CS 5100: Foundations of Artificial Intelligence

NLP & Support Vector Machines

Prof. Amy Sliva
December 8, 2011

Outline

- Discuss final exam
- POS tagging and parsing review
- NLP semantics, pragmatics, machine translation
- Neural networks and support vector machines

Final exam

- December 15, room 135 Shillman Hall

- Topics—cumulative exam, but heavily skewed toward post-midterm material
  1. Probability theory
  2. Bayesian networks
  3. Machine learning
     - Naïve Bayes
     - Decision trees (and information theory)
     - Support vector machines
  4. NLP
     - Parsing
     - POS tagging
     - Semantic analysis

Extra credit opportunity!

- **IBM Watson talk**
  - Distinguished Speaker Series
    Fri. 12/09/11, 10am–11am 90 Snell Library
  - What Is Watson?
    Michael P. Perrone, PhD
    Manager, Multicore Computing, IBM T.J. Watson Research Center
- Attend talk, write a reaction paper
  - Max 2 pages
  - Insightful thoughts about the application, technical and theoretical AI content, relevance to what you have learned in the course, etc.
- Up to 4 extra points on final grade!

Natural language processing (review)

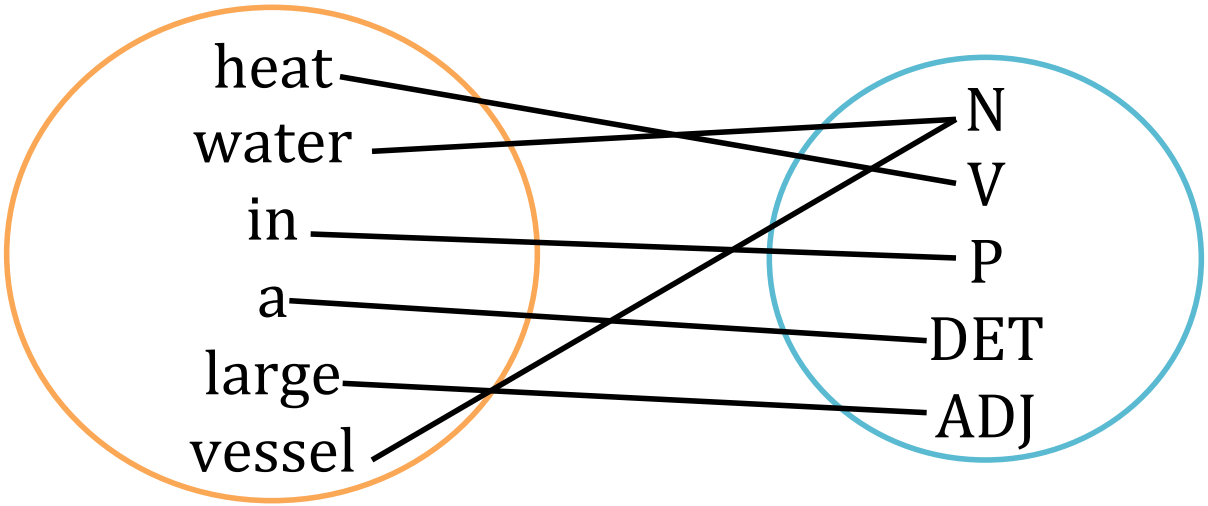
- Natural **language**
  - Language spoken by people
  - E.g., English, Japanese, Swahili, etc. as opposed to artificial languages like C++, Java, etc.

- Natural language **processing**
  - Applications that deal with natural language in one way or another

- Levels of analysis
  - Lexical
  - Syntactic
  - Semantic
  - Pragmatic

Part-of-speech tagging (lexical analysis)

- Process of assigning a part-of-speech (POS) to each word in a sentence



Significance of parts of speech

- Word's POS tells us a lot about the word (and its neighbors)
  - Limits range of **meanings** (deal), **pronunciation** (object vs object) or both (wind)
  - Limits range of **following words**
  - Help select nouns from a document for **summarization**
  - Parsers can build trees directly on the POS tags instead of maintaining a lexicon

Methods for POS tagging

- **Rule-based** POS tagging
  - E.g., ENGTWOL (Voutilainen, 1995)—large collection (> 1000) of constraints on what sequences of tags are allowable
- Transformation-based tagging
  - E.g., Brill's tagger (Brill, 1995)—sorry, I don't know anything about this…
- **Stochastic** (Probabilistic) tagging
  - Uses supervised learning
  - E.g., TNT (Brants, 2000)—we'll talk about this in more detail!

Supervised learning approach

- Algorithms "learn" from data by generalizing a set of examples
- **Training set**—examples trained on
- **Test set**—used for evaluating the algorithm
  - Must be separate from training set (otherwise you cheated!)
- "Gold" standard
  - Test set that a community has agreed on and uses as a common benchmark

Cross-validation learning algorithms

- Cross-validation set—part of training set

- Used for **tuning parameters** of the algorithm without "polluting" (tuning to) the test data
  - Train on *x*%, and then cross-validate on the remaining 1-*x*%
    - E.g., train on 90% training data, cross-validate on the remaining 10%
    - Repeat several times with different splits to get the best parameter estimation
  - Allows you to choose the best settings to then use on the real test set
    - Only evaluate on the test set at the very end after the algorithm is as good as possible from cross-validation

Strong baselines

- When designing NLP algorithms, must evaluate by comparing to others
- **Baseline** algorithm
  - Algorithm that is simple, but can be expected to do well
  - Should get the best score possible by doing the somewhat obvious thing

- **POS tagging baseline**—for each word, assign its most frequent tag in the training set
- Want our stochastic taggers to improve on this!

N-grams for POS tagging

- *N* stands for how many terms are used in conditional probability
  - Unigram: 1 term (0^{th} order) E.g., $P(X_i)$
  - Bigram: 2 terms (1^{st} order) E.g., $P(X_i | X_{i-1})$
  - Trigram: 3 terms (2^{nd} order) E.g., $P(X_i | X_{i-1}, X_{i-2})$

- Can use different kinds of terms
  - Character-based *n*-grams
  - Word-based *n*-grams
  - POS-based *n*-grams
- Helps determine **context** in which some linguistic phenomenon happens
  - E.g., what POS will a word have, given the preceding parts of speech?

First approach (unigram)

- Assign each word its **most likely** POS tag
- If w has tags $t_1,\ldots,t_k$ then use

$$P(t_i \mid w) = \frac{c(w, t_i)}{c(w, t_1) + \ldots + c(w, t_k)}$$

- where $c(w, t_i)$ = number of times w/t_i appears in the corpus
- Success: 91% for English!
- Example
  - heat::**noun/89**, **verb/5**

Second approach (bigram)

- Given: sequence of words (i.e., a **sentence**) W s.t.

$$W = w_1, w_2, \ldots, w_n$$

  - E.g., W = heat water in a large vessel

- Assign sequence of tags T s.t.

$$T = t_1, t_2, \ldots, t_n$$

- Find T that **maximizes** $P(T \mid W)$

Practical Stochastic Tagger

- By Bayes Rule:

$$P(T \mid W) = \frac{P(W \mid T)\, P(T)}{P(W)} = \alpha\, P(W \mid T)\, P(T)$$

- So find T that maximizes $P(W \mid T)\, P(T)$
  - Chain rule:

    $P(T) = P(t_1)P(t_2 \mid t_1)\, P(t_3 \mid t_1, t_2)\, P(t_3 \mid t_1, t_2, t_3) \ldots P(t_n \mid t_1, \ldots, t_{n-1})$

  - **Markov assumption**: as an approximation, use:

    $P(T) \approx P(t_1)P(t_2 \mid t_1)\, P(t_3 \mid t_2) \ldots P(t_n \mid t_{n-1})$

  - **Naïve Bayes assumption**: each word is dependent only on its own POS tag (given its POS tag, it is conditionally independent of the other words)

