Natural Language Processing

Lecture 7: Classification

Some Administrivia

- Read the book
  - (Really Read the Book!)
- Project Teams
  - Let us know if someone drops
- Read the book
  - (Really Read the Book!)

Acknowledging Collaboration

- *MUST* include explicit acknowledgements of people/software/websites in assignments/projects
- Included in written submissions
  - “No collaboration made” (if none)

Notation

- Training examples: $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$
- Their categories: $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)$
- A **classifier** C seeks to map x_i to y_i

$$x \rightarrow \boxed{C} \rightarrow y$$

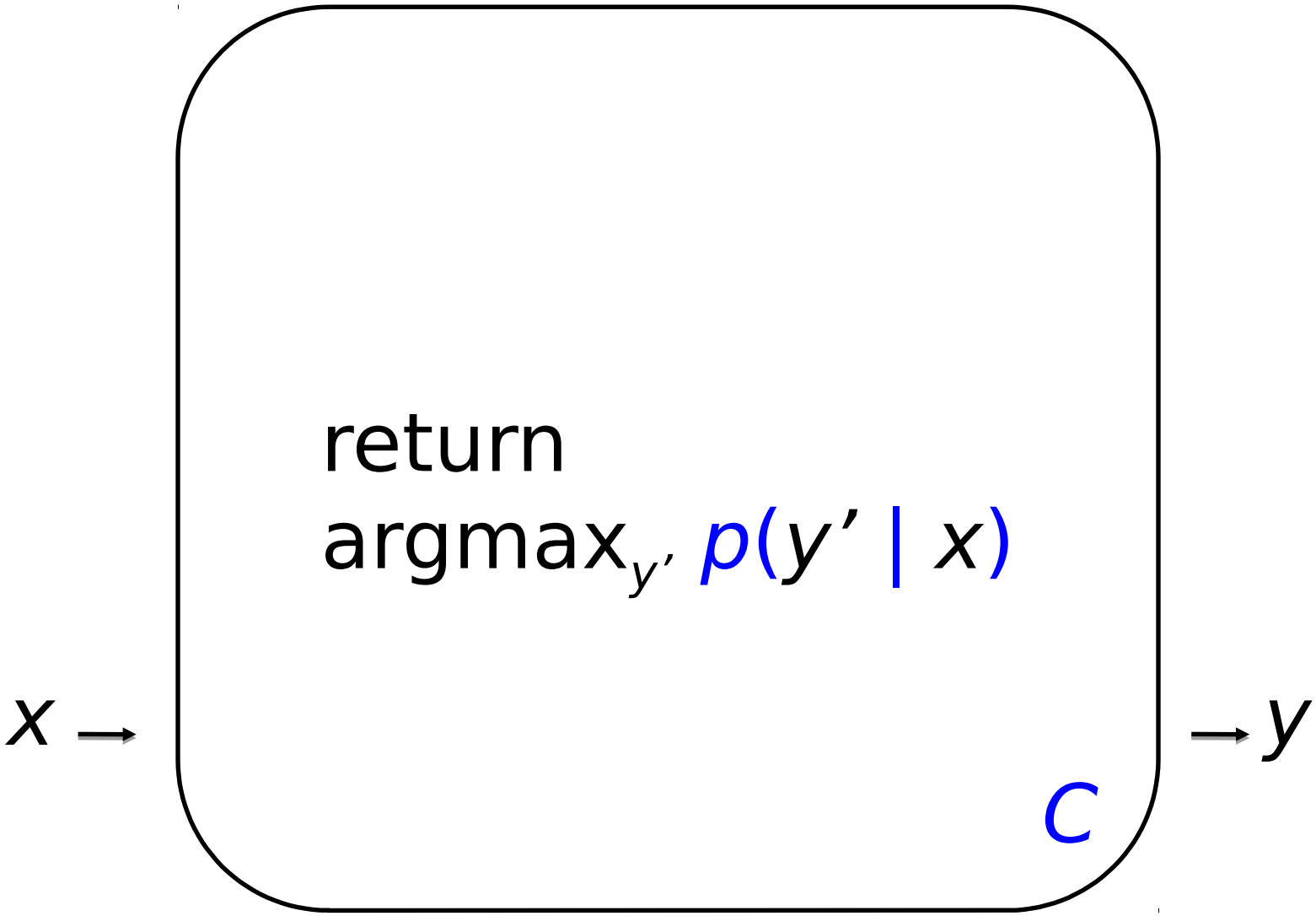
- A **learner** L infers C from $(\boldsymbol{x}, \boldsymbol{y})$

$$\begin{matrix} \boldsymbol{x} \rightarrow \\ \\ \boldsymbol{y} \rightarrow \end{matrix} \boxed{L} \rightarrow \boxed{C}$$

