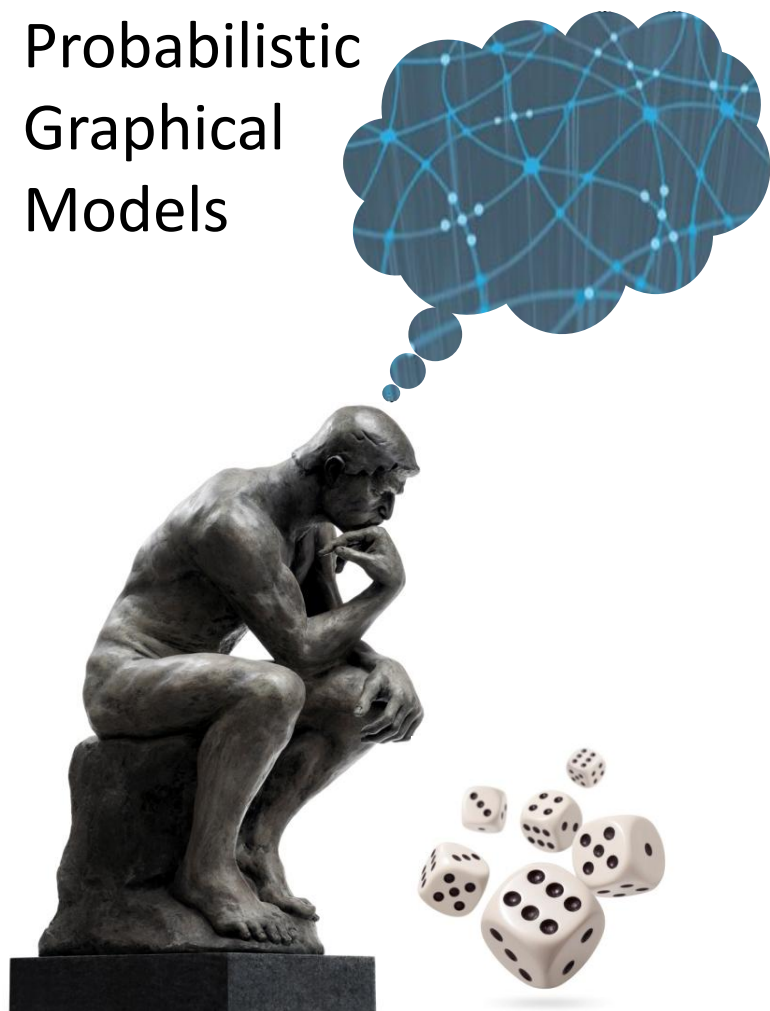


Probabilistic  
Graphical  
Models



Learning

---

Parameter Estimation

---

# Max Likelihood for Log-Linear Models

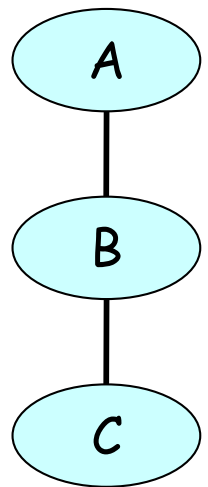
# Log-Likelihood for Markov Nets

$$P_{\theta}(a, b, c) = \frac{1}{Z} \phi_1(a, b) \cdot \phi_2(b, c)$$

$$\begin{aligned}\ell(\theta : \mathcal{D}) &= \sum_m (\ln \phi_1(a[m], b[m]) + \ln \phi_2(b[m], c[m]) - \ln Z(\theta)) \\ &= \sum_{a,b} M[a, b] \ln \phi_1(a, b) + \sum_{b,c} M[b, c] \ln \phi_2(b, c) - M \ln Z(\theta)\end{aligned}$$

