Natural Language Processing

Lecture 8: Classification

Three Spelling Problems

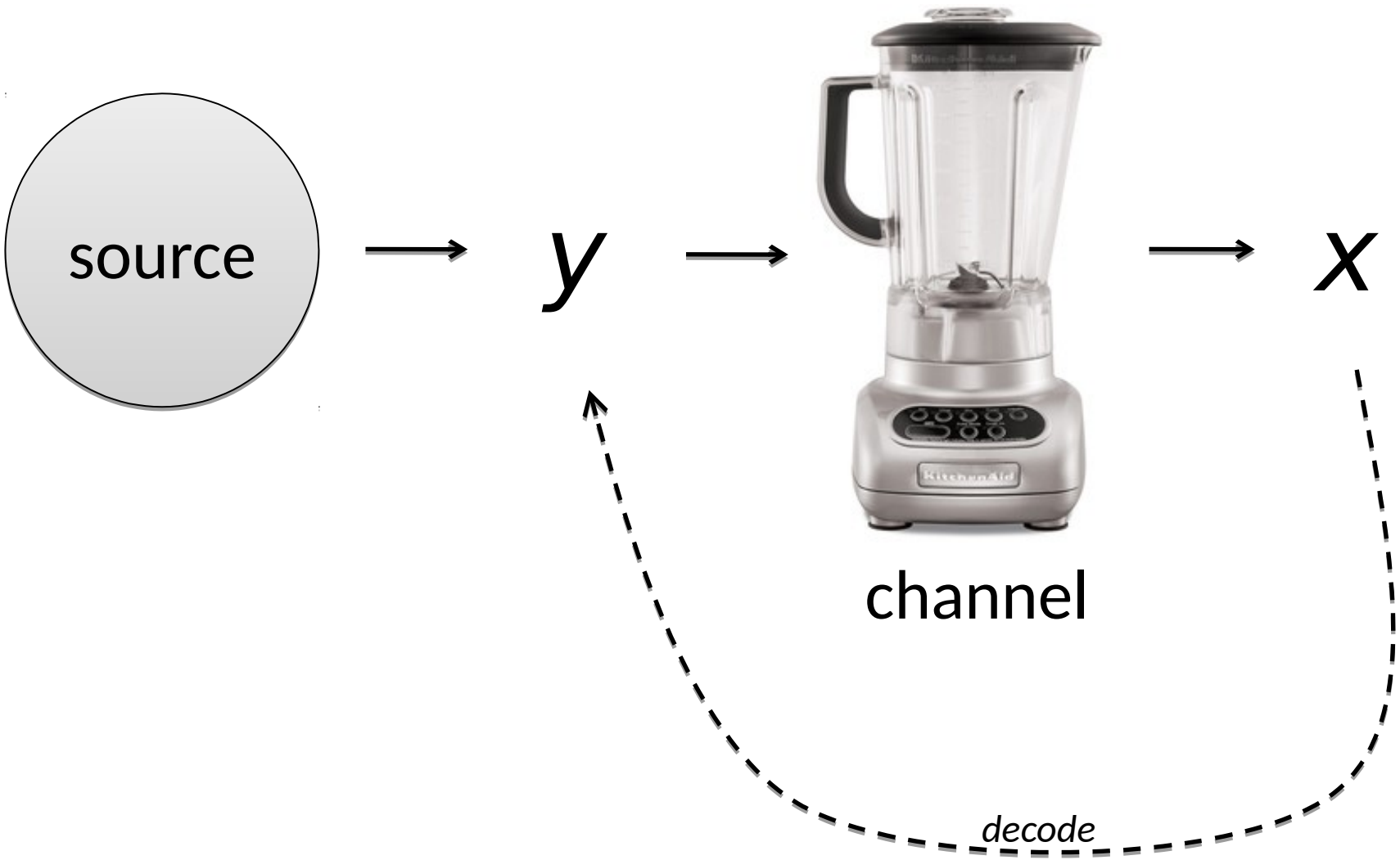
- ✓ Detecting isolated non-words
- ✓ Fixing isolated non-words
3. Fixing errors in context