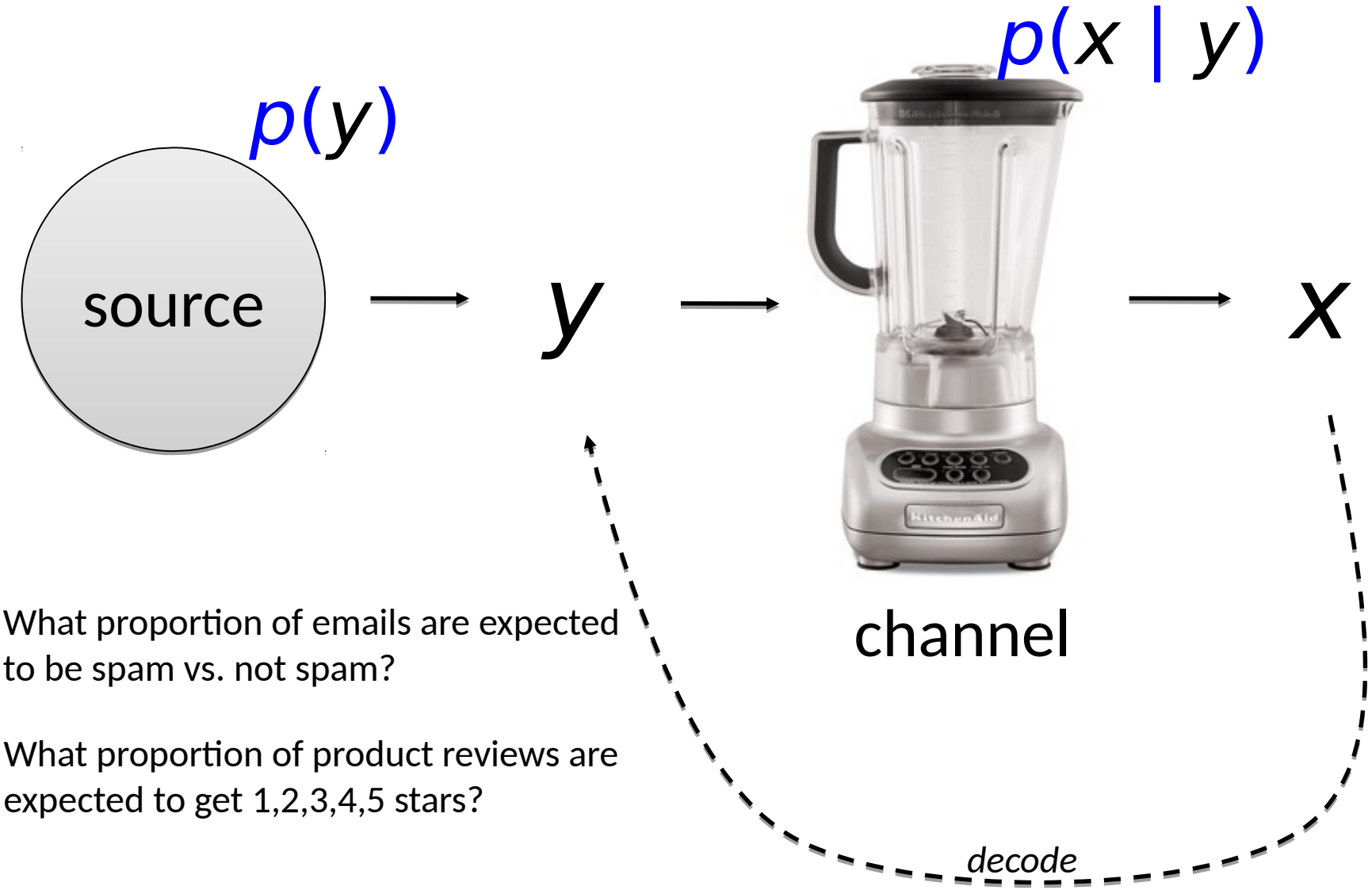
Three different classifiers

- Noisy Channel
- Naïve Bayes
- Linear Models
- Perceptron

Probabilistic Classifiers



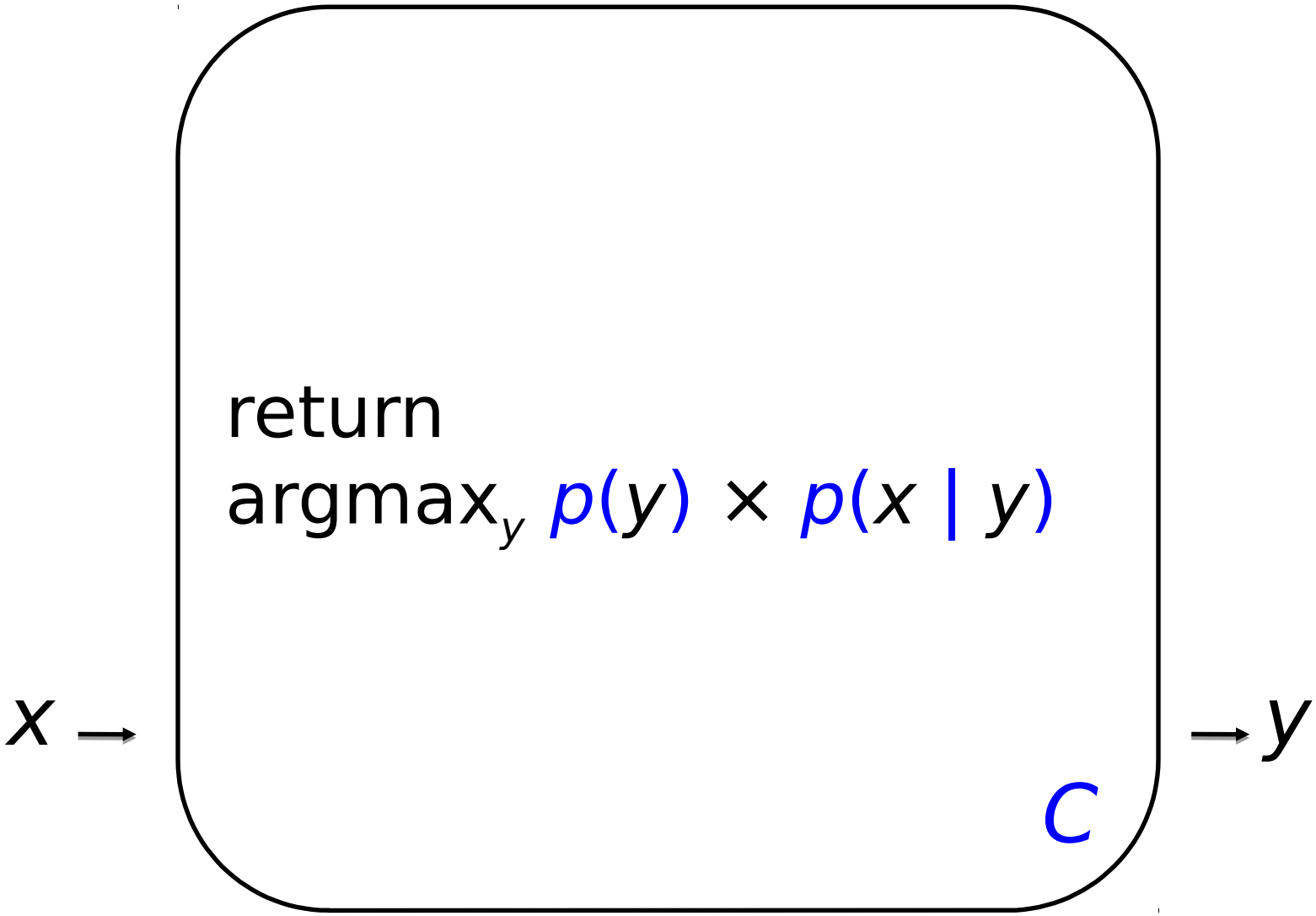
Noisy Channel Model (General)



What proportion of emails are expected to be spam vs. not spam?

What proportion of product reviews are expected to get 1,2,3,4,5 stars?

Noisy Channel Classifiers



Representing Text: *Features*

- Any object $x \in \mathcal{X}$ you might be given to classify can be represented as a vector in a **vector space**
  - Vectors of representing text are often **sparse** and **high-dimensional**
  - Word embeddings (word2vec, ELMo, Bert)
- Designing $\mathbf{\Phi}$ ("Feature engineering")
  - What information do you need to solve the problem?
  - What information do you need to avoid mistakes?
  - Very common: bag-of-words

Example: Spam Detection

- A training set of email messages
  - Spam or Not-Spam
- A set of features for each message
  - ???

