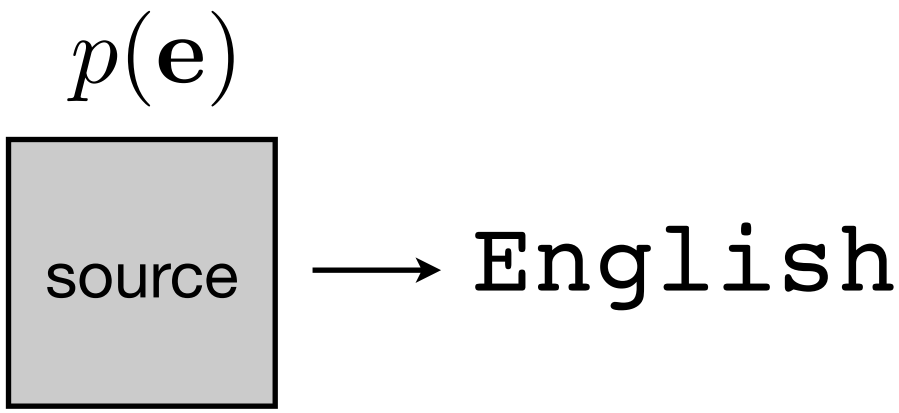
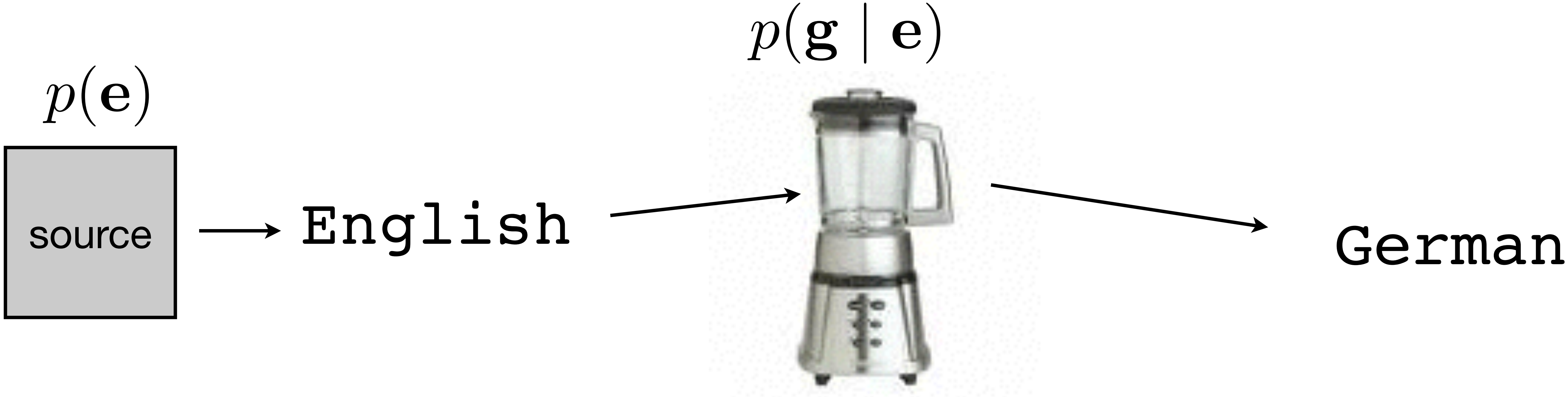
Discriminative Training

March 4, 2014

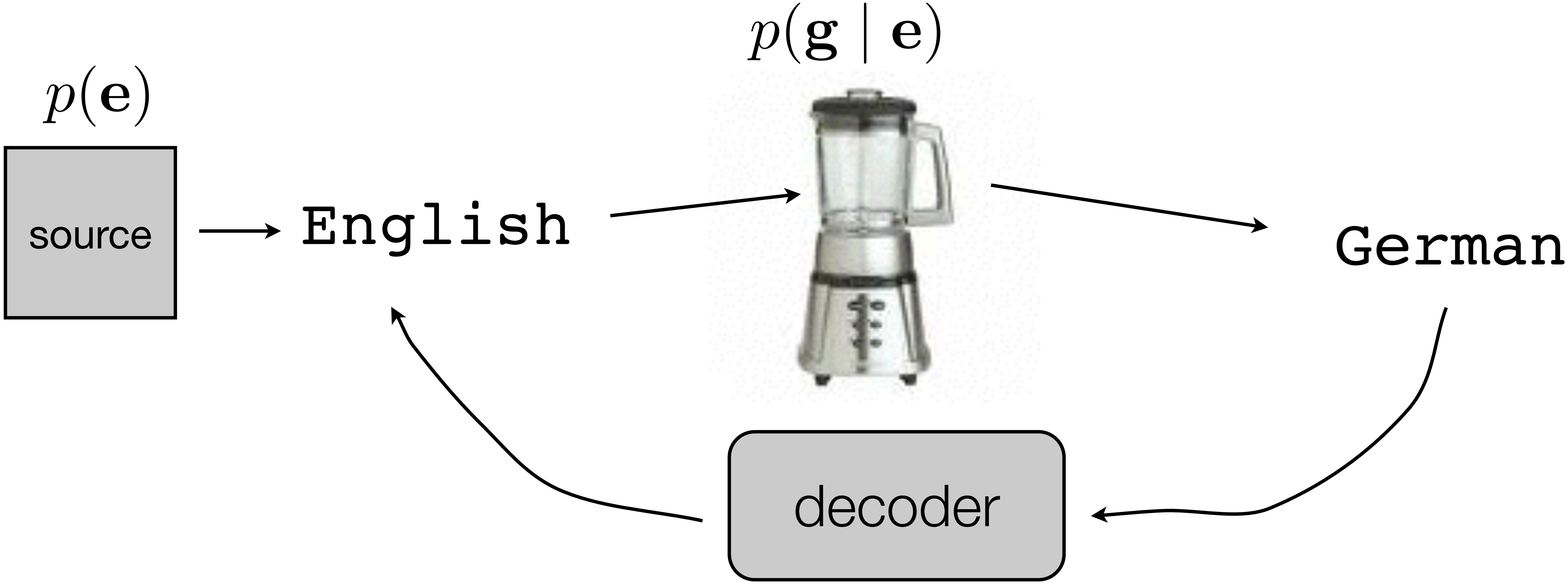
Noisy Channels Again



Noisy Channels Again



Noisy Channels Again



$$\begin{aligned}
\mathbf{e}^* &= \arg\max_{\mathbf{e}} p(\mathbf{e} \mid \mathbf{g}) \\
&= \arg\max_{\mathbf{e}} \frac{p(\mathbf{g} \mid \mathbf{e}) \times p(\mathbf{e})}{p(\mathbf{g})} \\
&= \arg\max_{\mathbf{e}} p(\mathbf{g} \mid \mathbf{e}) \times p(\mathbf{e})
\end{aligned}$$

Noisy Channels Again

$$
\begin{aligned}
\mathbf{e}^* &= \arg\max_{\mathbf{e}} p(\mathbf{e} \mid \mathbf{g}) \\
&= \arg\max_{\mathbf{e}} \frac{p(\mathbf{g} \mid \mathbf{e}) \times p(\mathbf{e})}{p(\mathbf{g})} \\
&= \arg\max_{\mathbf{e}} p(\mathbf{g} \mid \mathbf{e}) \times p(\mathbf{e})
\end{aligned}
$$

Noisy Channels Again

$$\begin{aligned}
\mathbf{e}^* &= \arg\max_{\mathbf{e}} p(\mathbf{e} \mid \mathbf{g}) \\
&= \arg\max_{\mathbf{e}} \frac{p(\mathbf{g} \mid \mathbf{e}) \times p(\mathbf{e})}{p(\mathbf{g})} \\
&= \arg\max_{\mathbf{e}} p(\mathbf{g} \mid \mathbf{e}) \times p(\mathbf{e}) \\
&= \arg\max_{\mathbf{e}} \log p(\mathbf{g} \mid \mathbf{e}) + \log p(\mathbf{e})
\end{aligned}$$

Noisy Channels Again

$$\begin{aligned}
\mathbf{e}^* &= \arg\max_{\mathbf{e}} p(\mathbf{e} \mid \mathbf{g}) \\
&= \arg\max_{\mathbf{e}} \frac{p(\mathbf{g} \mid \mathbf{e}) \times p(\mathbf{e})}{p(\mathbf{g})} \\
&= \arg\max_{\mathbf{e}} p(\mathbf{g} \mid \mathbf{e}) \times p(\mathbf{e}) \\
&= \arg\max_{\mathbf{e}} \log p(\mathbf{g} \mid \mathbf{e}) + \log p(\mathbf{e}) \\
&= \arg\max_{\mathbf{e}} \underbrace{\begin{bmatrix} 1 \\ 1 \end{bmatrix}^\top}_{\mathbf{w}^\top} \underbrace{\begin{bmatrix} \log p(\mathbf{g} \mid \mathbf{e}) \\ \log p(\mathbf{e}) \end{bmatrix}}_{\mathbf{h}(\mathbf{g},\mathbf{e})}
\end{aligned}$$

Log-linear Model

Discriminative modeling

- Depart from generative modeling
- Goal:
  - Directly optimize for translation performance by discriminating between good/bad translation, and adjusting our model to give preference to good translations

Discriminative modeling

- Possible translations of a sentence are represented using a set of features **h**
- Each feature $\mathbf{h_i}$ derives from one property of the translation
- Its feature weight $\mathbf{w_i}$ indicates its relative importance
- The feature weights and feature values are combined into an overall score

Discriminative modeling

- **Re-ranking** - a two stage process
  - 1: generate a candidate set of translations
  - 2: add additional features and re-score the candidates according to the discriminative model