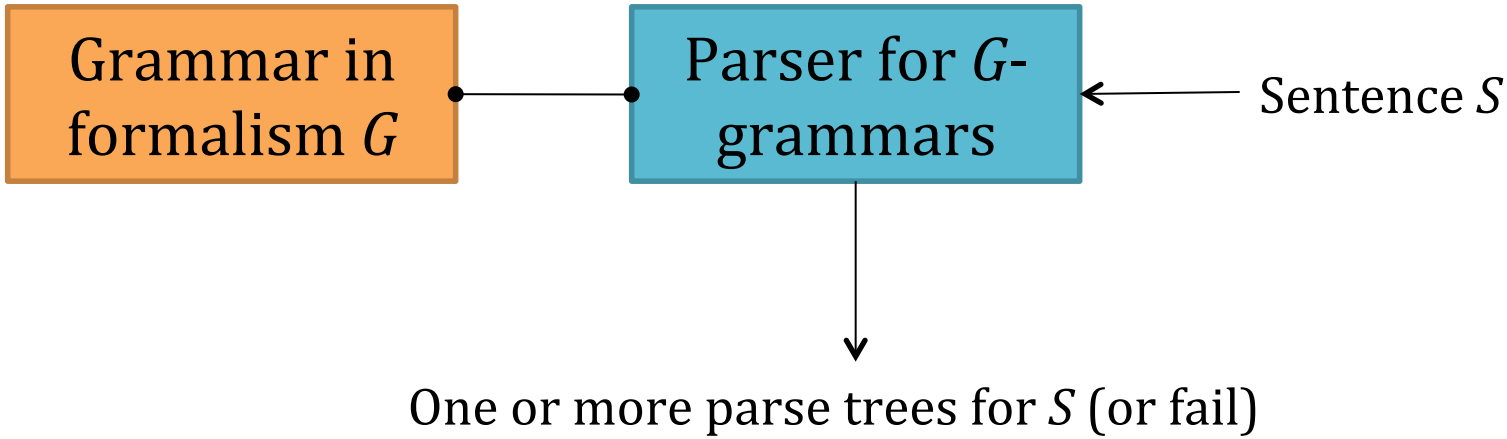
    $P(W \mid T) = P(w_1 \mid t_1)\, P(w_2 \mid t_2) \ldots P(w_n \mid t_n)$

- So

$$\mathbf{P(W \mid T)\, P(T) \approx P(w_1 \mid t_1)\, P(w_2 \mid t_2) \ldots P(w_n \mid t_n)\, P(t_1)\, P(t_2 \mid t_1)\, P(t_3 \mid t_2) \ldots P(t_n \mid t_{n-1})}$$

Syntactic analysis (parsing)

- Uses formal grammar and parsing algorithm to find structure
  - Create parse trees from sentences



- Limitations
  - Explosion of number of parse trees
  - Inability to handle ungrammatical input

Grammars for parsing

- Grammar—specifies the **compositional structure** of complex messages
  - E.g., speech (linear), text (linear), music (two-dimensional)

- A **formal language** is a set of **strings** of **terminal** symbols (actual words)
- Each string in the language can be **analyzed/generated** by the grammar

- Grammar is a set of rewrite rules (productions)

$S \rightarrow NP\ VP$
$Article \rightarrow the$ | a | an | …
$NP \rightarrow$ …
$VP \rightarrow$ …
…

Here ***S*** is the **sentence** symbol, ***NP*** and ***VP*** are **non-terminals**

Parse tree for a sentence

- Parse tree represents **grammatical structure** of sentence
  - May also indicate **semantic interpretation**

- Any sentence of formal language L has one or more parse trees
  - Show how it can be derived by repeated applications of production rules in the grammar

Parsing with CFGs (or PCFGs)

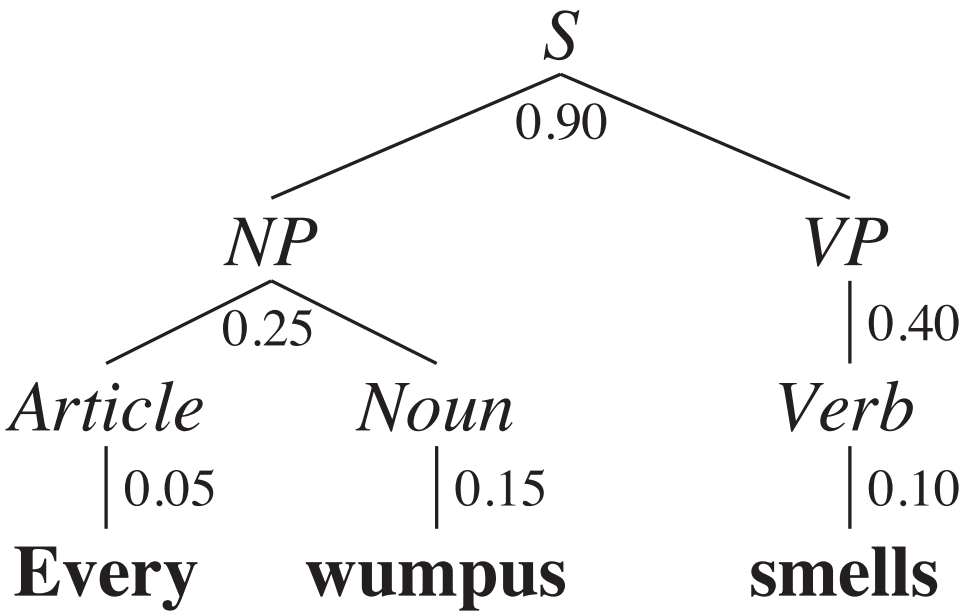
- Task of assigning correct trees to input strings
  - Tree covers **all and only the elements of the input** and has **an *S* a at the top**

- System may not be able to select the "correct" tree from among possible trees
  - Parsing involves search where choices must be made
  - Requires semantics to find the "right" tree!

Wumpus grammar

S → *NP VP*	[0.90]	I + feel a breeze
| *S Conj S*	[0.10]	I feel a breeze + and + it stinks
NP → *Pronoun*	[0.30]	I
| *Name*	[0.10]	John
| *Noun*	[0.10]	pits
| *Article Noun*	[0.25]	the + wumpus
| *Article Adjs Noun*	[0.05]	the + smelly dead + wumpus
| *Digit Digit*	[0.05]	3 4
| *NP PP*	[0.10]	the wumpus + in 1 3
| *NP RelClause*	[0.05]	the wumpus + that is smelly
VP → *Verb*	[0.40]	stinks
| *VP NP*	[0.35]	feel + a breeze
| *VP Adjective*	[0.05]	smells + dead
| *VP PP*	[0.10]	is + in 1 3
| *VP Adverb*	[0.10]	go + ahead
Adjs → *Adjective*	[0.80]	smelly
| *Adjective Adj*	[0.20]	smelly + dead
PP → *Prep NP*	[1.00]	to + the east
RelClause → *RelPro VP*	[1.00]	that + is smelly

Wumpus parse tree



- Total probability of the tree
  $= 0.9 \times 0.25 \times 0.40 \times 0.05 \times 0.15 \times 0.10 = 0.0000675$

Simple parsing exercise

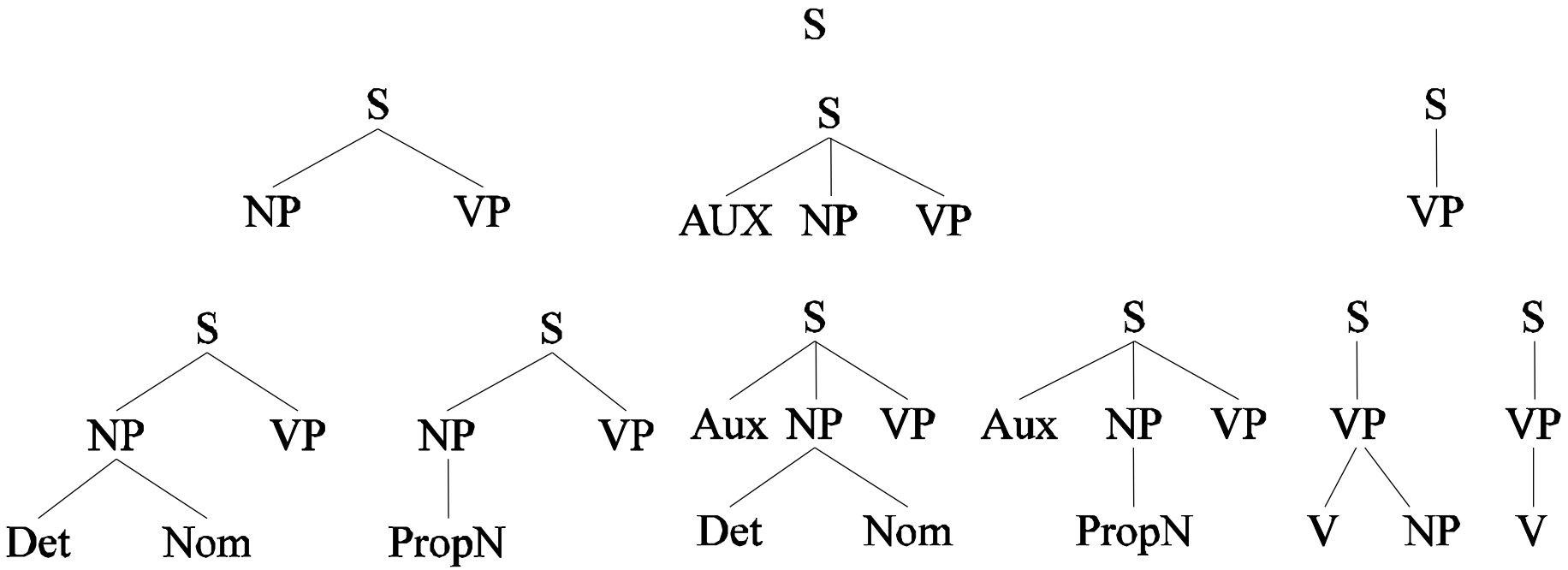
- Context-free grammar for arithmetic expressions