Kernighan's Model: A Noisy Channel

acress

c	freq(*c*)	*p*(*t* | *c*)	%
actress	1343	p(delete t)	37
cress	0	p(delete a)	0
caress	4	p(transpose a & c)	0
access	2280	p(substitute r for c)	0
across	8436	p(substitute e for o)	18
acres	2879	p(delete s)	21
acres	2879	p(delete s)	23

Noisy Channel Model (General)



Classification

Notation

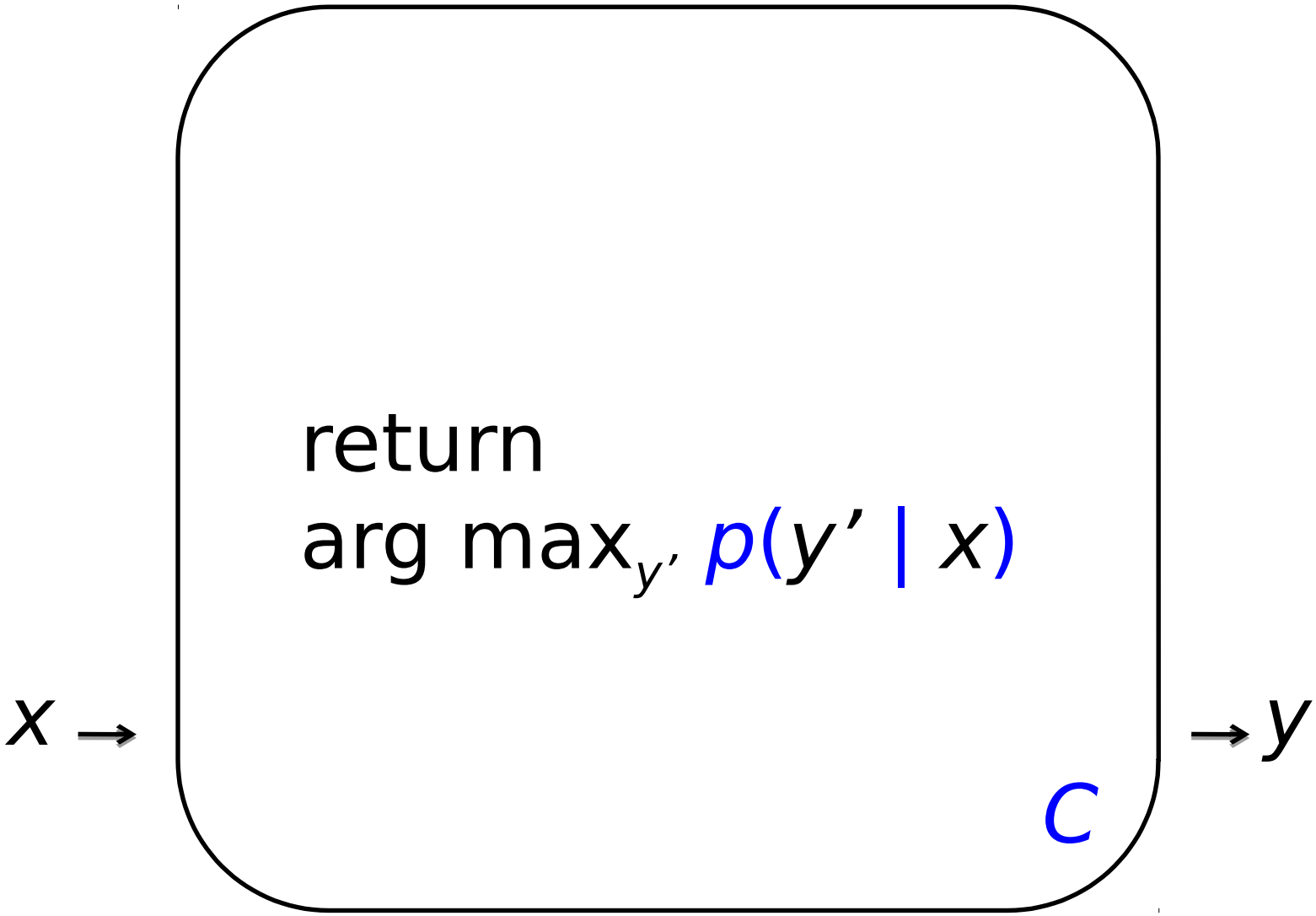
- Training examples: $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$
- Their categories: $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)$
- A **classifier** C seeks to map x_i to y_i