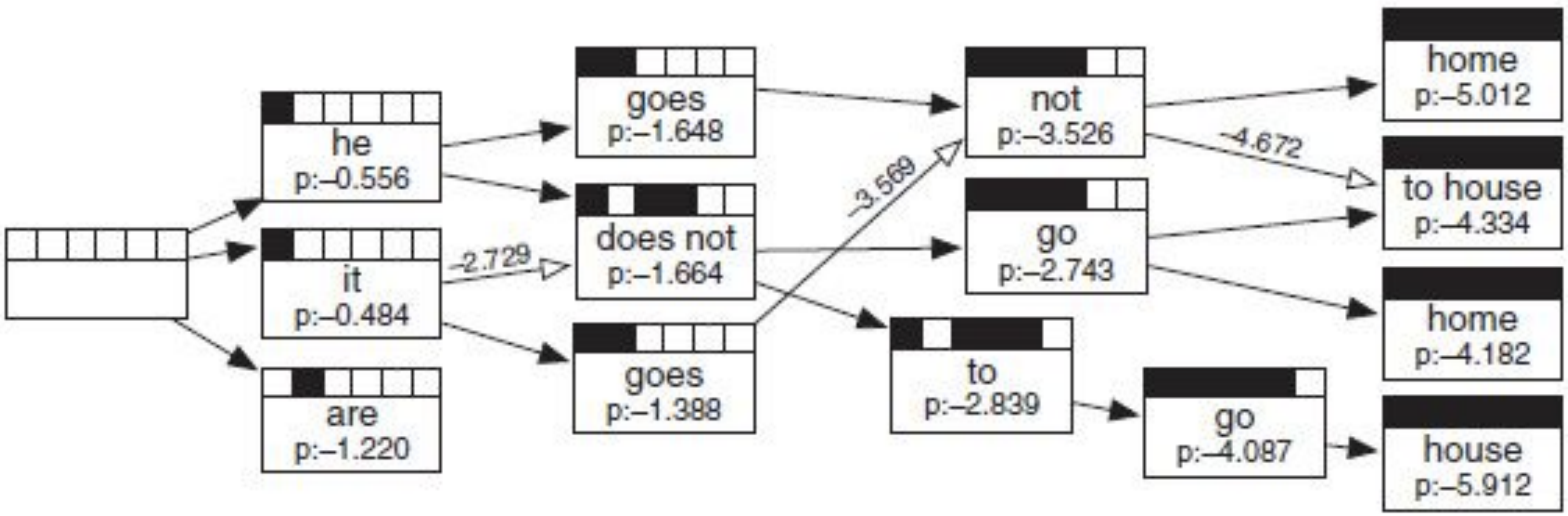
Discriminative modeling

- **Optimize the features used in decoding**
  - Use more features than just the language model and translation model
  - Tune their parameters and use the weights during decoding
- We can optimize a small handful of features, or we can use large-scale discriminative training for millions of features

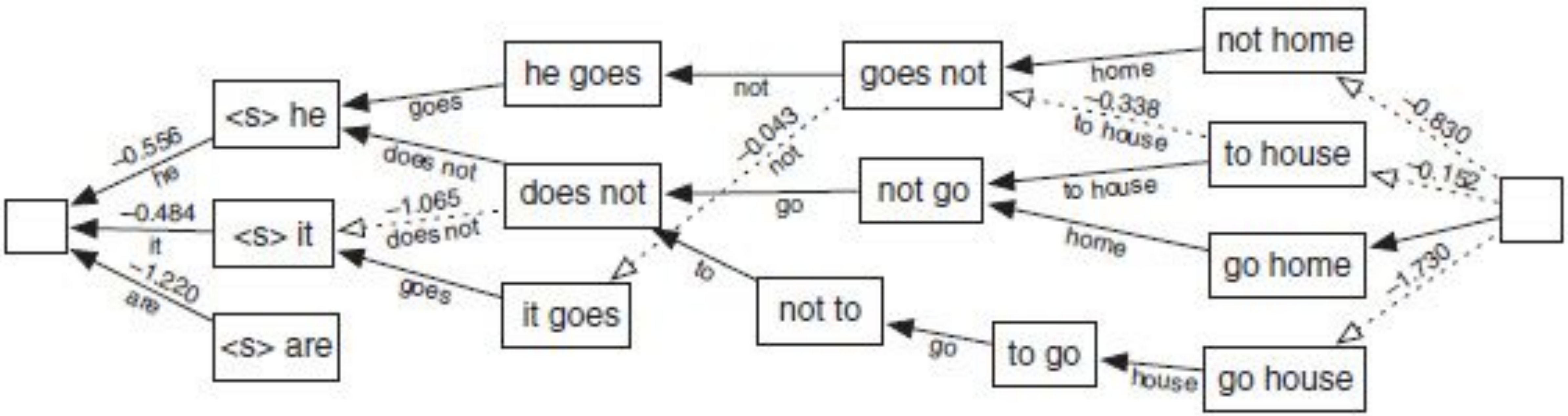
n-best translations

- Discriminative training operates on candidate translations of a sentence
- In theory we could enumerate all possible translations, but in practice there are too many
- Typically, we operate on the 1,000-best or the 10,000-best translations, or the n-best

