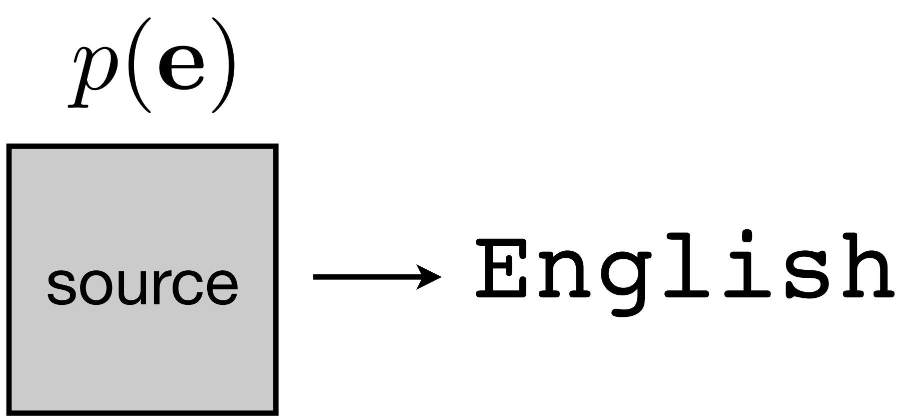
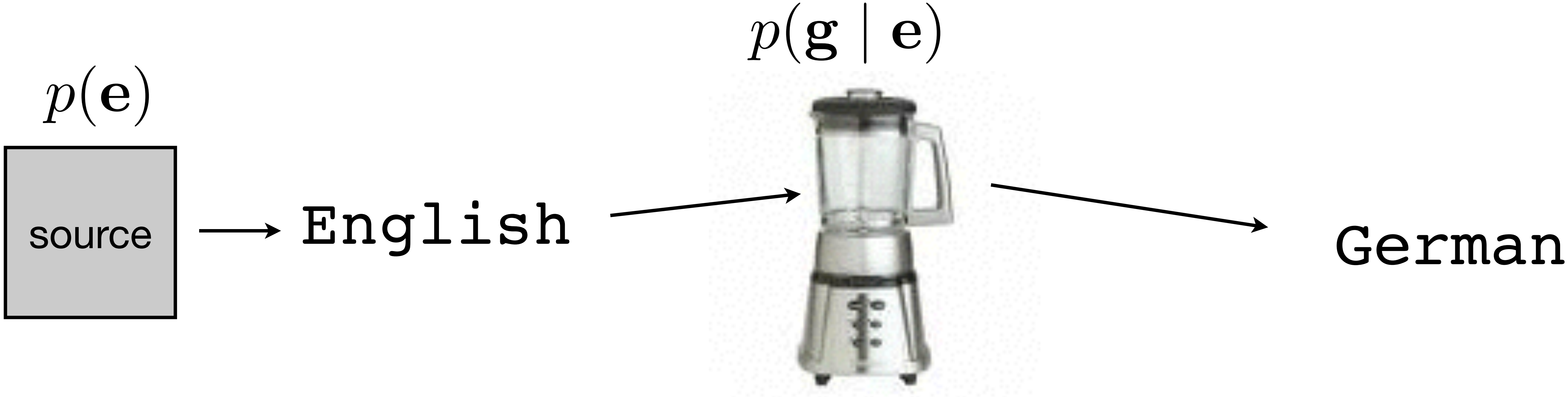
Discriminative Training I: Intro & PRO

April 3, 2014

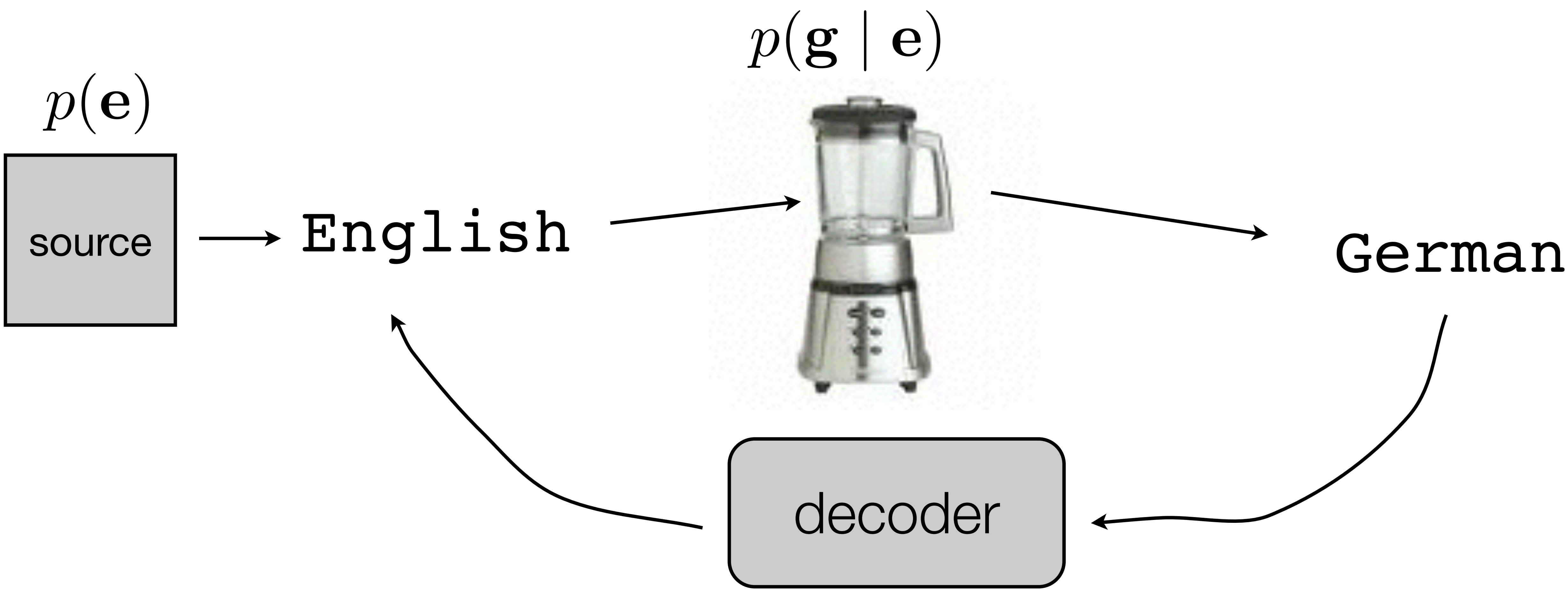
Noisy Channels Again



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

$$\mathbf{e}^* = \arg\max_{\mathbf{e}} p(\mathbf{e} \mid \mathbf{g})$$

$$= \arg\max_{\mathbf{e}} \frac{p(\mathbf{g} \mid \mathbf{e}) \times p(\mathbf{e})}{p(\mathbf{g})}$$

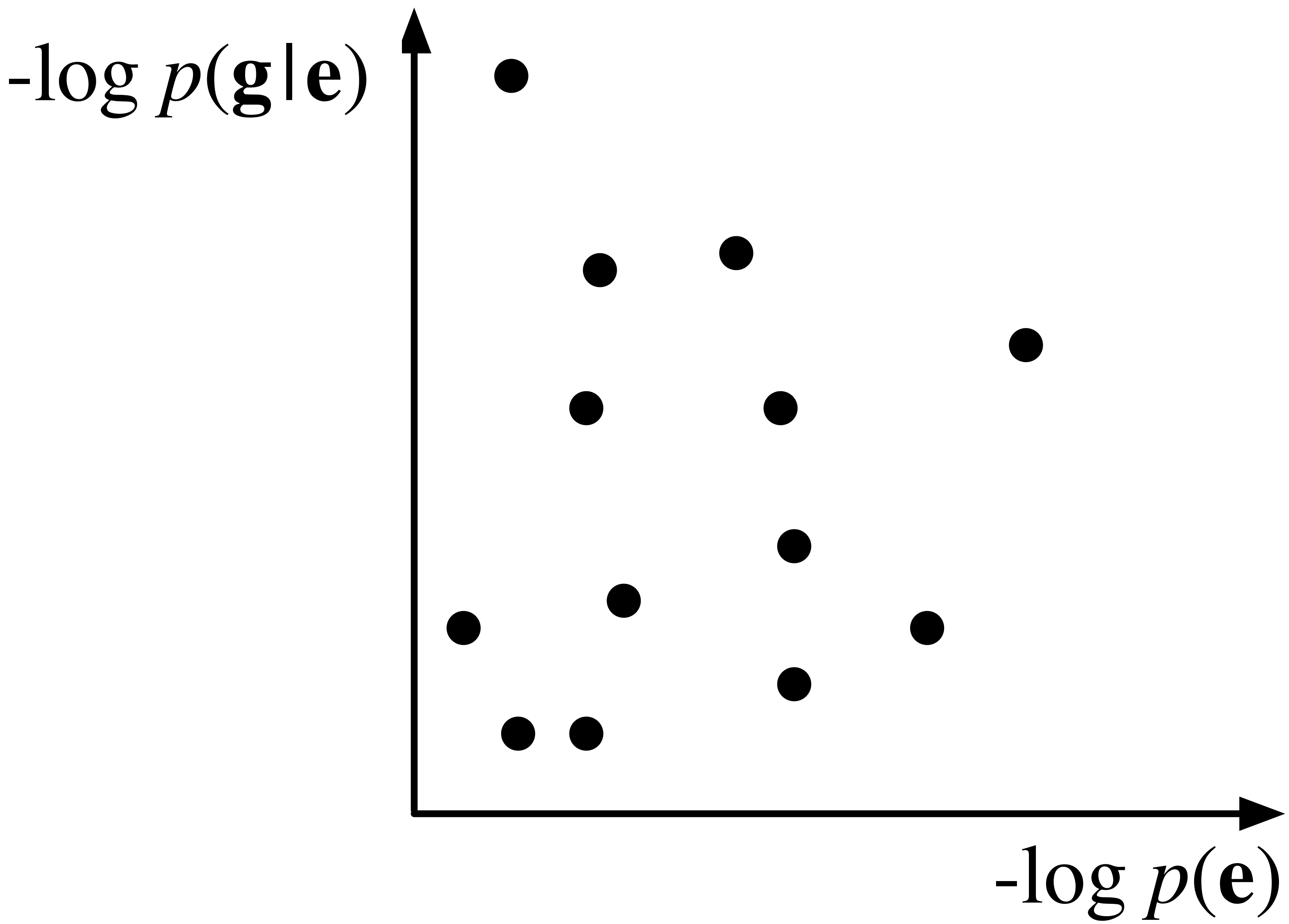
$$= \arg\max_{\mathbf{e}} p(\mathbf{g} \mid \mathbf{e}) \times p(\mathbf{e})$$

Does this look familiar?

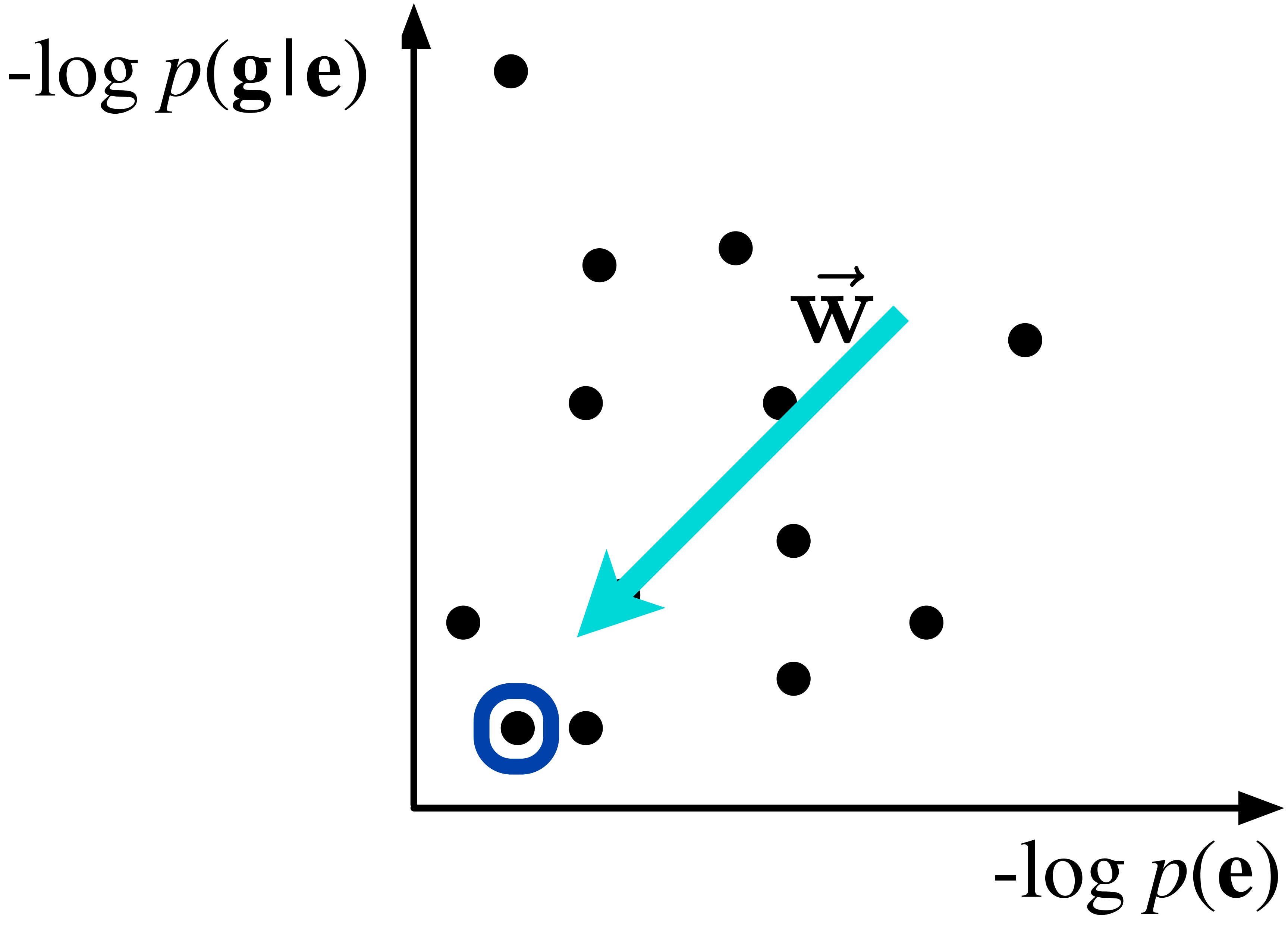
$$= \arg\max_{\mathbf{e}} \log p(\mathbf{g} \mid \mathbf{e}) + \log p(\mathbf{e})$$

$$\boxed{= \arg\max_{\mathbf{e}} \underbrace{\begin{bmatrix} 1 \\ 1 \end{bmatrix}^{\top}}_{\mathbf{w}^{\top}} \underbrace{\begin{bmatrix} \log p(\mathbf{g} \mid \mathbf{e}) \\ \log p(\mathbf{e}) \end{bmatrix}}_{\mathbf{h}(\mathbf{g},\mathbf{e})}}$$