Example: Spam Detection

- A training set of email messages
  - Spam or Not-Spam
- A set of features for each message
  - Bag of words
    - For each word: Number of occurrences
    - Nigerian Prince, email quota full, won ONE HUNDRED MILLION DOLLARS
  - From someone you know
  - A reply
  - From @andrew.cmu.edu (your domain)

Example: Movie Ratings

- A training set of movie reviews
  - Stars 1-5
- A set of features for each message
  - Bag of words
    - For each word: Number of occurrences
      - The: 234; of: 26 …; wars: 3; nebuchadnezzar: 0;
    - Excellent, sucks, blockbuster, biggest, Star Wars, Disney, Adam Sandler
  - …

Feature Representation

- Discrete vs Continuous
  - Finite number of categories
  - Floating point scale
- Many classifiers require continuous features
- Convert discrete to continuous
  - “very bad”, “bad”, “okay”, “good”, “very good” →
  - 1-5 linear scale, **if** its a scale
  - Or [1,0,0,0,0], [0,1,0,0,0], [0,0,1,0,0] …
  - “one hot” vector **if** its **not** a scale

Naïve Bayes Classifier

$\phi_j \leftarrow [\mathbf{\Phi}(x)]_j$
return
$\text{argmax}_{y'}\ p(y') \times \Pi_j\ p(\phi_j \mid y')$

$x \rightarrow$ $\rightarrow y$

C

Naïve Bayes Learner

$\boldsymbol{x} \rightarrow$

$\boldsymbol{y} \rightarrow$

$$\forall y,\ p(y) \leftarrow \frac{\mathrm{count}(y)}{N}$$

$$\forall y,\ \forall j,\ \forall f,\quad p(\phi_j(x) = f \mid y) \leftarrow \frac{\mathrm{count}(f, y)}{\mathrm{count}(y)}$$