  S → digit
  S → S + S
  S → S * S

- 3 * 4 + 5 has two valid parse trees with different semantics. What are they?

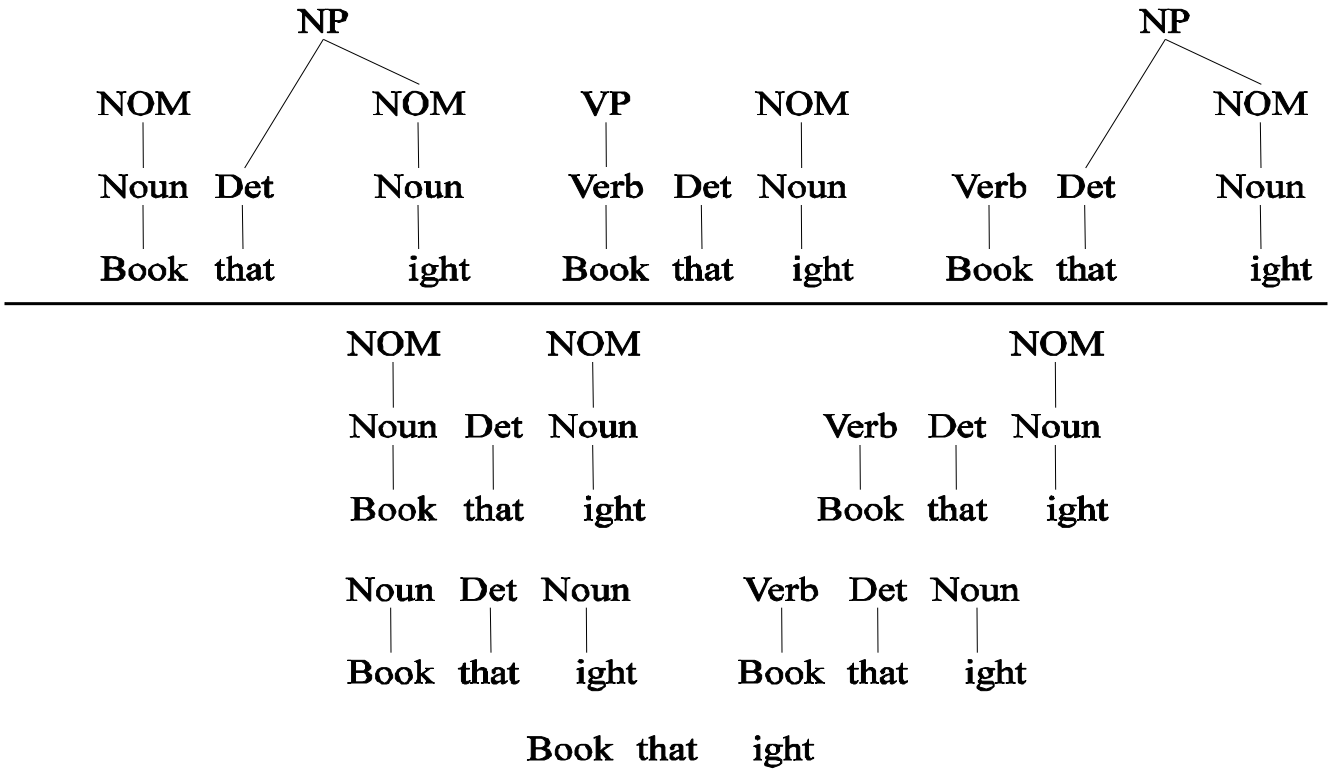
Top-down parsing

- Trying to find trees rooted with an S, so start with rules that give us an S
- Work your way down from there to the words



Bottom up parsing

- Since want trees that cover the input words, start with trees that match the words right away
- Work your way up from there



Top-down vs. bottom-up

- Top-down
  - Only searches for trees that can be answers, but suggests trees that are not consistent with the input words
  - Guarantees that tree starts with S as root
  - Does not guarantee that tree will match input words
- Bottom-up
  - Only forms trees consistent with the input words, but suggests trees that make no sense globally
  - Guarantees that tree matches input words
  - Does not guarantee that parse tree will lead to S as root
- Combine advantages of the two by doing a search constrained from both sides

Semantic analysis

- Determine the **meaning** of language

- Importance of semantics?
  - Machine translations: wrong translations
  - Information retrieval: wrong information
  - Anaphora resolution: wrong referents

- Biggest challenge: lexical **ambiguity**—words are ambiguous
  - "plant" = industrial plant
  - "plant" = living organism

Why do we need semantics?

- Machine translation example
  - **The sea is home to millions of plants and animals**

    English → French translation (commercial MT system)

    Le mer est a la maison de billlion des usines et des animaux

    French → English

    The sea is at the home for billions of factories and animals

- Hmmm…

Lexical ambiguity

- Extreme case—two words with the **same spelling**
  - Wound, wound
- More frequent case—words that can be a noun, verb, adjective, etc.
  - Time, phone
- Many English words have multiple meanings even within one part of speech (POS)
  - Set, head, can, bear, …
- WordNet—public domain lexical-semantic net
  - **Demo!! (http://wordvis.com/)**
- SemEval—a periodic "shared task" activity to evaluate semantic analysis tools

How to learn the meaning of words?

- How do we get a training set of semantic examples?

- From dictionaries: **word sense**
  - plant, works, industrial plant—(buildings for carrying on industrial labor; “They built a large plan to manufacture automobiles.”)

    plant, flora, plant life—(a living organism lacking the power of locomotion)

- Can these definitions help disambiguate all uses?
  - They are producing about 1,000 automobiles in the new plant.
  - The sea flora consists of 1,000 different plant species.
  - The plant was close to the farm.

How to learn the meaning of words? (cont.)

- Learn from annotated examples
  - Assume 100 examples containing “plant” previously tagged by a human
  - Train a learning algorithm to classify future instances of “plant”

- How to choose the learning algorithm?
- How to obtain the 100 tagged examples?

Fillmore's case grammar

- Assign semantically based **cases** to distinguish a word's "role" and disambiguate
- Charles Fillmore, "The Case for Case," 1968
  - Produced more than one version

Case grammar semantics

- Treats the verb as a predicate and the subject, objects, and other subordinate clauses as the "arguments"

- Labels arguments with their **relationship** to the verb-predicate (called **cases**)
  - E.g., John sold his car—agent and object cases
    John sold his car to Mary—agent, object, and recipient cases

Fillmore's list of cases

- **Agentive** (A)—case of the typically animate perceived instigator of the action identified by the verb
- **Instrumental** (I)—case of inanimate force or object causally involved in the action of state identified by the verb
- **Dative** (D) (later **Experiencer** (E))—the case of the animate being affected by the state or action identified by the verb
- **Factive** (F) (later **Goal** (G))—the case of the object or being resulting from the action or state identified by the verb, or understood as a part of the meaning of the verb
- **Locative** (L)—the case identifying location or spatial orientation of the state or action identified by the verb
- **Objective** (O)—the semantically most neutral case; anything representable by a noun whose role in the action or state identified by the verb is identified by the semantic interpretation of the verb itself

Analysis of case semantics

- Strengths
  - Only one Noun Phrase occupies each case role in relation to a particular verb
  - Can classify verbs in terms of which case roles they took
    E.g., "open"—O, {A}, {I}
    "shout"—A, O, {E}

- Weaknesses
  - Researchers cannot agree on standard set of cases!
  - Not easy to classify Noun Phrases as cases in practice
  - Tendency to use the "Objective" case whenever it gets difficult

More issues in semantic analysis

- Reference resolution
  "Josh sold a book to Tom. **He** was happy."
  "Mary and I went out to dinner. **It** was fun."