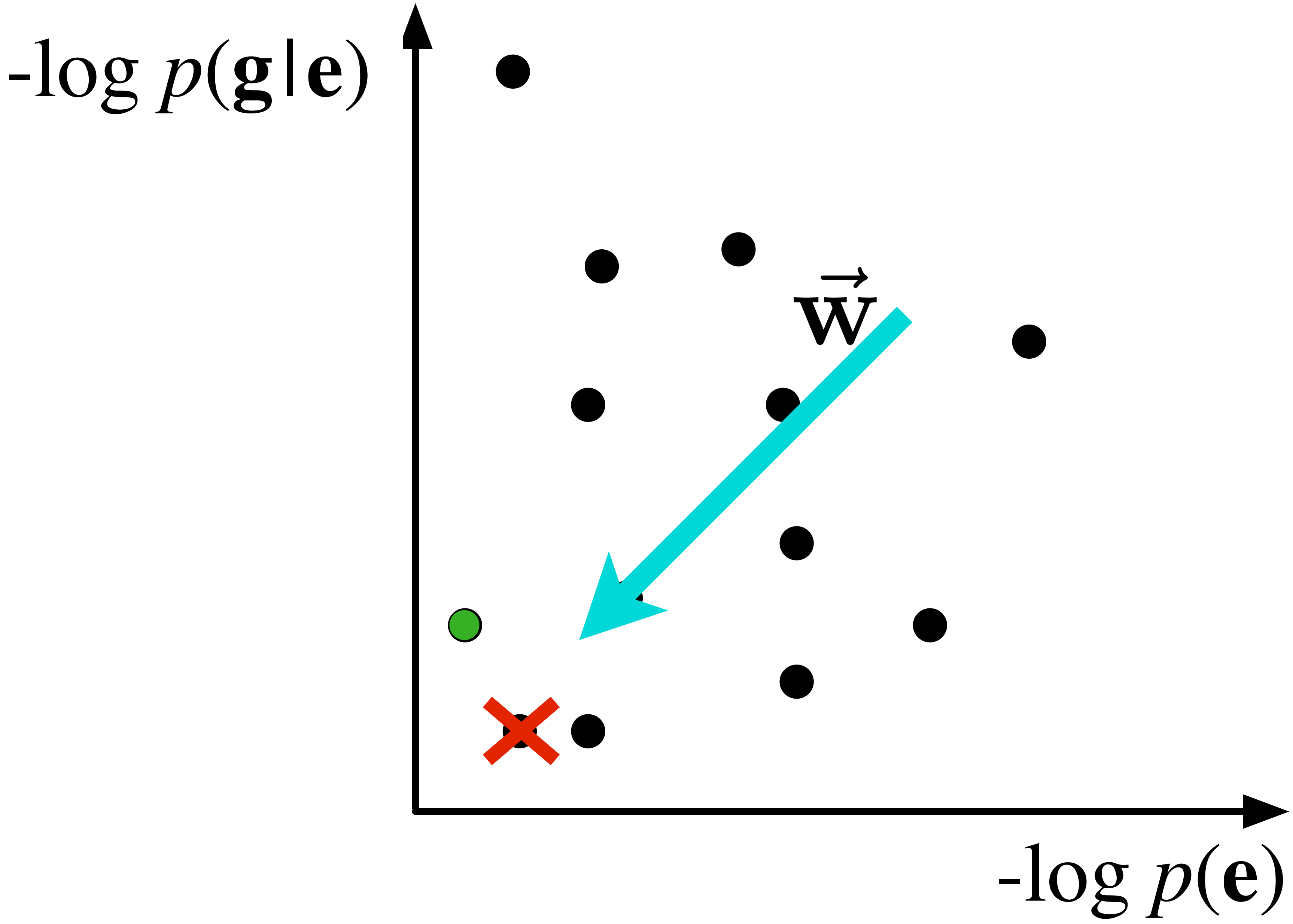
The Noisy Channel



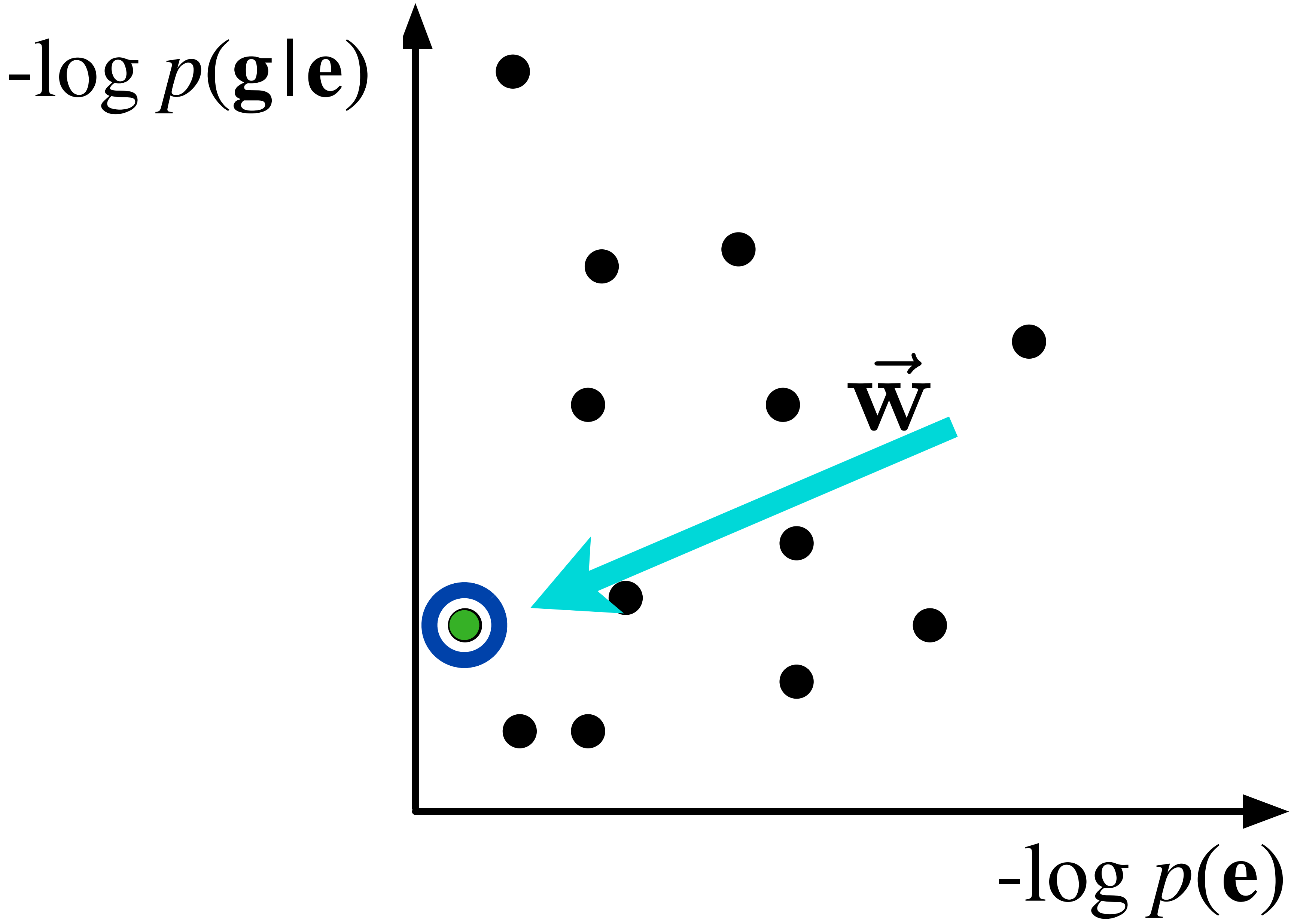
As a Linear Model



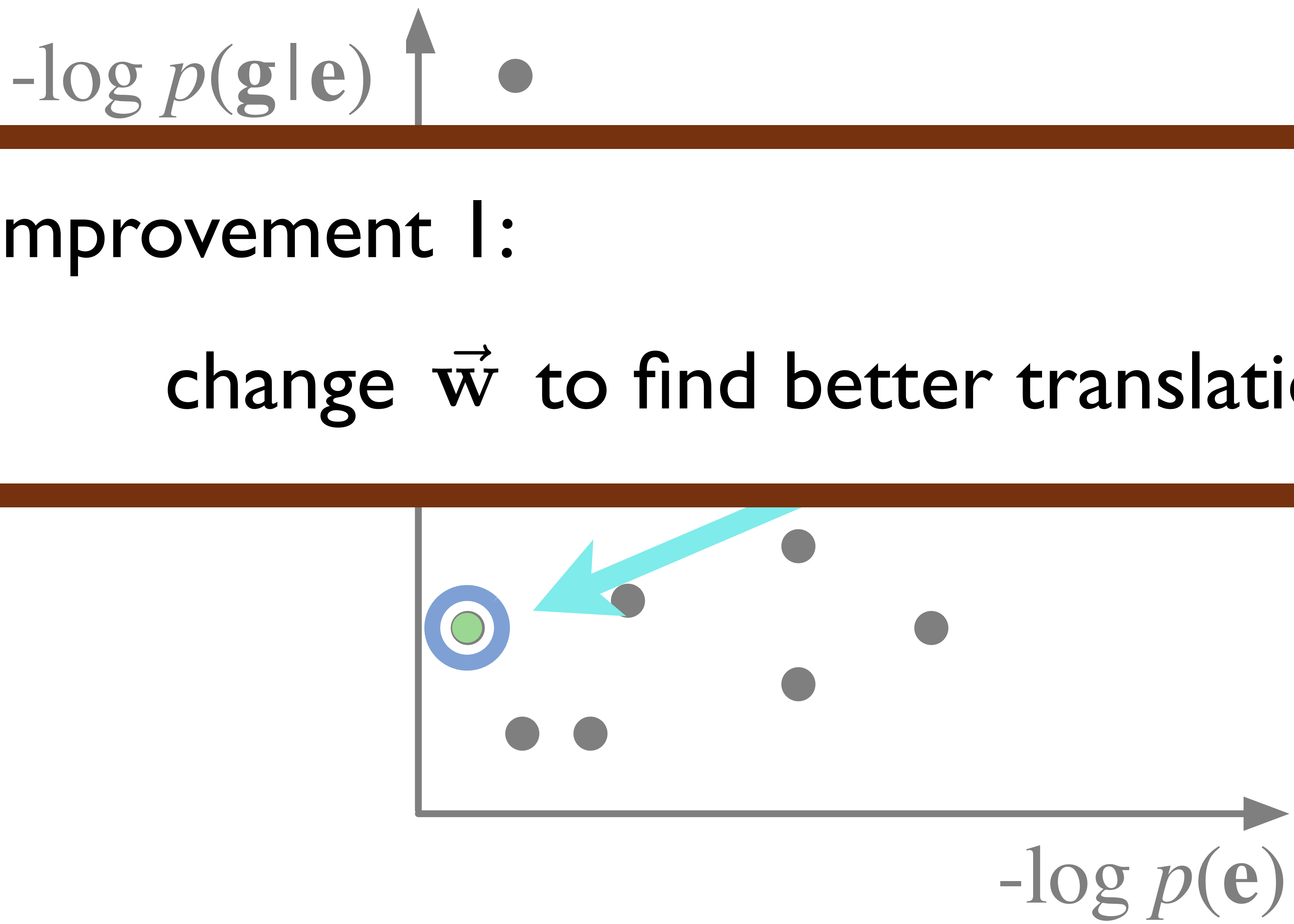
As a Linear Model



As a Linear Model



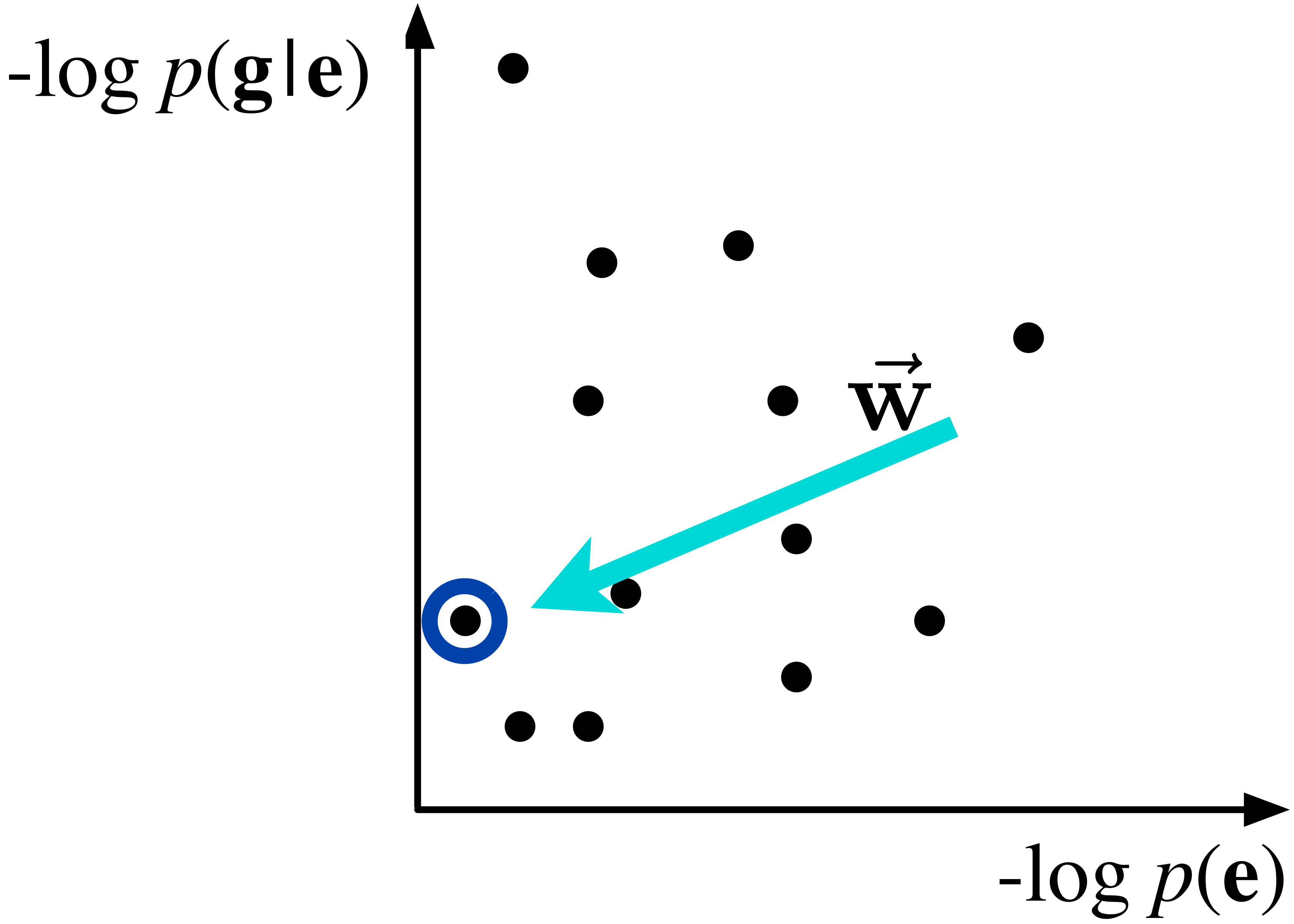
As a Linear Model



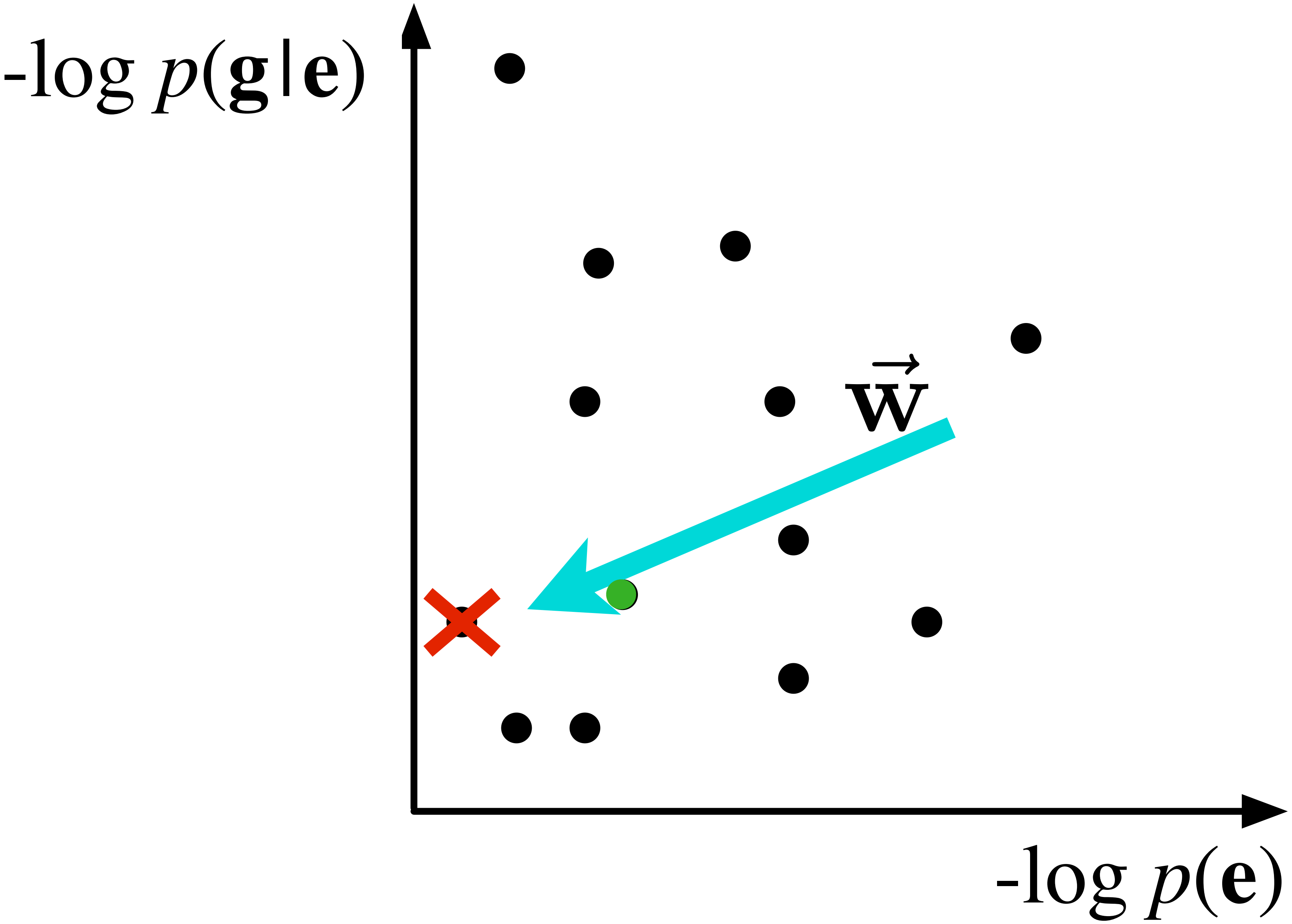
Improvement 1:

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

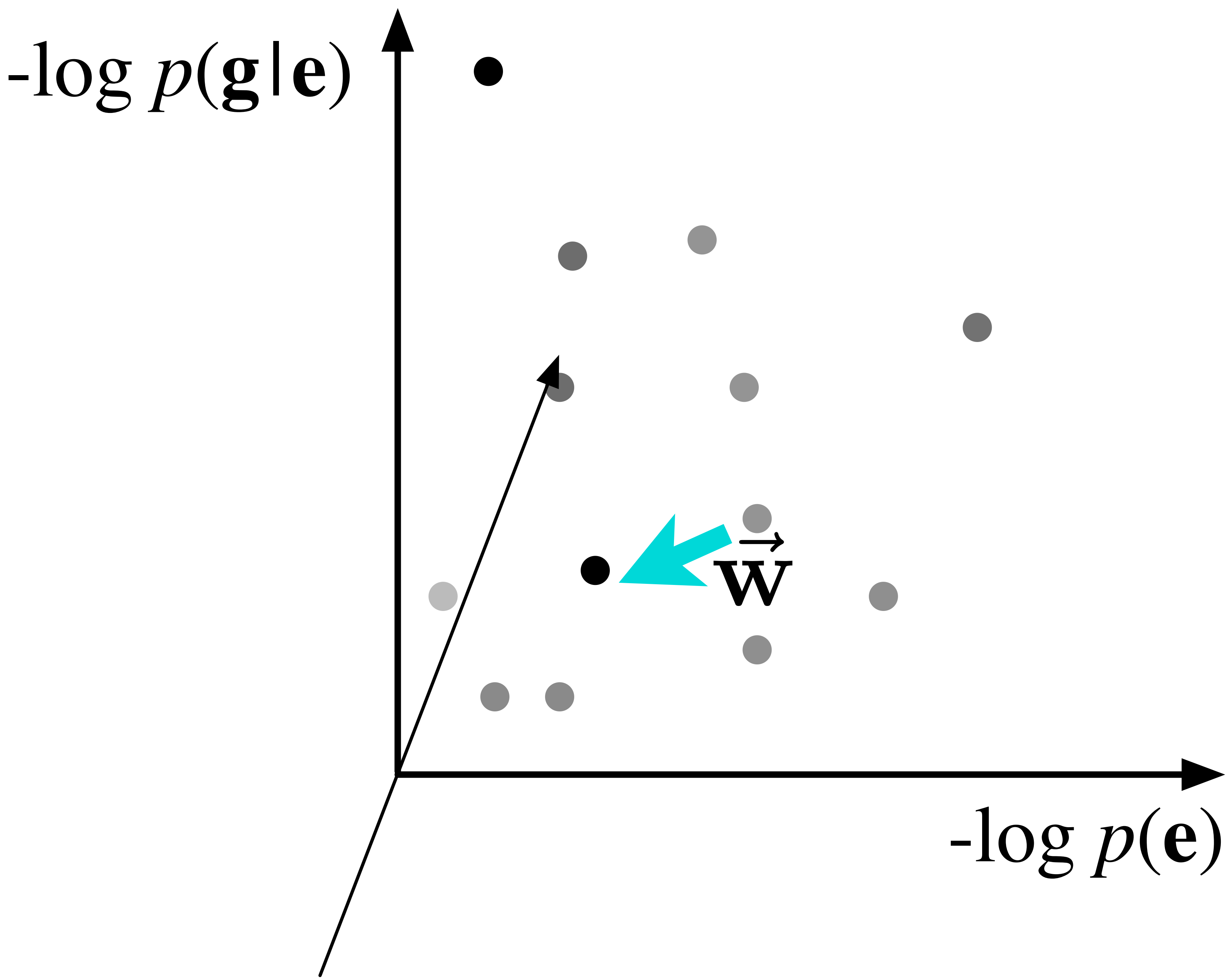
As a Linear Model



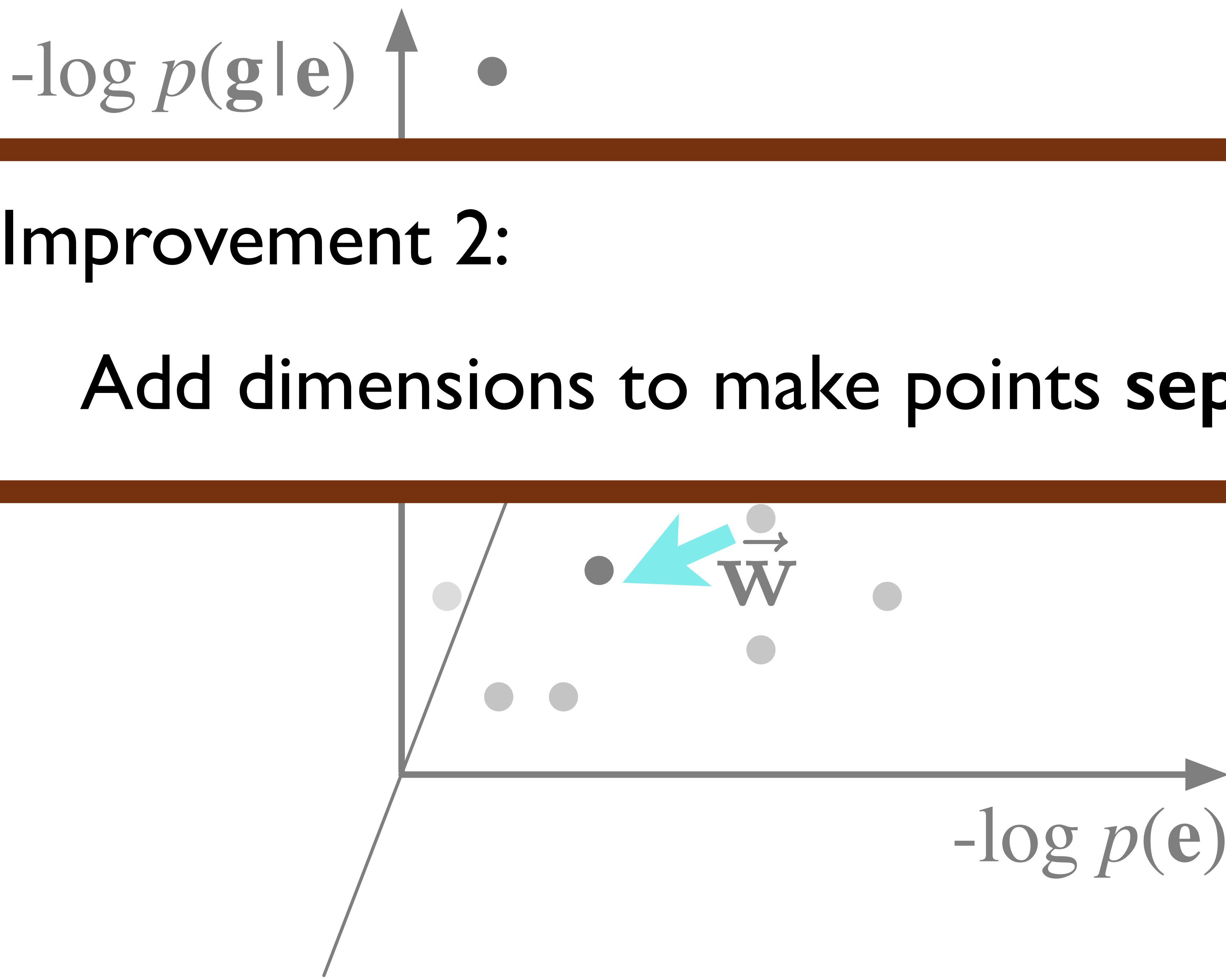
As a Linear Model



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

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

Mann	beißt	Hund
man	bites	cat

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

Feature Locality

- Dynamic programming recombination assumes that features are “rule local”
  - The must have the same value independent of the other rules that are used around them
- Features that look at “large amounts of structure” are expensive to compute
- Language models are “medium sized” features

Why do this?

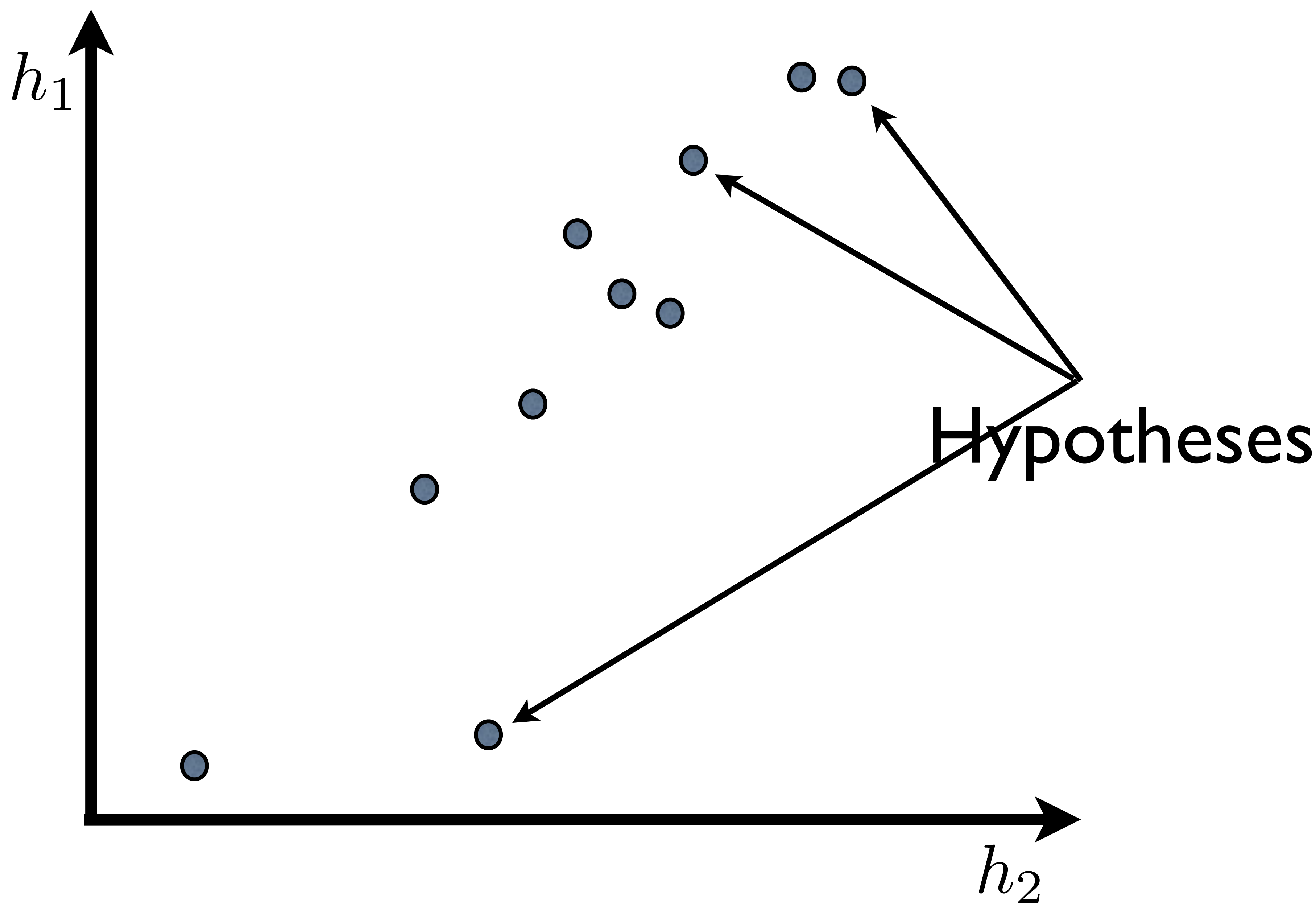
Table 2: Effect of maximum entropy training for alignment template approach (WP: word penalty feature, CLM: class-based language model (five-gram), MX: conventional dictionary).

	objective criteria [%]					subjective criteria [%]	
	SER	WER	PER	mWER	BLEU	SSER	IER
Baseline($\lambda_m = 1$)	86.9	42.8	33.0	37.7	43.9	35.9	39.0
ME	81.7	40.2	28.7	34.6	49.7	32.5	34.8
ME+WP	80.5	38.6	26.9	32.4	54.1	29.9	32.2
ME+WP+CLM	78.1	38.3	26.9	32.1	55.0	29.1	30.9
ME+WP+CLM+MX	77.8	38.4	26.8	31.9	55.2	28.8	30.9

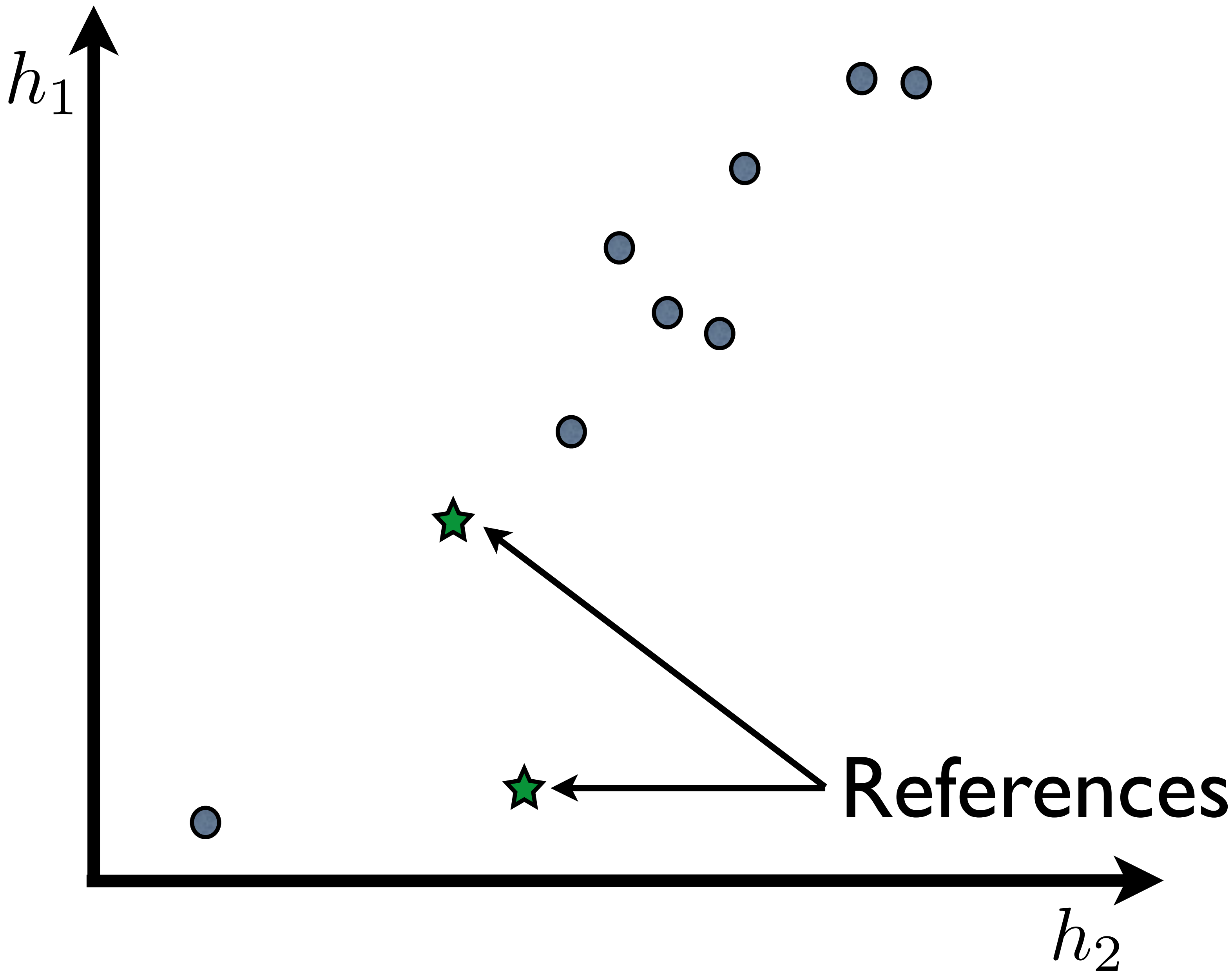
Discriminative

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:

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

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?**

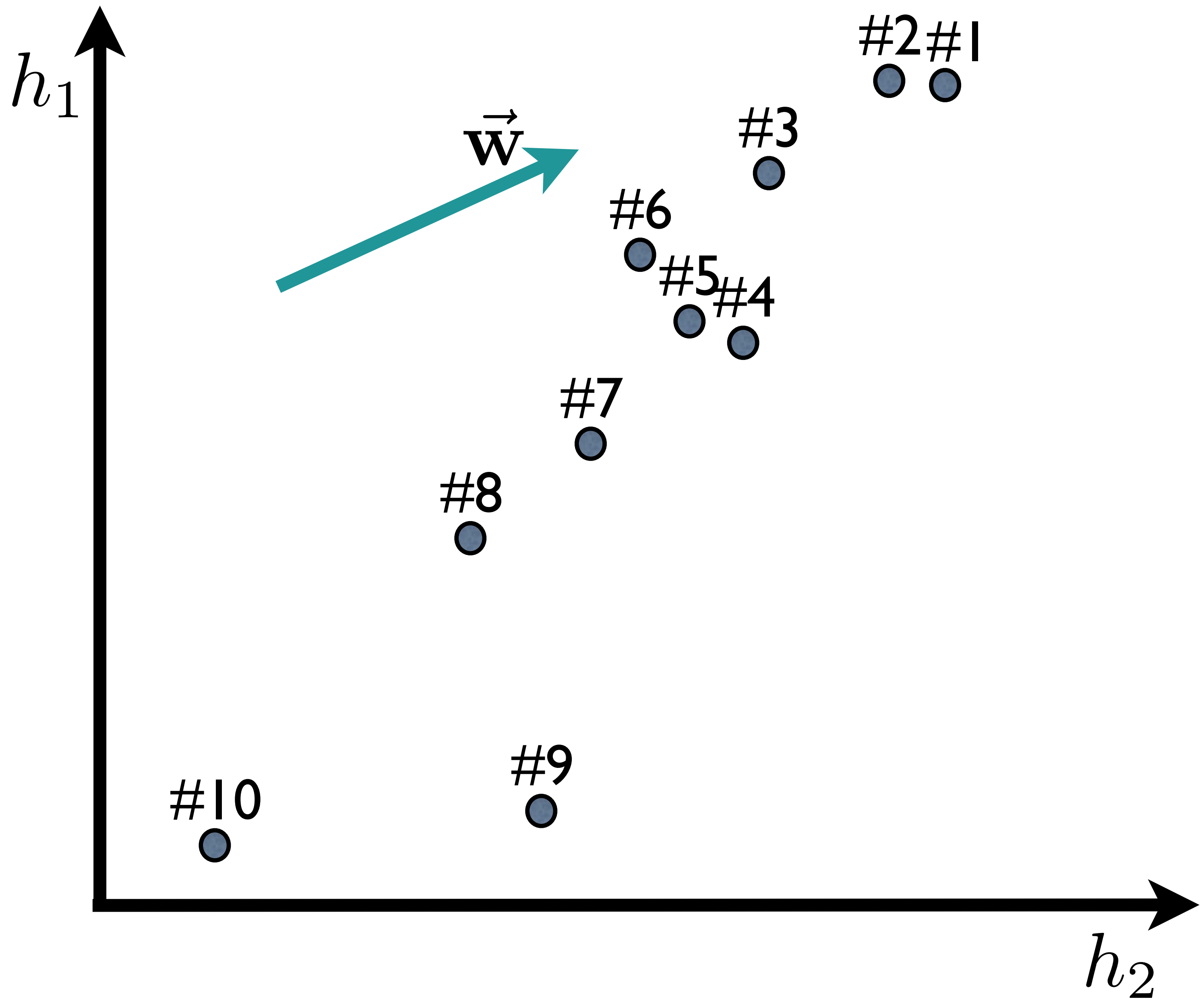
Cost-Sensitive Training

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

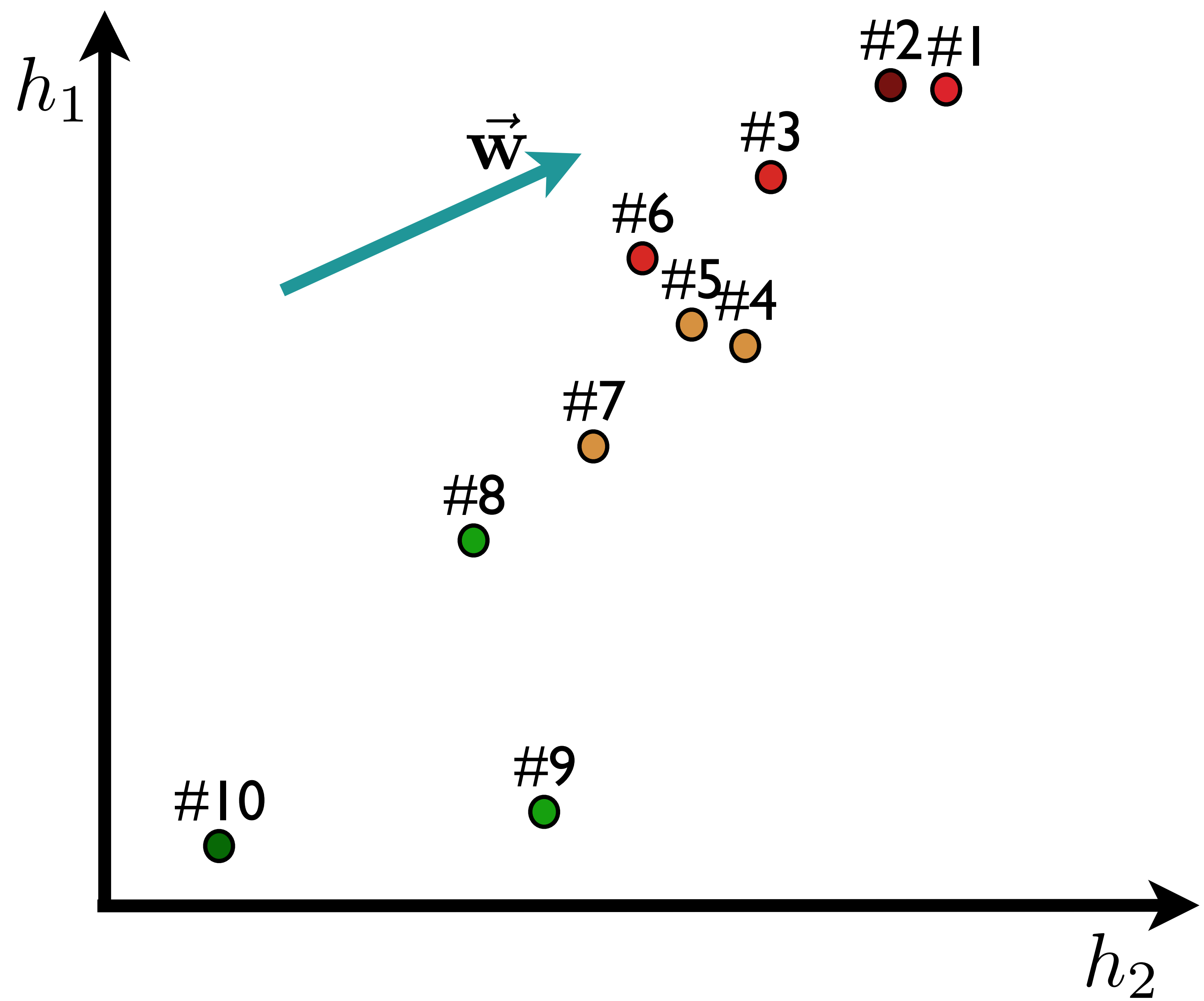
$$\ell(\hat{\mathbf{e}}, \mathcal{E}) \mapsto [0, 1]$$

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

K-Best List Example

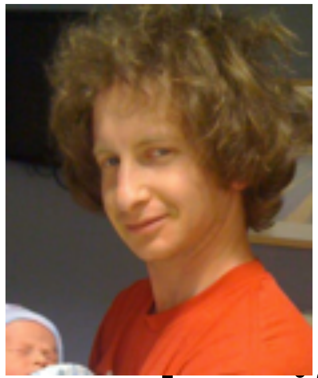
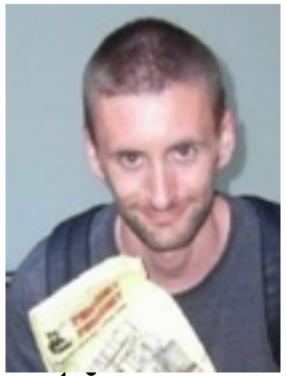


K-Best List Example

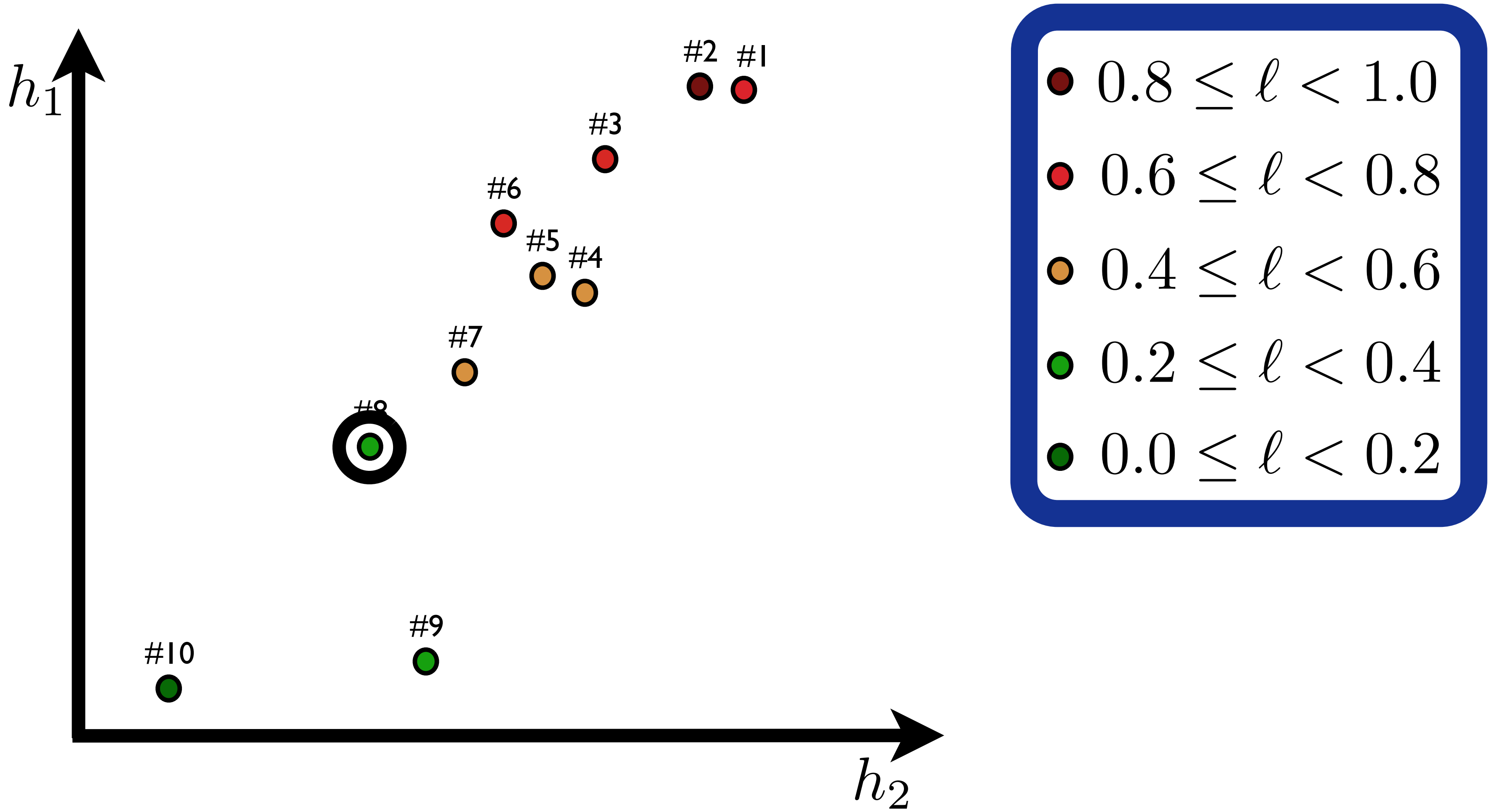


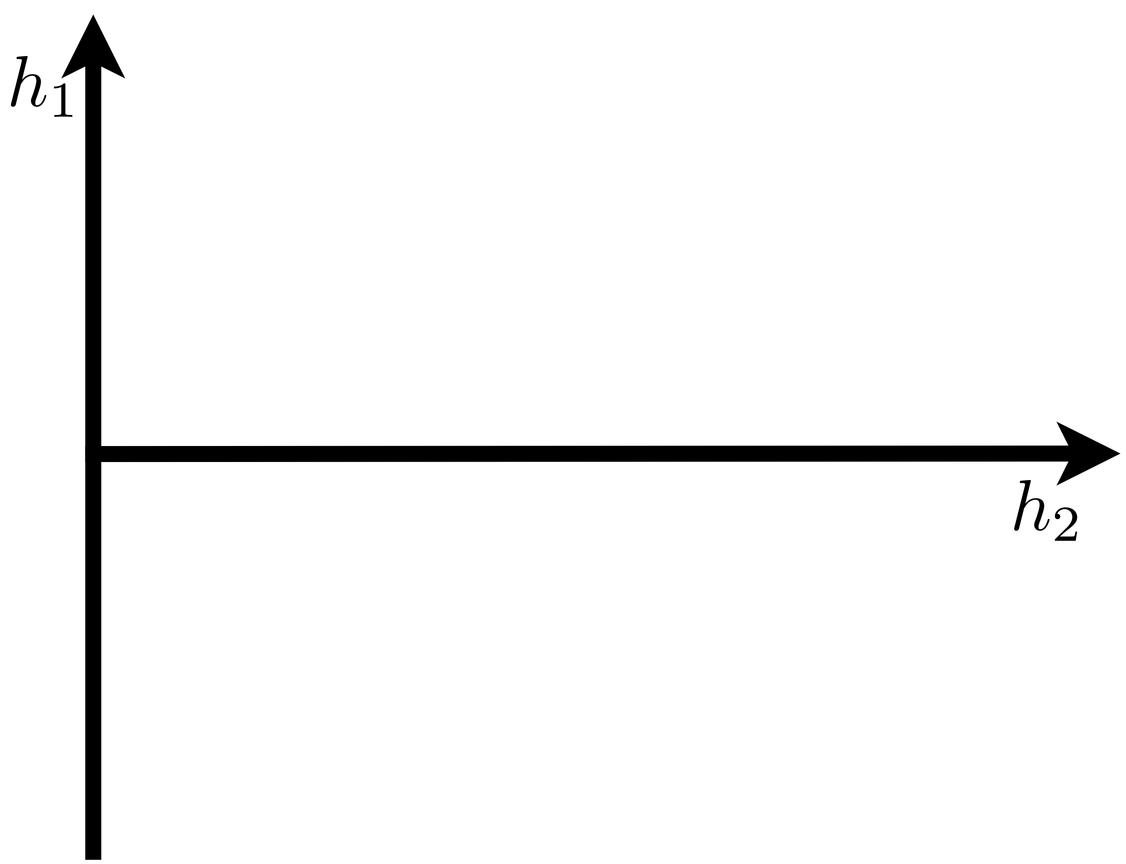
- $0.8 \leq \ell < 1.0$
- $0.6 \leq \ell < 0.8$
- $0.4 \leq \ell < 0.6$
- $0.2 \leq \ell < 0.4$
- $0.0 \leq \ell < 0.2$

