Language and Statistics II

Lecture 9: Log-linear models (learning)

Quick Review

Input/observable space $\mathcal{X}$

Output/label space $\mathcal{Y}$

Feature function: $f_j : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R} \quad (\forall j \in \{1, 2, ..., d\})$

Feature vector function: $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$

Weight vector: $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix}$

Score: $\boldsymbol{\theta}^\top \mathbf{f}(x, y) \quad (\forall x \in \mathcal{X}, \forall y \in \mathcal{Y})$

Positive score: $\exp\left(\boldsymbol{\theta}^\top \mathbf{f}(x, y)\right) \quad (\forall x \in \mathcal{X}, \forall y \in \mathcal{Y})$

Probability: $\exp\left(\boldsymbol{\theta}^\top \mathbf{f}(x, y)\right) \Big/ z(x, \boldsymbol{\theta}) \quad (\forall x \in \mathcal{X}, \forall y \in \mathcal{Y})$

Maximum Likelihood Estimation

- Given a model family, pick the parameters to maximize

  p(data | model)

- Examples:
  - Gaussian:

$$\hat{\mu} = \sum_{i=1}^{N} x_i \bigg/ N$$

$$\hat{\sigma} = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{\mu})^2}{N}}$$

$$\hat{p} = \frac{N_{\text{success}}}{N}$$

  - Bernoulli:
  - Multinomial:

$$\hat{p}(x) = \sum_{i=1}^{N} \delta(x_i, x) \bigg/ N = N_x / N$$

closed form solution

Maximum Likelihood Estimation

- Given a model family, pick the parameters to maximize

  p(data | model)

- Examples:
  - Gaussian:

$$\hat{\mu} = \sum_{i=1}^{N} x_i \Big/ N$$

$$\hat{\sigma} = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{\mu})^2}{N}}$$

closed form solution

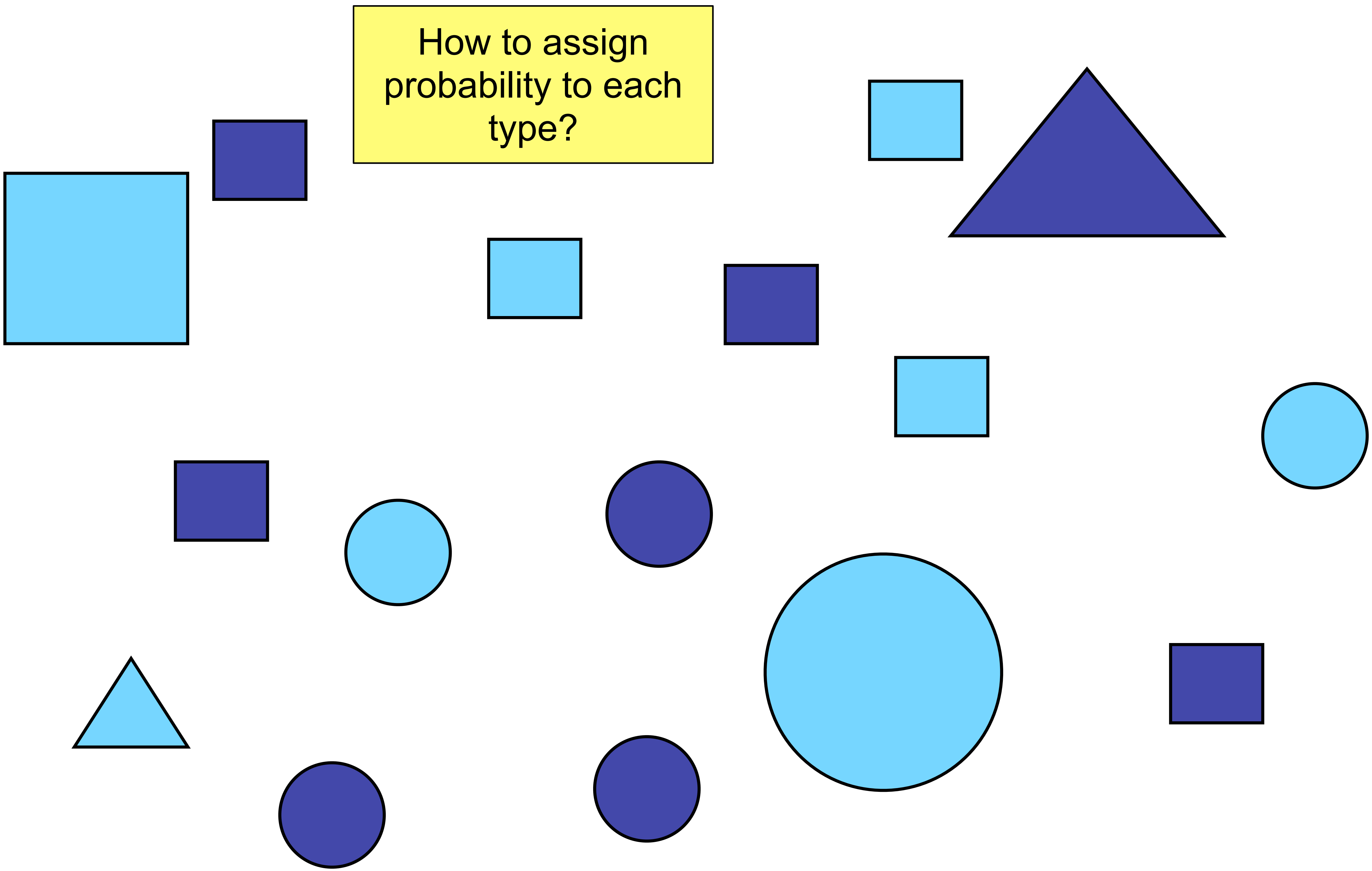
$$\hat{p} = \frac{N_{\text{success}}}{N}$$

  - Bernoulli:
  - Multinomial:

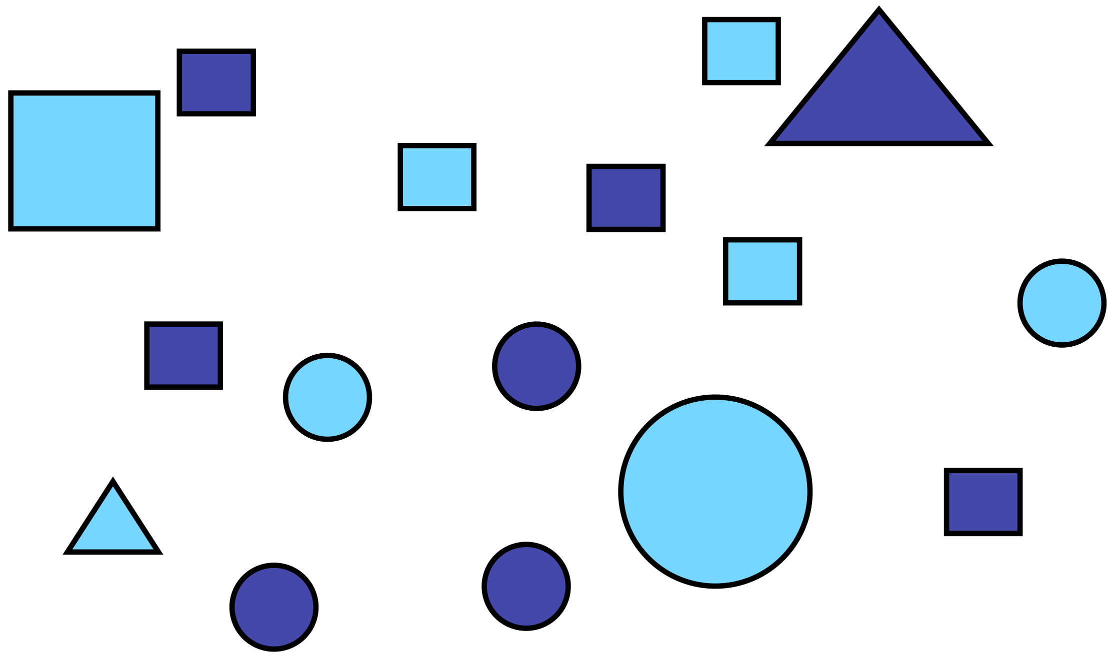
$$\hat{p}(x) = \sum_{i=1}^{N} \delta(x_i, x) \Big/ N = N_x / N$$

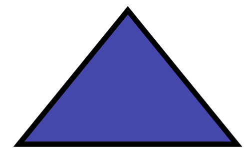
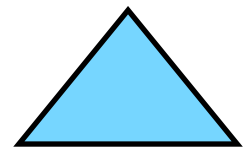
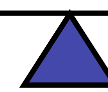
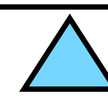
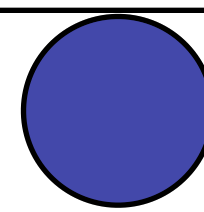
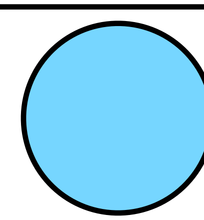
  – *n*-gram model? HMM?