- The pronoun (or other **anaphoric** noun phrase) may reference something that is implicitly mentioned by does not have a specific antecedent.

Pragmatics—semantics and **context**

- **Classical** view (pre-1953)—language consists of sentences that are true/false (like logic)
  - Why?
    To modify the beliefs of other agents

- **Modern** view (post-1953)—language is a form of action
  - Why?
    To change the actions of other agents

Speech act theory of pragmatics

- Examples: "I pronounce you husband and wife"
  "I sentence you to five years"

SITUATION

Speaker → **Utterance** → **Hearer**

- Speech acts achieve the speaker's goals:
  - Inform "There is a pit in front of you"
  - Query "Can you see the gold?"
  - Command "Pick it up"
  - Promise "I'll share the gold with you"
  - Acknowledge "OK"

- Speech act planning requires knowledge of
  - Situation
  - Semantic and syntactic conventions
  - Hearer's goals, knowledge base, rationality

Machine translation

- Text to text machine translations
- Speech to speech machine translations

- Most of the work has addressed pairs of widely spread languages like English-French, English-Chinese

Issues in machine translations

- How to translate text?
  - Learn from previously translated data
  - Need **parallel corpora**
    - French-English, Chinese-English have the Hansards (transcripts of parliamentary debates)
- Reasonable translations?
  - Application dependent—do we need the general idea or precise language?
- Lack of data = lack of tools
  - Chinese-Hindi—no translator available!

Speech to speech translation challenges

- Stages in communication (informing)
  - **Intention** — *S* wants to inform *H* that ***P***
  - **Generation** — *S* selects words ***W*** to express ***P*** in context ***C***
  - **Synthesis** — *S* utters words ***W***

  - **Perception** — *H* perceives ***W'*** in context ***C'***
  - **Analysis** — *H* infers possible meanings $P_1,\ldots,P_n$
  - **Disambiguation** — *H* infers intended meaning P_i
  - **Incorporation** — *H* incorporates P_i into KB

- How could this go wrong?
  - Insincerity (S doesn't believe ***P***)
  - Speech wreck ignition (**recognition!**) failure
  - Ambiguous utterance
  - Differing understanding of current context ($C \neq C'$)

Information extraction

- Extract information and detect new patterns in data
  - Detect hacking, hidden information, etc.
- Government and military put a lot of money into IE research!

- Example:
  "There was a group of about 8-9 people close to the entrance on Highway 75"
  - **Who?** "8-9 people"
  - **Where?** "Highway 75"

Information retrieval

- General model—Huge collection of **texts** and a **query**
- Tasks
  - Find **documents** that are relevant to the given query—Create an index, like the index in a book
  - Retrieve specific information—**question answering**
    "What is the height of Mount Everest?" 11,000 feet

- Types of models
  - Vector-space models
  - Boolean models

- Examples—Google, Yahoo, etc.

Cross-language information retrieval

- Find information across languages!

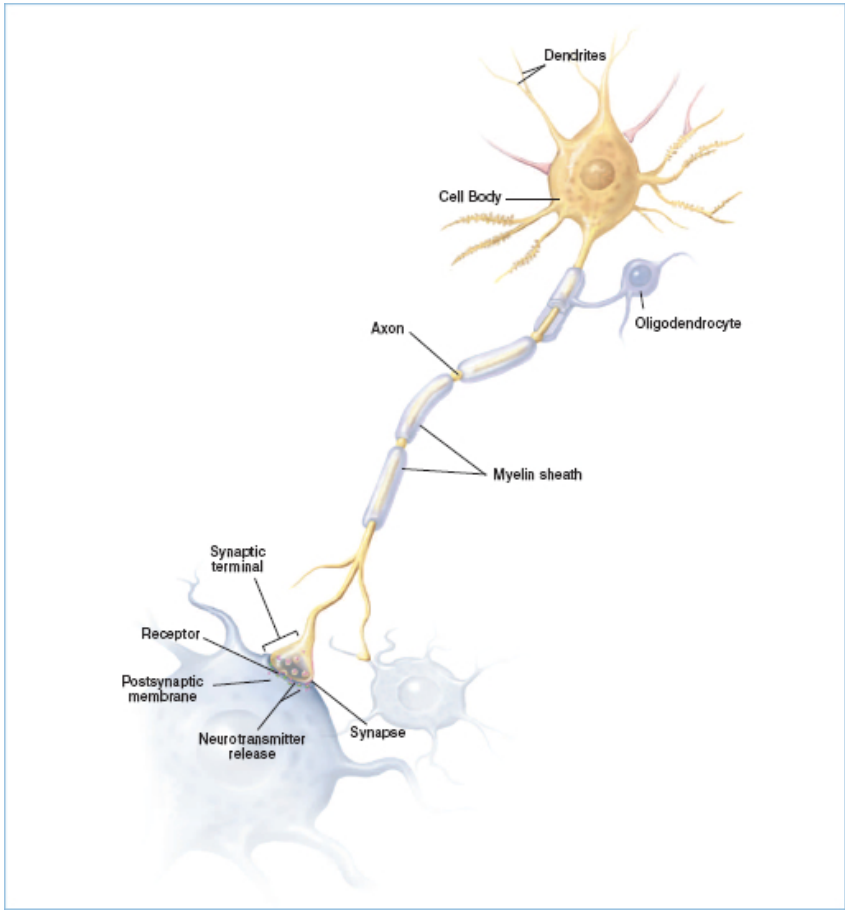
- Example:
  "What is the minimum age requirement for car rental in Italy?"
  - Search English and also Italian texts for "eta minima per noleggio macchine"

- Integrate large number of languages and into performant IR engines

Perceptrons and neural networks

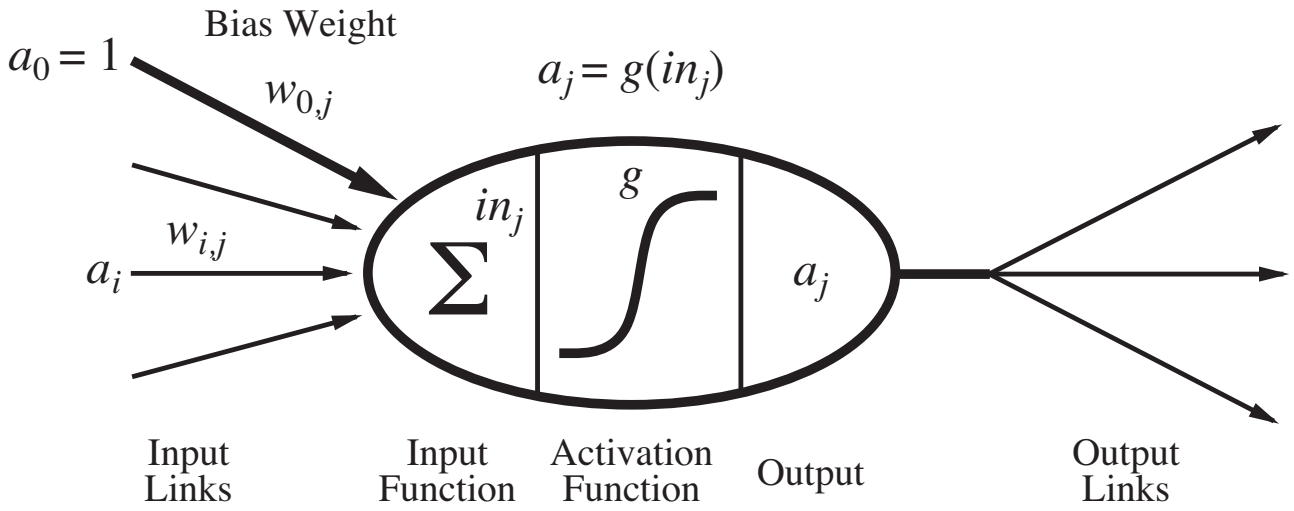
- Another supervised learning approach—mathematical model of neurons