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

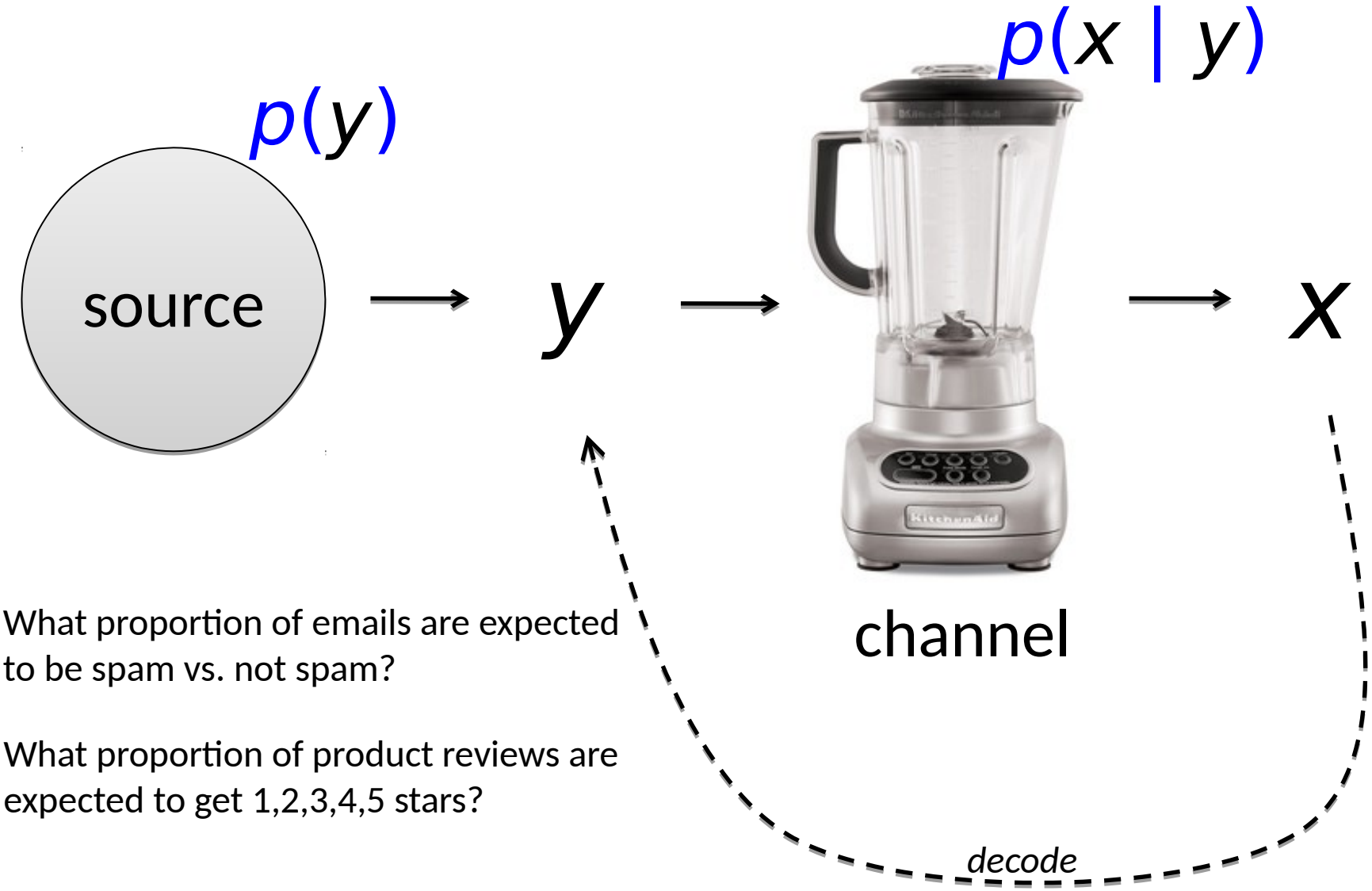
- A **learner** L infers C from $(\boldsymbol{x}, \boldsymbol{y})$

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

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

$x \rightarrow$

return
$\text{argmax}_y \; p(y) \times p(x \mid y)$

C

$\rightarrow y$

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**
- 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