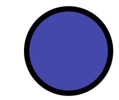
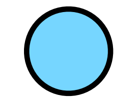
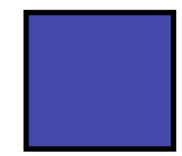
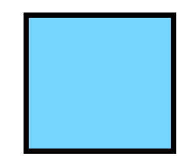
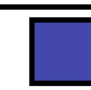
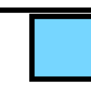
Data



Maximum Likelihood (Multinomial)



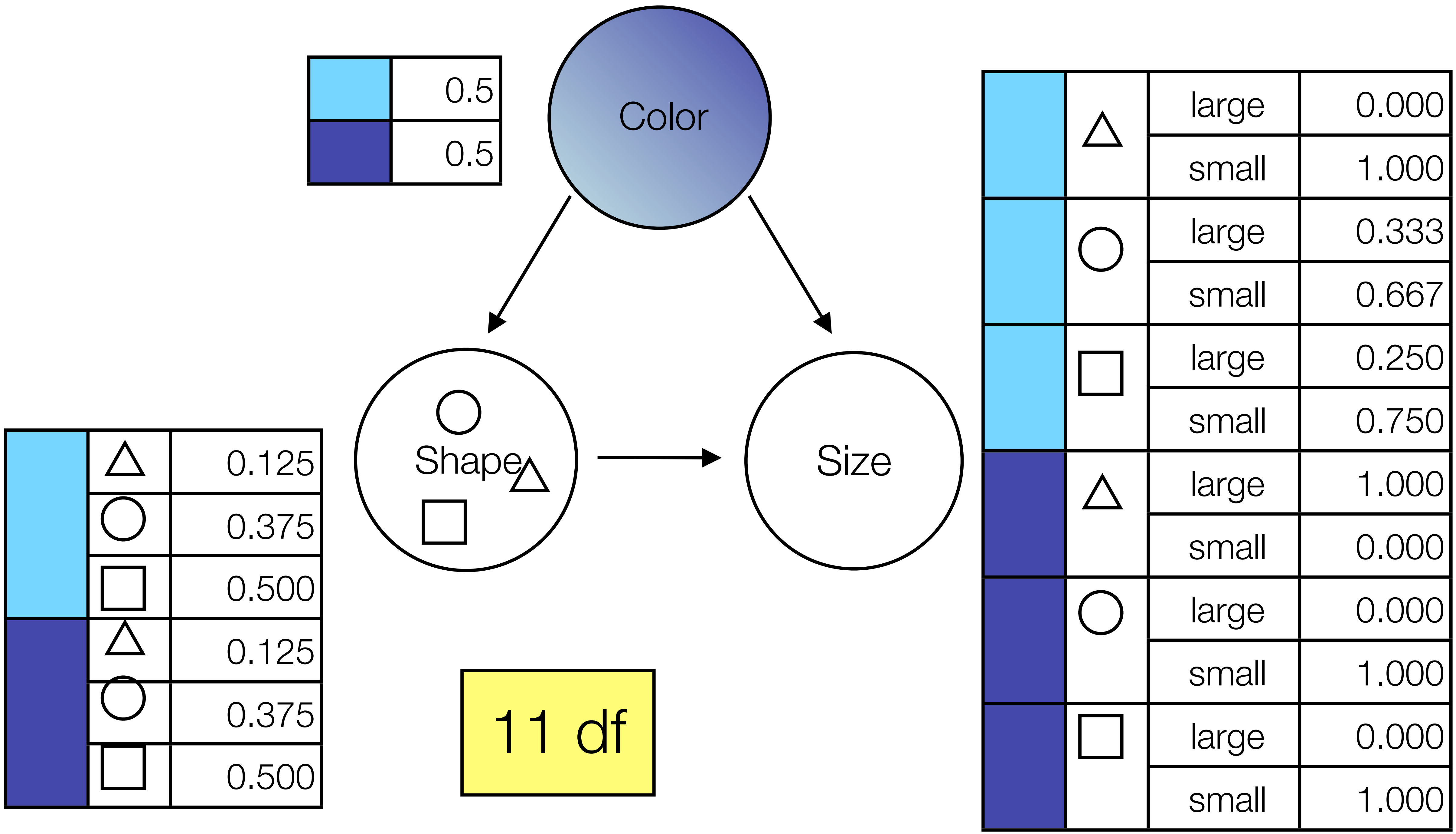
	1	0.06
	0	0
	0	0
	1	0.06
	0	0
	1	0.06

	3	0.19
	2	0.12
	0	0
	1	0.06
	4	0.25
	3	0.19

Overfitting?

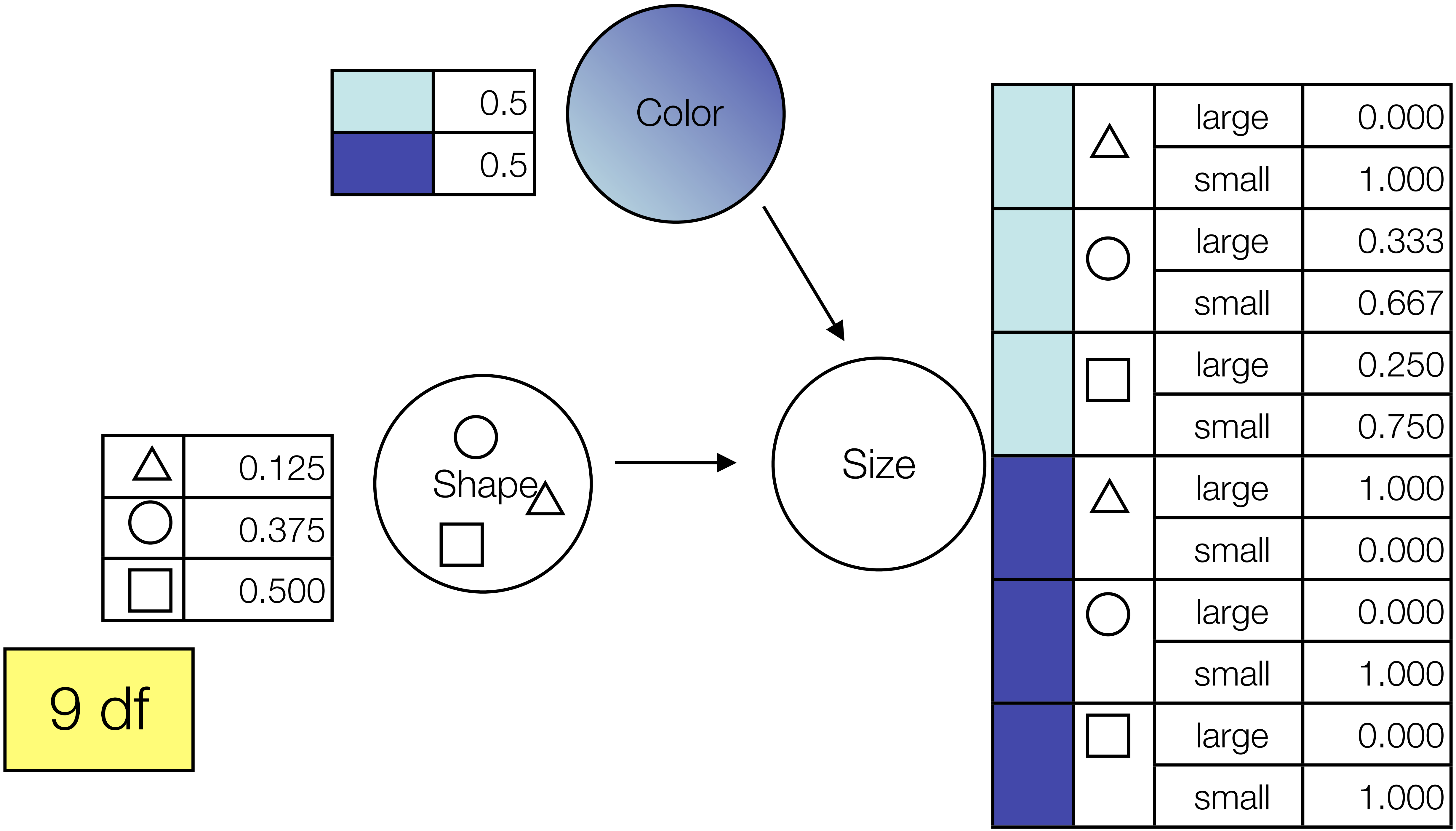
11 df

Using the Chain Rule



Pr(Color, Shape, Size) = Pr(Color) • Pr(Shape | Color) • Pr(Size | Color, Shape)