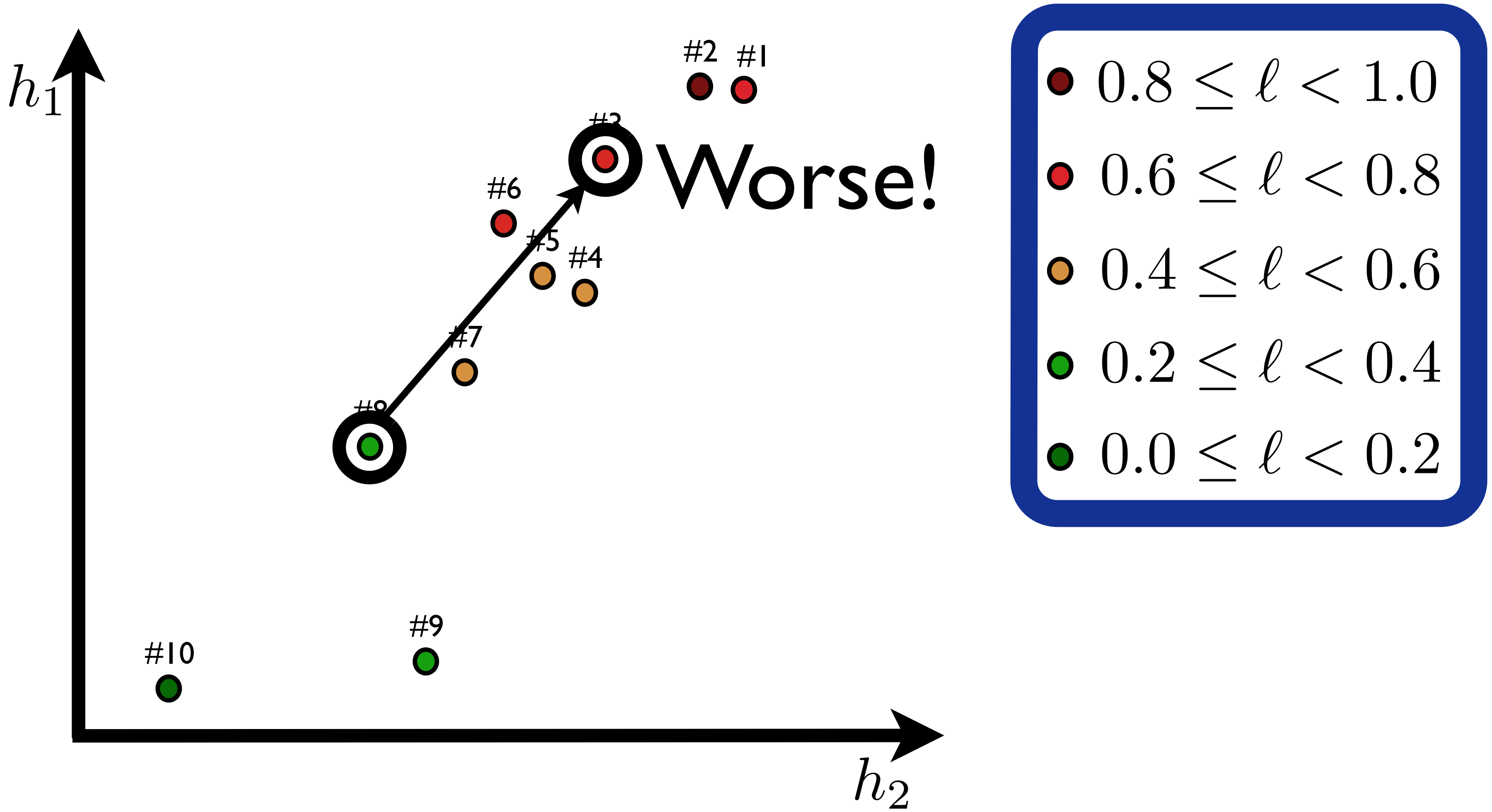
Training as Classification

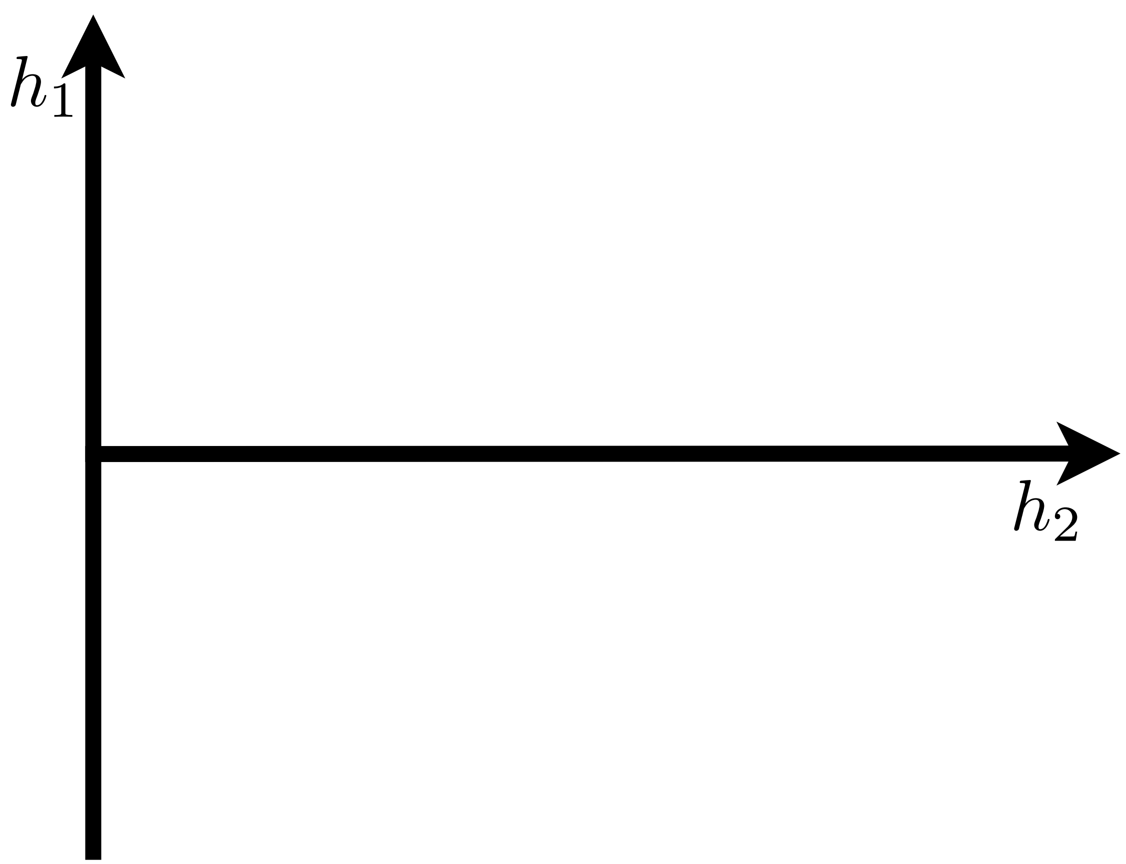



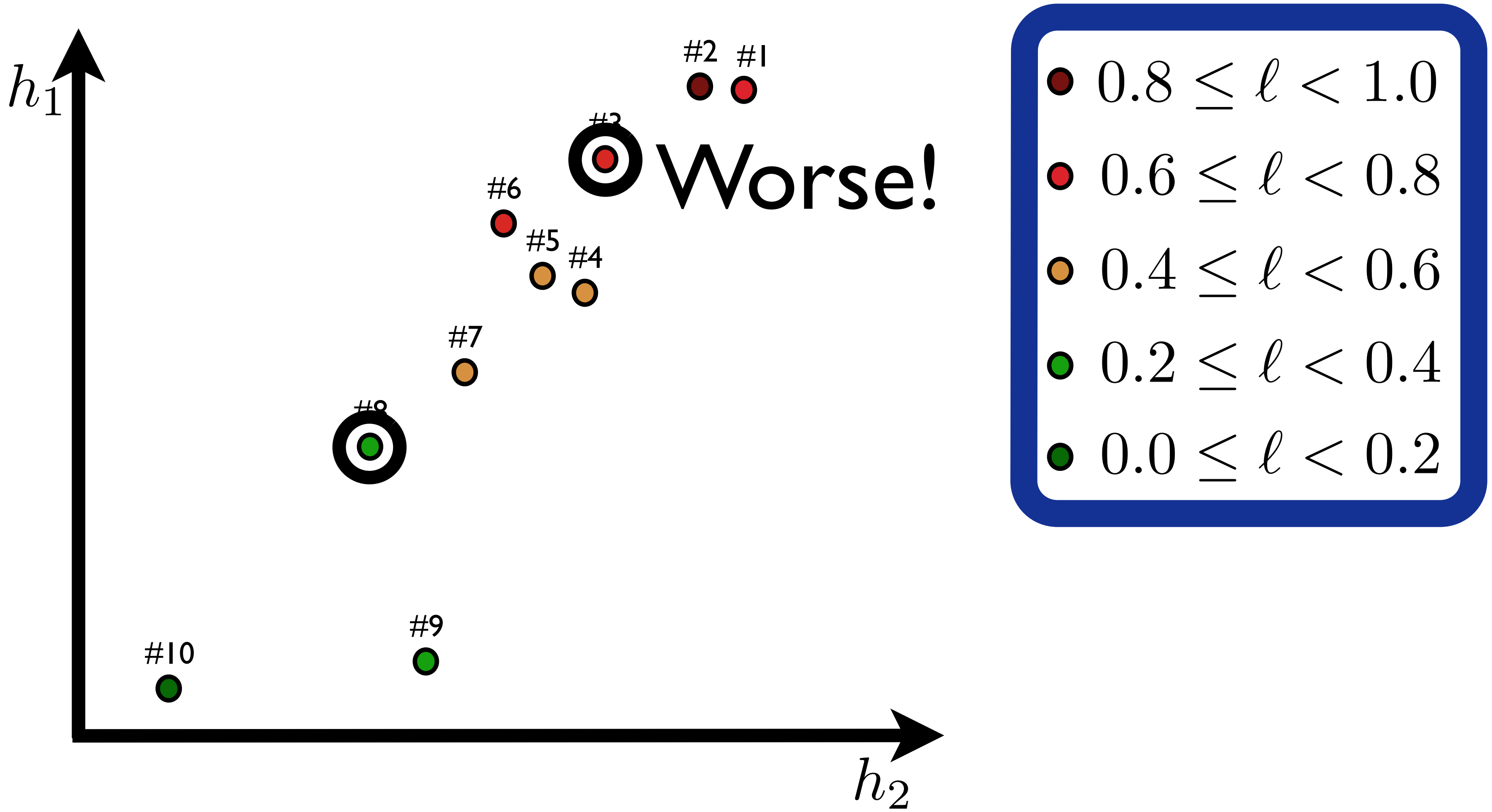
- **P**airwise **R**anking **O**ptimization
  - Reduce training problem to **binary classification** with a **linear model**
- **Algorithm**
  - For i=1 to N
    - Pick random pair of hypotheses (A,B) from K-best list
    - Use cost function to determine if is A or B better
    - Create ith training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

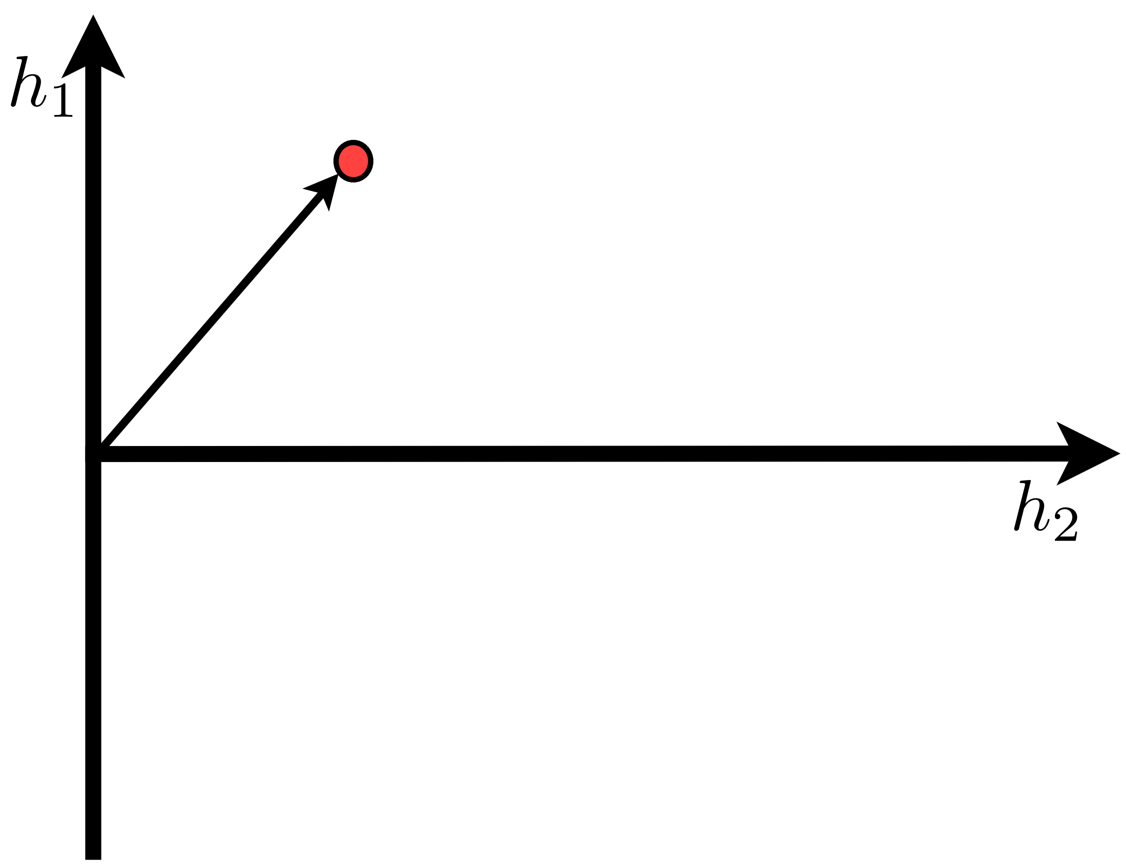

h_1
h_2
#1
#2
#3
#4
#5
#6
#7
#8
#9
#10
$0.8 \leq \ell < 1.0$
$0.6 \leq \ell < 0.8$
$0.4 \leq \ell < 0.6$
$0.2 \leq \ell < 0.4$
$0.0 \leq \ell < 0.2$