L

$\rightarrow$ p

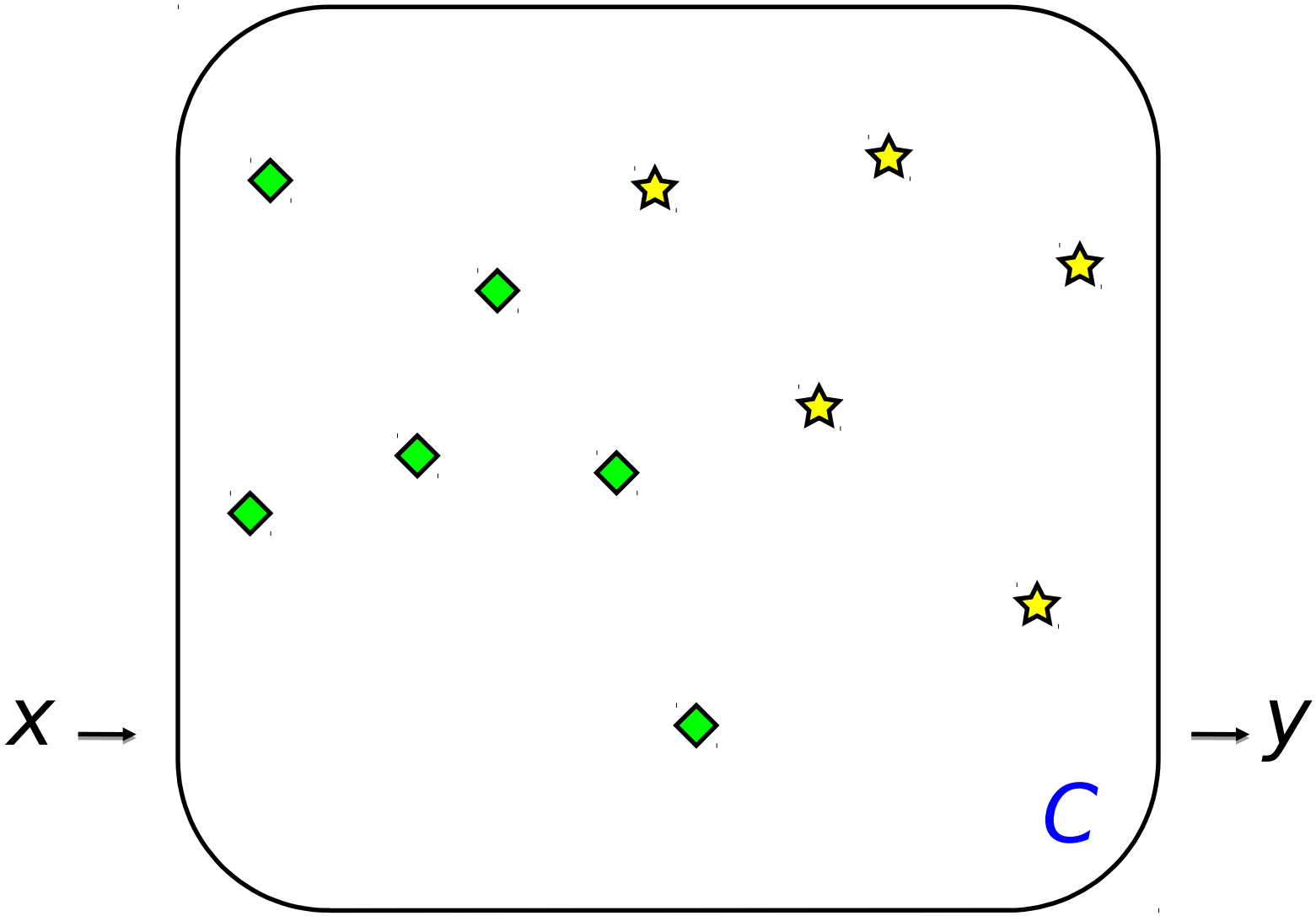
Linear Classifiers

C:

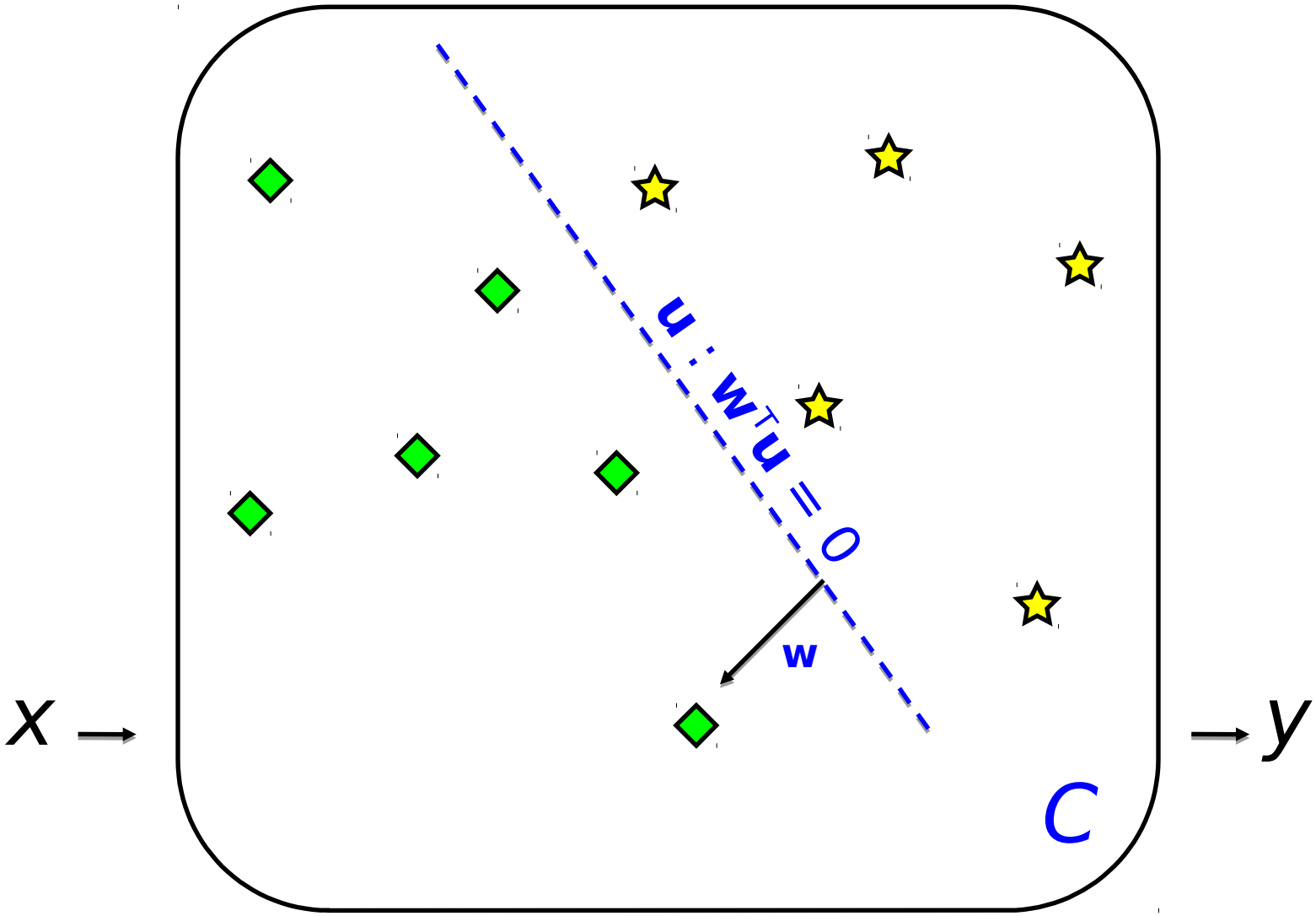
1. Use $\mathbf{\Phi}(x)$ to map x onto a **real-valued** feature space.
2. Calculate the linear score $z = \mathbf{w}^{\top} \mathbf{\Phi}(x)$.
3. If $z > 0$, then return y = **YES**, else y = **NO**.