Naïve Bayes Classifier

$\phi_j \leftarrow [\mathbf{\Phi}(x)]_j$

return

$\arg\max_{y'} p(y') \times \Pi_j \, p(\phi_j \mid y')$

$x \rightarrow$ $\rightarrow y$

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

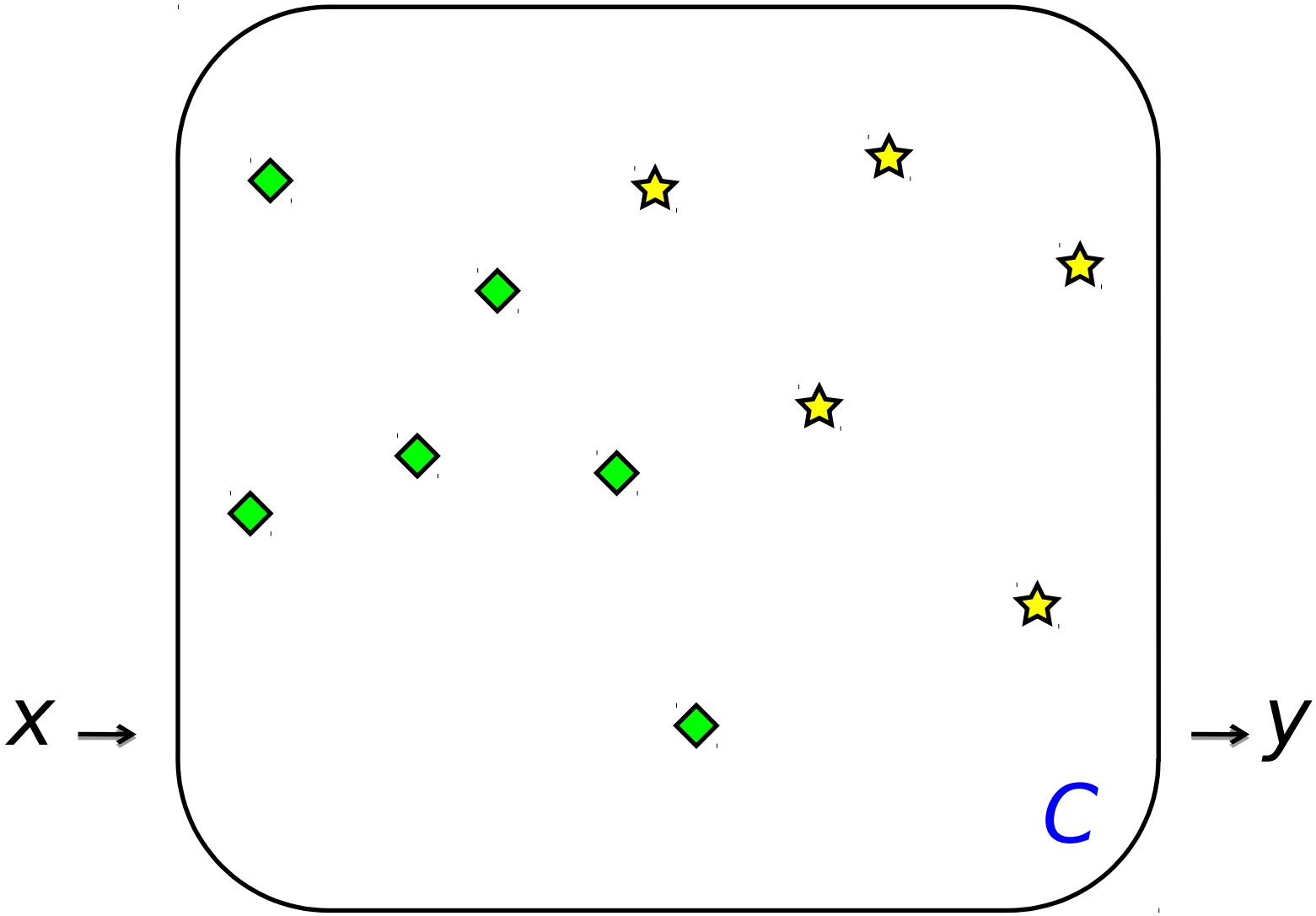
Linear Classifiers

C:

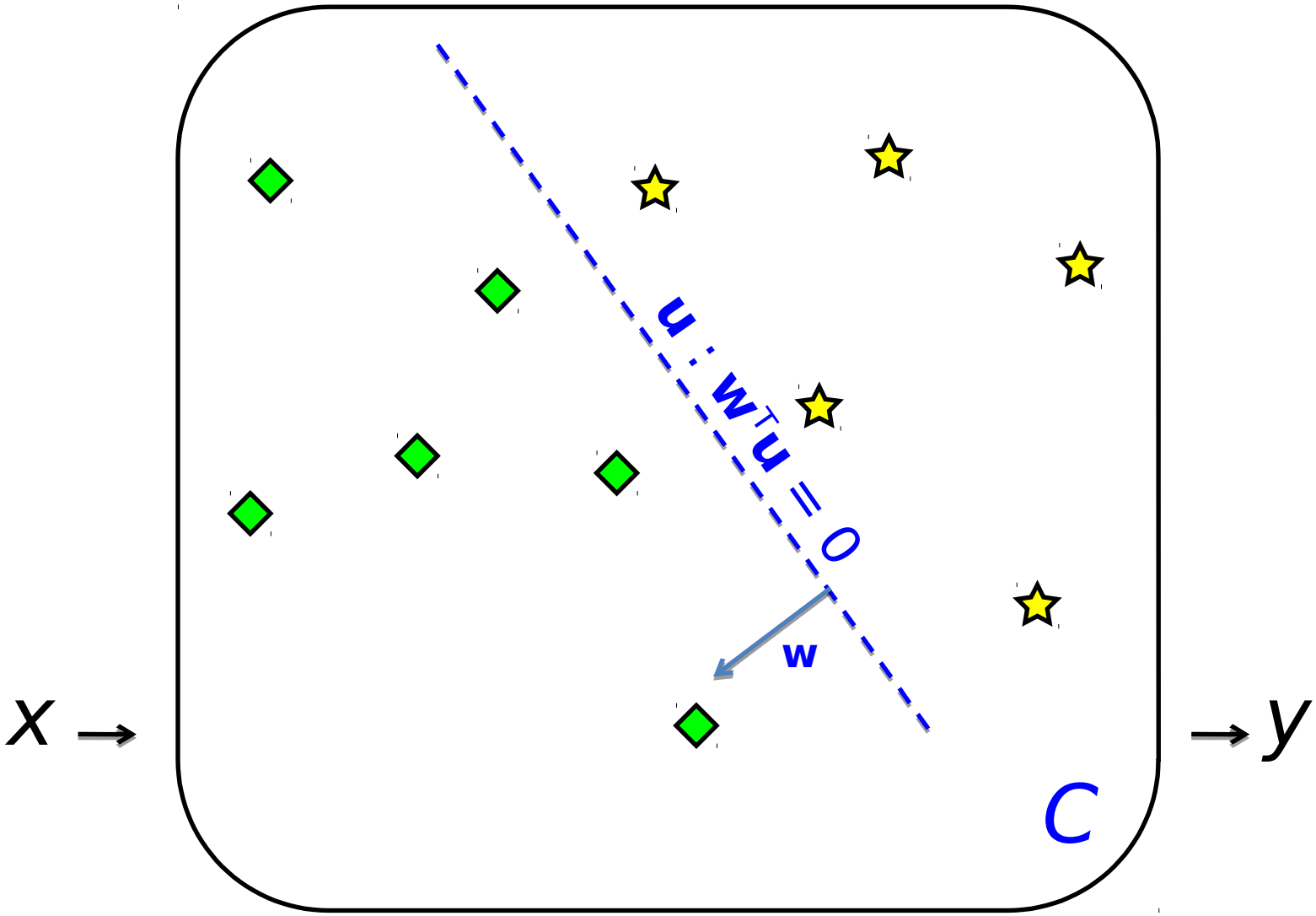
1. Use $\mathbf{\Phi}(x)$ to map x onto a **real-valued** feature space.
2. Calculate the linear score $z = \mathbf{w}^{\top} \mathbf{\Phi}(x)$.
3. If $z > 0$, then return y = **YES**, else y = **NO**.