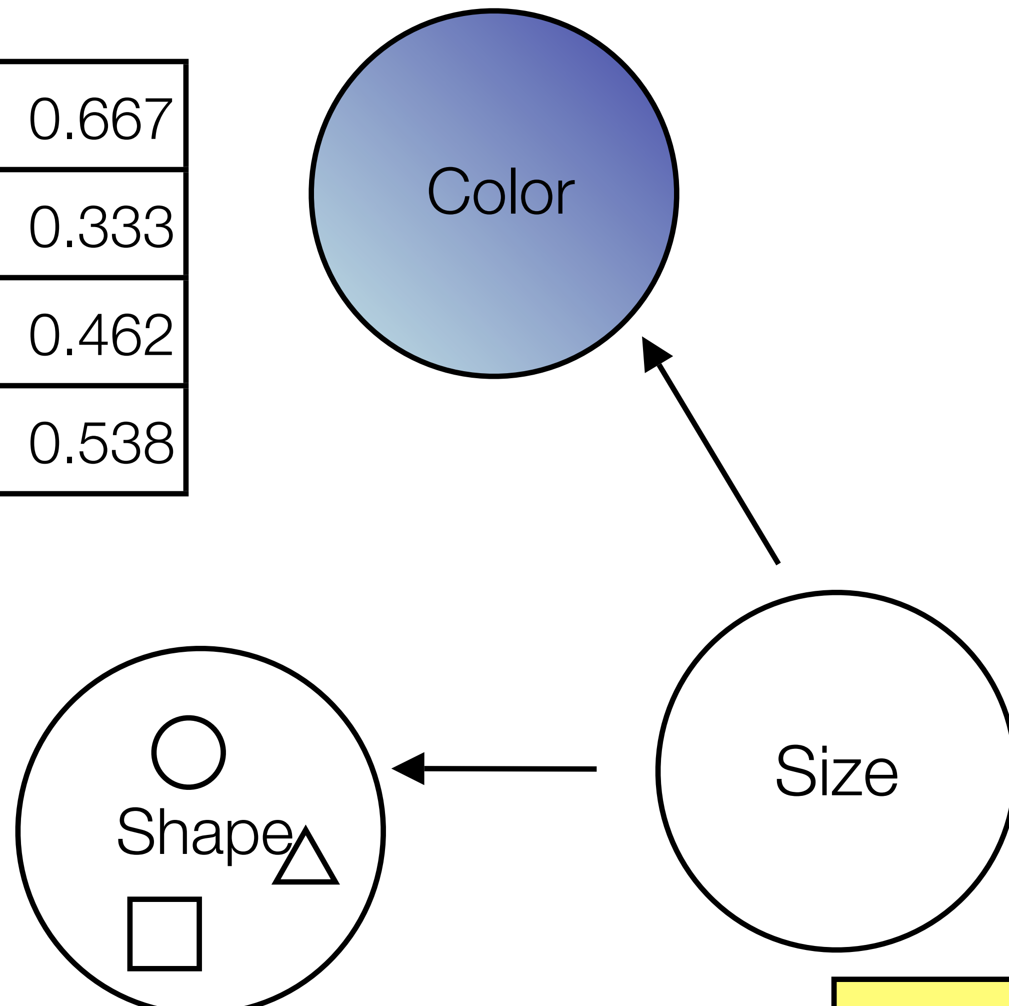
Add an Independence Assumption?



Pr(Color, Shape, Size) = Pr(Color) • Pr(Shape) • Pr(Size | Color, Shape)

Reverse Arrows?

large		0.667
		0.333
small		0.462
		0.538



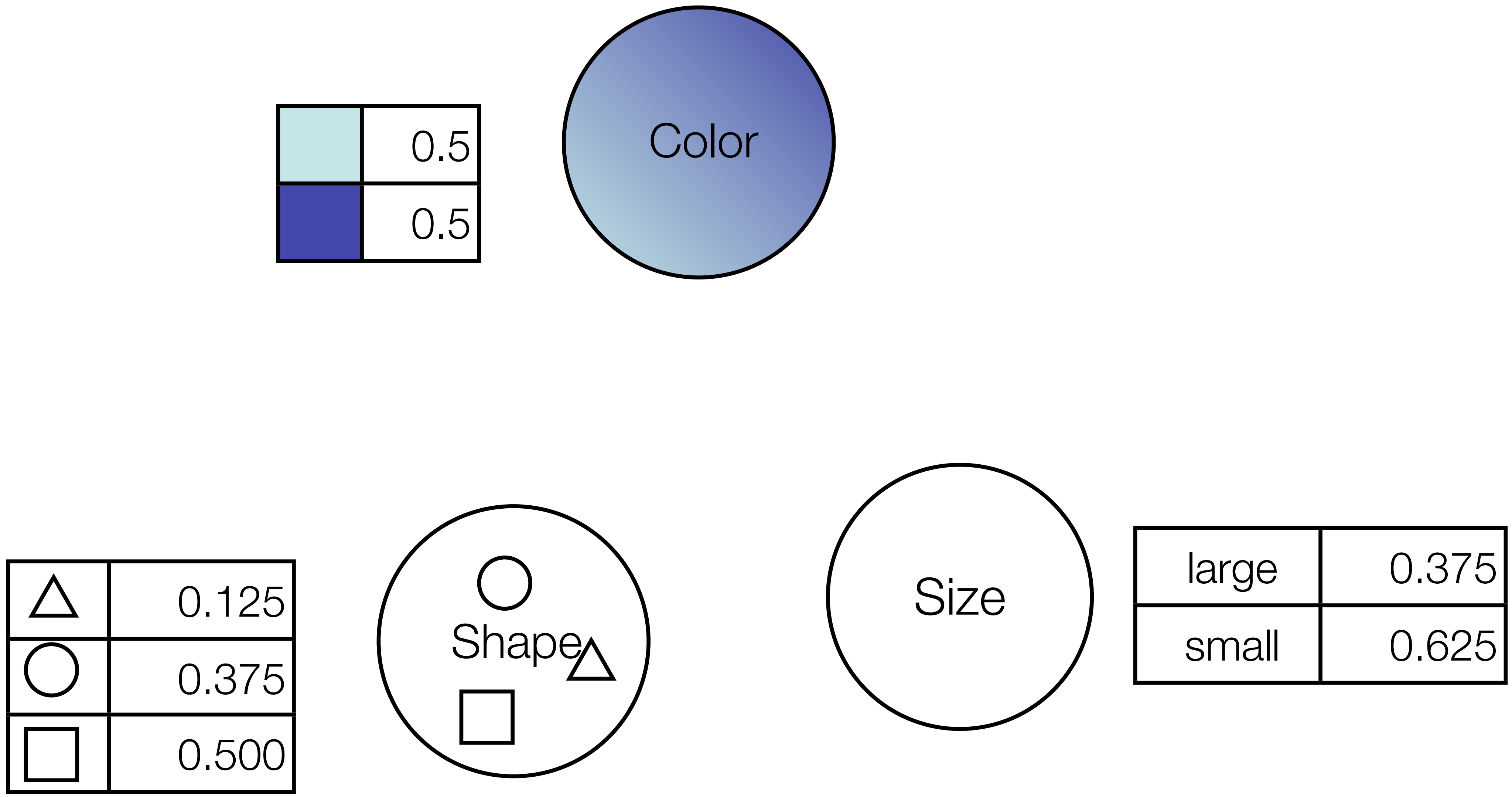
large	△	0.333
	○	0.333
	□	0.333
small	△	0.077
	○	0.385
	□	0.538

large	0.375
small	0.625

7 df

Pr(Color, Shape, Size) = Pr(Size) • Pr(Shape | Size) • Pr(Color | Size)

Strong Independence?



4 df

Pr(Color, Shape, Size) = Pr(Size) • Pr(Shape) • Pr(Color)

This Is Hard!

- Different **factorizations** affect
  - Model size (e.g., number of parameters or df)
  - Complexity of inference
  - "Interpretability"
  - Goodness of fit to the data
  - Generalization
  - Smoothing methods
- How would it change if we used **log-linear** models?
- Arguable: some major "innovations" in NLP involved really good choices about independence assumptions, directionality, and smoothing!

A Log-Linear Shape Model

How do we set the weights?

How do we pick the features?