$$x \rightarrow \boxed{C} \rightarrow y$$

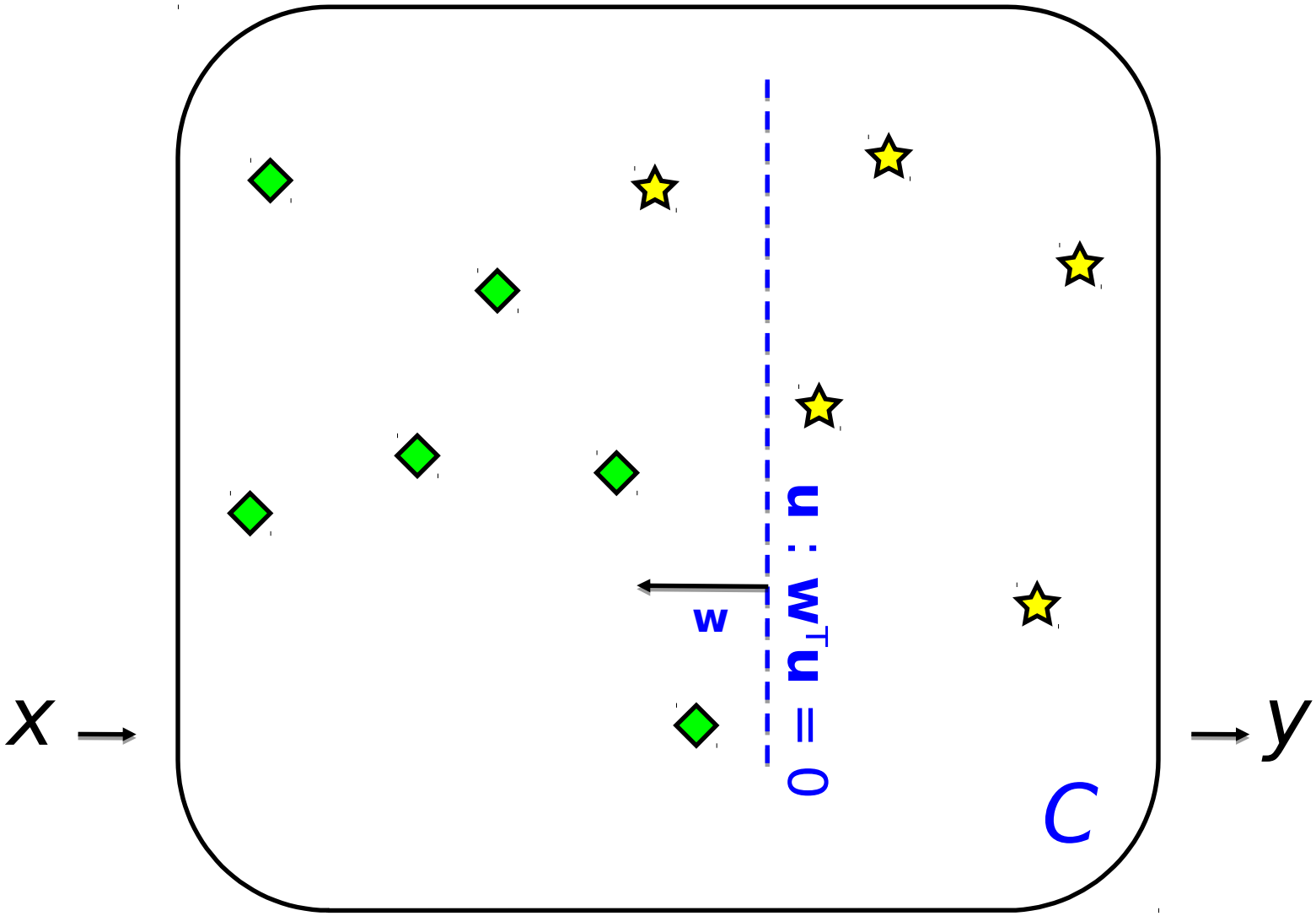
Linear Classifiers



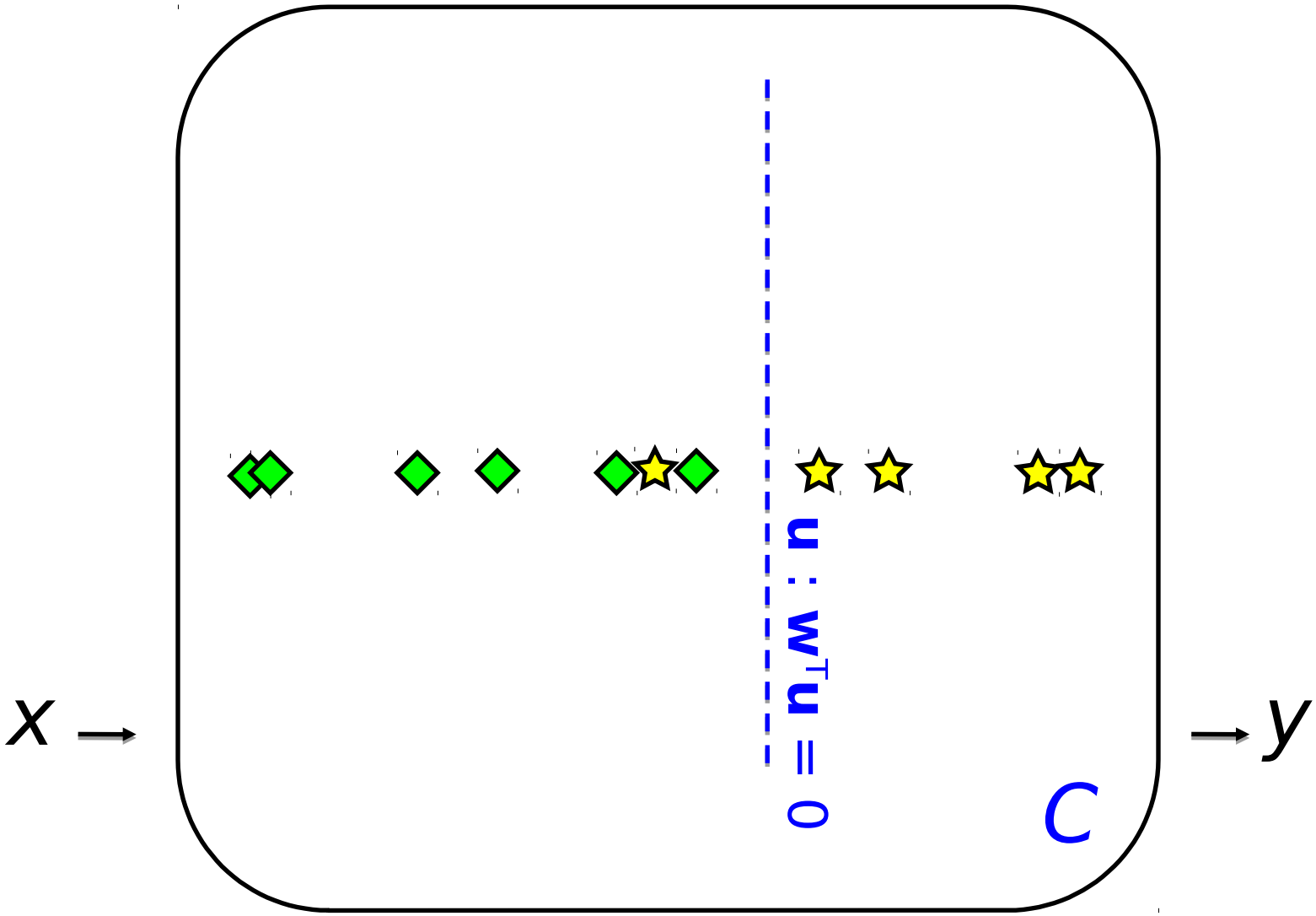
Linear Classifiers



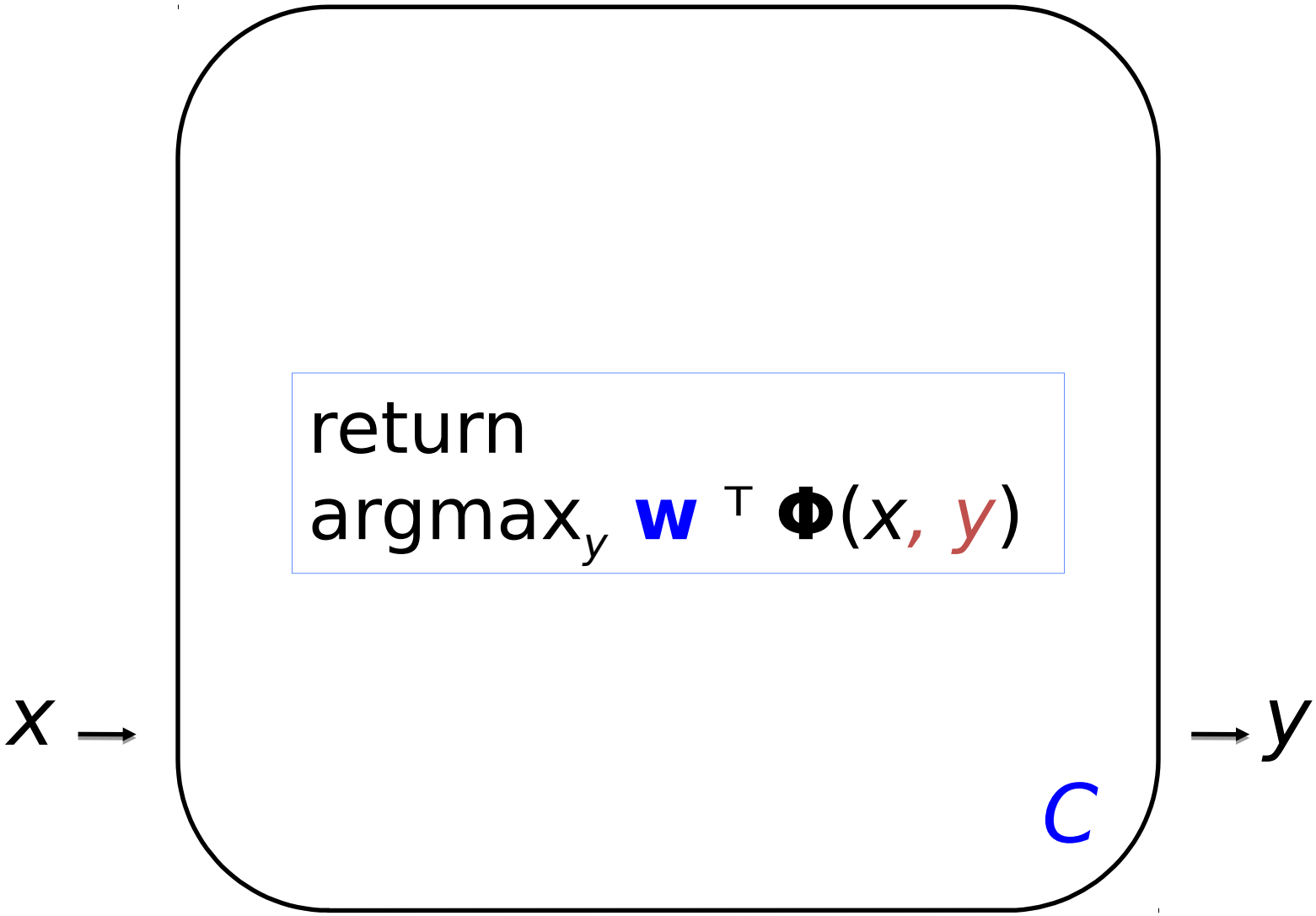
Linear Classifiers



Linear Classifiers



Linear Classifiers (> 2 Classes)



Perceptron Learner

$\boldsymbol{x} \rightarrow$

$\boldsymbol{y} \rightarrow$

$\mathbf{w} \leftarrow \mathbf{0}$
for $t = 1 \ldots T$:
    select (x_t, y_t)
    # run current classifier
    $y \leftarrow \text{argmax}_{y'}\ \mathbf{w}^{\top} \boldsymbol{\Phi}(x, y')$

    if $y \mathrel{!=} y_t$ then *# mistake*
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\, [\boldsymbol{\Phi}(x_t, y_t) - \boldsymbol{\Phi}(x_t, y)]$
return $\mathbf{w}$

L

$\rightarrow$ $\mathbf{w}$

Perceptrons

- Single Layer Perceptron
  - Multilayer perceptrons used in neural networks
  - Deep Neural Nets are > 3ish layers.
- Weights are found iteratively
  - May converge (or not) depending on problem
  - May find local maximum
  - May take longer to train

Which Classifier to Use?

- The Best one for the Problem
- Machine Learning Specialist
  - Best algorithm to train from given features
- Language Technologist
  - Feature engineer the best features
- NLP Specialist
  - Care about *both*
  - What features should be there, and what algorithm will be able to exploit those features
- Engineer
  - Why didn’t it work (what info is missing)