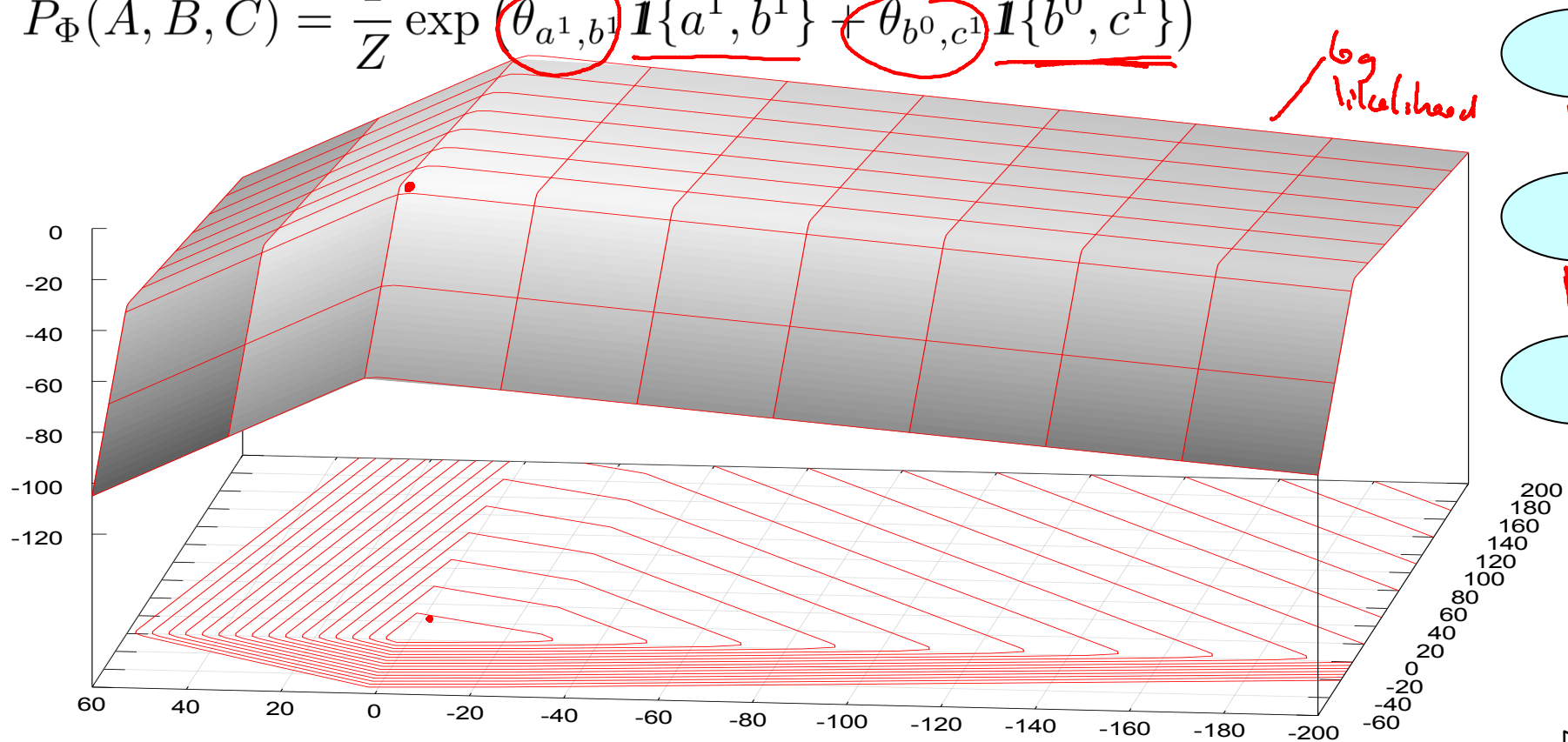
$$Z(\theta) = \sum_{a,b,c} \phi_1(a, b) \phi_2(b, c)$$

- Partition function couples the parameters
  - No decomposition of likelihood
  - No closed form solution



# Example: Log-Likelihood Function

$$P_{\Phi}(A, B, C) = \frac{1}{Z} \exp(\underbrace{\theta_{a^1, b^1}}_{\text{log likelihood}} \mathbf{1}\{a^1, b^1\} + \underbrace{\theta_{b^0, c^1}}_{\text{log likelihood}} \mathbf{1}\{b^0, c^1\})$$



# Log-Likelihood for Log-Linear Model

$$\underline{P(X_1, \dots, X_n : \boldsymbol{\theta})} = \frac{1}{Z(\boldsymbol{\theta})} \exp \left\{ \sum_{i=1}^k \theta_i f_i(\underline{D_i}) \right\}$$