- Human brains
  - **10^{11}** neurons of **> 20** types, **10^{14}** synapses, 1ms-10ms cycle time
  - Signals are noisy "spike trains" of **electrical potential**



McCulloch-Pitts "unit"

- Simple mathematical model for a neuron

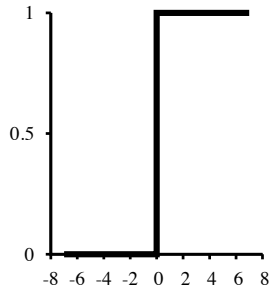


- Neural network—units connected by directed links
  - Propagates activation a_i from i to j
  - Network is a function $h_{\boldsymbol{w}}(\boldsymbol{a})$ parameterized by weights
- Output is a "squashed" linear function of the inputs

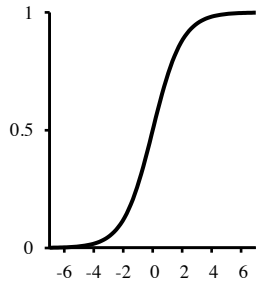
$$a_j = g(in_j) = g(\Sigma_i w_{i,j} a_i)$$

Activation functions

- **Perceptron**—hard threshold (step function) activation function
  - Changing the bias weight moves the threshold loction



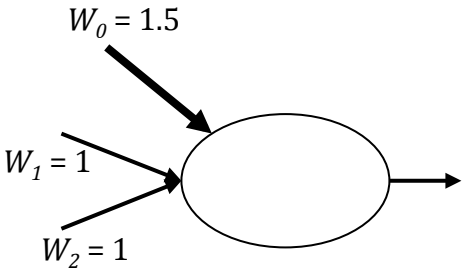
- **Sigmoid**—logistic activation function



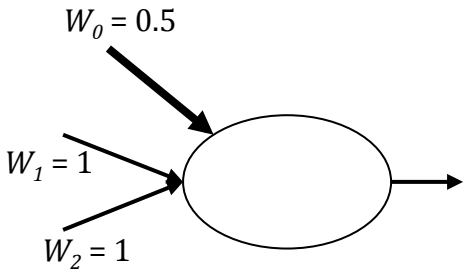
Perceptrons and threshold logic

- Perceptron—single layer neural network with only one neuron
  - Neuron unit calculates input through threshold activation function
  - Sometimes called **threshold logic unit** (TLU)
    - Discriminates data depending on whether sum is greater than threshold
  - $g(in) = 1$ iff $in >$ threshold, 0 otherwise

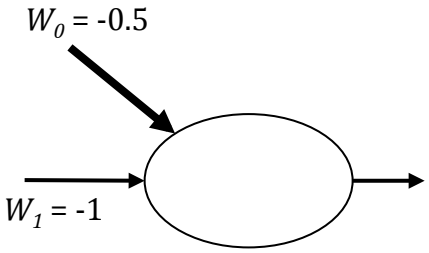
- Can implement every **boolean function**



AND



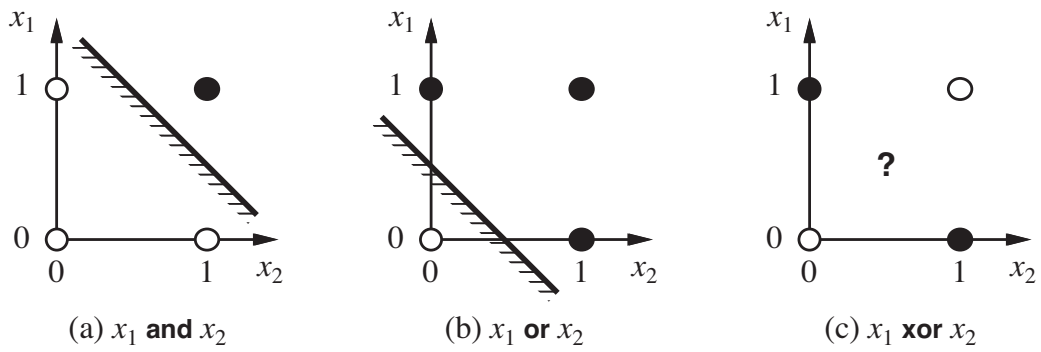
OR



NOT

Perceptron learning

- Provides linear discriminant for classification



(a) x_1 **and** x_2 (b) x_1 **or** x_2 (c) x_1 **xor** x_2

- Main learning task is to **learn the weights** so we know how to classify new inputs
- Several possible algorithms for **training** single-layer perceptrons using
  - Perceptron rule
  - Gradient descent rule
  - Delta rule

Error-correction learning

- All rules for perceptron learning based on **error-correction**

1. Assign random weights (or set all to 0)
2. Cycle through input until change < target
3. Let α be the "learning coefficient"
4. For each input:
   - If perceptron gives correct answer, do nothing
   - If perceptron says yes when answer should be no, decrease weights on all units that "fired" by α
   - If perceptron says no when answer should be yes, increase weights by α

Perceptron rule

- Simple rule for updating the weights when perceptron answer is incorrect

$$w_i = w_i + \alpha(y - h_w(x_i)) \, x_i$$

- **Perceptron convergence theorem**
  - For any data set which is linearly separable the perceptron learning rule is **guaranteed** to find a solution in a **finite** number of steps

Perceptron learning example

- Suppose a perceptron accepts two inputs $x_1 = 2$ and $x_2 = 1$, with weights $w_1 = 0.5$ and $w_2 = 0.3$ and $w_0 = -1$ (meaning that the threshold is 1)

- The output of the perceptron is :

$$h_w(x) = 2 * 0.5 + 1 * 0.3 - 1 = 0.3 \text{ which is } > 0$$

- Therefore the output is 1. If the correct output however is -1, the weights will be adjusted according to the Perceptron rule as follows:

$$w_1 = 0.5 + 0.1 * (-1 - 1) * 2 = 0.1$$
$$w_2 = 0.3 + 0.1 * (-1 - 1) * 1 = 0.1$$
$$w_0 = -1 + 0.1 * (-1 - 1) * 1 = -1.2$$

- The new weights would classify this input as follows:

$$h_w(x) = 2 * 0.1 + 1 * 0.1 – 1.2 = - 0.9$$

- Therefore we have done "error correction"

How to optimize the search

- Learn by adjusting weights to reduce **error** on the training set
- The squared error for an example with input x and true output y

$$E = \frac{1}{2} Err^2 \approx \frac{1}{2}(y - h_w(x))^2$$

- Gradient descent, hill climbing, simulated annealing
  - Optimization techniques that search for weights to reduce error faster
  - Find a new adjusted activation function g'
- Optimized perceptron learning rule

$$w_i = w_i + \alpha \times \text{Err} \times g'(x_i) \times x_i$$

Hill climbing optimization

- Given function *F,* find the x that gives the best *F(x)*

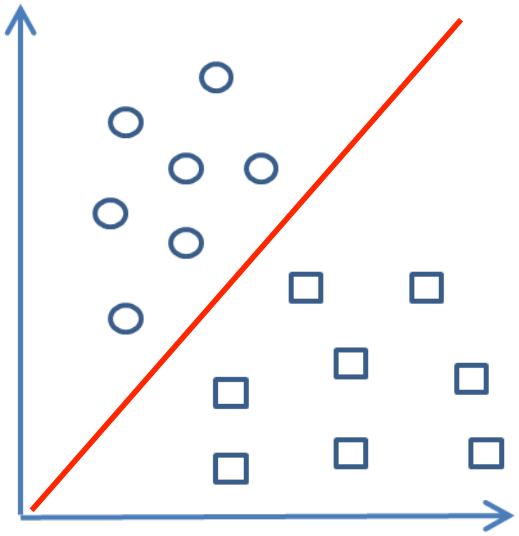
- How it works
  - Choose point in *n*-dimensional space to search as current "guess" $\boldsymbol{x} = x_1 \ldots x_n$ (i.e., our current amount of error)
  - Take a small step in *k* directions
  - Choose the direction that results in maximum improvement in $F(x)$ i.e., find a new value for $g(in)$ s.t. the error $y - g(in)$ is lower
  - Make that the new guess
  - Repeat until no more improvement is possible or desirable

- Simulated annealing—variant that randomly jumps at intervals to find a better region

- Rather than incremental improvements, we can **reduce error faster** and get closer to the actual function