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

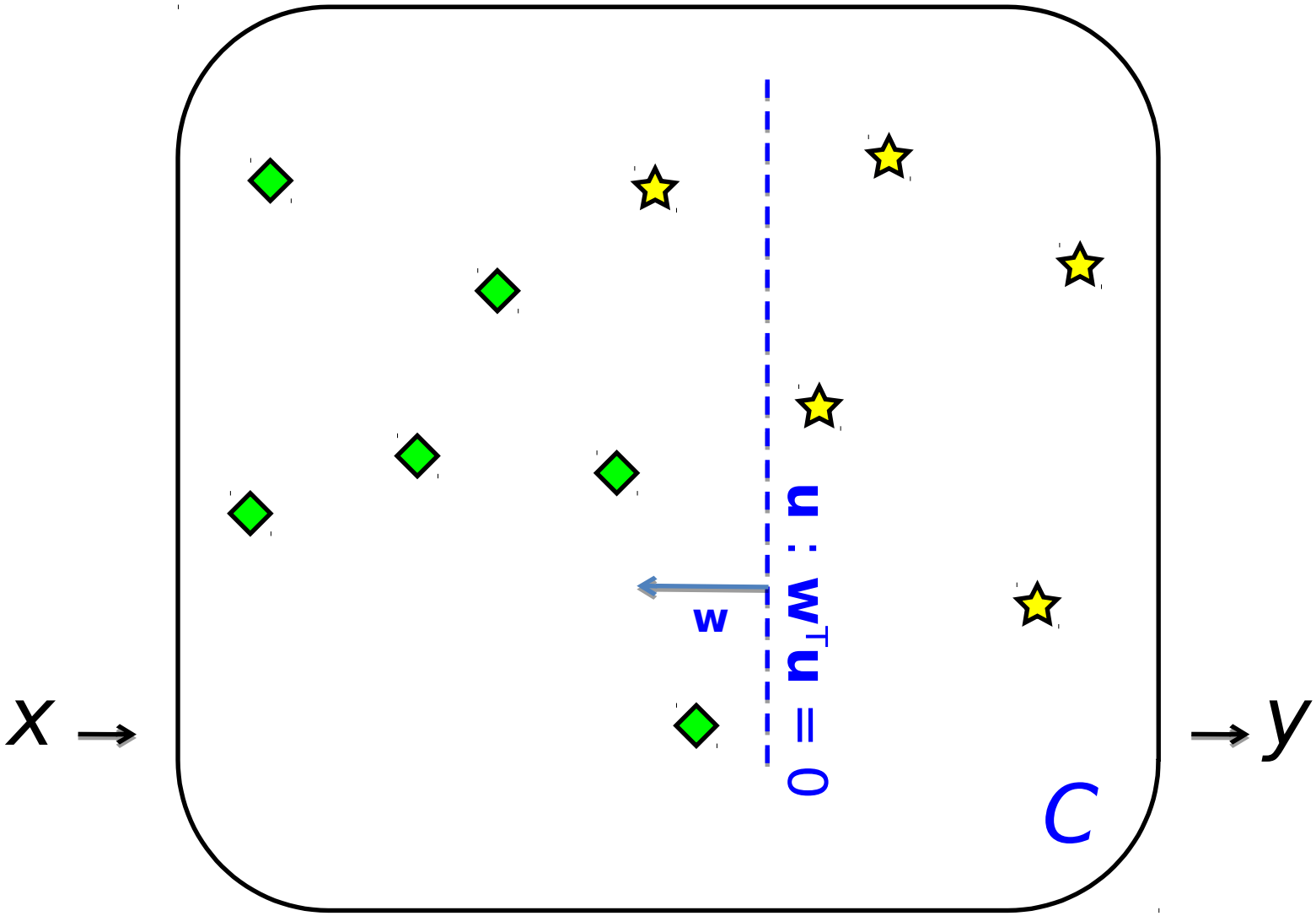
Linear Classifiers



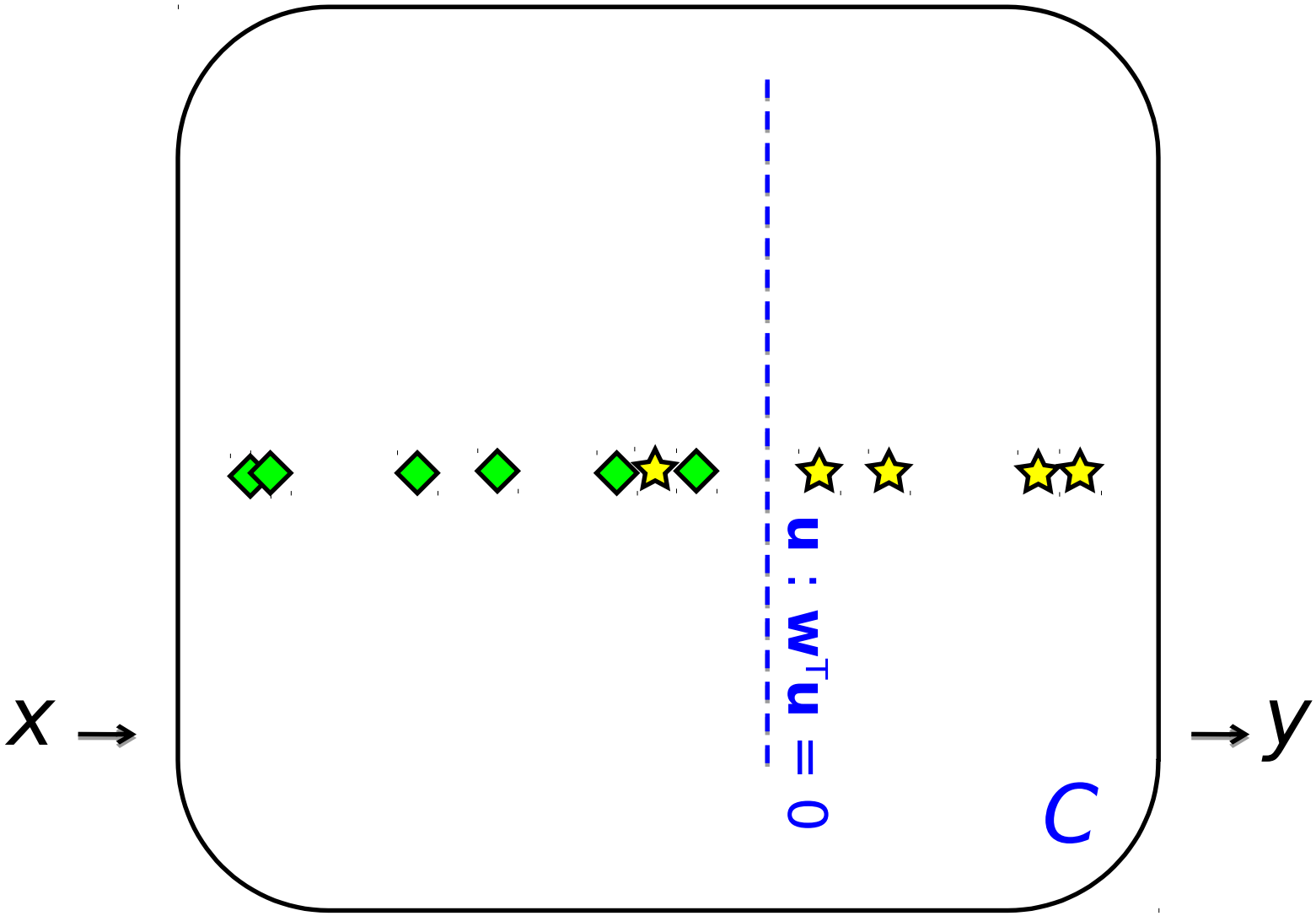
Linear Classifiers



Linear Classifiers



Linear Classifiers



Linear Classifiers (> 2 Classes)

$x \rightarrow$

return
$\arg\max_y \mathbf{w}^\top \mathbf{\Phi}(x, y)$

C

$\rightarrow y$

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

    $y \leftarrow \arg\max_{y'} \mathbf{w}^{\top} \boldsymbol{\Phi}(x, y')$

    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

$\rightarrow$ $\mathbf{w}$

Classifiers

- Noisy Channel: Two components:
  - Most probable times prior probability
- Linear Classifiers: feature times weight
- Naive Bayes: Independent feature/weights
- Perceptron: back-propagate loss to input weights
- Choose your input features well
- Use the one that works best