*parameters* (pointing to  $\theta_i$ )  
*features* (pointing to  $f_i$ )  
*partition function* (pointing to  $Z(\boldsymbol{\theta})$ )

$$\ell(\boldsymbol{\theta} : \mathcal{D}) = \sum_i \theta_i \left( \sum_m f_i(\underline{x[m]}) \right) - M \ln Z(\boldsymbol{\theta})$$

*feature  $f_i$  applied to the  $m$ th instance* (pointing to  $f_i(\underline{x[m]})$ )

$$\ln Z(\boldsymbol{\theta}) = \ln \sum_{\underline{x}} \exp \left\{ \sum_i \theta_i f_i(\underline{x}) \right\}$$

*exponentially large space* (pointing to  $\sum_{\underline{x}}$ )  
*log-sum-exp* (pointing to the outer  $\ln$ )

# The Log-Partition Function

Theorem:  $\frac{\partial}{\partial \theta_i} \ln Z(\theta) = E_{\theta}[f_i]$

*vector of derivatives* *expectation of  $f_i$  relative to  $P_{\theta}$*

$\sum_x P_{\theta}(x) f_i(x)$

*matrix (Hessian)*  $\frac{\partial^2}{\partial \theta_i \partial \theta_j} \ln Z(\theta) = \text{Cov}_{\theta}[f_i; f_j]$

Proof:  $\frac{\partial}{\partial \theta_i} \ln Z(\theta) = \frac{1}{Z(\theta)} \sum_x \frac{\partial}{\partial \theta_i} \exp \left\{ \sum_j \theta_j f_j(x) \right\}$

$= \frac{1}{Z(\theta)} \sum_x f_i(x) \exp \left\{ \sum_j \theta_j f_j(x) \right\}$

$= \sum_x \left[ \frac{1}{Z(\theta)} \exp \left\{ \sum_j \theta_j f_j(x) \right\} \right] f_i(x) = \sum_x P_{\theta}(x) f_i(x)$

$\frac{\partial}{\partial \theta_i} \theta_j f_j = \begin{cases} 0 & i \neq j \\ f_i & i = j \end{cases}$

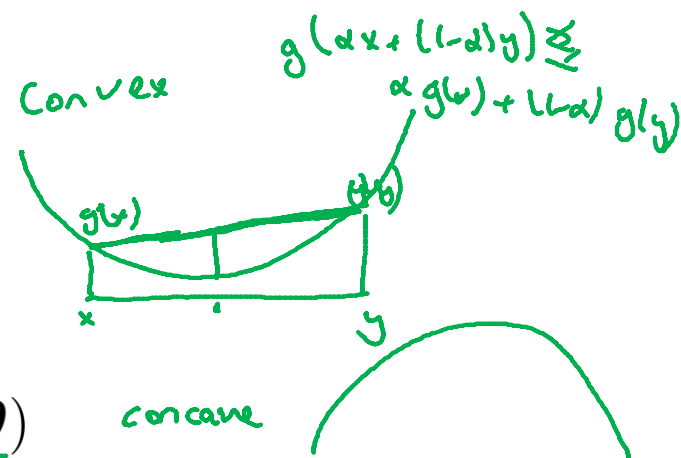
# The Log-Partition Function

Theorem:  $\frac{\partial}{\partial \theta_i} \ln Z(\theta) = E_{\theta}[f_i]$

Hessian:  $\frac{\partial^2}{\partial \theta_i \partial \theta_j} \ln Z(\theta) = \text{Cov}_{\theta}[f_i; f_j]$

$$\ell(\theta : \mathcal{D}) = \sum_i \theta_i \left( \sum_m f_i(\mathbf{x}[m]) \right) - \underline{M \ln Z(\theta)}$$