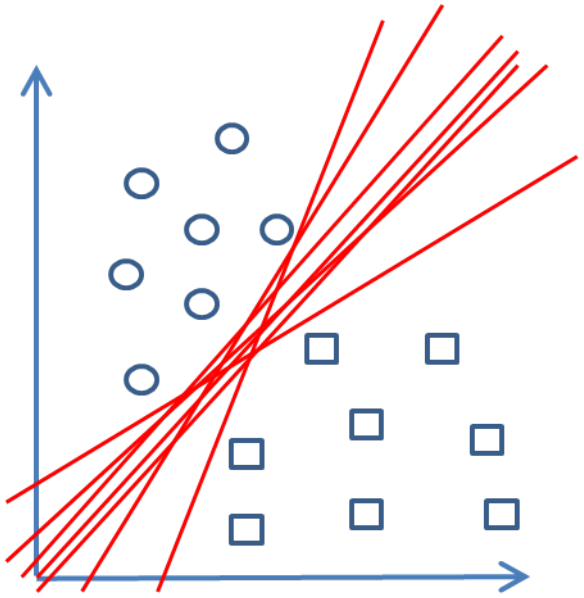
Brief overview of support vector machines

- SVMs are also linear classifiers



Brief overview of SVMs

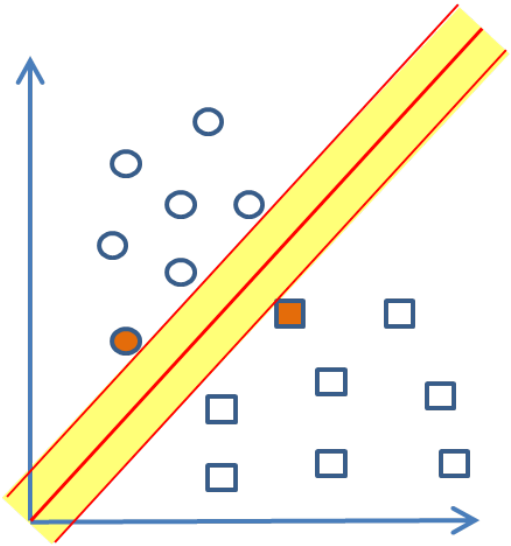
- But, which linear separator do we want to use?



- Any is fine, but which is the best?

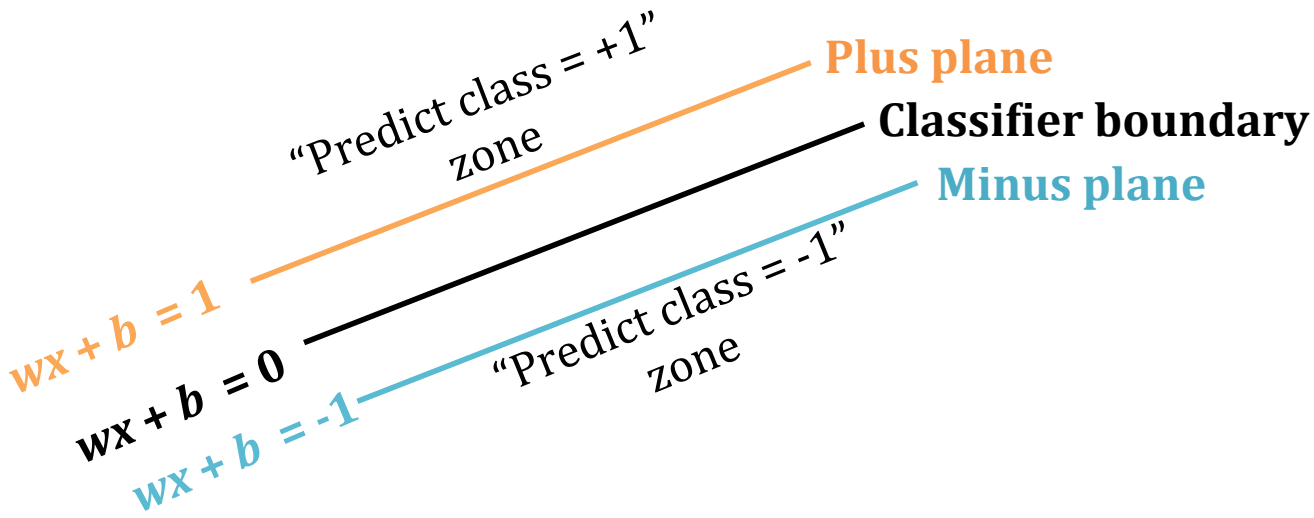
Maximum margin linear classifier

- **Margin** of a linear classifier is the width the boundary can be increased before hitting a data point
- **Support vectors**—data points that the margin pushes against



- Want to use the linear classifier with the **maximum margin**
  - **Conservative** estimation—if we have an error in the boundary, higher margin means less chance of misclassificaiton
  - Robust to outliers—strong generalization ability
- Simplest kind of SVM—empirically works really well!

Mathematically specifying a line and margin

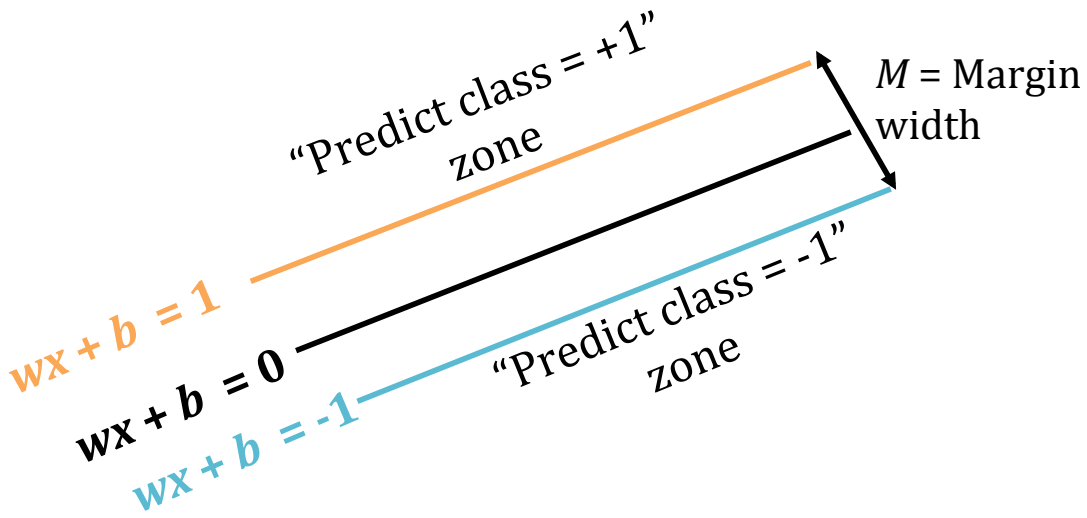


- Plus-plane = {***x*: *wx* + *b* = +1}**
- Minus-plane = {***x*: *wx* + *b* = -1}**

- Classify as

+1	if	***wx* + *b* >= 1**
-1	if	***wx* + *b* <= -1**
universe explodes	if	**-1 < *wx* + *b* < 1**

Computing the margin width

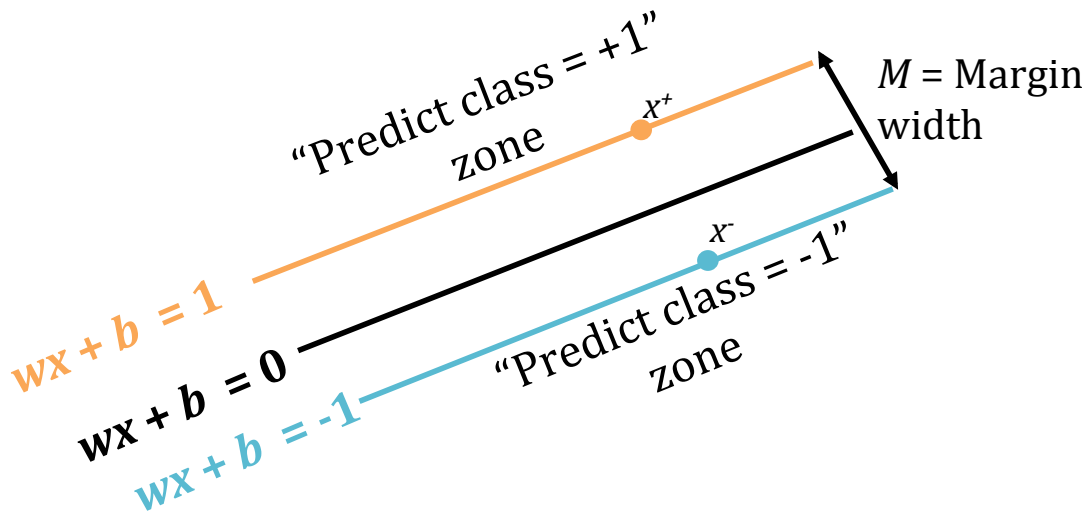


- Plus-plane = $\{\boldsymbol{x}: \boldsymbol{w}\boldsymbol{x} + b = +1\}$
- Minus-plane = $\{\boldsymbol{x}: \boldsymbol{w}\boldsymbol{x} + b = -1\}$

- How do we compute the margin M in terms of w and b?