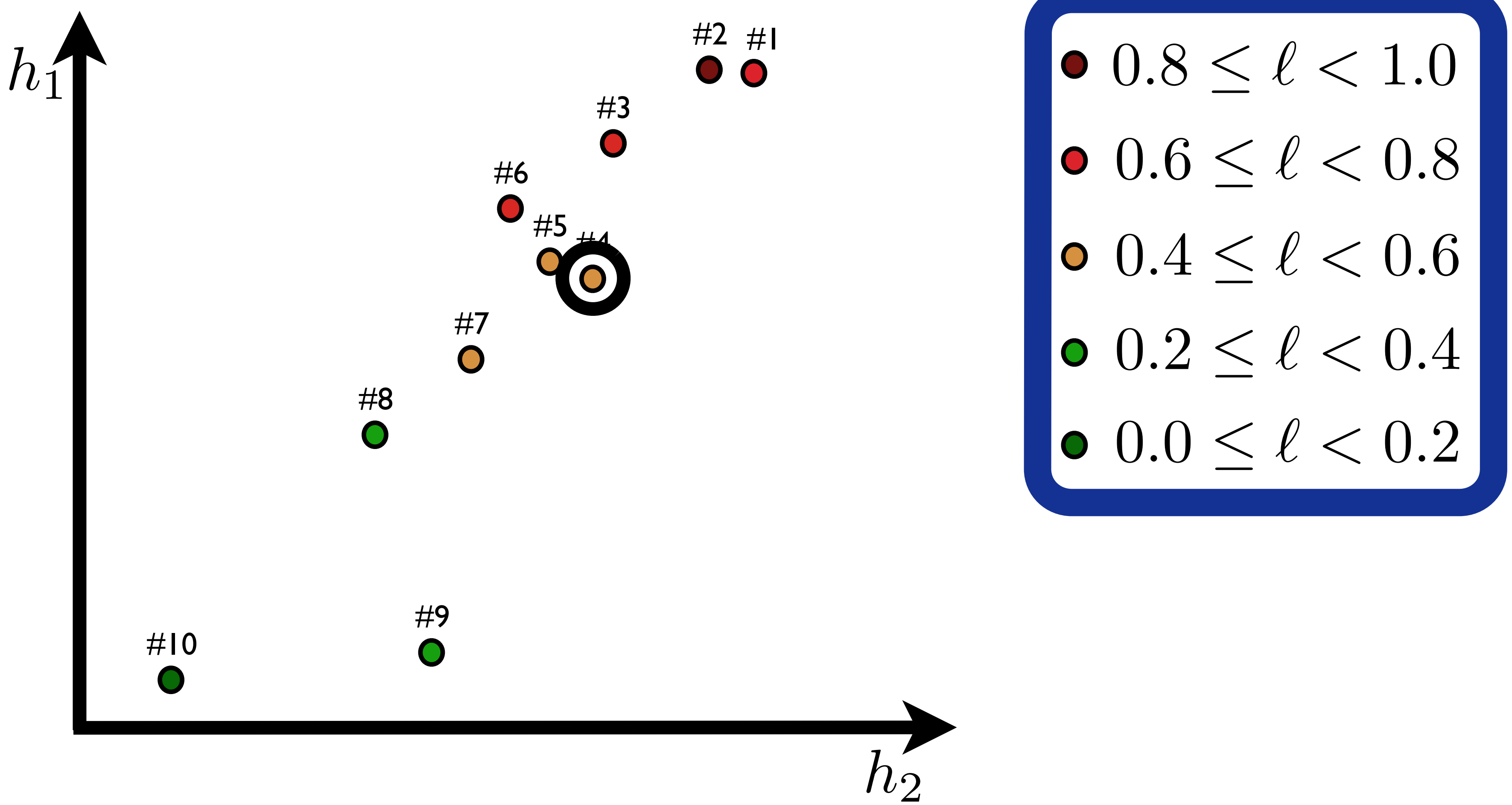

h_1
h_2

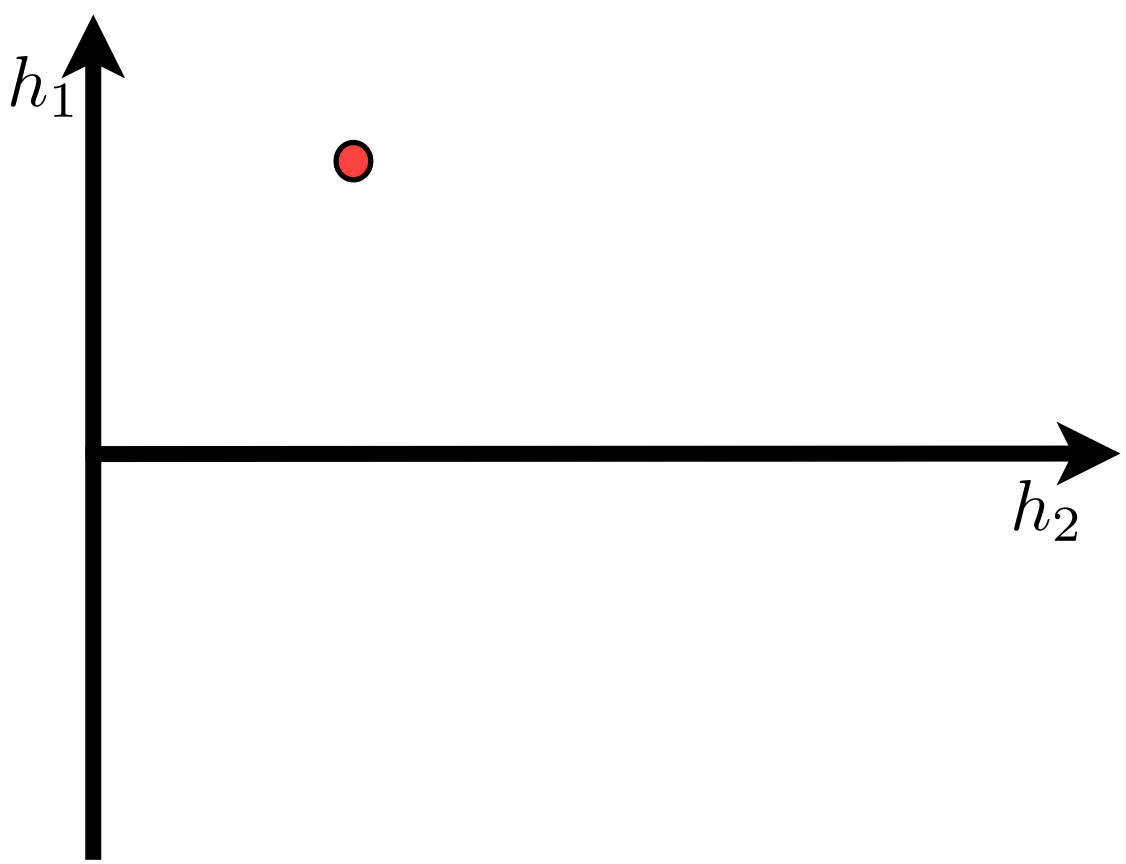

h_1
h_2
#1
#2
#4
#5
#6
#7
#9
#10
Worse!
$0.8 \leq \ell < 1.0$
$0.6 \leq \ell < 0.8$
$0.4 \leq \ell < 0.6$
$0.2 \leq \ell < 0.4$
$0.0 \leq \ell < 0.2$

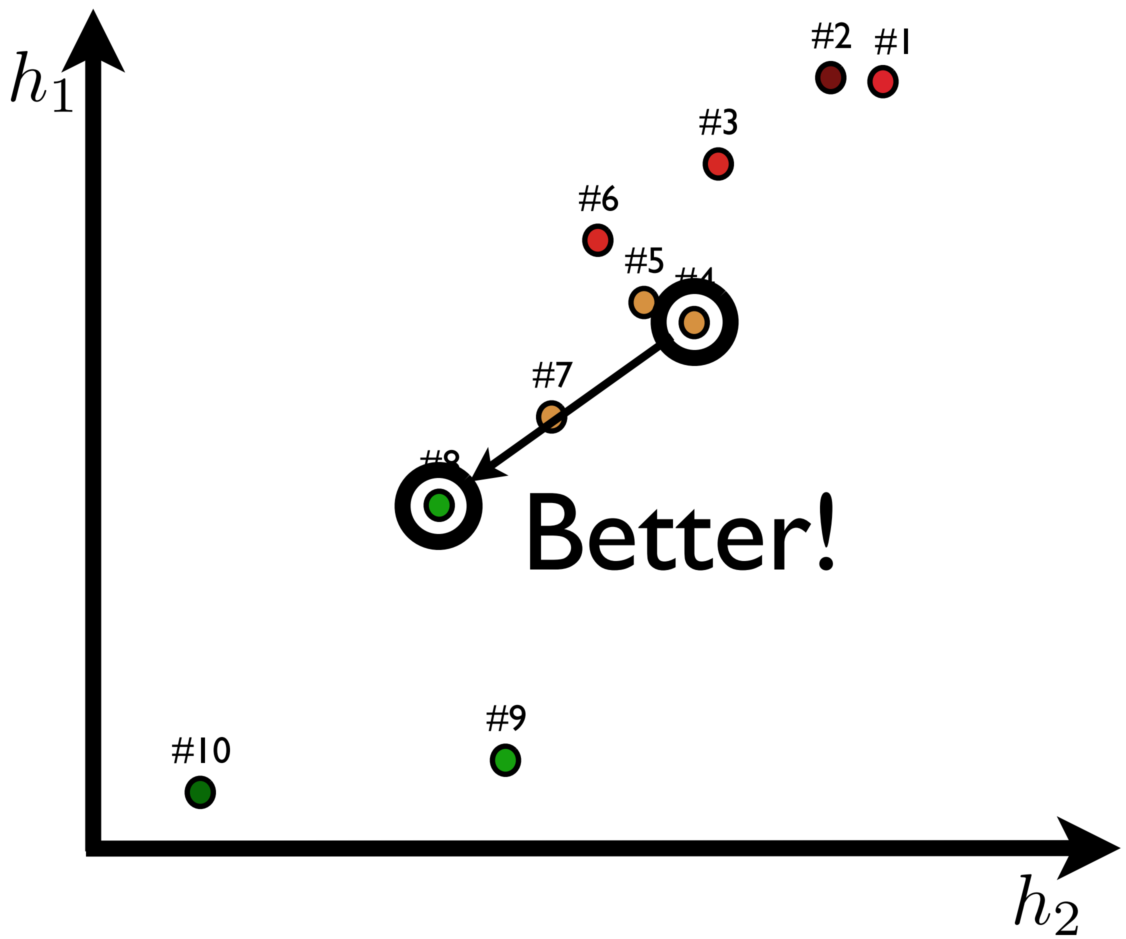

h_1
h_2


h_1
#2
#1
Worse!
#6
#5
#4
#7
#9
#10
h_2
$0.8 \leq \ell < 1.0$
$0.6 \leq \ell < 0.8$
$0.4 \leq \ell < 0.6$
$0.2 \leq \ell < 0.4$
$0.0 \leq \ell < 0.2$

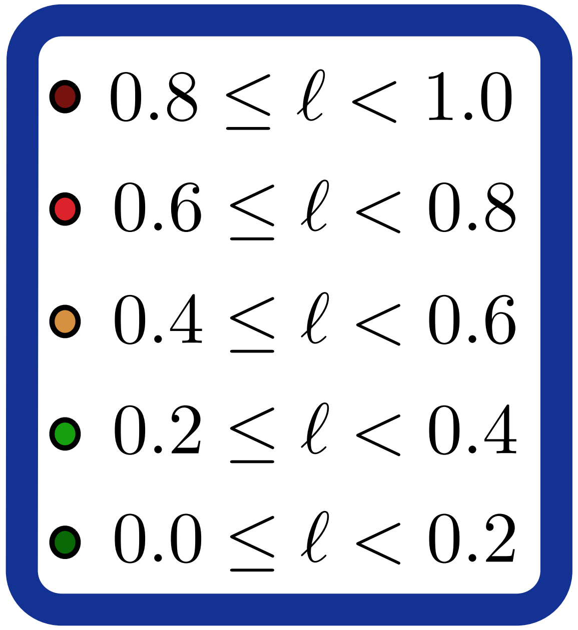

h_1
h_2


h_1
#2
#1
#3
#6
#5
#7
#8
#9
#10
h_2
$0.8 \leq \ell < 1.0$
$0.6 \leq \ell < 0.8$
$0.4 \leq \ell < 0.6$
$0.2 \leq \ell < 0.4$
$0.0 \leq \ell < 0.2$