- Log likelihood function
  - No local optima
  - Easy to optimize



# Maximum Likelihood Estimation

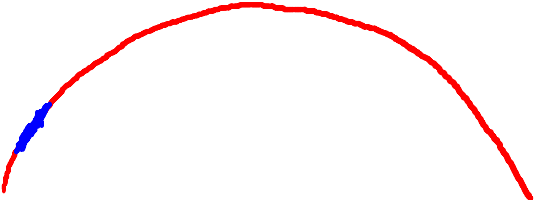
$$\frac{1}{M} \ell(\boldsymbol{\theta} : \mathcal{D}) = \sum_i \theta_i \left( \underbrace{\frac{1}{M} \sum_m f_i(x[m])}_{\text{empirical expectation of } f_i \text{ in } \mathcal{D}} \right) - \ln Z(\boldsymbol{\theta})$$

$$\frac{\partial}{\partial \theta_i} \frac{1}{M} \ell(\boldsymbol{\theta} : \mathcal{D}) = \underbrace{E_{\mathcal{D}}[f_i(X)]}_{\text{empirical expectation of } f_i \text{ in } \mathcal{D}} - \underbrace{E_{\boldsymbol{\theta}}[f_i]}_{\text{expectation of } f_i \text{ in } p_{\boldsymbol{\theta}}}$$

Theorem:  $\hat{\boldsymbol{\theta}}$  is the MLE if and only if

$$\forall i \quad \underbrace{E_{\mathcal{D}}[f_i(X)]}_{\text{expectation in } \mathcal{D}} = \underbrace{E_{\hat{\boldsymbol{\theta}}}[f_i]}_{\text{expectation relative to } \hat{\boldsymbol{\theta}}}$$

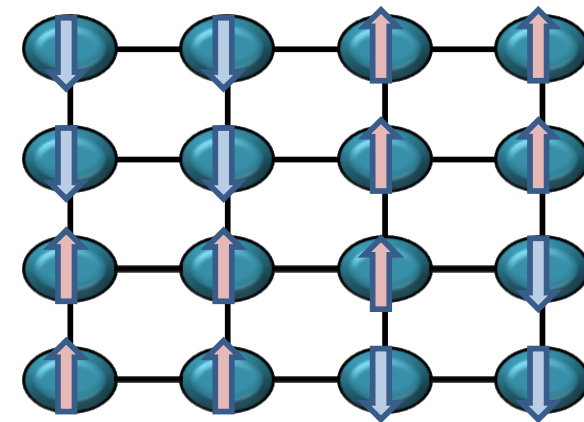
# Computation: Gradient Ascent

$$\frac{\partial}{\partial \theta_i} \frac{1}{M} \ell(\theta : \mathcal{D}) = \underbrace{E_{\mathcal{D}}[f_i(\mathbf{X})]}_{\text{in data}} - \underbrace{E_{\theta}[f_i]}_{\text{relative to current model}}$$


- Use gradient ascent:
  - typically L-BFGS - a quasi-Newton method
- For gradient, need expected feature counts:
  - in data
  - relative to current model
- Requires inference at each gradient step



# Example: Ising Model



$$x_i \in \{-1, +1\}$$

$$E(x_1, \dots, x_n) = - \sum_{i < j} w_{i,j} x_i x_j - \sum_i u_i x_i$$

$$\frac{\partial}{\partial \theta_i} \frac{1}{M} \ell(\theta : \mathcal{D}) = \mathbf{E}_{\mathcal{D}}[f_i(\mathbf{X})] - \mathbf{E}_{\theta}[f_i]$$