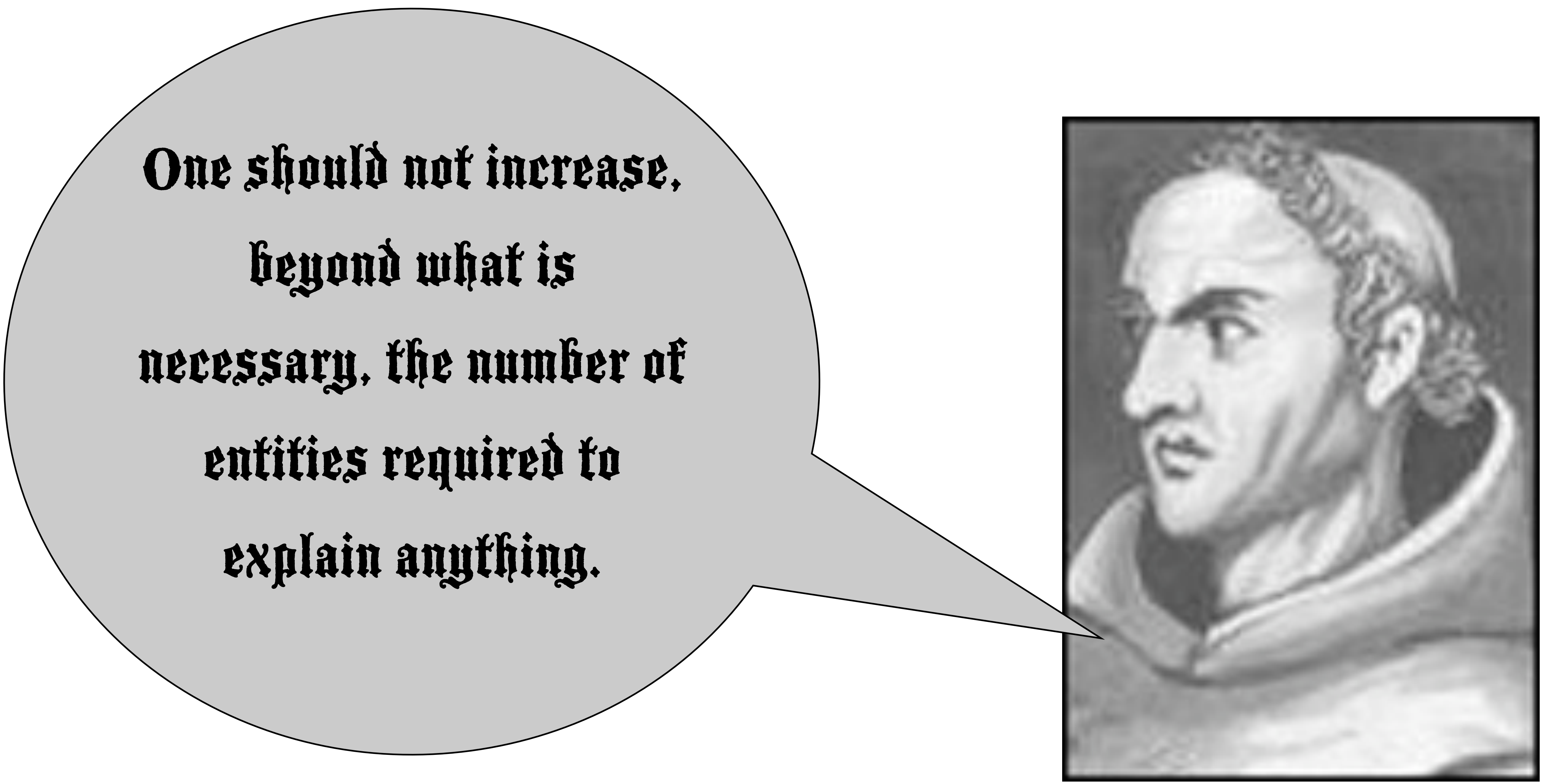
$$p(\text{shape}) = \frac{\exp \boldsymbol{\theta}^\top \mathbf{f}(\text{shape})}{Z(\boldsymbol{\theta})}$$

Desideratum: after we pick features, picking the weights should be the computer's job!

Some Intuitions

- Simpler models are better
  - (E.g., fewer degrees of freedom)
  - Why?
- Want to fit the data
- Don’t want to assume that an unobserved event has probability 0

Occam's Razor



Uniform Model

	△	○	□
small	0.083	0.083	0.083
small	0.083	0.083	0.083
large	0.083	0.083	0.083
large	0.083	0.083	0.083

Constraint: Pr(small) = 0.625

	△	○	□
small	0.104	0.104	0.104
small	0.104	0.104	0.104
large	0.063	0.063	0.063
large	0.063	0.063	0.063

0.625

Where did the constraint come from?

Constraint: Pr(small,△) = 0.048

0.048

	△	○	□
small	0.024	0.144	0.144
small	0.024	0.144	0.144
large	0.063	0.063	0.063
large	0.063	0.063	0.063

0.625

Constraint: Pr(large,) = 0.125

0.048

	△	○	□
small	0.024	0.144	0.144
small	0.024	0.144	0.144
large	0.042	0.042	0.042
large	0.083	0.083	0.083

0.625

0.125

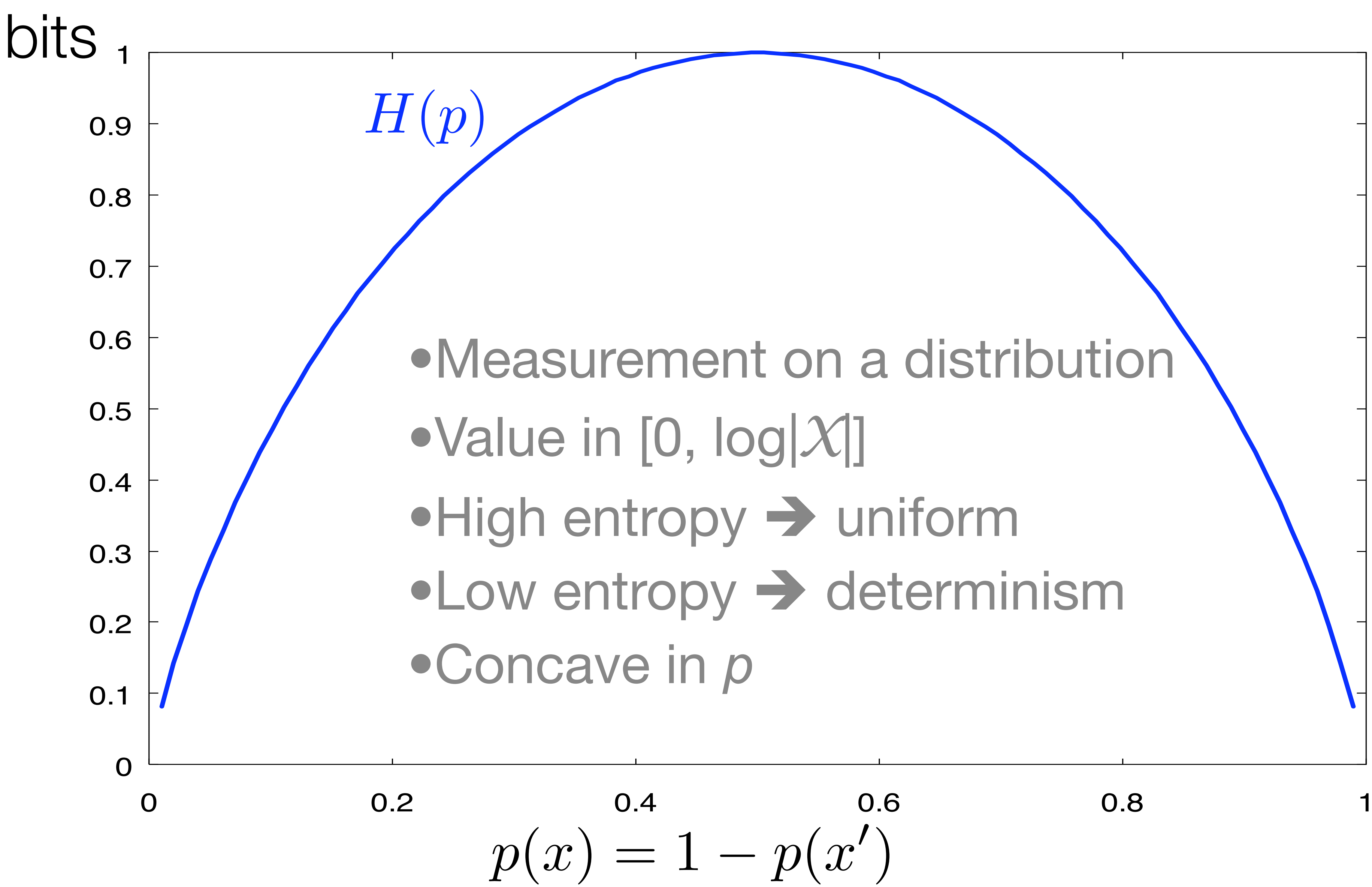
“As Evenly As Possible”

(given the constraints)

Shannon Entropy Review

- Measurement on a distribution
- Value in $[0, \log|\mathcal{X}|]$
- High entropy ➔ uniform
- Low entropy ➔ determinism
- Concave in p

Shannon Entropy Review



- Measurement on a distribution
- Value in [0, log$|\mathcal{X}|$]
- High entropy ➔ uniform
- Low entropy ➔ determinism
- Concave in *p*