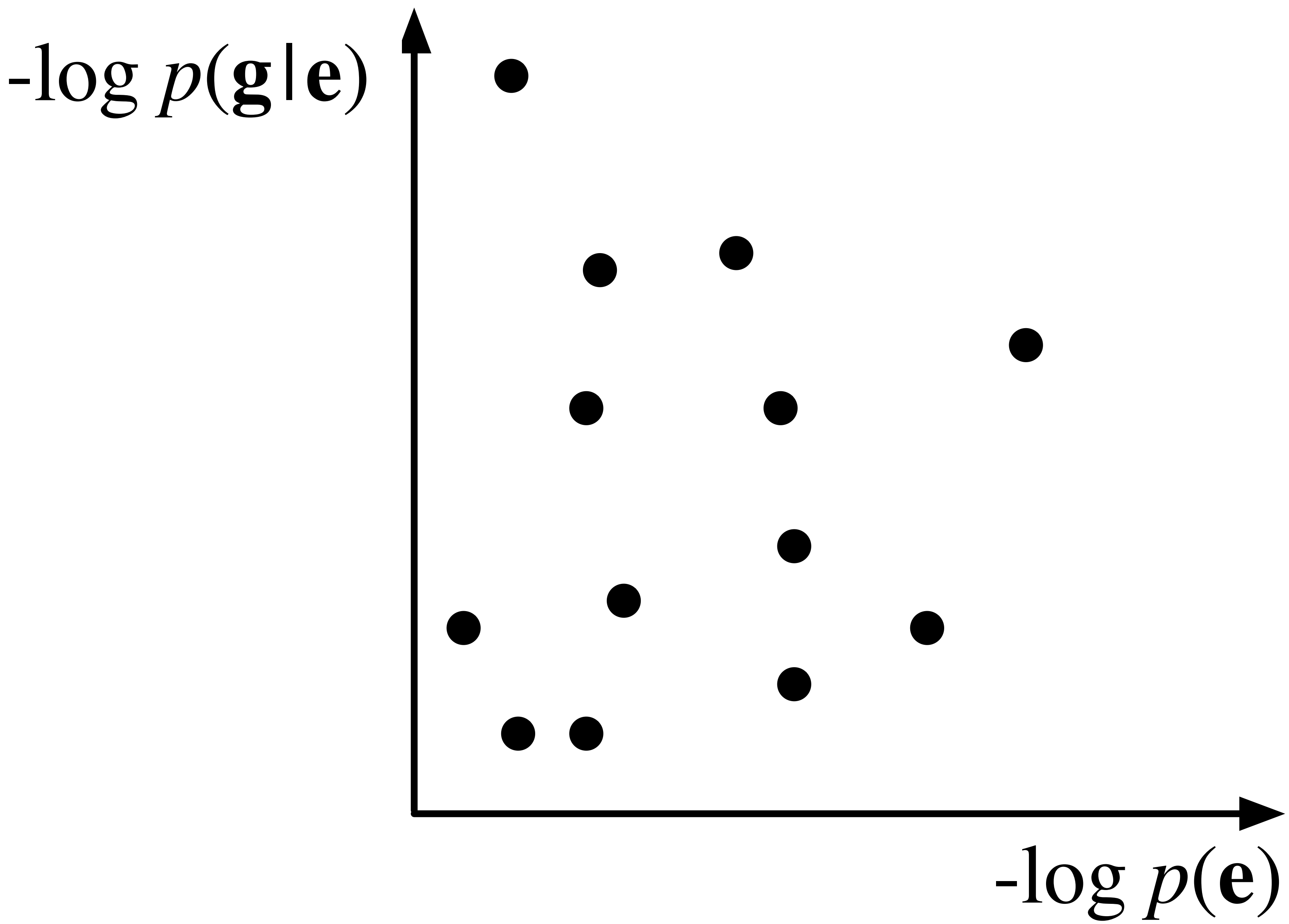
n-best translations



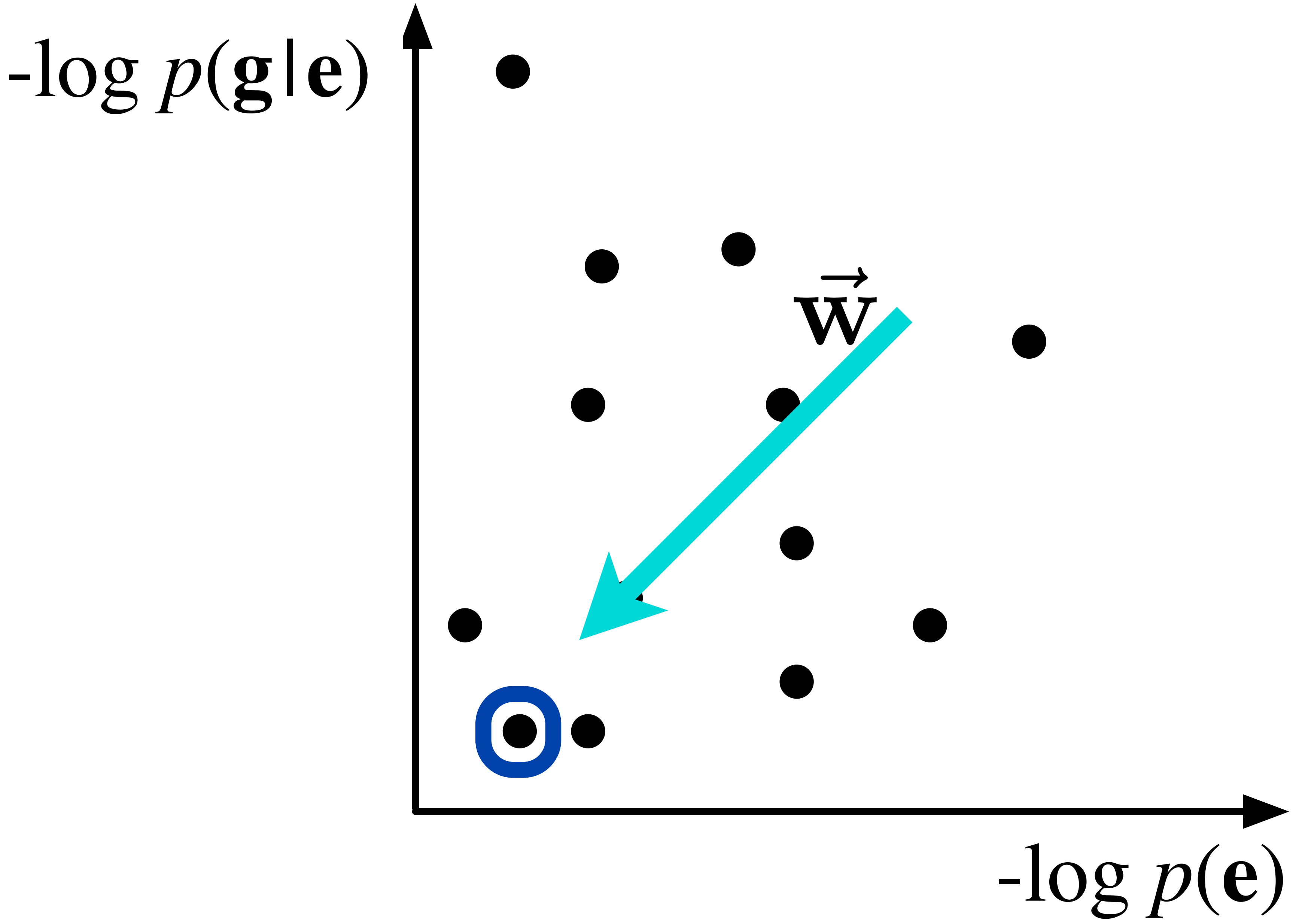
n-best translations



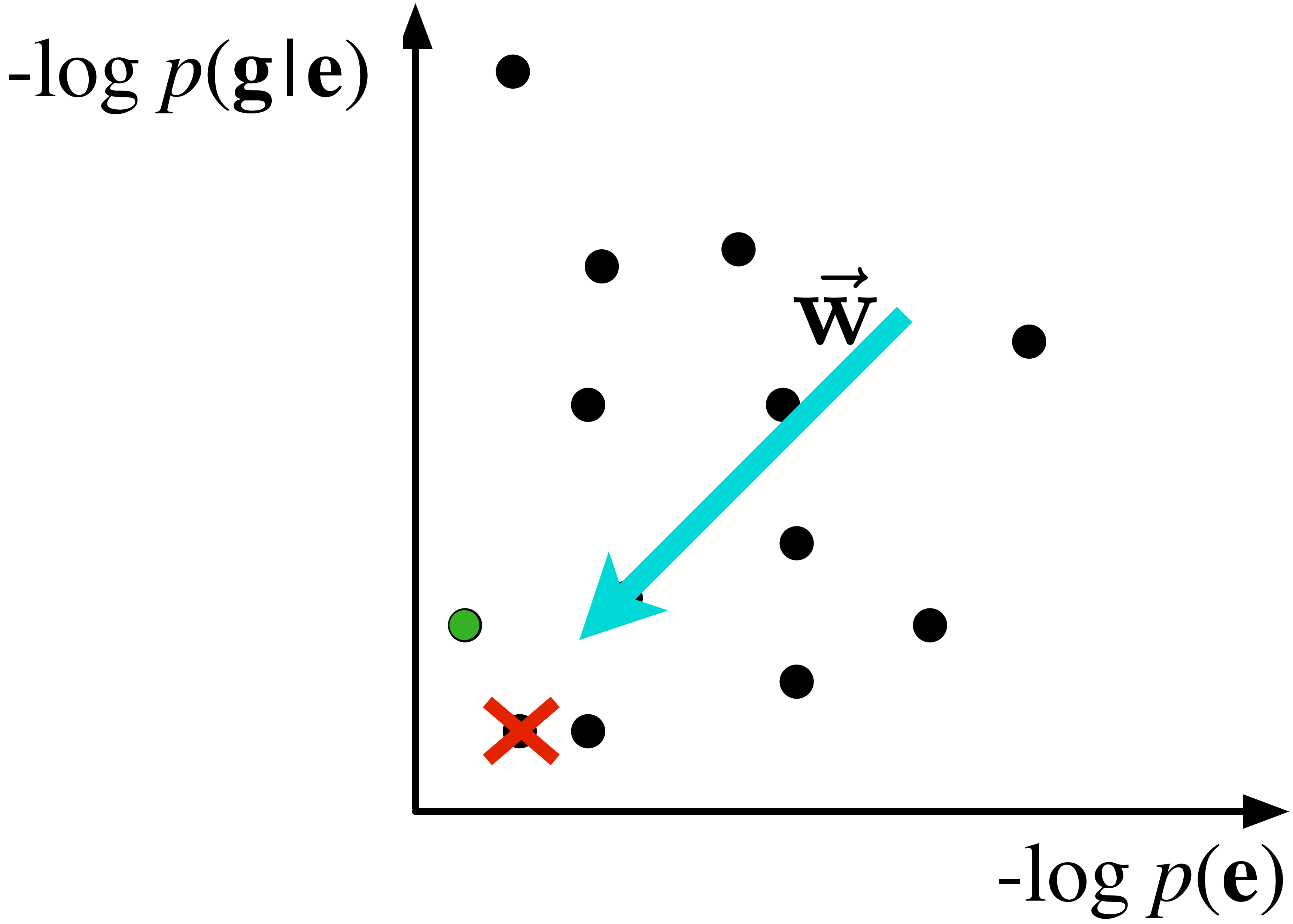
The Noisy Channel



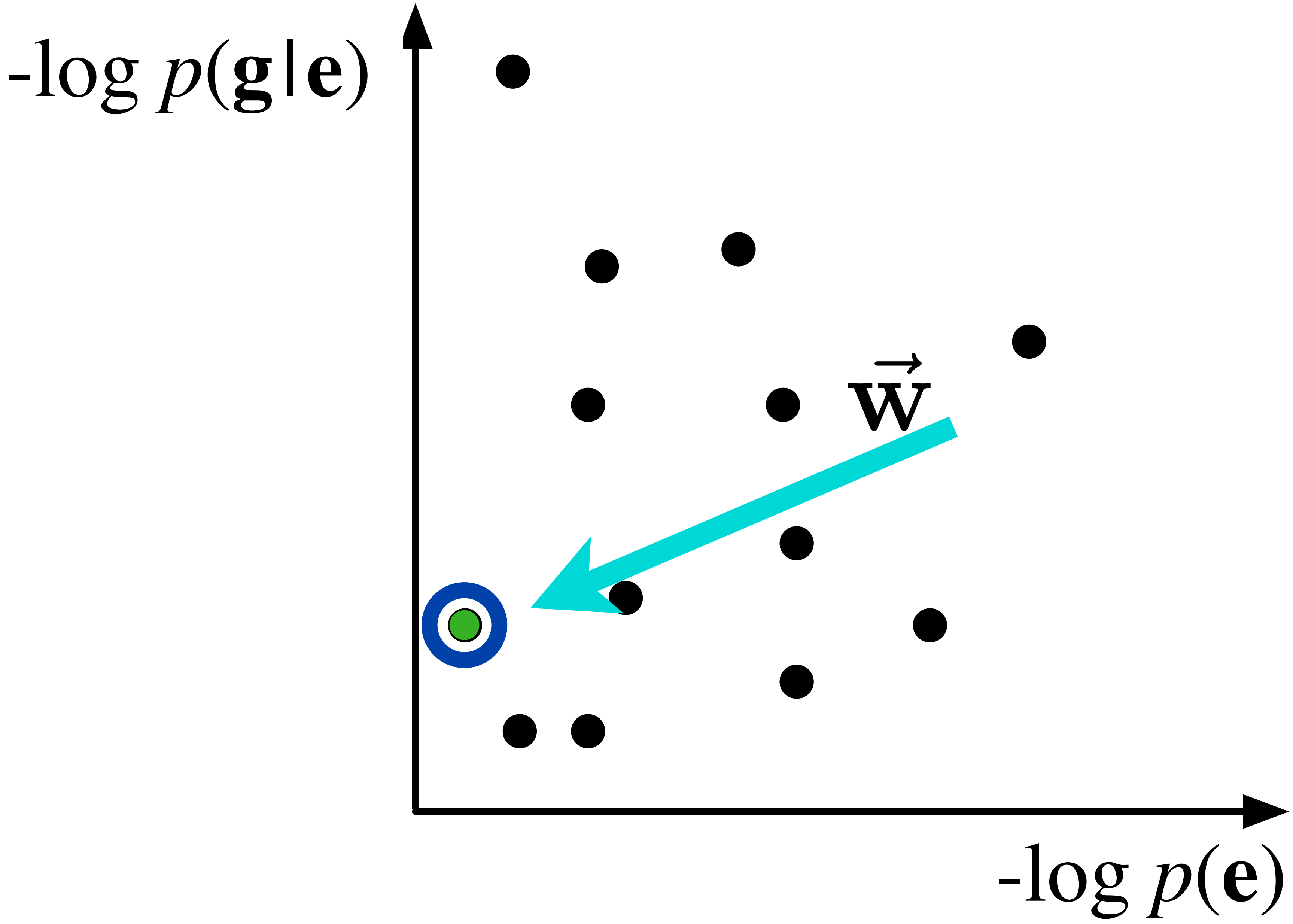