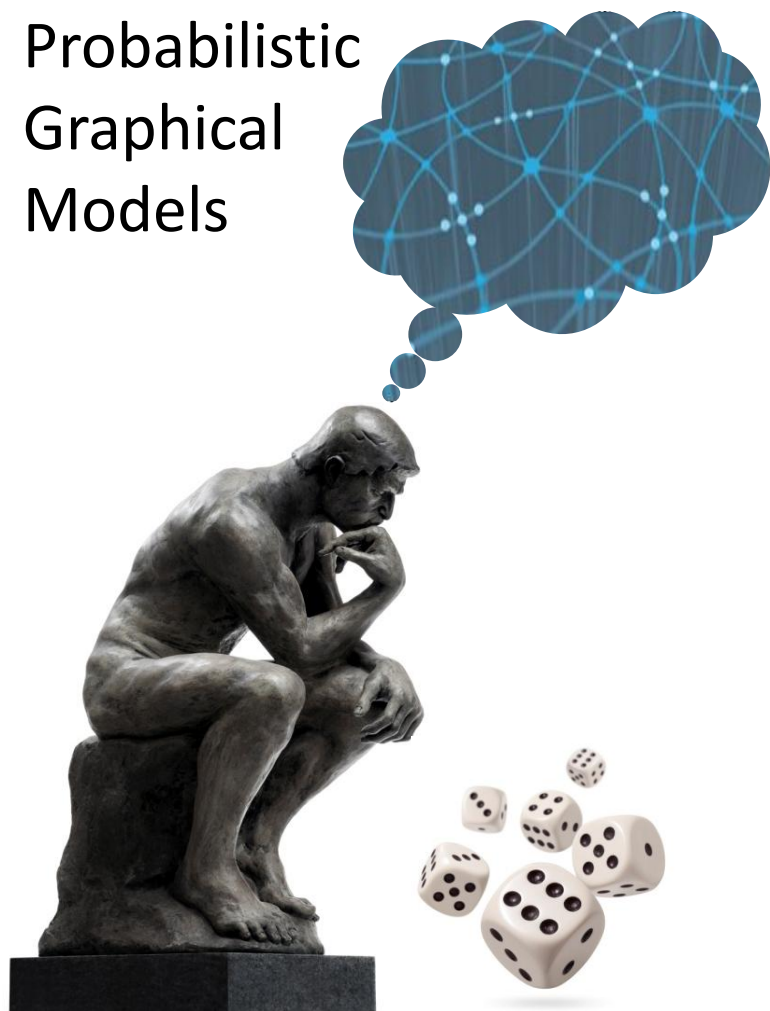
$$\frac{\partial}{\partial u_i} = \frac{1}{M} \sum_m \underbrace{x_i[m]}_{+1} - (P_{\theta}(X_i = 1) - P_{\theta}(X_i = -1))$$

$$\frac{\partial}{\partial w_{ij}} = \frac{1}{M} \sum_m \underbrace{x_i[m] x_j[m]}_{+1} - \left( \frac{P_{\theta}(X_i = 1, X_j = 1) + P_{\theta}(X_i = -1, X_j = -1)}{-P_{\theta}(X_i = 1, X_j = -1) - P_{\theta}(X_i = -1, X_j = -1)} \right)$$

# Summary

- Partition function couples parameters in likelihood
- No closed form solution, but convex optimization
  - Solved using gradient ascent (usually L-BFGS) *global opt.*
- Gradient computation requires inference at each gradient step to compute expected feature counts
- Features are always within clusters in cluster-graph or clique tree due to family preservation
  - One calibration suffices for all feature expectations

Probabilistic  
Graphical  
Models



Learning

---

Parameter Estimation

---

# Max Likelihood for CRFs

# Estimation for CRFs

$$P_{\theta}(\mathbf{Y} | \mathbf{x}) = \frac{1}{Z_{\mathbf{x}}(\theta)} \tilde{P}_{\theta}(\mathbf{x}, \mathbf{Y})$$

$$Z_{\mathbf{x}}(\theta) = \sum_{\mathbf{Y}} \tilde{P}_{\theta}(\mathbf{x}, \mathbf{Y})$$

log conditional likelihood

$$\mathcal{D} = \{(\mathbf{x}[m], \mathbf{y}[m])\}_{m=1}^M$$

$$\ell_{\mathbf{Y}|\mathbf{X}}(\theta : \mathcal{D}) = \sum_{m=1}^M \ln P_{\theta}(\mathbf{y}[m] | \mathbf{x}[m], \theta)$$

$$\ell_{\mathbf{Y}|\mathbf{X}}(\theta : (\mathbf{x}[m], \mathbf{y}[m])) = \left( \sum_i \theta_i f_i(\mathbf{x}[m], \mathbf{y}[m]) \right) - \ln Z_{\mathbf{x}[m]}(\theta)$$