The Maximum Entropy Problem

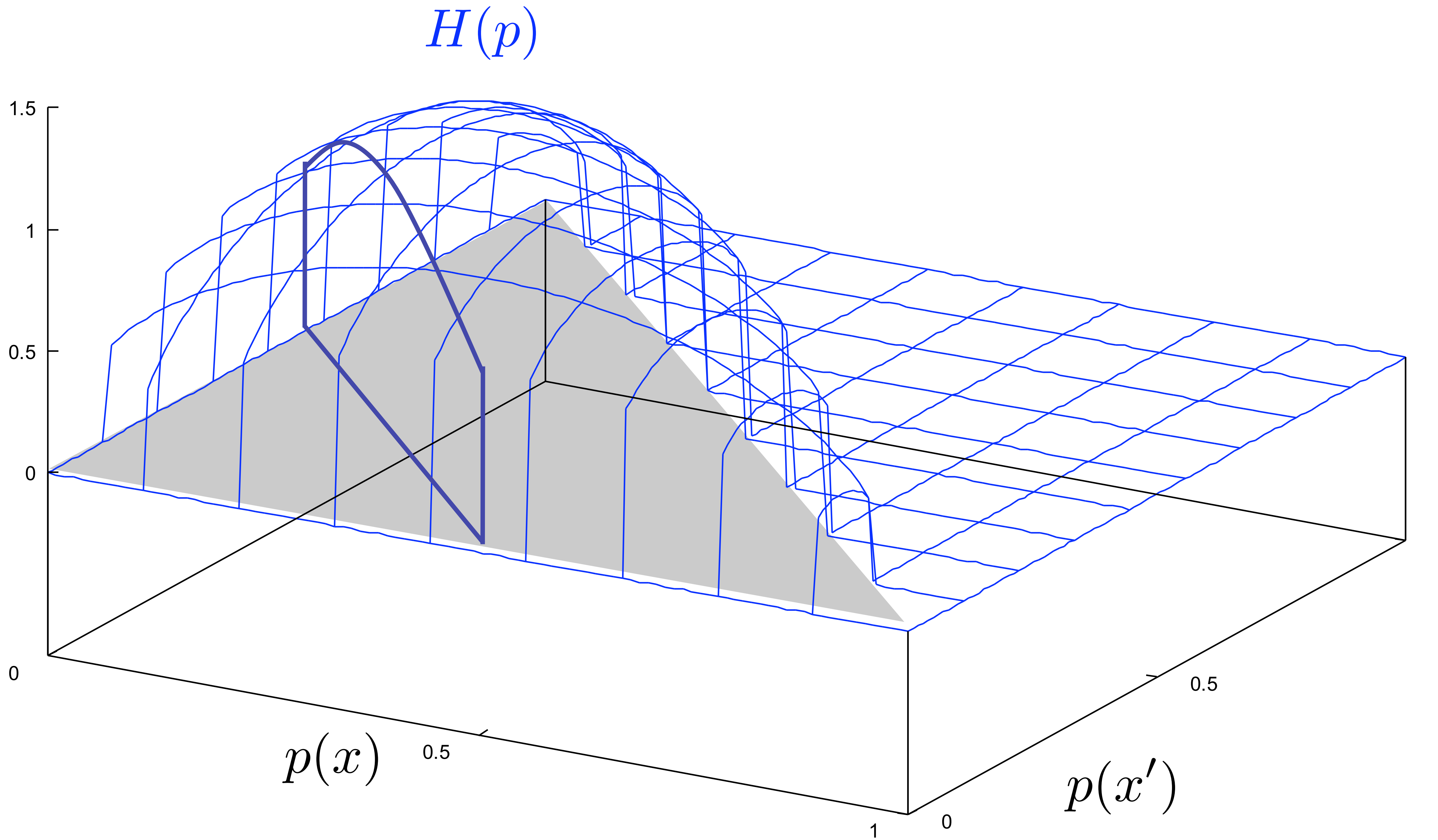
$$\max_p H(p) \equiv -\sum_y p(y)\log p(y) \equiv \mathbb{E}_{p(Y)}[-\log p(Y)]$$

$$\text{such that} \quad \sum_y p(y) = 1$$

$$\forall y, p(y) \geq 0$$

$$\forall j, \underbrace{\mathbb{E}_{p(Y)}[f_j(Y)]}_{\sum_y p(y) f_j(y)} = \underbrace{\alpha_j}_{\frac{1}{N}\sum_{i=1}^{N} f_j(y_i)}$$

Max Ent



Questions Worth Asking

- Does a solution always exist?
- What to do if it doesn't?
- How to find the solution?

The Maximum Entropy Problem

$$p^* = \arg\max_p H(p)$$

such that

$$\sum_{x \in \mathcal{X}} p(x) = 1,$$

$$(\forall x \in \mathcal{X}) \quad p(x) \geq 0$$

$$(\forall j \in \{1, 2, ..., d\}) \quad \sum_{x \in \mathcal{X}} p(x) f_j(x) = \frac{1}{N} \sum_{i=1}^{N} f_j(x_i)$$

The Maximum Entropy Problem

$$p^* = \arg\max_{p \in \mathcal{P}} H(p)$$

$$\mathcal{P} = \{p : \text{all empirical constraints satisfied}\}$$

The Maximum Entropy Problem

$$p^* = \arg\max_{p \in \mathcal{P}} H(p)$$

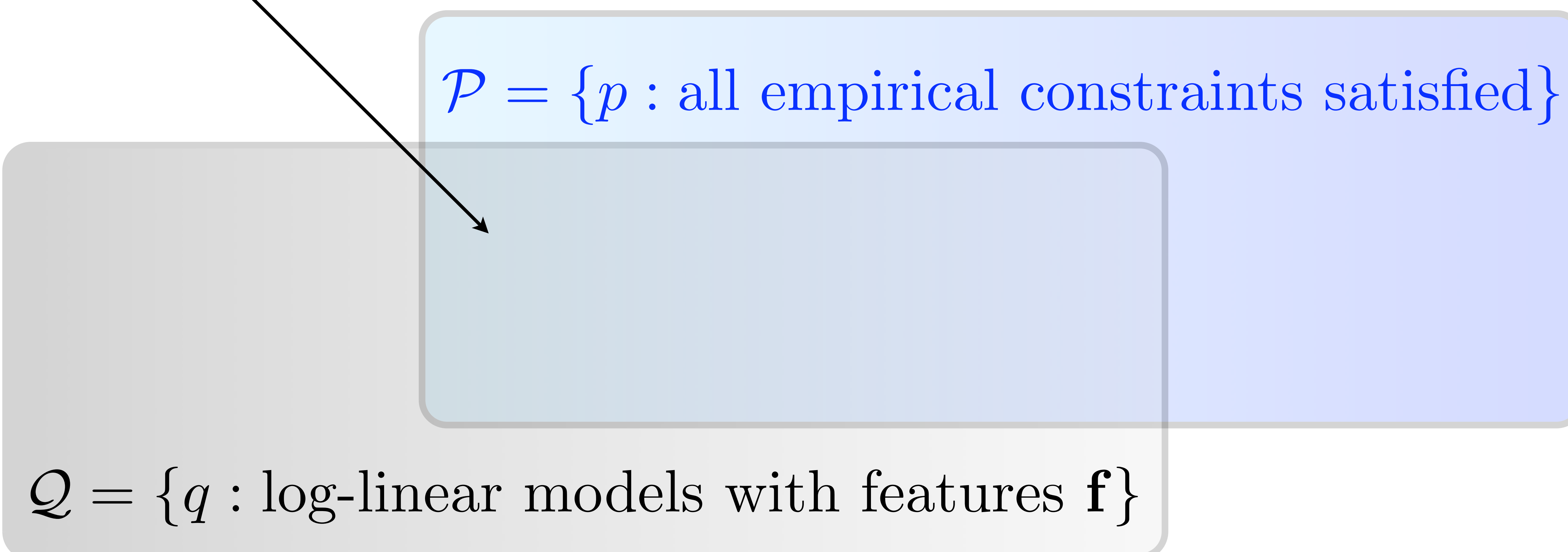
$\mathcal{P} = \{p : \text{all empirical constraints satisfied}\}$

$\mathcal{Q} = \{q : \text{log-linear models with features } \mathbf{f}\}$

Claim 1

If $\mathcal{P} \cap \mathcal{Q}$ is nonempty, it consists of only p^*, which is unique.

$$p^* = \arg\max_{p \in \mathcal{P}} H(p)$$



Claim 2

If $\mathcal{P} \cap \mathcal{Q}$ is nonempty, it consists of only $\hat{q}$, which is unique.

$$p^* = \arg\max_{p \in \mathcal{P}} H(p)$$

$$\mathcal{P} = \{p : \text{all empirical constraints satisfied}\}$$

$$\mathcal{Q} = \{q : \text{log-linear models with features } \mathbf{f}\}$$

$$\hat{q} = \arg\max_{q \in \mathcal{Q}} \prod_{i=1}^{N} q(x_i)$$

Result

Maximum entropy (with empirical constraints on **f**)

=

Maximum likelihood (over log-linear models on **f**)

The Magic of Lagrange Multipliers

constrained
$|\mathcal{X}|$ variables (p)
concave in p

*un*constrained
d variables (θ)
concave in θ

Mathematical Magic

For details: see handout on course page.