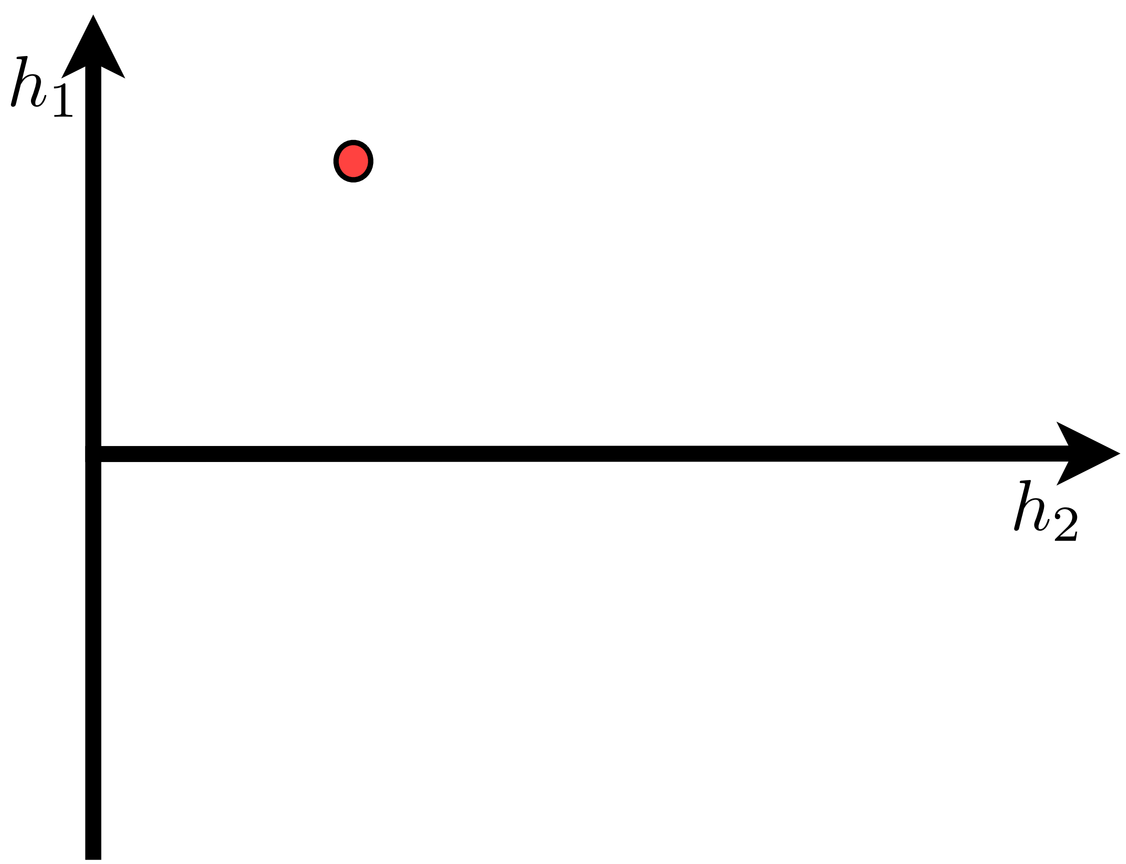

h_1
h_2



h_1
#2
#1
#3
#6
#5
#4
#7
#8
Better!
#10
#9
h_2

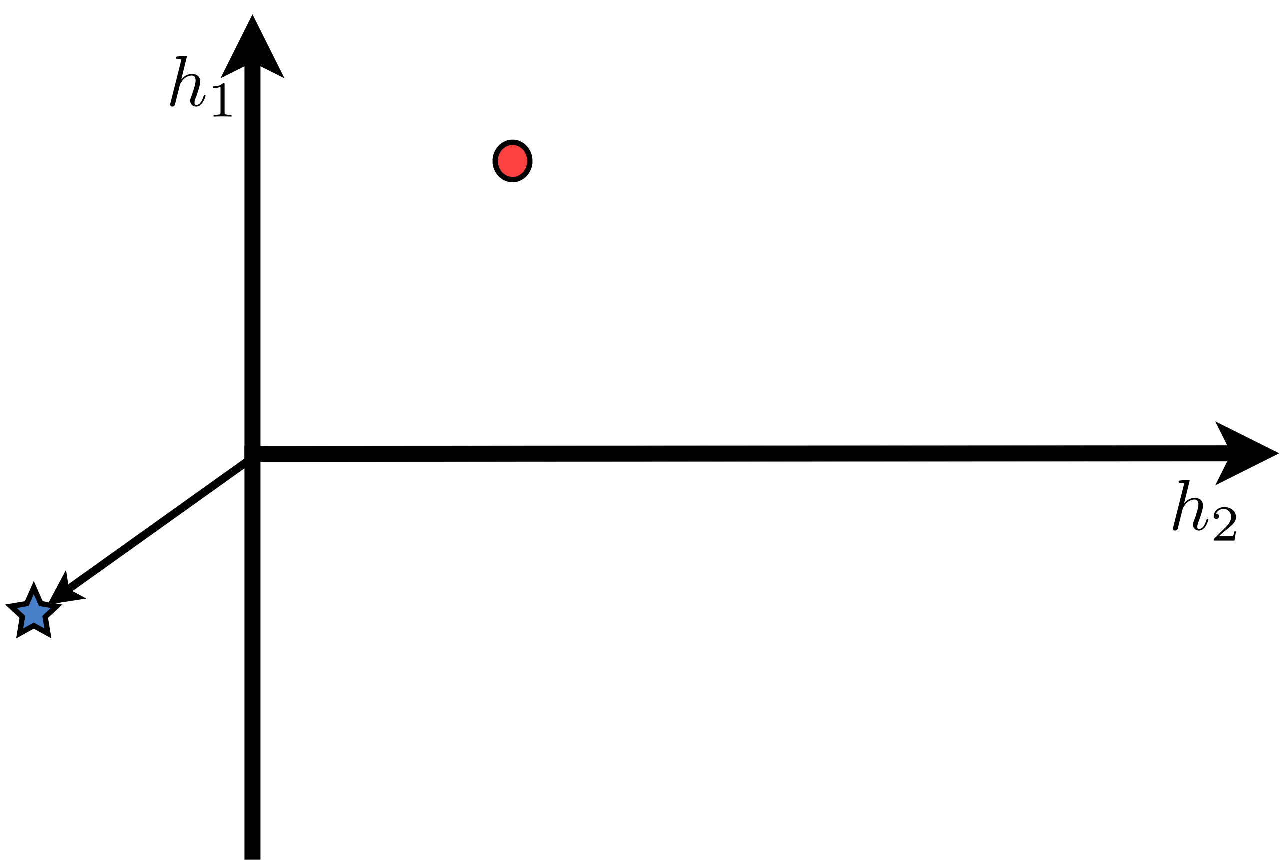


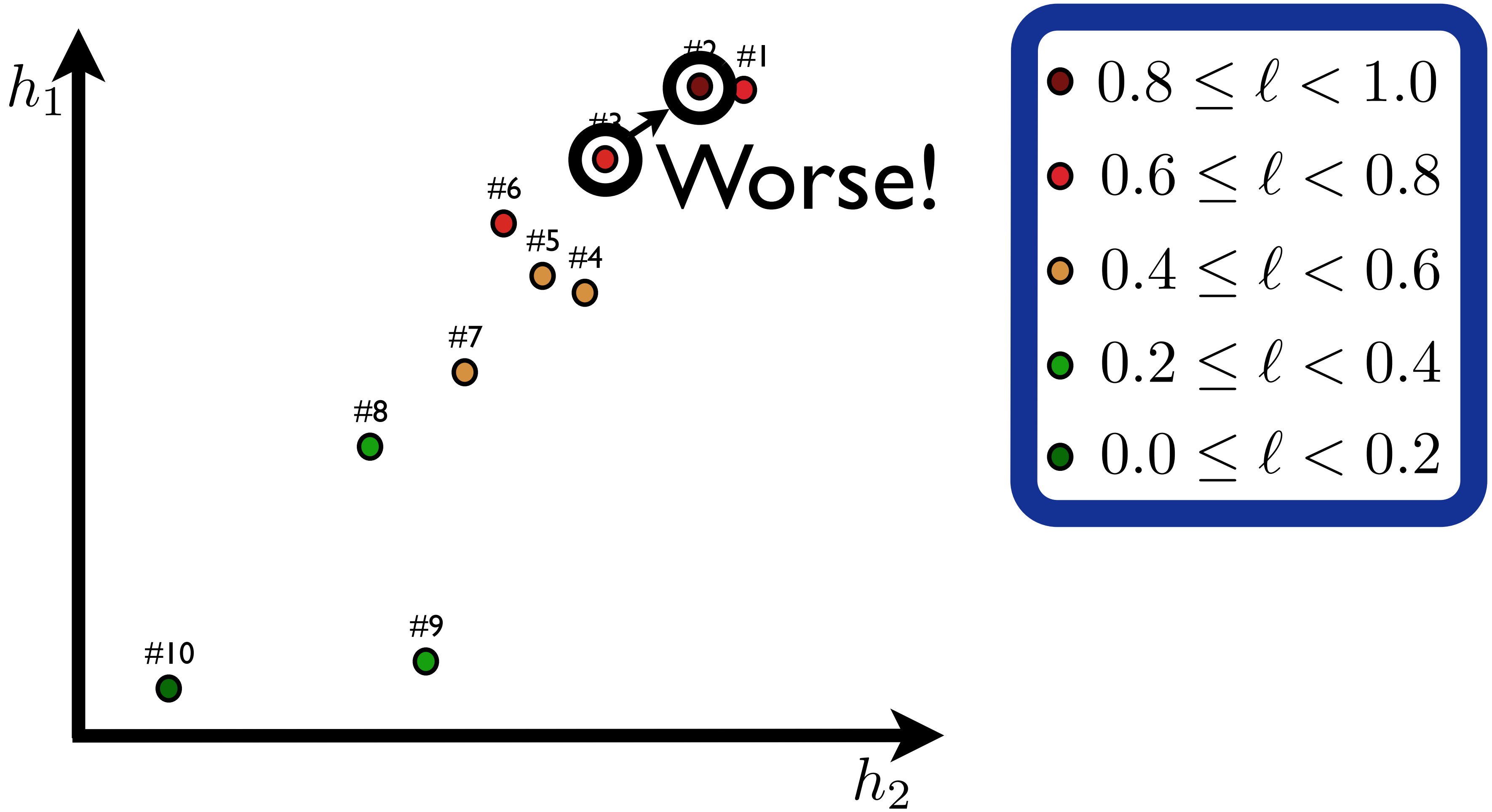
$0.8 \leq \ell < 1.0$
$0.6 \leq \ell < 0.8$
$0.4 \leq \ell < 0.6$
$0.2 \leq \ell < 0.4$
$0.0 \leq \ell < 0.2$