As a Linear Model



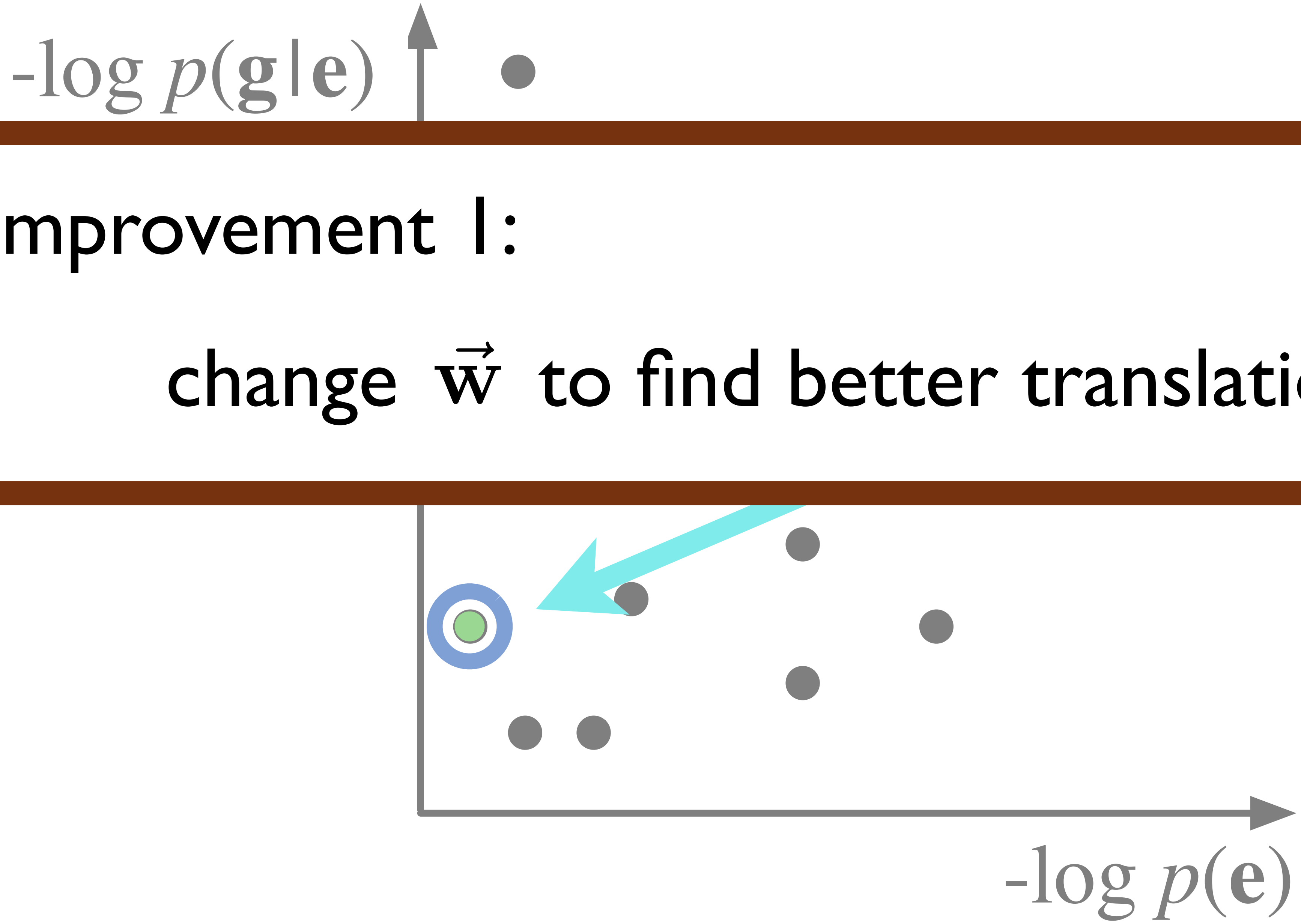
As a Linear Model



As a Linear Model



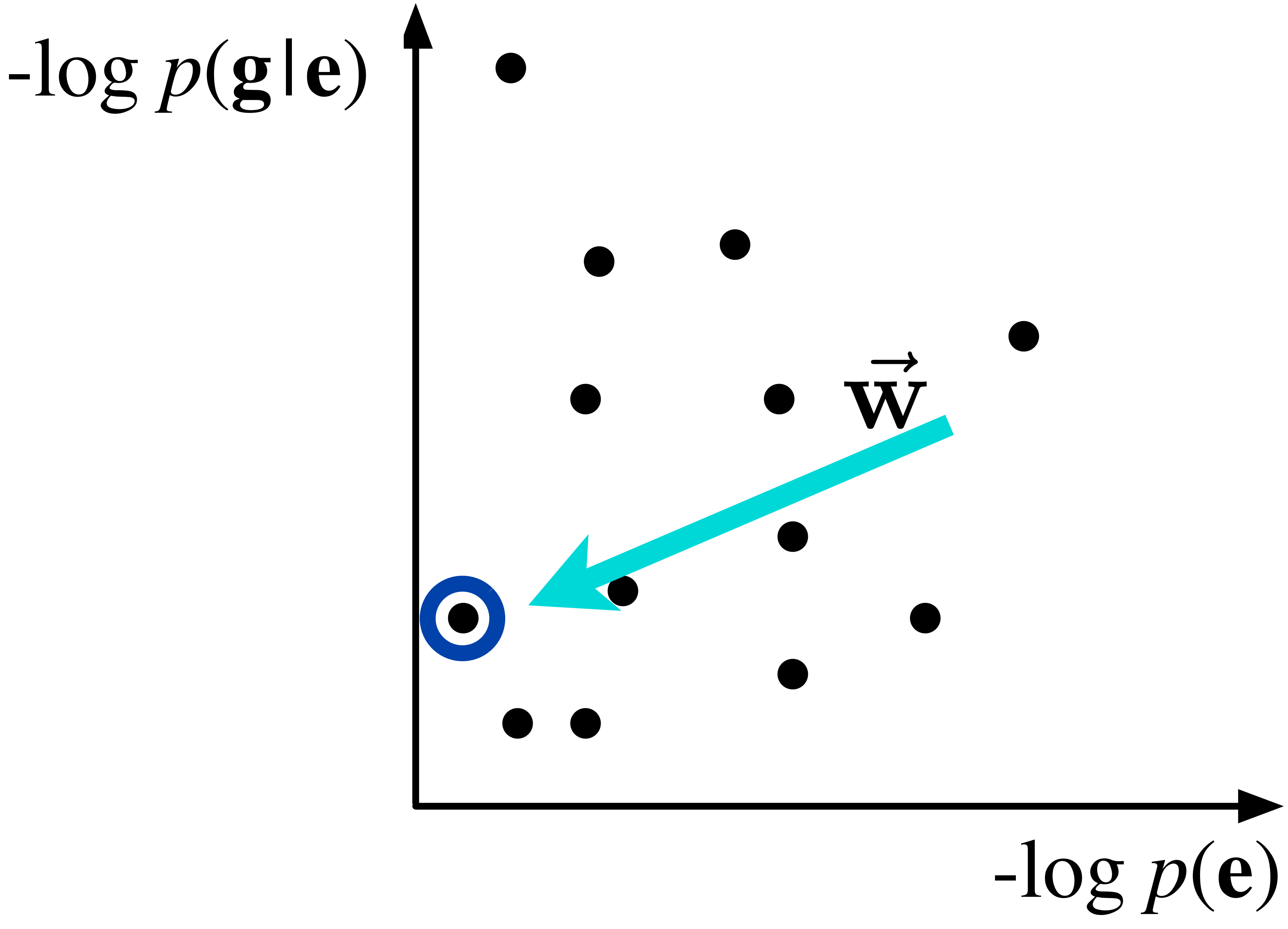
As a Linear Model



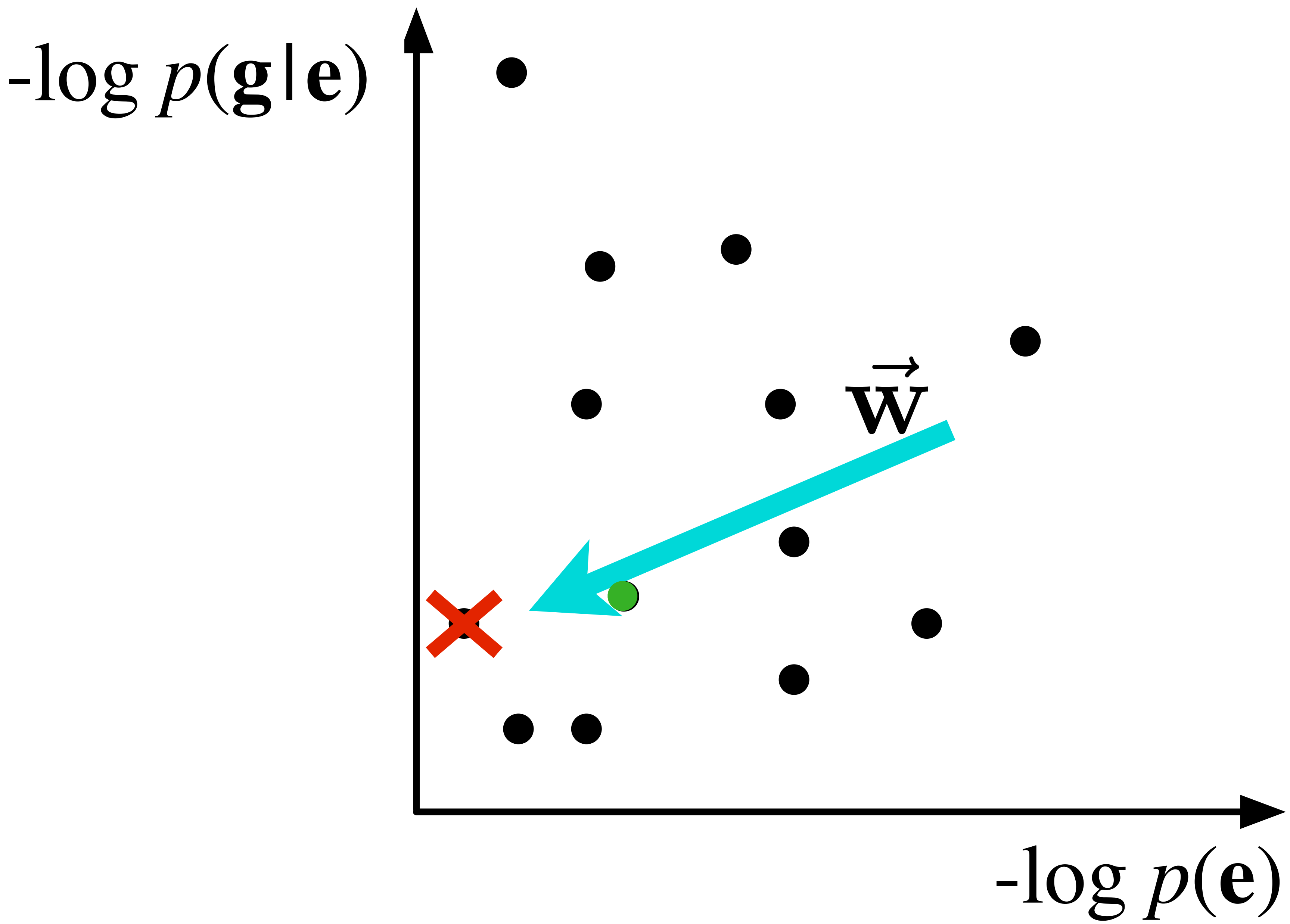
Improvement 1:

change $\vec{w}$ to find better translations

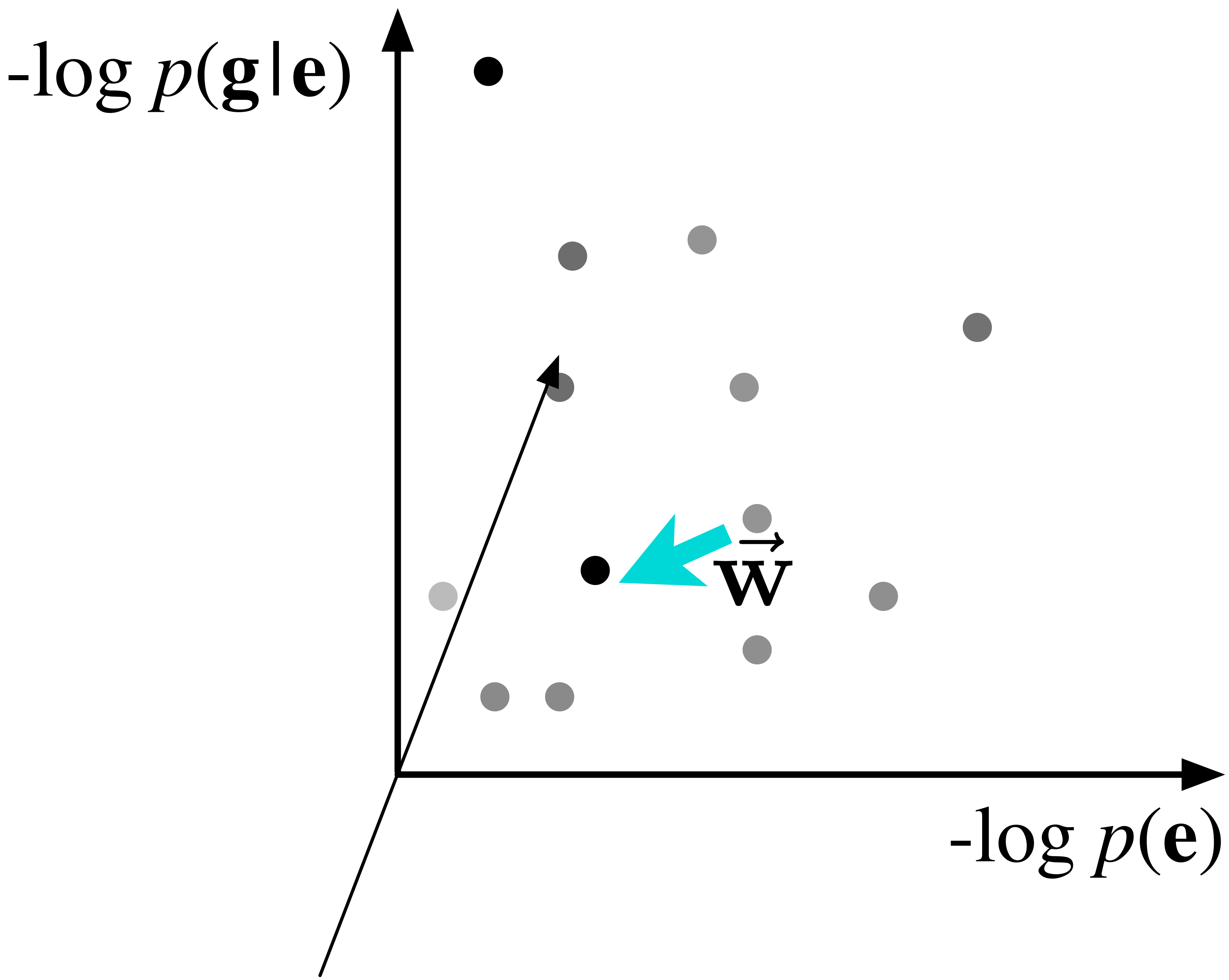
As a Linear Model



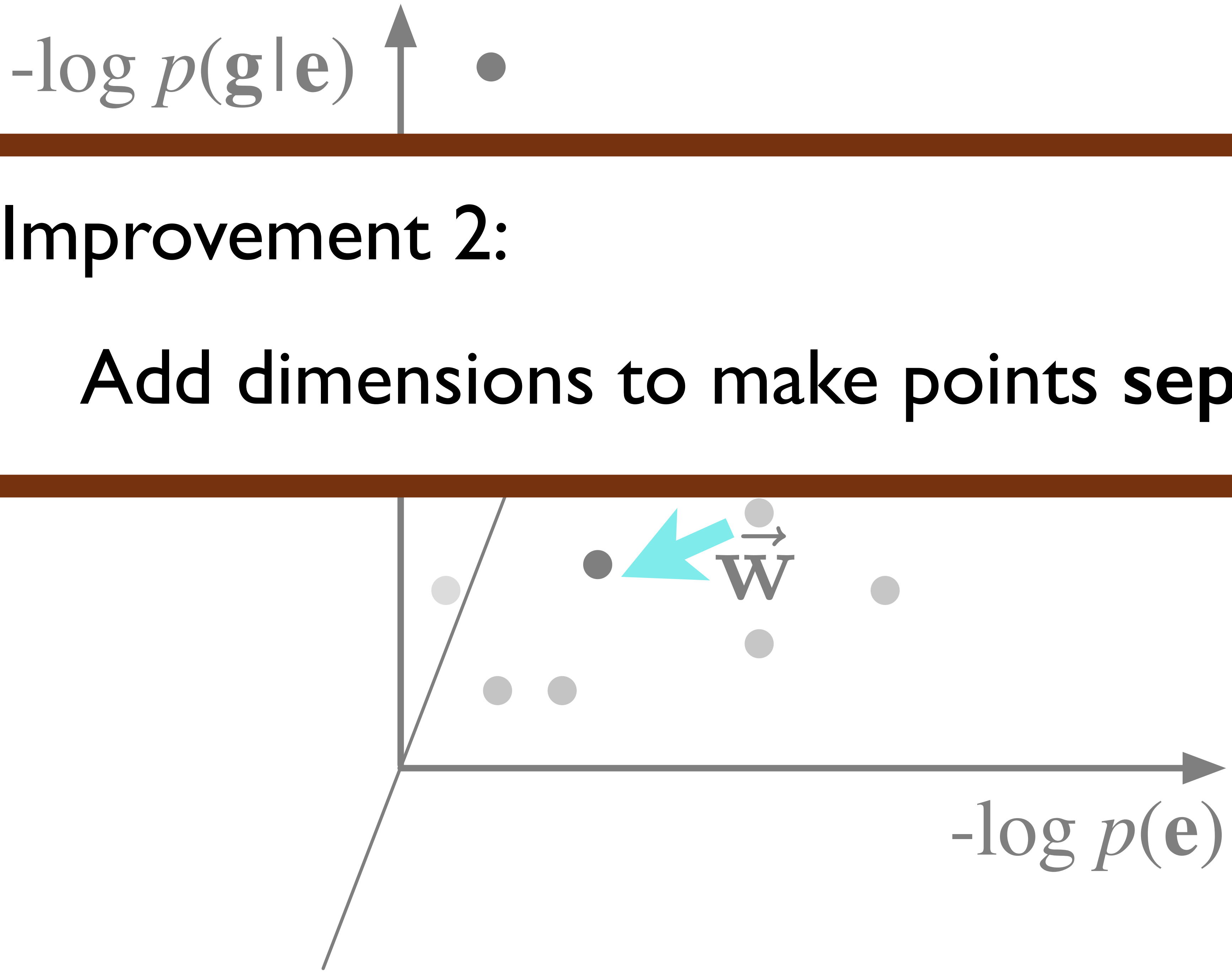
As a Linear Model



As a Linear Model



As a Linear Model



Improvement 2:

Add dimensions to make points **separable**

Linear Models

$$\mathbf{e}^* = \arg\max_{\mathbf{e}} \mathbf{w}^\top \mathbf{h}(\mathbf{g}, \mathbf{e})$$

- Improve the modeling capacity of the noisy channel in two ways
  - Reorient the weight vector
  - Add new dimensions (***new features***)
- Questions
  - What features? $\mathbf{h}(\mathbf{g}, \mathbf{e})$
  - How do we set the weights? $\mathbf{w}$

Mann beißt Hund

x BITES y

Mann beißt Hund

x BITES y

Mann	beißt	Hund
man	bites	cat
man	bite	cat
dog	bites	man

Mann	beißt	Hund
man	chase	dog
man	bite	dog
man	**bites**	**dog**

Feature Classes

Lexical

Are lexical choices appropriate?

bank = "River bank" vs. "Financial institution"

Configurational

Are semantic/syntactic relations preserved?

"Dog bites man" vs. "Man bites dog"

Grammatical

Is the output fluent / well-formed?

"Man *bites* dog" vs. "Man *bite* dog"

What do lexical features look like?

Mann	beißt	Hund
man	bites	cat

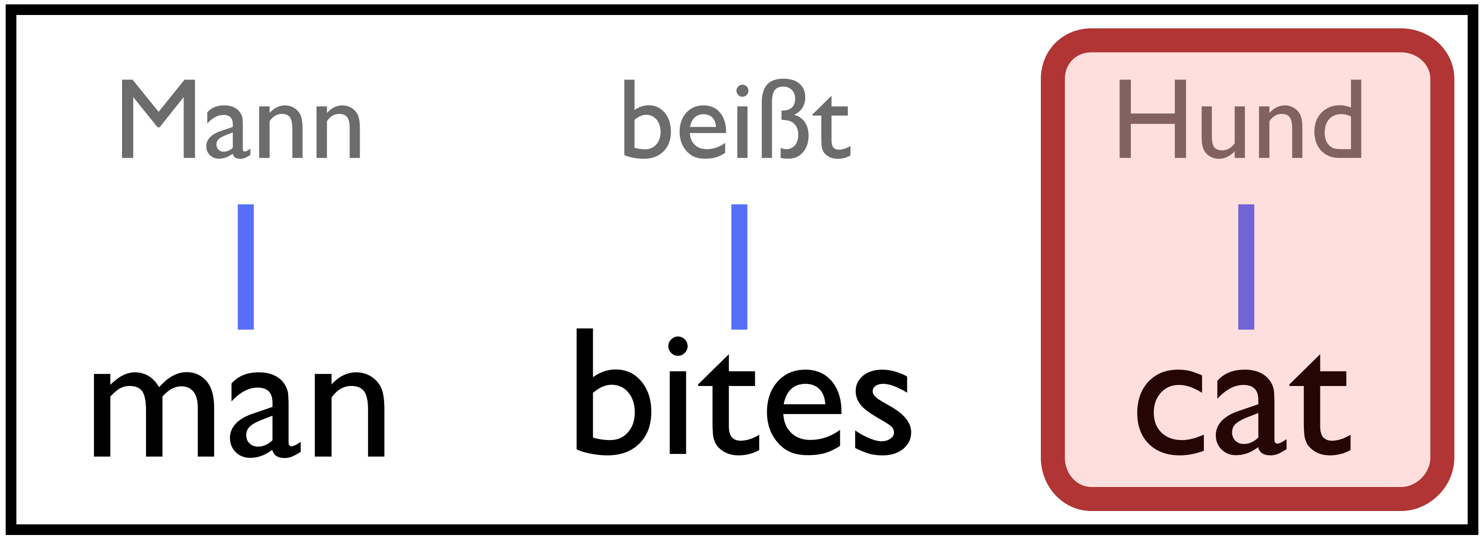
First attempt:

$$score(\mathbf{g},\mathbf{e}) = \mathbf{w}^\top \mathbf{h}(\mathbf{g},\mathbf{e})$$

$$h_{15,342}(\mathbf{g},\mathbf{e}) = \begin{cases} 1, & \exists i,j : g_i = Hund, e_j = cat \\ 0, & \text{otherwise} \end{cases}$$

*But what if a **cat** is being chased by a **Hund**?*

What do lexical features look like?



$\boxed{\text{Latent variables}}$ enable more precise features:

$$score(\mathbf{g}, \mathbf{e}, \boxed{\mathbf{a}}) = \mathbf{w}^\top \mathbf{h}(\mathbf{g}, \mathbf{e}, \boxed{\mathbf{a}})$$

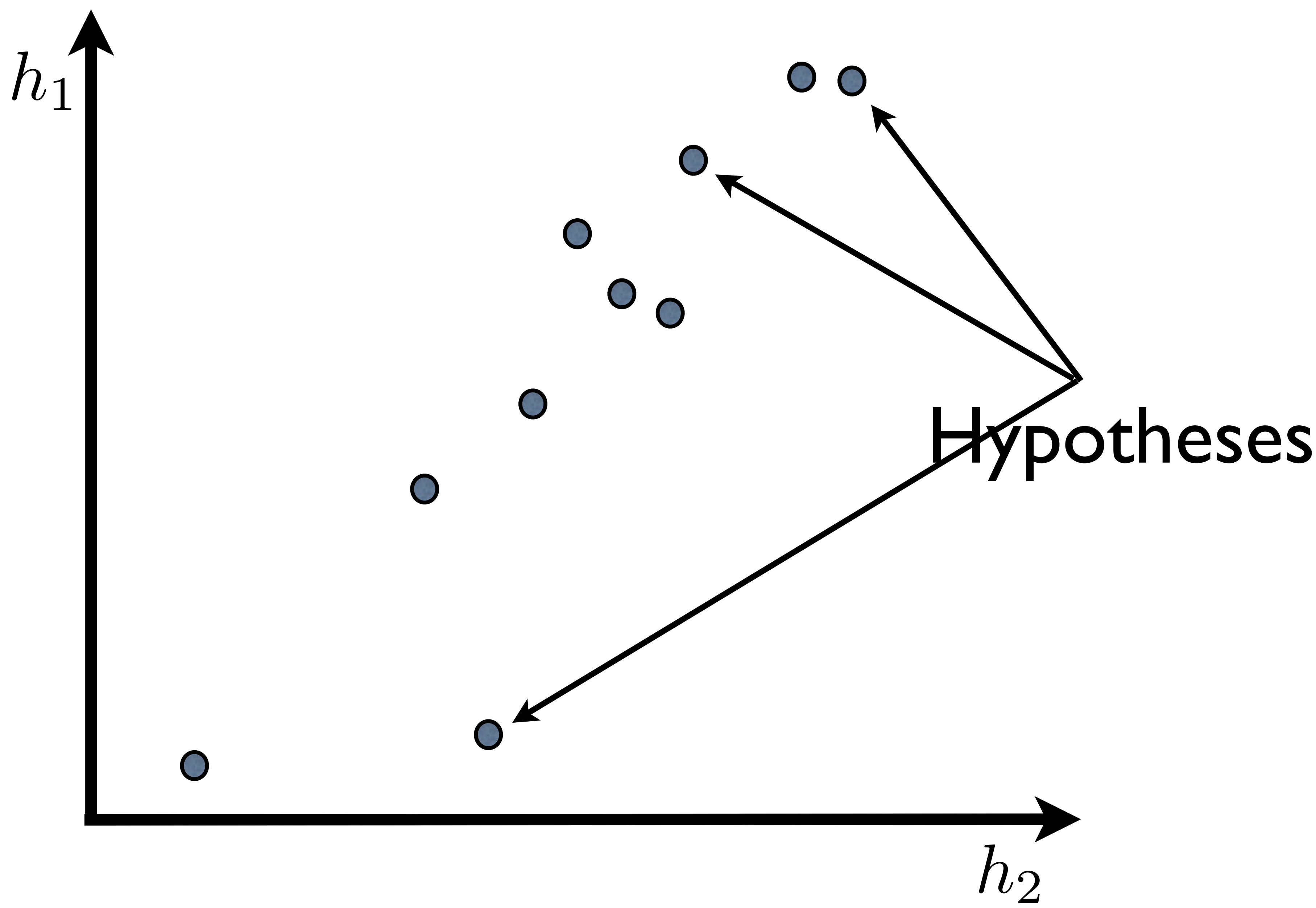
$$h_{15,342}(\mathbf{g}, \mathbf{e}, \boxed{\mathbf{a}}) = \sum_{(i,j)\in \boxed{\mathbf{a}}} \begin{cases} 1, & \text{if } g_i = Hund, e_j = cat \\ 0, & \text{otherwise} \end{cases}$$