1. Use Lagrangean multipliers (one per constraint).
2. Take the gradient, set equal to zero.
3. Algebra …
4. Voilà! Maximum likelihood problem!

Additional Point

- If the constraints are empirical, then they are satisfiable (solution exists).

- So there is a **unique** solution to:

Max Ent = Log-linear MLE

Slightly More General View

- Instead of "maximize entropy," can describe this as "minimize divergence" to a **base** distribution p_0 (which happens so far to be uniform, but needn't have been).

$$D(p \| p_0) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{p_0(x)}$$

- Everything goes through pretty much the same.

Training the Weights

- Old answer: “iterative scaling”
  - Specialized method for this problem
  - Later versions: Generalized IS (Darroch and Ratliff, 1972) and Improved IS (Della Pietra, Della Pietra, and Lafferty, 1995)
- More recent answers:
  - It’s unconstrained, convex optimization!
  - See Malouf (2002) for comparison.
  - Or use stochastic gradient descent.
- A newer answer:
  - Dualize the problem and optimize “p” instead, using exponentiated gradient.

Training Log-Linear Models

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &= \sum_{i=1}^{n} \boldsymbol{\theta}^\top \mathbf{f}(x_i, y_i) - \log z(x_i, \boldsymbol{\theta}) \\
&= \boldsymbol{\theta}^\top \sum_{i=1}^{n} \mathbf{f}(x_i, y_i) - \sum_{i=1}^{n} \log z(x_i, \boldsymbol{\theta}) \\
\frac{\partial \mathcal{L}}{\partial \theta_j} &= \sum_{i=1}^{n} f_j(x_i, y_i) - \sum_{i=1}^{n} \frac{\partial \log z(x_i, \boldsymbol{\theta})}{\partial \theta_j} \\
&= \sum_{i=1}^{n} f_j(x_i, y_i) - \sum_{i=1}^{n} \mathbb{E}_{p(Y \mid x_i, \boldsymbol{\theta})}[f_j(x_i, Y)]
\end{aligned}$$

Maximum Mutual Information
(Or, the speech people had the same idea!)

$$I(X;Y) = \mathbb{E}_{p(X,Y)}\left[\log \frac{p(X,Y)}{p(X)\cdot p(Y)}\right]$$

Assume empirical dist. $$\approx \mathbb{E}_{\tilde{p}(X,Y)}\left[\log \frac{p(X,Y)}{p(X)\cdot p(Y)}\right]$$

$$= \mathbb{E}_{\tilde{p}(X,Y)}\left[\log \frac{p(Y\mid X)}{p(Y)}\right]$$

Assume p(Y) is uniform. $$\approx \mathbb{E}_{\tilde{p}(X,Y)} \log p(Y\mid X)$$

$$= \frac{1}{N}\sum_{i=1}^{N} \log p(y_i \mid x_i)$$

Also Related: Conditional Estimation

- Even if the model doesn't take the form $p(y \mid x)$, it's possible to nonetheless **optimize** $p(y \mid x)$.
    - Computationally tricky
    - Why do this? (Very important idea here!)
- Of course, when the model does not define $p(x, y)$ (because it's inherently a **conditional model**) we can't optimize "joint likelihood" $p(x, y)$!
    - If you really want to do this, redefine X and Y.

Avoiding Overfitting

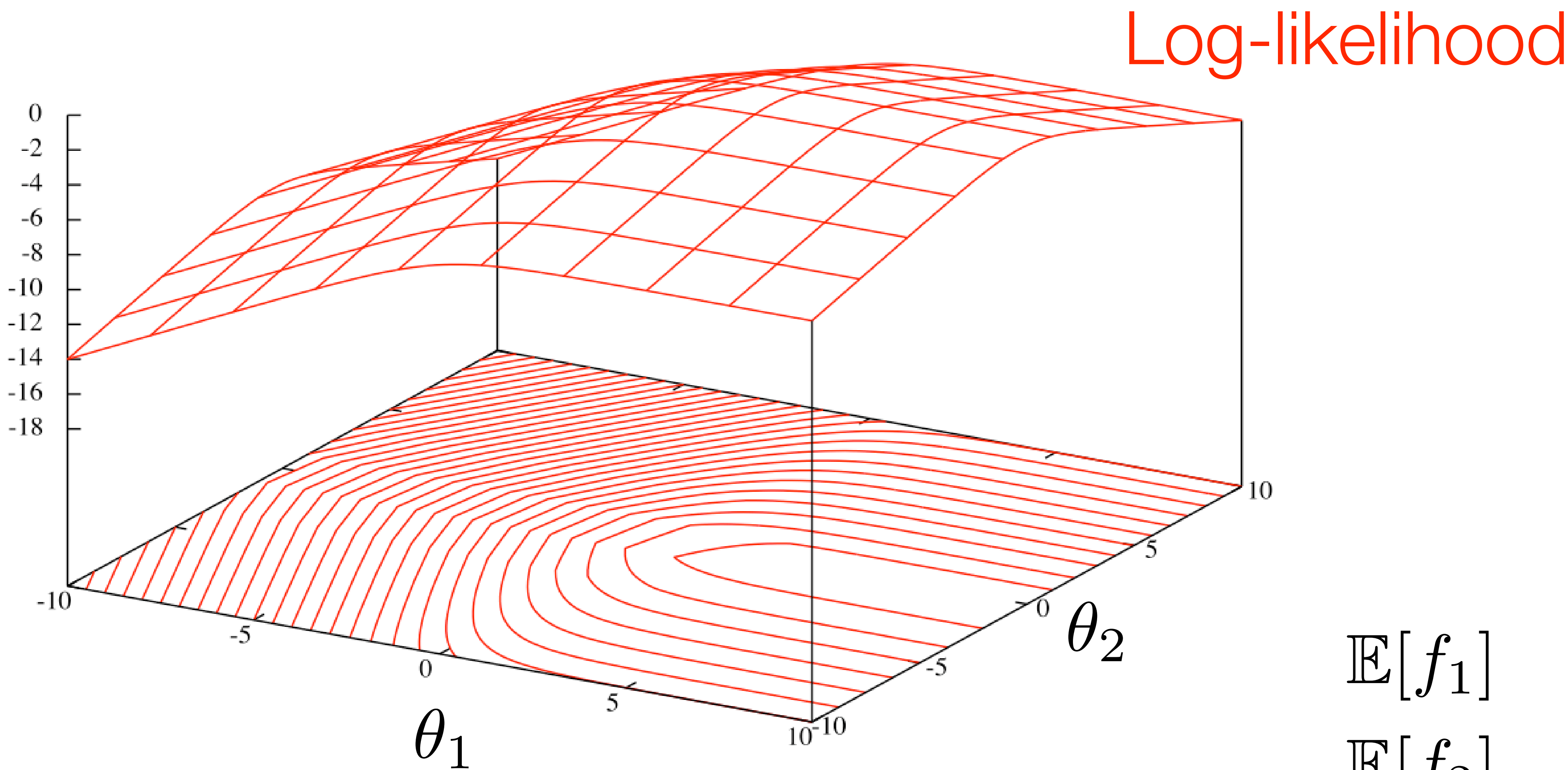
Example

- Suppose we're building a conditional model over the next *character* given the previous one (bigram character model)
- Two of our features:

  $f_{342}(c, c') = [c = \texttt{q} \text{ and } c' = \texttt{u}]$

  $f_{343}(c, c') = [c = \texttt{q} \text{ and } c' = \texttt{v}]$
- In training, `q` is *always* followed by `u`
- This happens 52 times
- To maximize likelihood: $p(\texttt{u} \mid \texttt{q})$ should go to 1, and $p(\texttt{v} \mid \texttt{q})$ should go to 0
- Is this what we really want?
- Consider the classic bigram model, which estimates $p(\texttt{u} \mid \texttt{q}) = 52/52$ and $p(\texttt{v} \mid \texttt{q}) = 0/52$