- **Claim**: the vector $\boldsymbol{w}$ is perpendicular to the plus plane (and minus plane)
  - Suppose $\boldsymbol{u}$ and $\boldsymbol{v}$ are two vectors on the plus plane
    $\boldsymbol{w} \cdot (\boldsymbol{u} - \boldsymbol{v}) = 0$

Computing the margin width

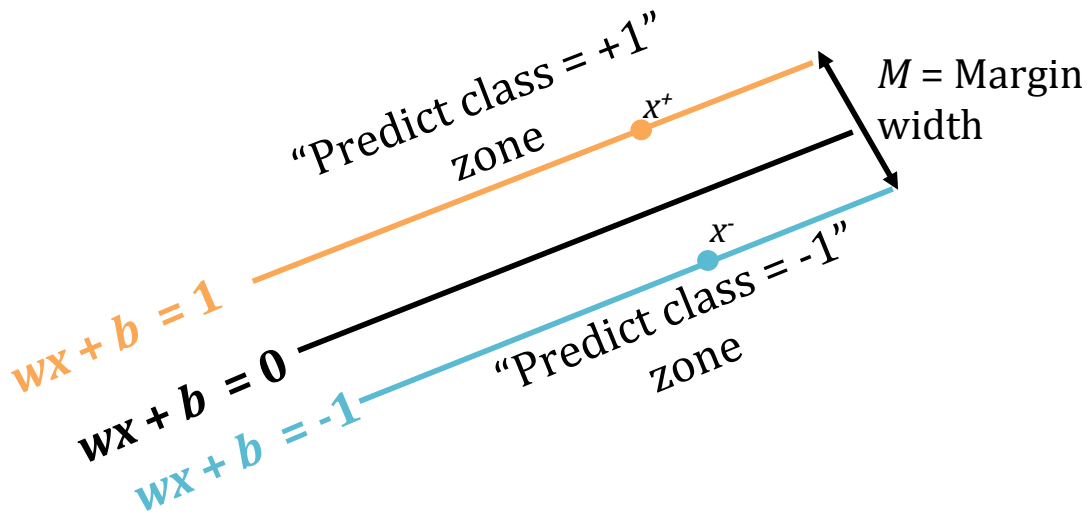


- Plus-plane = $\{x: wx + b = +1\}$
- Minus-plane = $\{x: wx + b = -1\}$

- How do we compute the margin M in terms of w and b?

- $\boldsymbol{w}$ is perpendicular to the Plus Plane
  - Let $\boldsymbol{x}^-$ be a point on the Minus Plane
    Let $\boldsymbol{x}^+$ be the closest point to x- on the Plus Plane
  - **Claim**: $\boldsymbol{x}^+ = \boldsymbol{x}^- + \lambda \boldsymbol{w}$ for some value of λ. **Why?**

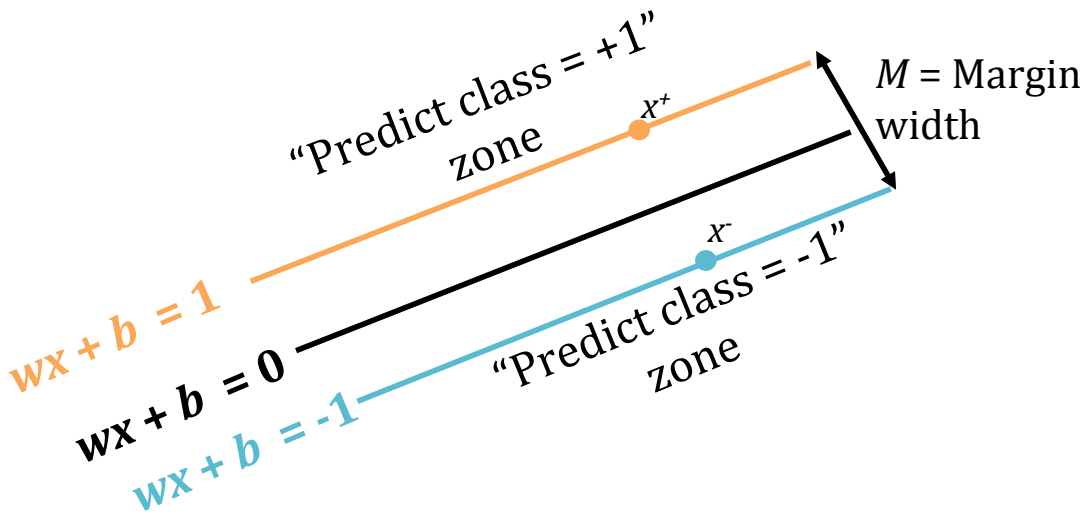
Computing the margin width



- Plus-plane = $\{\boldsymbol{x}: \boldsymbol{w}\boldsymbol{x} + b = +1\}$
- Minus-plane = $\{\boldsymbol{x}: \boldsymbol{w}\boldsymbol{x} + b = -1\}$

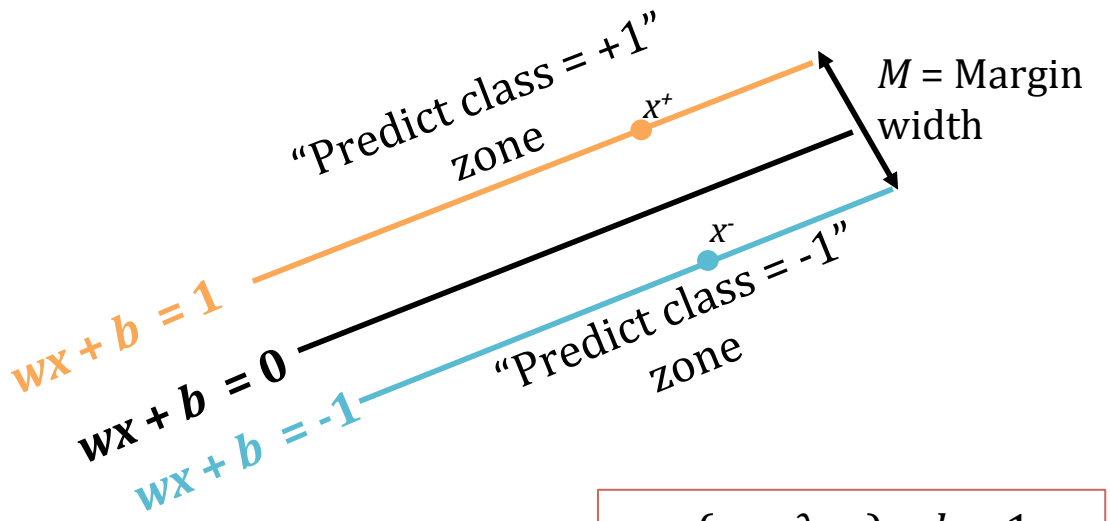
- $\boldsymbol{w}$ is perpendicular to the Plus Plane
  - Let $\boldsymbol{x}^-$ be a point on the Minus Plane
    Let $\boldsymbol{x}^+$ be the closest point to x- on the Plus Plane
  - **Claim**: $\boldsymbol{x}^+ = \boldsymbol{x}^- + \lambda \boldsymbol{w}$ for some value of λ. **Why?**
    The line from $\boldsymbol{x}^+$ to $\boldsymbol{x}^-$ is perpendicular to the planes, so travel some distance along $\boldsymbol{w}$ to get from one to another