Standard Features

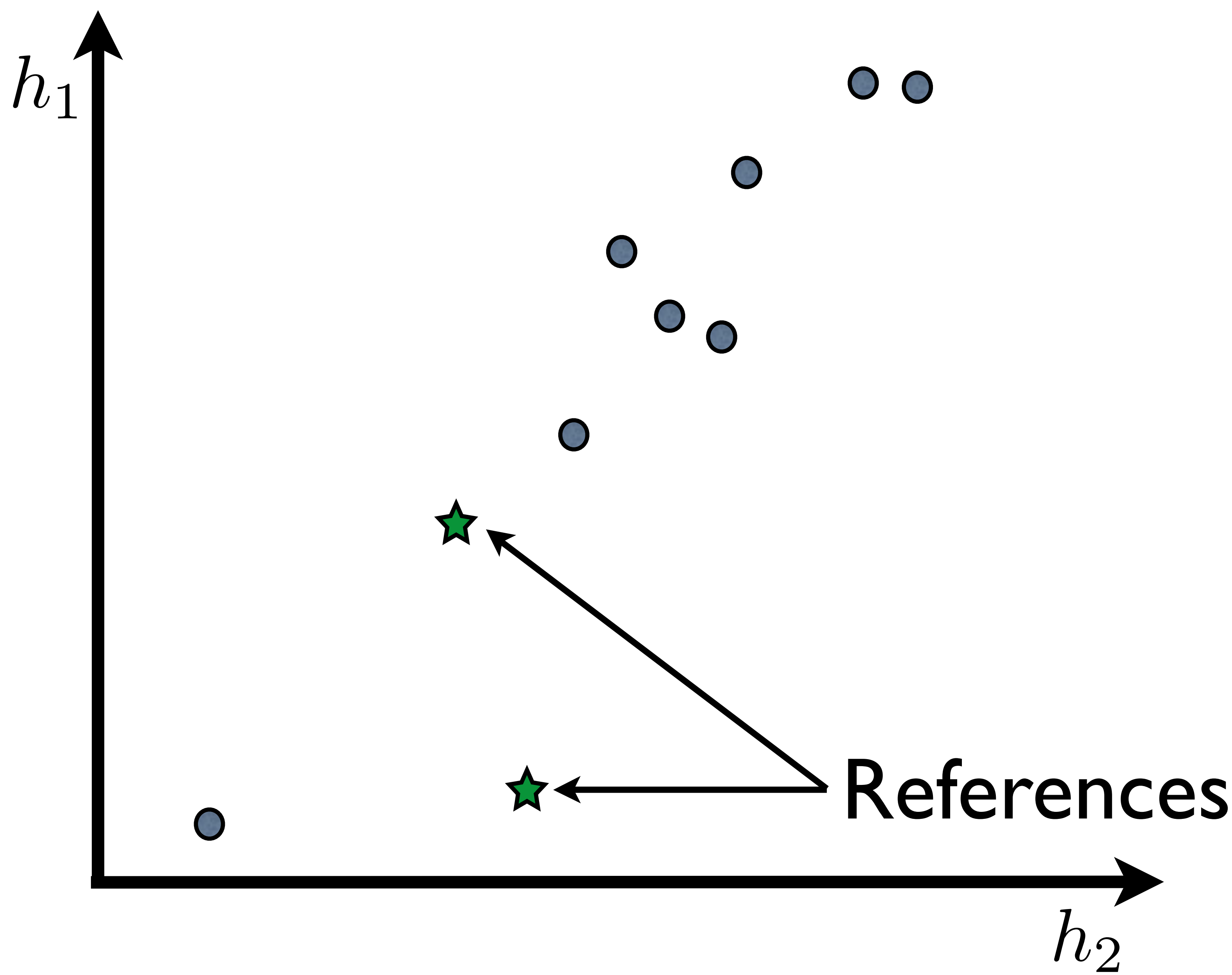
- **Target side features**
  - log p(e) [*n*-gram language model]
  - Number of words in hypothesis
  - Non-English character count
- **Source + target features**
  - log relative frequency e|*f* of each rule [log #(e,f) - log #(f)]
  - log relative frequency *f*|e of each rule [log #(e,f) - log #(e)]
  - "lexical translation" log probability e|*f* of each rule [$\approx \log p_{\text{model1}}(e|f)$]
  - "lexical translation" log probability *f*|e of each rule [$\approx \log p_{\text{model1}}(f|e)$]
- **Other features**
  - Count of rules/phrases used
  - Reordering pattern probabilities

Parameter Learning

Hypothesis Space



Hypothesis Space



Preliminaries

We assume a **decoder** that computes:

$$\langle \mathbf{e}^*, \boxed{\mathbf{a}^*} \rangle = \arg\max_{\langle \mathbf{e}, \boxed{\mathbf{a}} \rangle} \mathbf{w}^\top \mathbf{h}(\mathbf{g}, \mathbf{e}, \boxed{\mathbf{a}})$$

And *K*-**best lists** of, that is:

$$\{\langle \mathbf{e}_i^*, \boxed{\mathbf{a}_i^*} \rangle\}_{i=1}^{i=K} = \arg i^{\mathrm{th}}\text{-}\max_{\langle \mathbf{e}, \boxed{\mathbf{a}} \rangle} \mathbf{w}^\top \mathbf{h}(\mathbf{g}, \mathbf{e}, \boxed{\mathbf{a}})$$

Standard, efficient algorithms exist for this.

Learning Weights

- Try to match the reference translation *exactly*
  - **Conditional random field**
    - Maximize the conditional probability of the reference translations
    - “Average” over the different latent variables

Problems

- These methods give “full credit” when the model *exactly* produces the reference and no credit otherwise
- **What is the problem with this?**

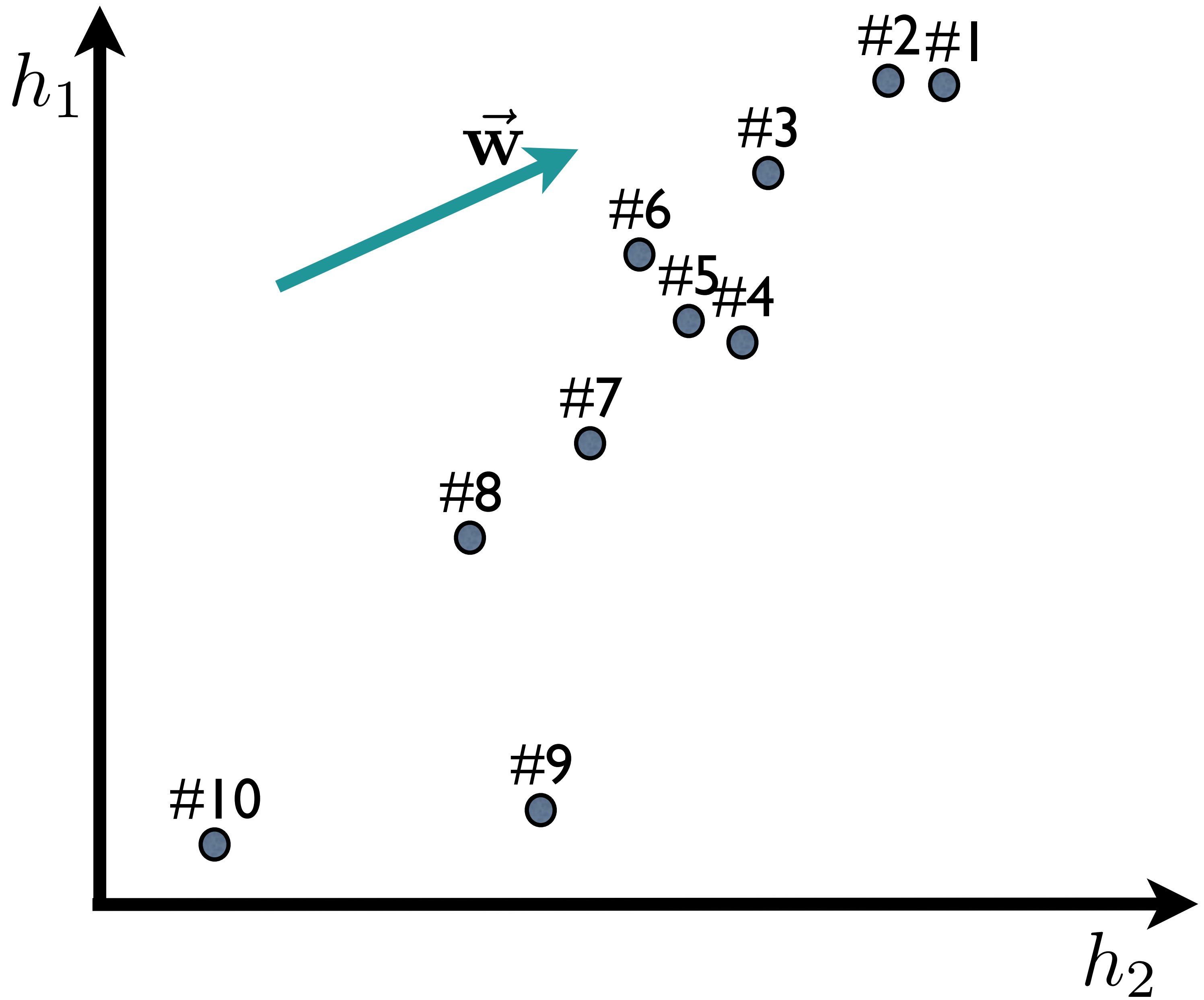
Cost-Sensitive Training

- Assume we have a **cost function** that gives a score for how good/bad a translation is