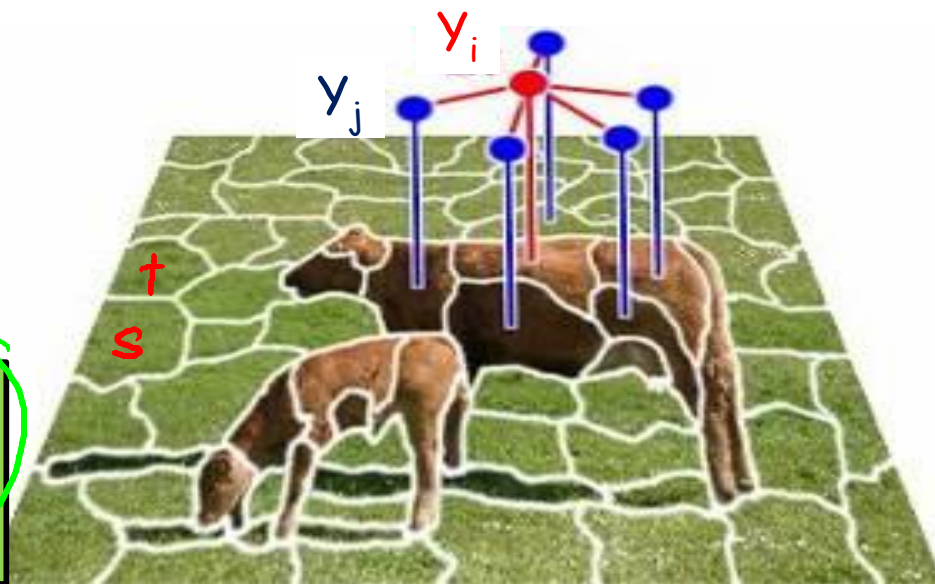
$$\frac{\partial}{\partial \theta_i} \frac{1}{M} \ell_{\mathbf{Y}|\mathbf{X}}(\theta : \mathcal{D}) = \frac{1}{M} \sum_{m=1}^M (f_i(\mathbf{x}[m], \mathbf{y}[m]) - E_{\theta}[f_i(\mathbf{x}[m], \mathbf{Y})])$$

# Example

$$\underline{f_1(Y_s, X_s) = \underline{1(Y_s = g)} \times G_s}$$

$$\underline{f_2(Y_s, Y_t) = \underline{1(Y_s = Y_t)}}$$

average intensity of  
green channel for  
pixels in superpixel  $s$



$$\frac{\partial}{\partial \theta_i} \ell_{\mathbf{Y}|\mathbf{X}}(\boldsymbol{\theta} : (\mathbf{x}[m], \mathbf{y}[m])) = (f_i(\mathbf{x}[m], \mathbf{y}[m]) - \mathbf{E}_{\boldsymbol{\theta}}[f_i(\mathbf{x}[m], \mathbf{Y})])$$

$$\frac{\partial}{\partial \theta_1} = \sum_s \underline{1\{y_s[m] = g\} G_s[m]} - \sum_s \underline{P_{\boldsymbol{\theta}}(Y_s = g | \mathbf{x}[m]) G_s[m]}$$

$$\frac{\partial}{\partial \theta_2} = \sum_{(s,t) \in \mathcal{N}} \underline{1\{y_s[m] = y_t[m]\}} - \sum_{(s,t) \in \mathcal{N}} \underline{P_{\boldsymbol{\theta}}(Y_s = Y_t | \mathbf{x}[m])}$$

# Computation

**MRF**  $\frac{\partial}{\partial \theta_i} \frac{1}{M} \ell(\theta : \mathcal{D}) = \mathbf{E}_{\mathcal{D}}[f_i(\mathbf{X})] - \underline{\mathbf{E}_{\theta}[f_i]}$

- Requires inference at each gradient step

---

**CRF**  $\frac{\partial}{\partial \theta_i} \frac{1}{M} \ell_{\mathbf{Y}|\mathbf{X}}(\theta : \mathcal{D}) = \frac{1}{M} \sum_{m=1}^M (f_i(\underline{\mathbf{x}[m]}, \mathbf{y}[m]) - \mathbf{E}_{\theta}[f_i(\underline{\mathbf{x}[m]}, \mathbf{Y})])$

- Requires inference for each  $\mathbf{x}[m]$  at each gradient step  
 $M = \# \text{ training instances}$

