsymbol{x}_t) \cdot \boldsymbol{\phi}(\boldsymbol{x}_s) + 1)^d$$

- Gaussian radial basis kernel (infinite feature space representation!)

$$K(\boldsymbol{x}_t, \boldsymbol{x}_s) = exp(\frac{-||\boldsymbol{\phi}(\boldsymbol{x}_t) - \boldsymbol{\phi}(\boldsymbol{x}_s)||^2}{2\sigma})$$

- String kernels
- Tree kernels

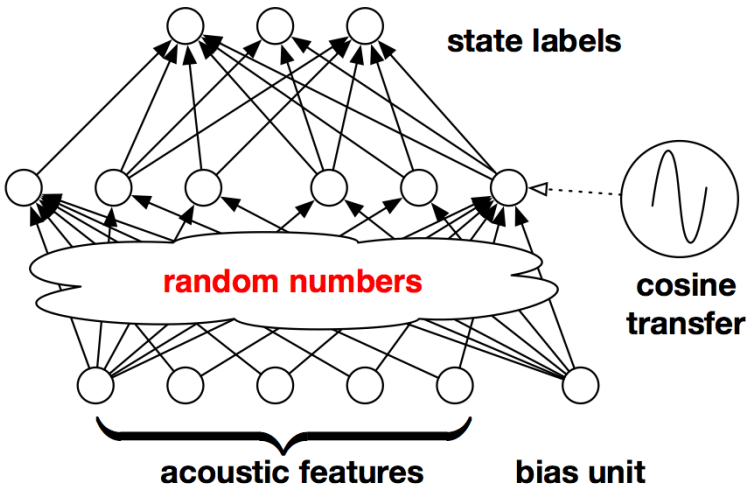
Kernels Summary

- Can turn a linear model into a non-linear model
- Kernels project feature space to higher dimensions
  - Sometimes exponentially larger
  - Sometimes an infinite space!
- Can "kernelize" algorithms to make them non-linear
- Convex optimization methods still applicable to learn parameters

Kernels for Large Training Sets

- Exact kernel methods depend polynomially on the number of training examples - infeasible for large datasets
- Alternative: Explicit randomized feature map

  [Rahimi and Recht 2007]
  - Shallow neural network by random Fourier transformation:
    - Random weights from input to hidden units
    - Cosine as transfer function
    - Linear learning of weights from hidden to output units



Wrap up and time for questions

Summary

Basic principles of machine learning:

- To do learning, we set up an objective function that tells the fit of the model to the data
- We optimize with respect to the model (weights, probability model, etc.)
- Can do it in a batch or online (preferred!) fashion

What model to use?

- One example of a model: linear model
- Can kernelize/randomize these models to get non-linear models
- Convex optimization applicable for both types of model

Further Reading

- **Introductory Example:**
- J. Y. Lettvin, H. R. Maturana, W. S. McCulloch, and W. H. Pitts. 1959. What the frog's eye tells the frog's brain. Proc. Inst. Radio Engr., 47:1940–1951.
- **Naive Bayes:**
- Pedro Domingos and Michael Pazzani. 1997. On the optimality of the simple bayesian classifier under zero-one loss. Machine Learning, (29):103–130.
- **Logistic Regression:**
- Bradley Efron. 1975. The efficiency of logistic regression compared to normal discriminant analysis. Journal of the American Statistical Association, 70(352):892–898.
- Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. Computational Linguistics, 22(1):39–71.
- Stefan Riezler, Detlef Prescher, Jonas Kuhn, and Mark Johnson. 2000.

Lexicalized Stochastic Modeling of Constraint-Based Grammars using Log-Linear Measures and EM Training. In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00), Hong Kong.

- **Perceptron:**
- Albert B.J. Novikoff. 1962.
  On convergence proofs on perceptrons. Symposium on the Mathematical Theory of Automata, 12:615–622.
- Yoav Freund and Robert E. Schapire. 1999.
  Large margin classification using the perceptron algorithm. Journal of Machine Learning Research, 37:277–296.
- Michael Collins. 2002.
  Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP'02), Philadelphia, PA.
- **SVM:**
- Vladimir N. Vapnik. 1998.
  Statistical Learning Theory. Wiley.
- Olivier Chapelle. 2007.

Training a support vector machine in the primal. Neural Computation, 19(5):1155–1178.

- Ben Taskar, Carlos Guestrin, and Daphne Koller. 2003.
  Max-margin markov networks. In Advances in Neural Information Processing Systems 17 (NIPS'03), Vancouver, Canada.

- Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004.
  Support vector machine learning for interdependent and structured output spaces. In Proceedings of the 21st International Conference on Machine Learning (ICML'04), Banff, Canada.

- **Kernels and Regularization:**

- Bernhard Schölkopf and Alexander J. Smola. 2002.
  Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond. The MIT Press.

- Ali Rahimi and Ben Recht. 2007.
  Random features for large-scale kernel machines. In Advances in Neural Information Processing Systems (NIPS), Vancouver, B.C., Canada.

- Zhiyun Lu, Dong Guo, Alireza Bagheri Garakani, Kuan Liu, Avner May, Aurelien Bellet, Linxi Fan, Michael Collins, Brian Kingsbury, Michael Picheny, and Fei Sha. 2016.
  A comparison between deep neural nets and kernel acoustic models for speech recognition. In IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).

- **Convex and Non-Convex Optimization:**

- Yurii Nesterov. 2004.
  Introductory lectures on convex optimization: A basic course. Springer.

- Stephen Boyd and Lieven Vandenberghe. 2004.
  Convex Optimization. Cambridge University Press.

- Dimitri P. Bertsekas and John N. Tsitsiklis. 1996.
  Neuro-Dynamic Programming. Athena Scientific.

- **Online/Stochastic Optimization:**

- Herbert Robbins and Sutton Monro. 1951.
  A stochastic approximation method. Annals of Mathematical Statistics, 22(3):400–407.

- Boris T. Polyak. 1964.

Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics, 4(5):1 – 17.

- Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. 2004. On the generalization ablility of on-line learning algorithms. IEEE Transactons on Information Theory, 50(9):2050–2057.
- Nicolas LeRoux, Mark Schmidt, and Francis Bach. 2012. A stochastic gradient method with an exponential convergence rate for finite training sets. In Advances in Neural Information Processing Systems (NIPS), Lake Tahoe, CA.
- Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In Proceedings of International Conference on Machine Learning (ICML), pages 1139–1147.