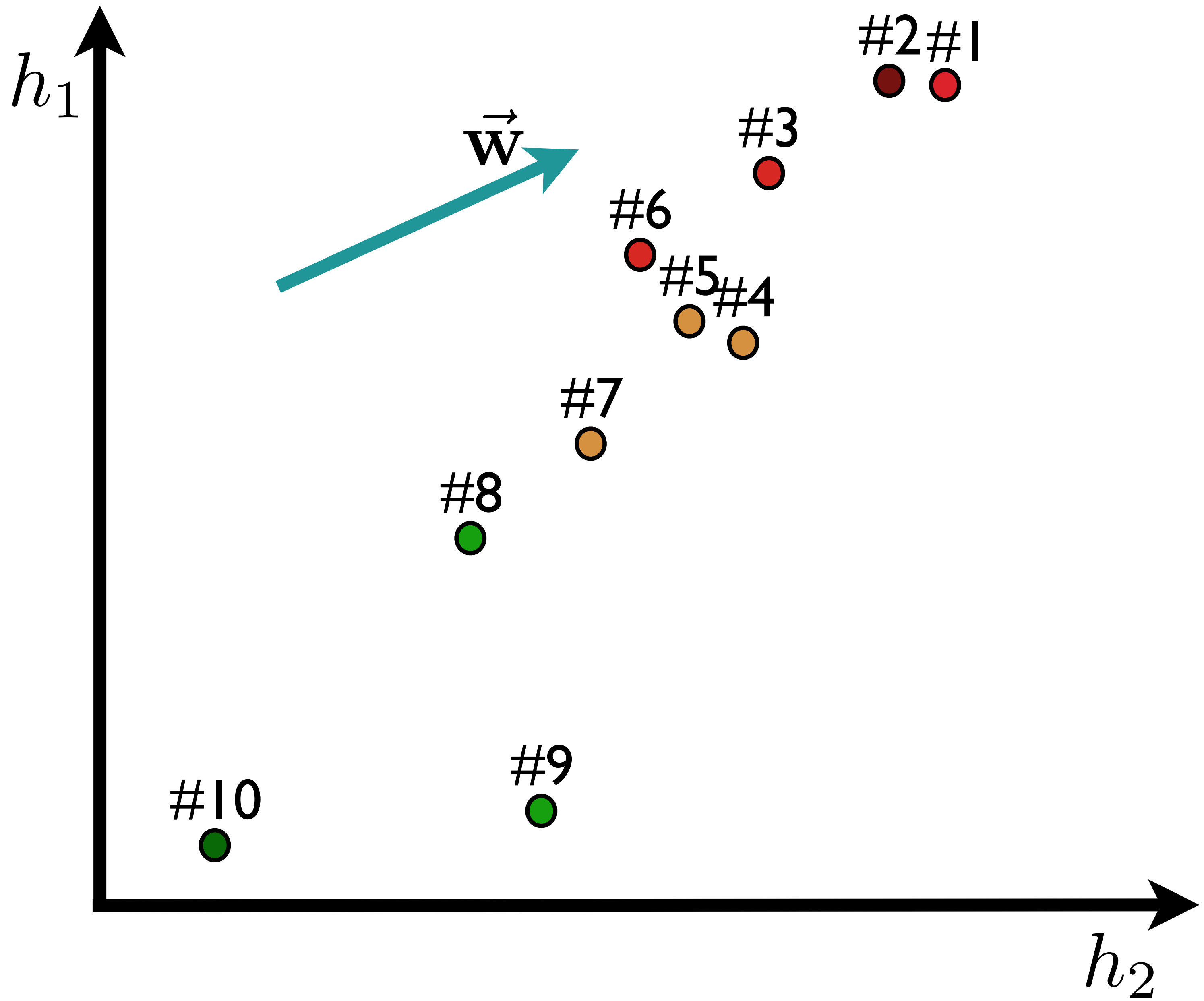
$$\ell(\hat{\mathbf{e}}, \mathcal{E}) \mapsto [0, 1]$$

- Optimize the weight vector by making reference to this function
  - We will talk about two ways to do this

K-Best List Example

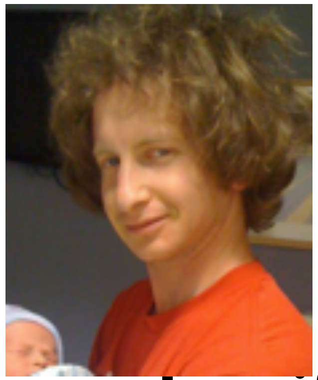


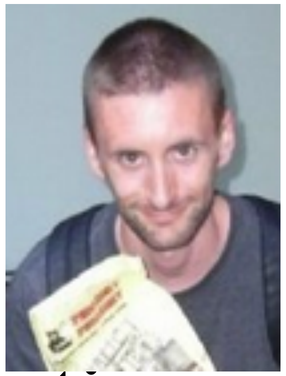
K-Best List Example



- $0.8 \leq \ell < 1.0$
- $0.6 \leq \ell < 0.8$
- $0.4 \leq \ell < 0.6$
- $0.2 \leq \ell < 0.4$
- $0.0 \leq \ell < 0.2$

Training as Classification

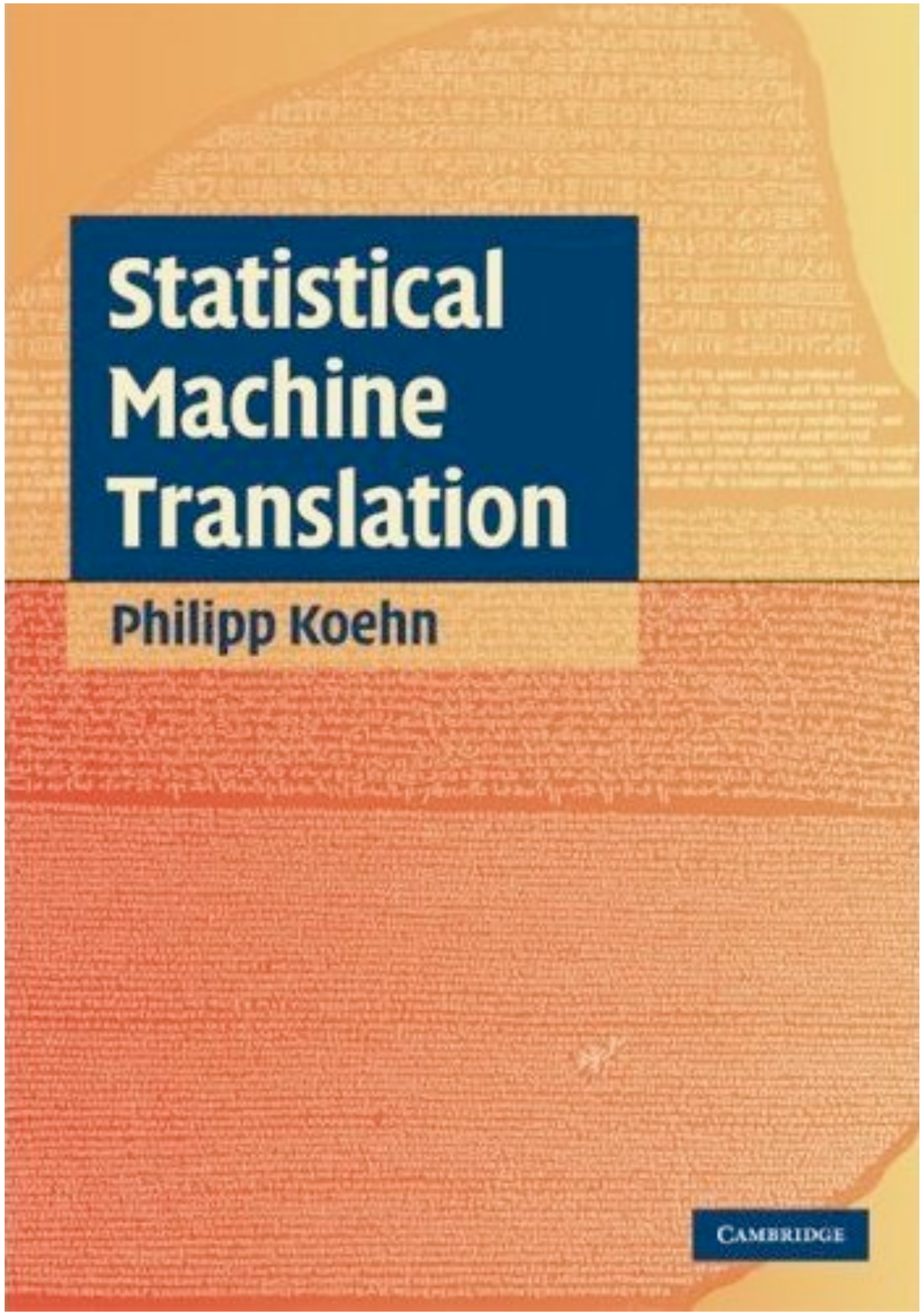




- **P**airwise **R**anking **O**ptimization
  - Reduce training problem to **binary classification** with a **linear model**
- **Algorithm**
  - For *i*=1 to *N*
    - Pick random pair of hypotheses (A,B) from *K*-best list
    - Use cost function to determine if is A or B better
    - Create *i*th training instance
  - Train binary linear classifier

Reading

- Read 9 from the textbook



Announcements

- HW3 due on Thursday at 11:59pm
- Jonny has office hours tomorrow 2-3pm
- 1st term project is due by the end of Spring break
- Upcoming language-in-10
  - Thursday: **Anshul** - Japanese