Computing the margin width



- What we know now:
  $\boldsymbol{w}\boldsymbol{x}^+ + b = 1$
  $\boldsymbol{w}\boldsymbol{x}^- + b = -1$
  $\boldsymbol{x}^+ - \boldsymbol{x}^- = \lambda \boldsymbol{w}$
  $|\boldsymbol{x}^+ - \boldsymbol{x}^-| = M$
- Now its easy to get M in terms of $\boldsymbol{w}$ and b

Computing the margin width



- What we know now:
  $\boldsymbol{w}\boldsymbol{x}^+ + b = 1$
  $\boldsymbol{w}\boldsymbol{x}^- + b = -1$
  $\boldsymbol{x}^+ - \boldsymbol{x}^- = \lambda \boldsymbol{w}$
  $|\boldsymbol{x}^+ - \boldsymbol{x}^-| = M$
- Now its easy to get M in terms of $\boldsymbol{w}$ and b

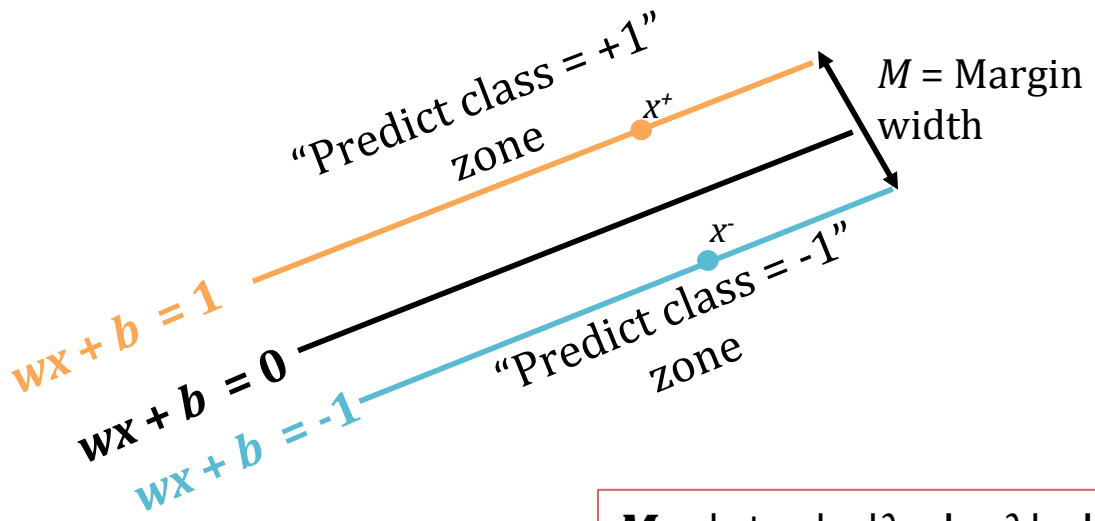
$$\boldsymbol{w} \cdot (\boldsymbol{x}^- + \lambda \boldsymbol{w}) + b = 1$$

$$(\boldsymbol{w} \cdot \boldsymbol{x}^- + b) + \lambda \boldsymbol{w} \cdot \boldsymbol{w} = 1$$

$$-1 + \lambda \boldsymbol{w} \cdot \boldsymbol{w} = 1$$

$$\lambda = \frac{2}{\boldsymbol{w} \cdot \boldsymbol{w}}$$

Computing the margin width



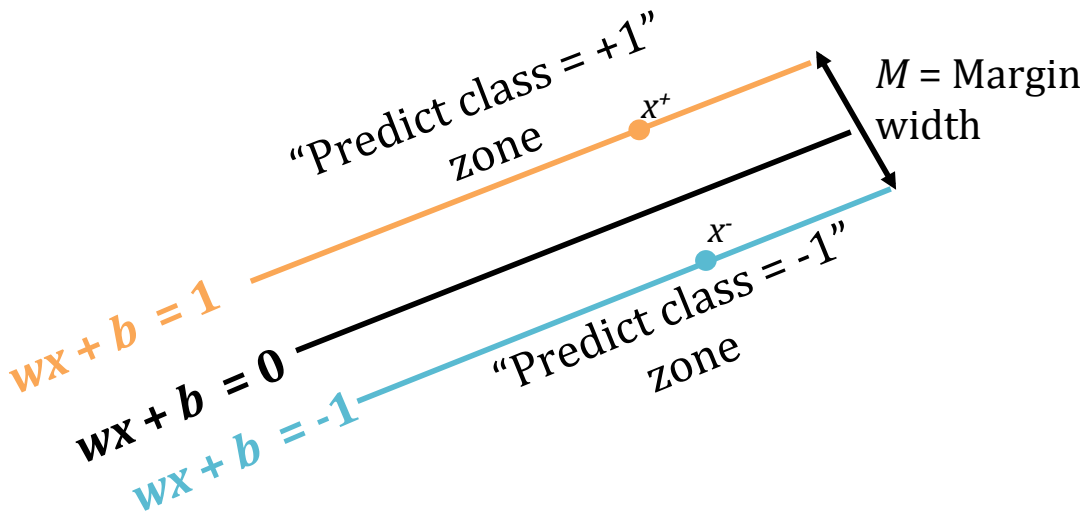
- What we know now:
  $\boldsymbol{w}\boldsymbol{x}^+ + b = 1$
  $\boldsymbol{w}\boldsymbol{x}^- + b = -1$
  $\boldsymbol{x}^+ - \boldsymbol{x}^- = \lambda \boldsymbol{w}$
  $|\boldsymbol{x}^+ - \boldsymbol{x}^-| = M$
  $\lambda = \dfrac{2}{\boldsymbol{w} \cdot \boldsymbol{w}}$

$$\boldsymbol{M} = |\boldsymbol{x}^+ - \boldsymbol{x}^-| = |\lambda \boldsymbol{w}| = \lambda |\boldsymbol{w}|$$

$$= \lambda \sqrt{(\boldsymbol{w} \cdot \boldsymbol{w})}$$

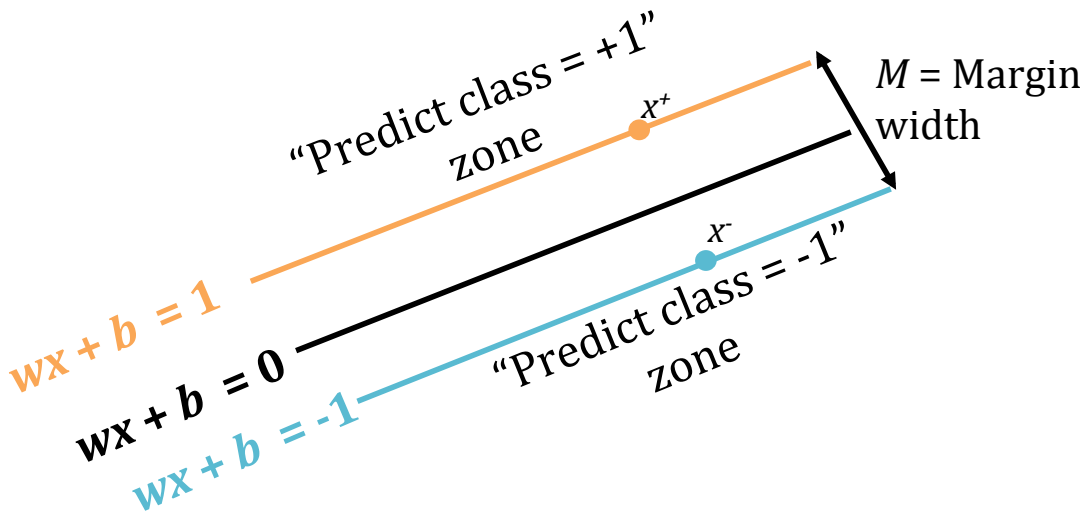
$$= \frac{2\sqrt{(\boldsymbol{w} \cdot \boldsymbol{w})}}{\boldsymbol{w} \cdot \boldsymbol{w}} = \frac{2}{\sqrt{(\boldsymbol{w} \cdot \boldsymbol{w})}}$$

Computing the margin width



- Yay! Just maximize $M = \dfrac{2}{\sqrt{(\boldsymbol{w} \cdot \boldsymbol{w})}}$
- **Wait…what about the data?!?**

Computing the margin width



- Given a guess of $\boldsymbol{w}$ and b we can
  - Compute whether all data points are in the correct half-planes
  - Compute the **width** of the margin
- So now we need to write a program to **search** the space of w's and b's that finds the widest margin that fits all data points
  - Gradient descent, simulated annealing, etc.