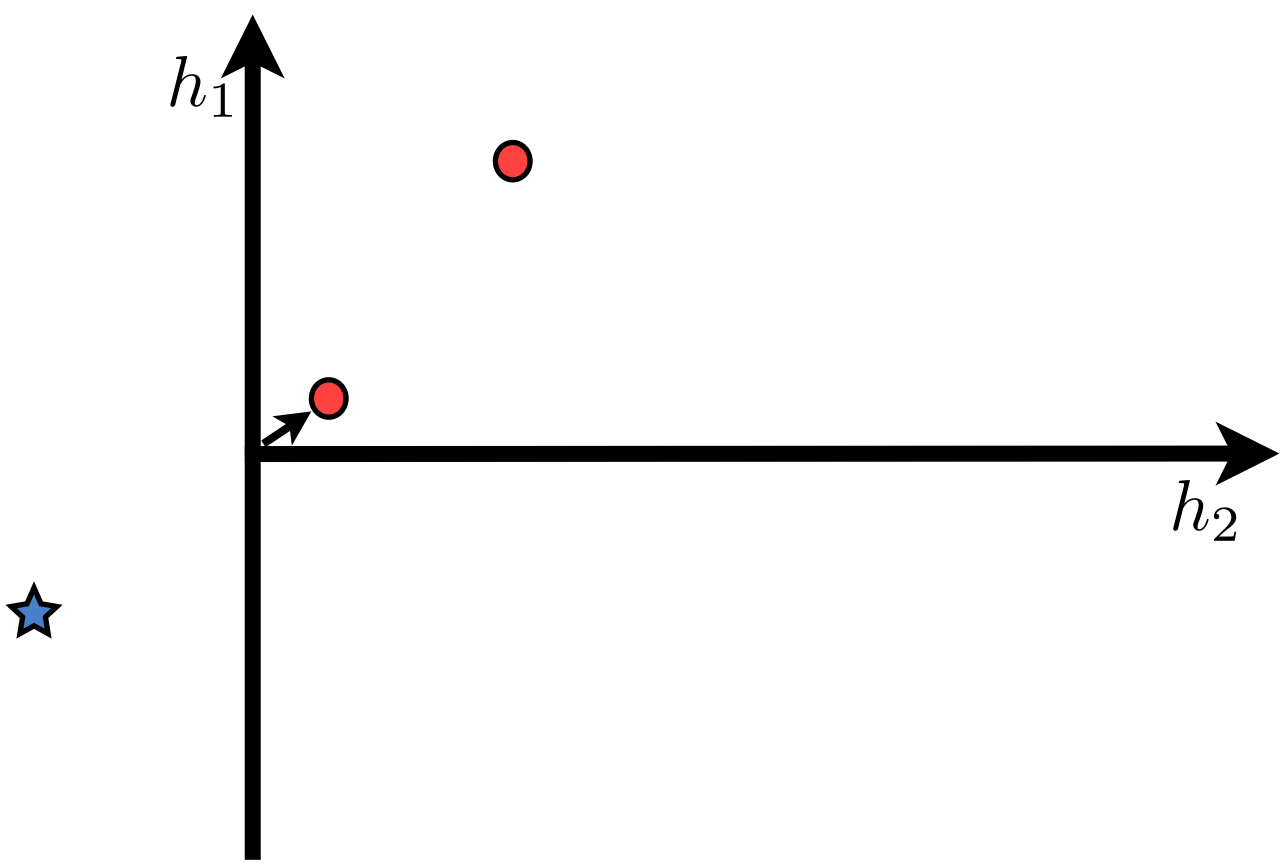
h_1
h_2

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

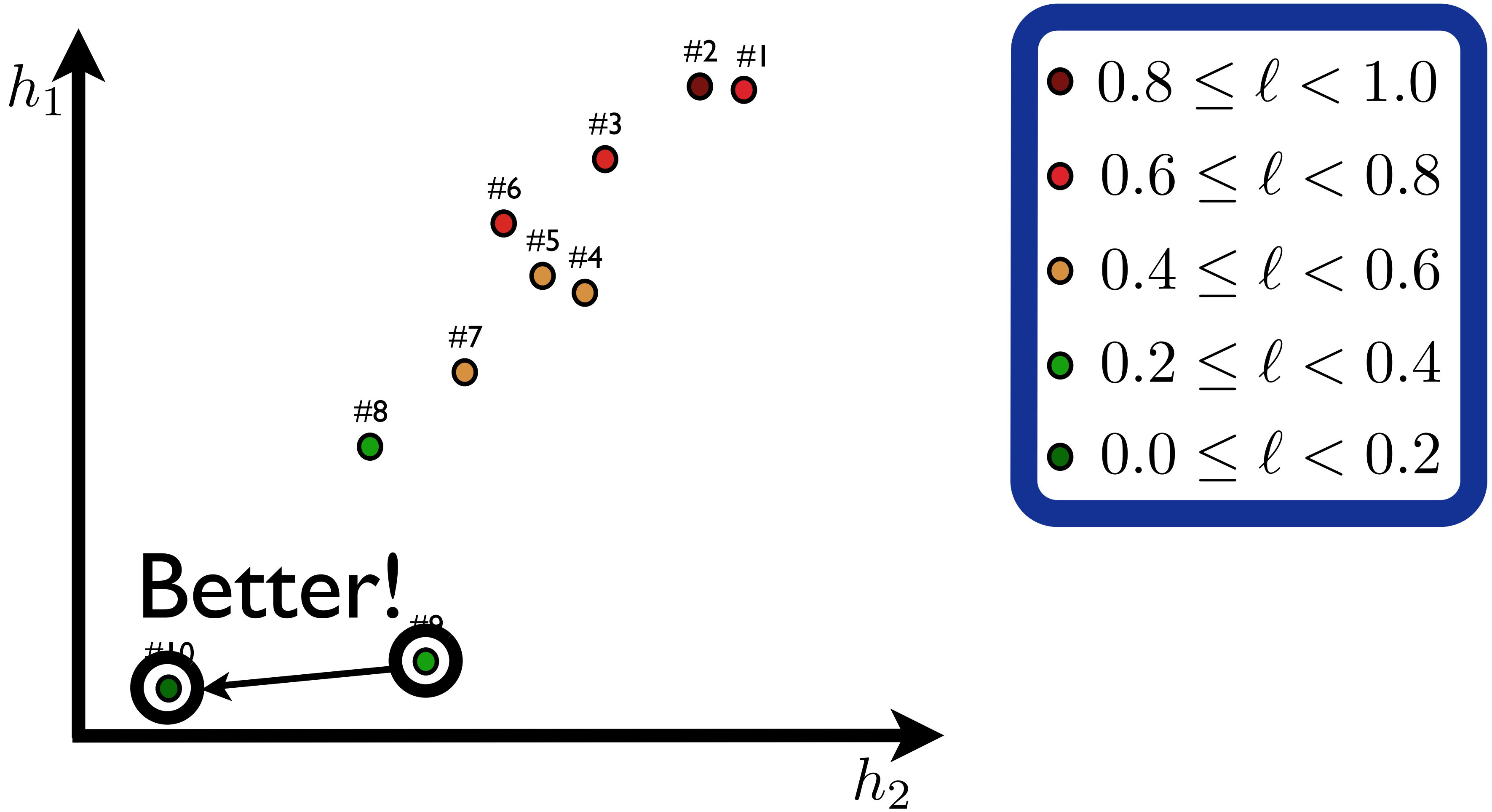


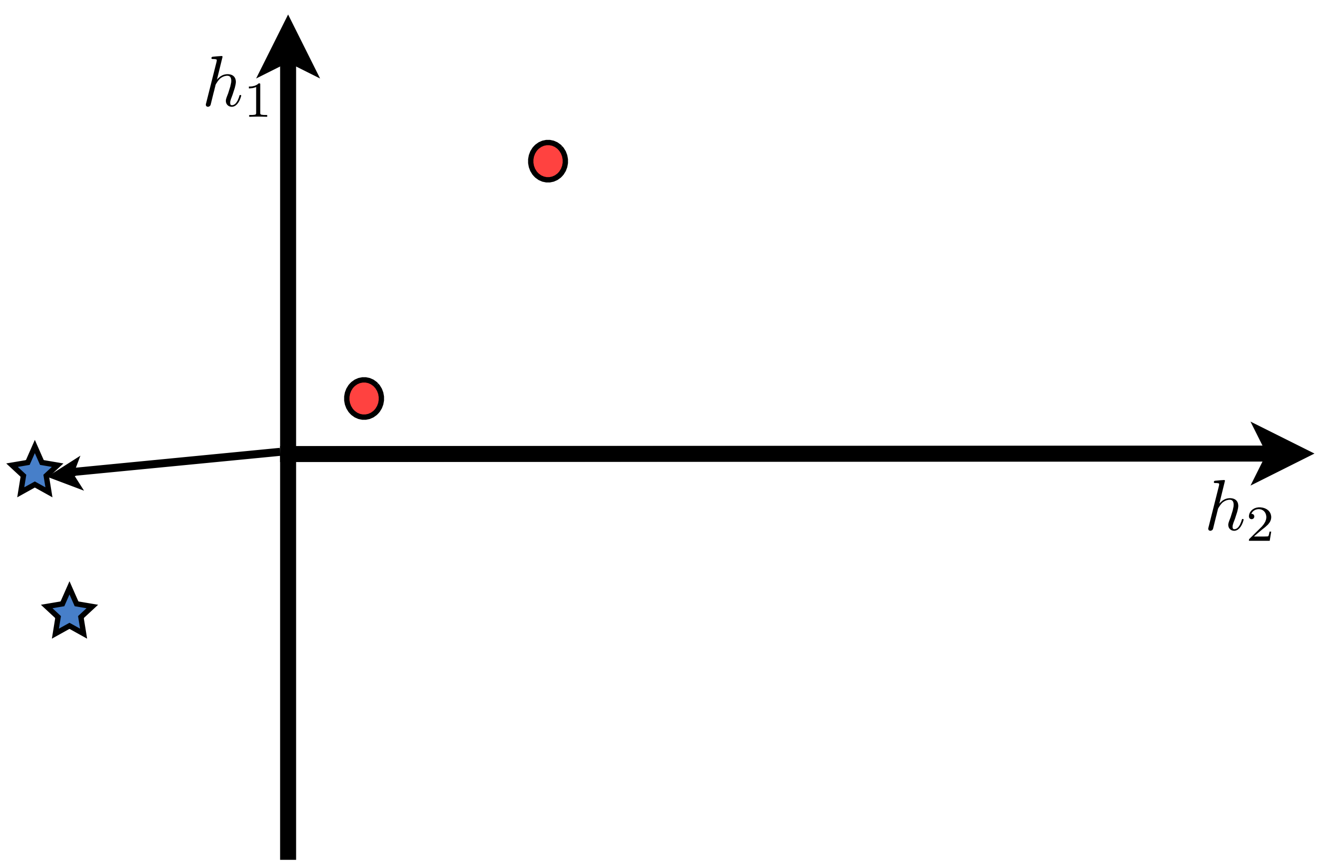


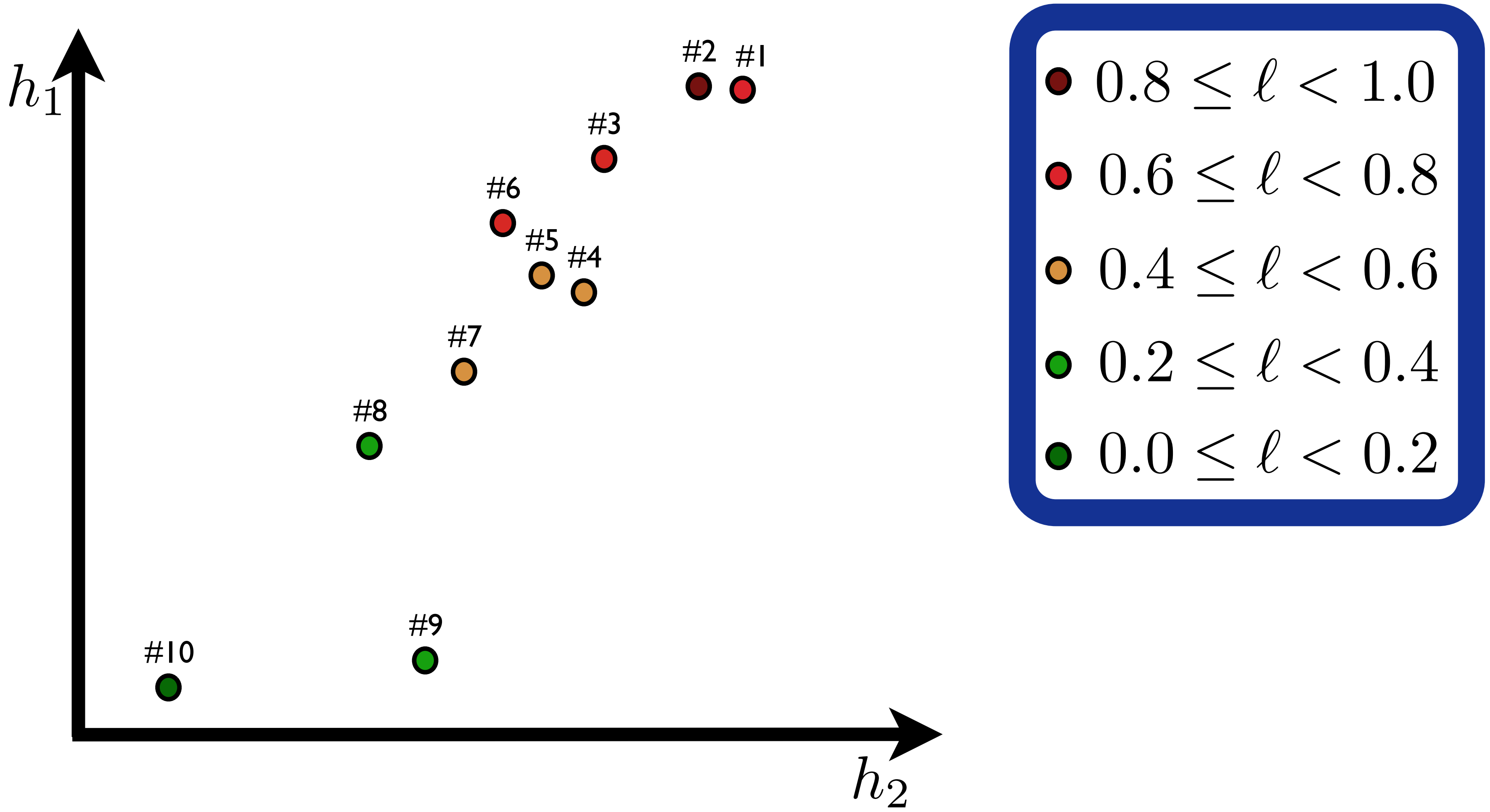
h_1
#1
Worse!
#6
#5
#4
#7
#8
#9
#10
h_2
$0.8 \leq \ell < 1.0$
$0.6 \leq \ell < 0.8$
$0.4 \leq \ell < 0.6$
$0.2 \leq \ell < 0.4$
$0.0 \leq \ell < 0.2$

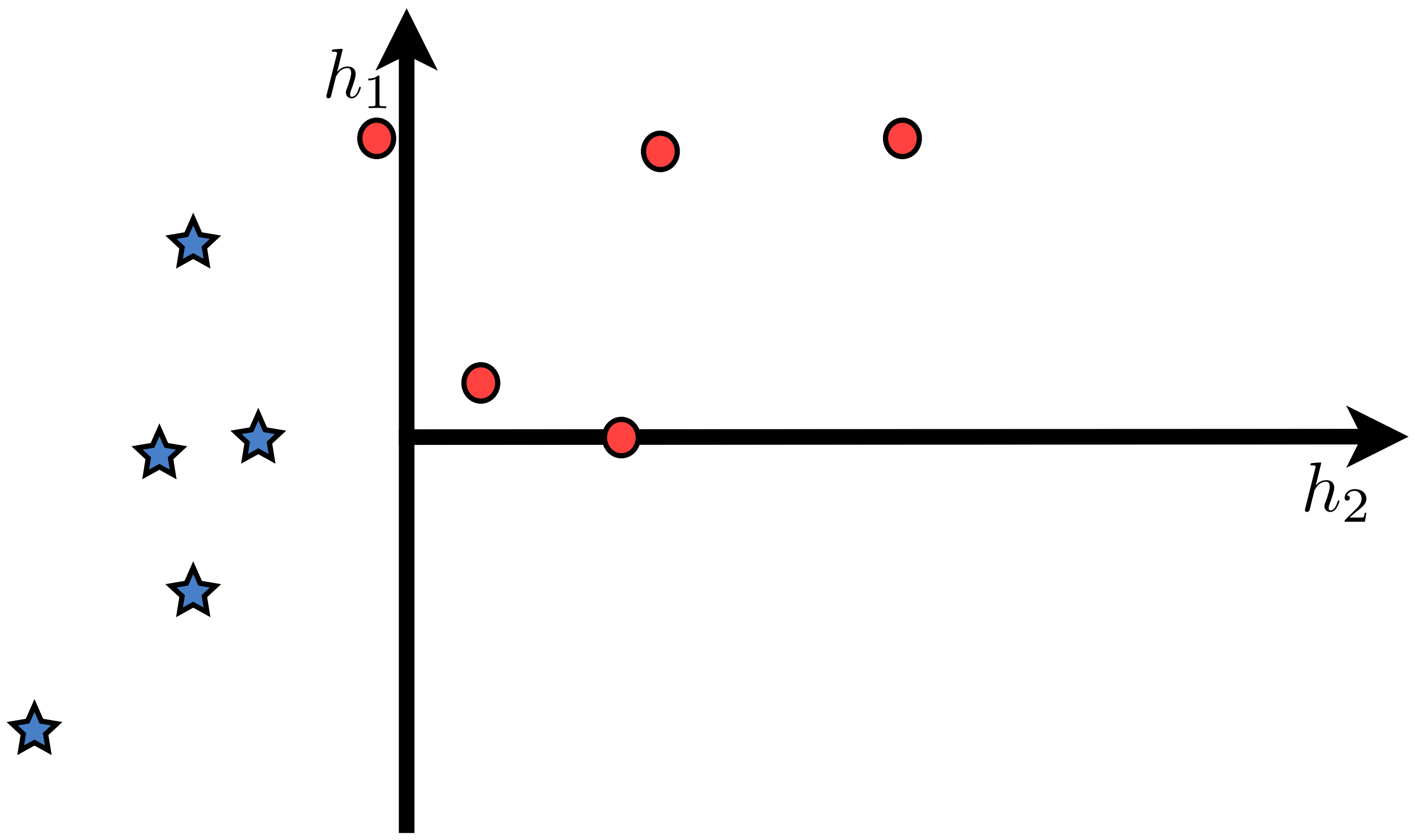


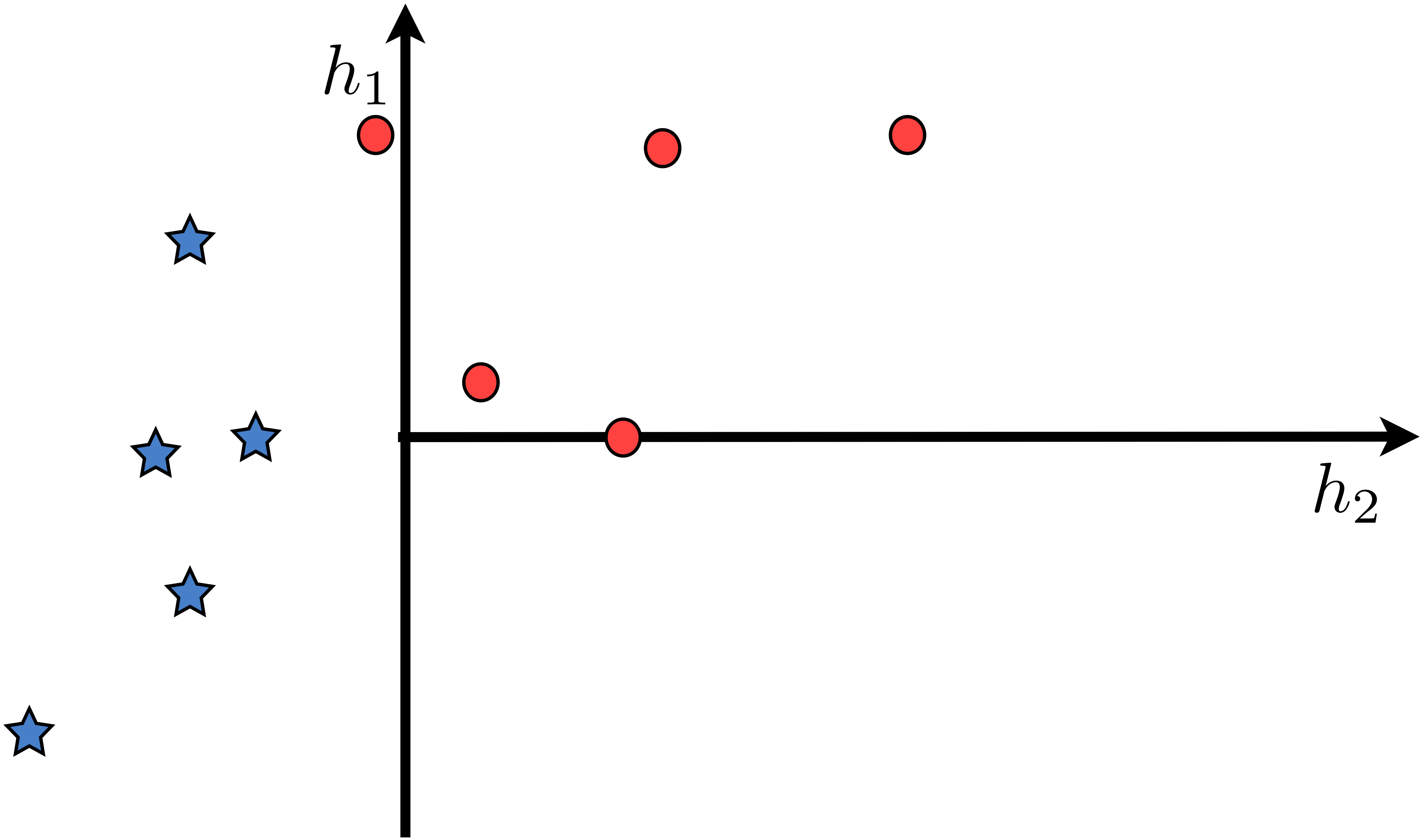
h_1
h_2


h_1
h_2
#1
#2
#3
#4
#5
#6
#7
#8
Better!
$0.8 \leq \ell < 1.0$
$0.6 \leq \ell < 0.8$
$0.4 \leq \ell < 0.6$
$0.2 \leq \ell < 0.4$
$0.0 \leq \ell < 0.2$

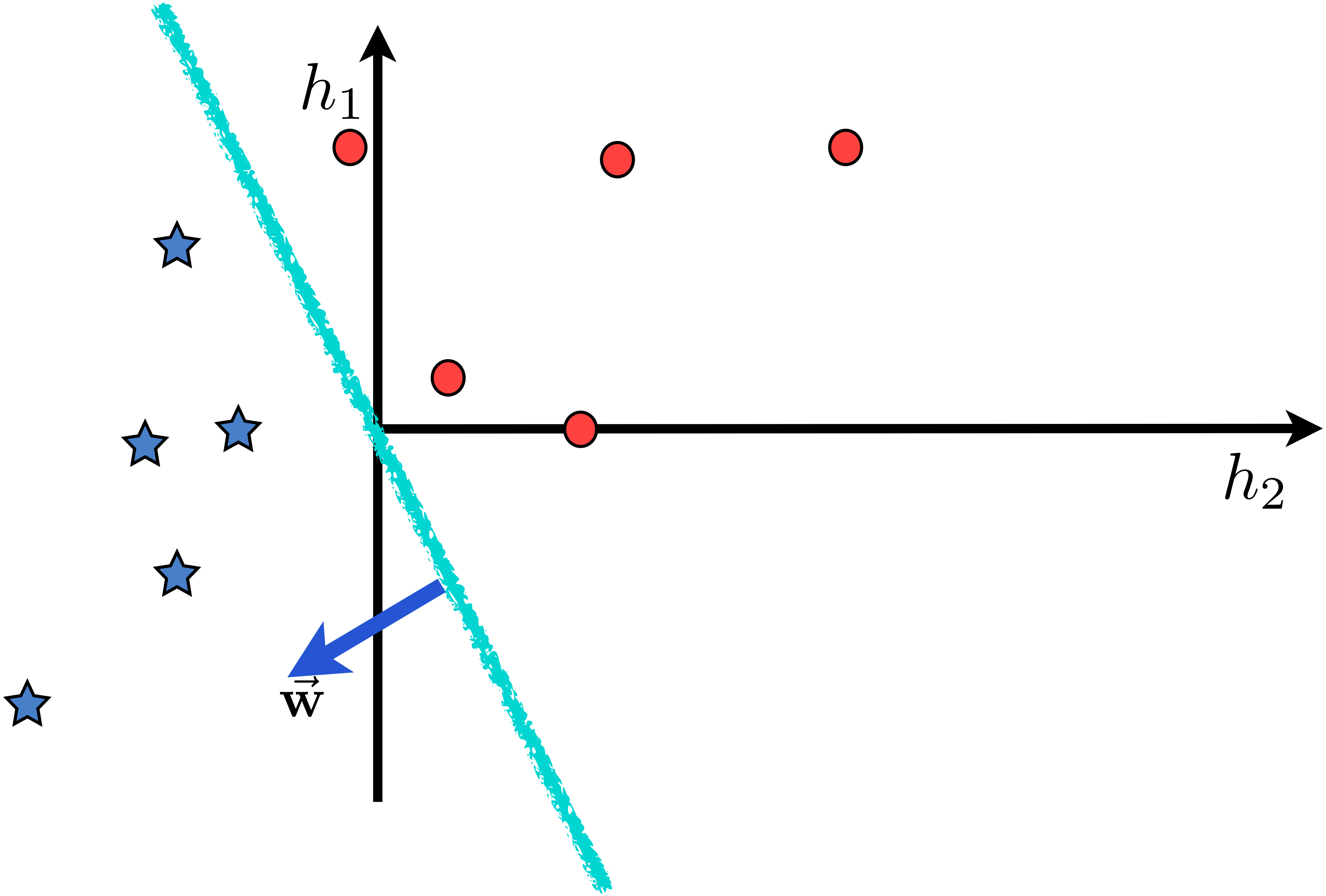

h_1
h_2


h_1
h_2
#1
#2
#3
#4
#5
#6
#7
#8
#9
#10
$0.8 \leq \ell < 1.0$
$0.6 \leq \ell < 0.8$
$0.4 \leq \ell < 0.6$
$0.2 \leq \ell < 0.4$
$0.0 \leq \ell < 0.2$


h_1
h_2



Fit a linear model



Fit a linear model

K-Best List Example