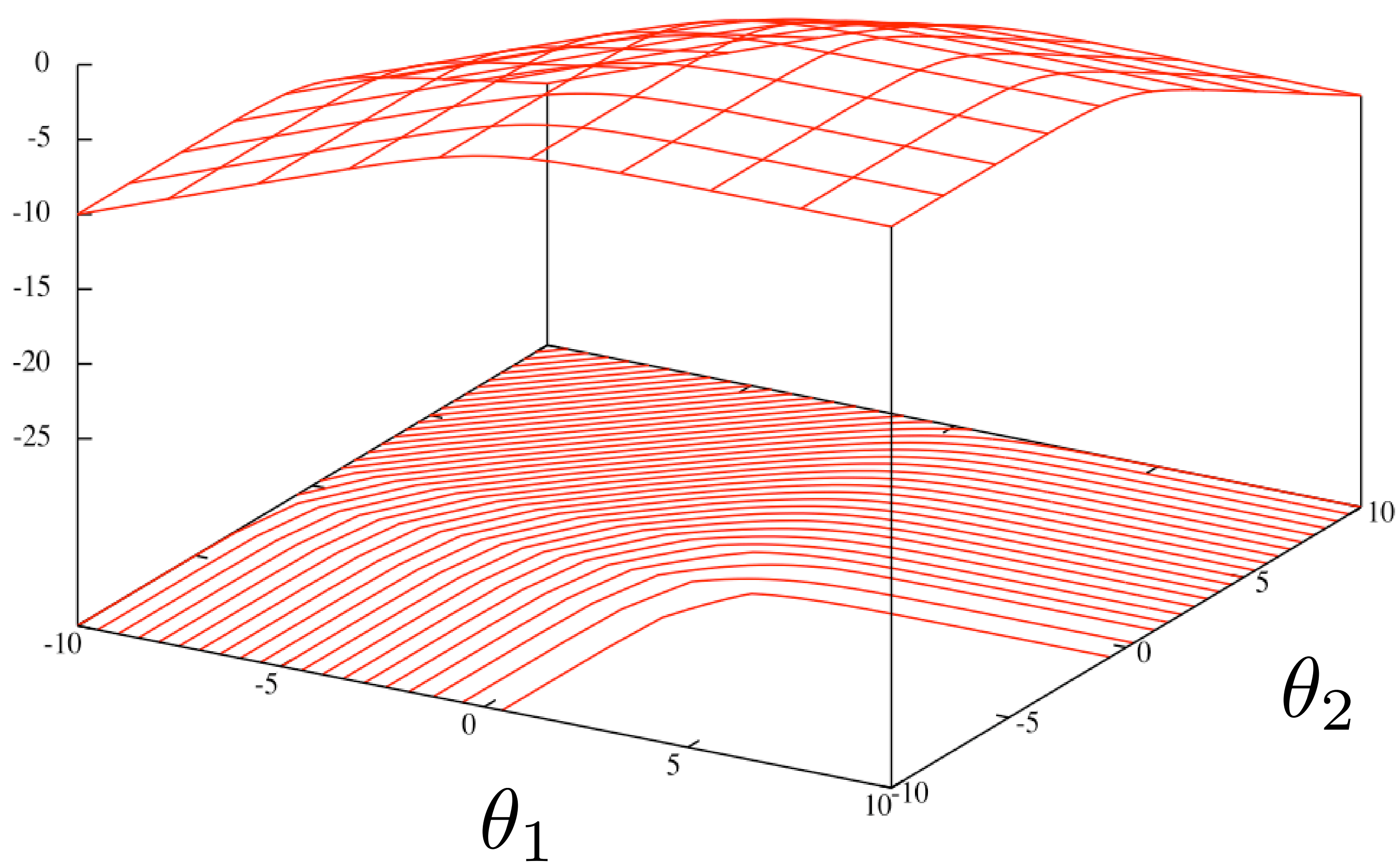
The Infinity Problem



$$\begin{aligned} \mathbb{E}[f_1] &= 1 \\ \mathbb{E}[f_2] &= 0.4 \end{aligned}$$

The Infinity Problem

Log-likelihood



$$
\begin{aligned}
\mathbb{E}[f_1] &= 1 \\
\mathbb{E}[f_2] &= 0.
\end{aligned}
$$

Overfitting in “Max Ent”

- We’re still doing MLE!
- Our models have the potential to be very expressive (more features)
  - More expressive power leads to greater potential for overfitting.
- Avoiding overfitting: **regularization** and **feature selection**

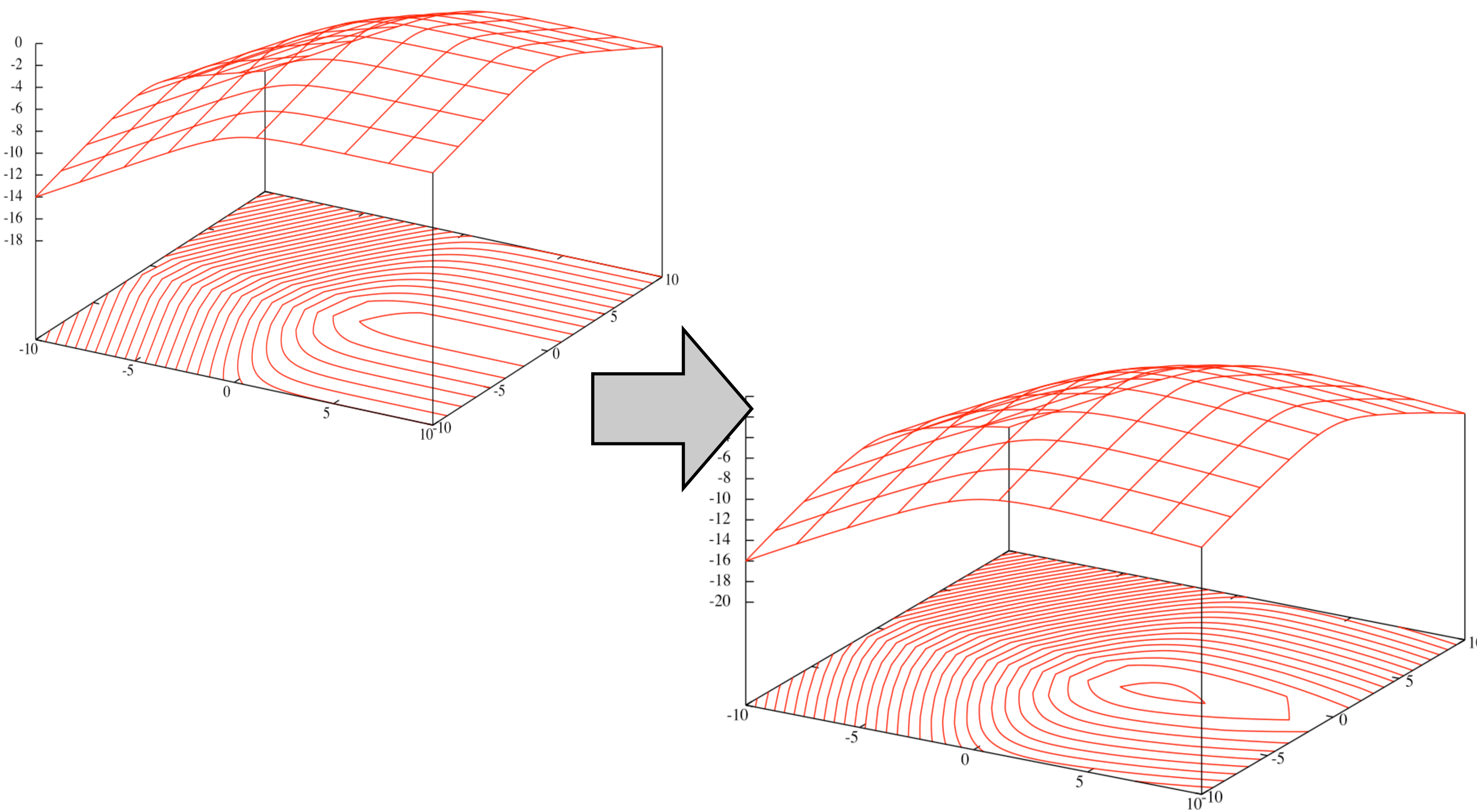
Regularization

- Idea borrowed from neural networks: penalize “extreme” models.
- L_2 regularization for log-linear models:

$$\arg\max_{\boldsymbol{\theta}} \left(\sum_{i=1}^{N} \boldsymbol{\theta}^\top \mathbf{f}(x_i, y_i) - \log z(x_i, \boldsymbol{\theta}) \right) - c \sum_{j=1}^{d} \theta_j^2$$

- Can also do L_1: $-c \sum_{j=1}^{d} |\theta_j|$

L_2 Regularization



A Probabilistic Interpretation

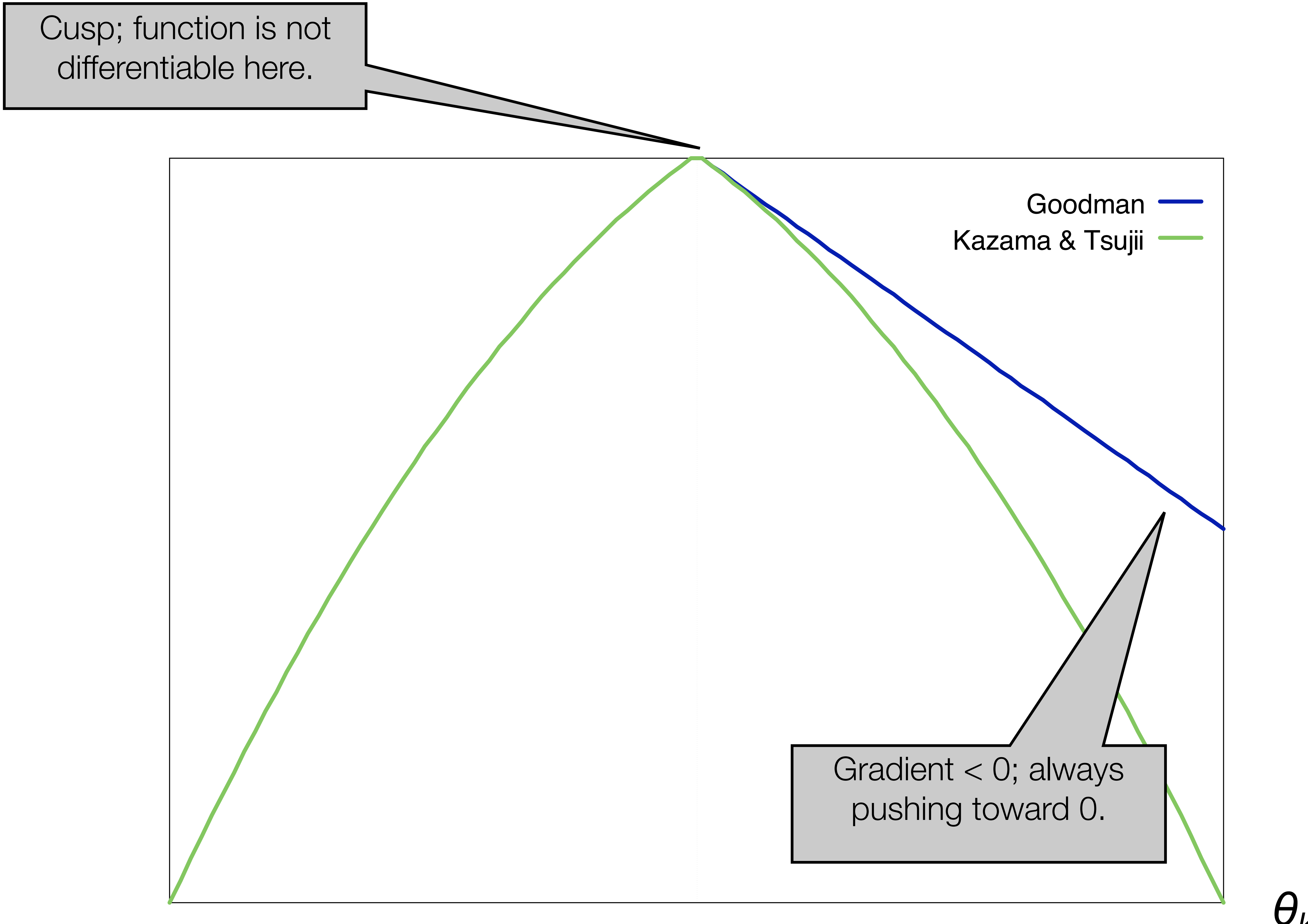
- Maximum likelihood est.: $$\max_{\text{model}} p(\text{data} \mid \text{model})$$
- Maximum *a posteriori* est.: $$\max_{\text{model}} p(\text{data} \mid \text{model}) \cdot p(\text{model})$$

$$\equiv \max_{\text{model}} \log p(\text{data} \mid \text{model}) + \log p(\text{model})$$

- So L_2 regularization is equivalent to MAP with a zero-mean Gaussian *prior* on each parameter. See Chen & Rosenfeld.

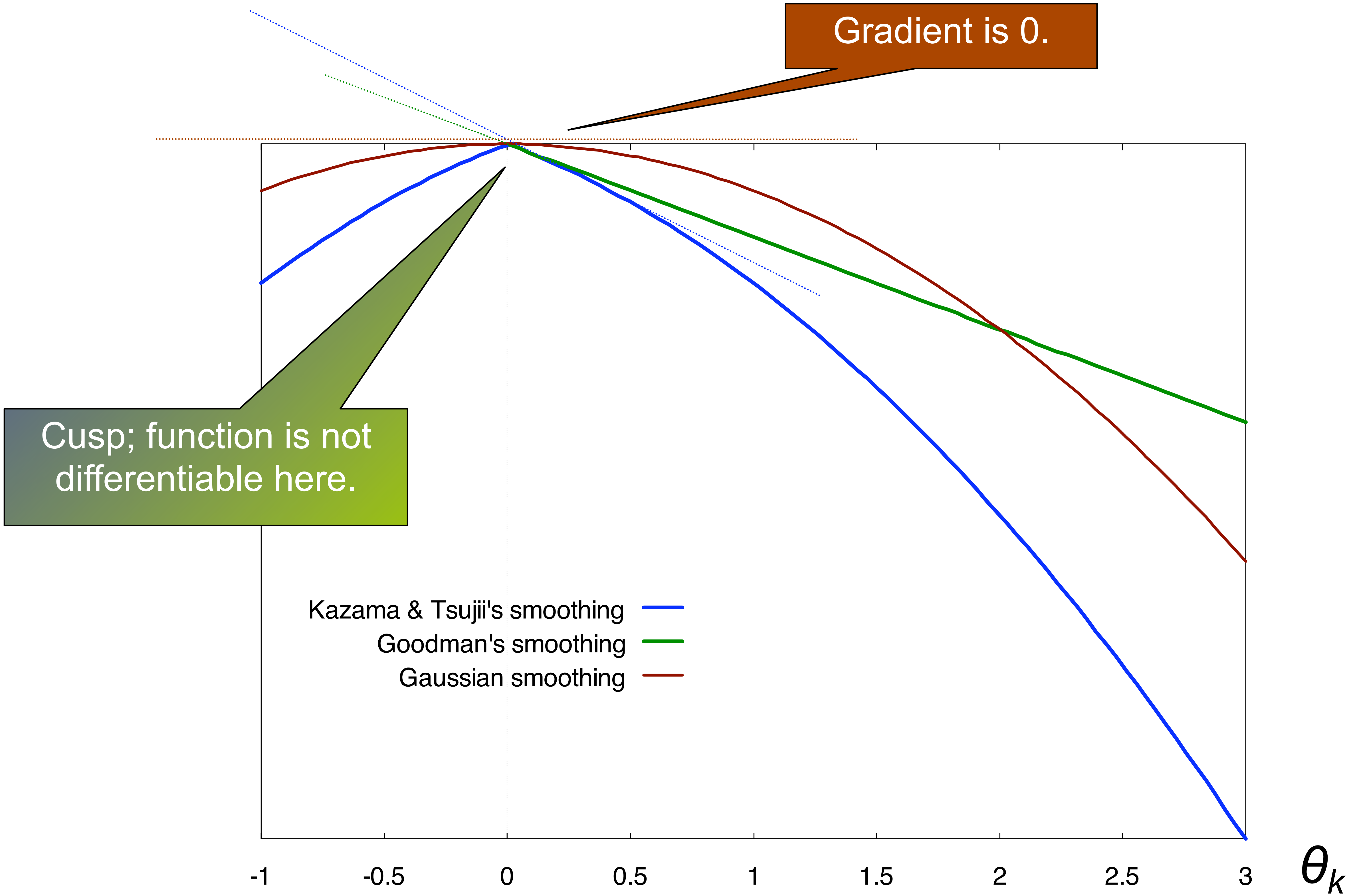
More on Regularization

- Goodman (2003): Laplacian prior corresponds to L_1 regularization. There's also an exponential prior.
- Related: Kazama & Tsuji'i (2003) and Khudanpur (2005): "relaxed" constraints.
- Added bonus for L_1 regularization: **sparsity**
- As you strengthen the prior, more of the global optimum's coordinates go to zero.



This shows the additive component resulting from a single parameter.

Sparsity



Gradient is 0.
Cusp; function is not differentiable here.
Kazama & Tsujii's smoothing
Goodman's smoothing
Gaussian smoothing
-1
-0.5
0
0.5
1
1.5
2
2.5
3
θ_k

On Feature Selection

- “Sparse” priors give you a kind of automatic feature selection
  - Drawback: have to start with all of them thrown in
- Ratnaparkhi (1996): count cutoff - include a feature iff it’s observed 5+ times in the training data
- Della Pietra et al. (1997): greedy algorithm

Della Pietra, Della Pietra, and Lafferty (1997): Feature Induction

1. Start with no active features (maximum entropy!).
2. Consider candidates:
   - "Atomic" features
   - Conjoined features (1 active AND 1 atomic)
3. Pick the candidate *g* with the greatest gain
   - Gain = upper bound on improvement to likelihood, for any value of *g*'s weight, assuming other weights are fixed.
   - Closed-form solution for gain if features are binary!
4. Add *g* to the model.
5. Retrain.
6. Go to 2.