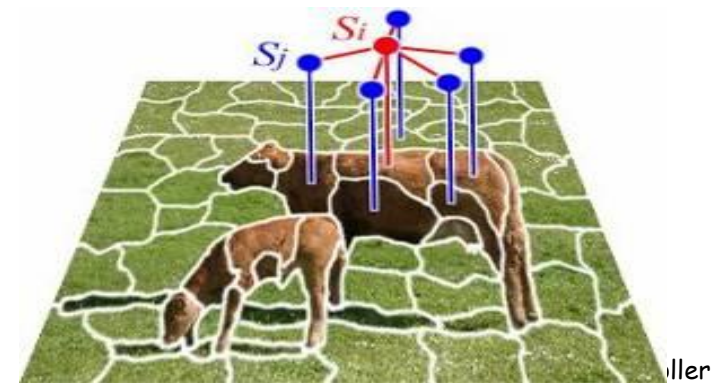
# However...

- For inference of  $P(Y | x)$ , we need to compute distribution only over  $Y$
- If we learn an MRF, need to compute  $P(Y, X)$ , which may be much more complex

$$f_1(Y_s, X_s) = \mathbf{1}(Y_s = g) \times G_s$$

$$f_2(Y_s, Y_t) = \mathbf{1}(Y_s = Y_t)$$

average intensity of  
green channel for  
pixels in superpixel  $i$

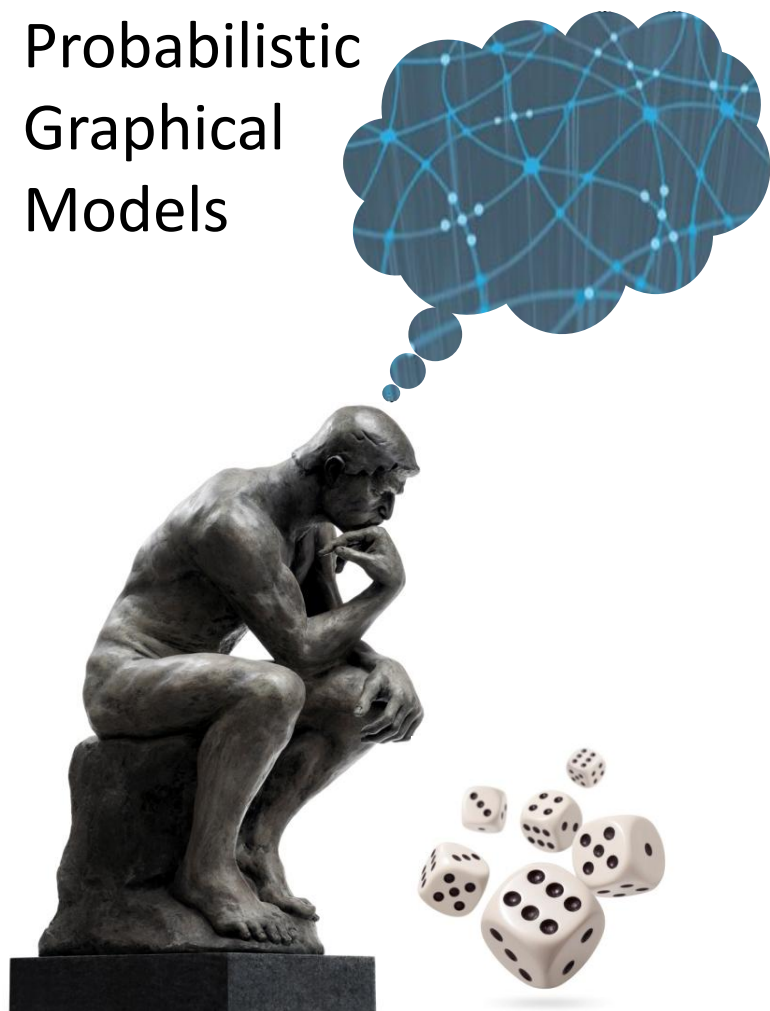


# Summary

- CRF learning very similar to MRF learning
  - Likelihood function is concave
  - Optimized using gradient ascent (usually L-BFGS)
- Gradient computation requires inference: one per gradient step, data instance
  - c.f., once per gradient step for MRFs
- But conditional model is often much simpler, so inference cost for CRF, MRF is not the same



Probabilistic  
Graphical  
Models



Learning

---

Parameter Estimation

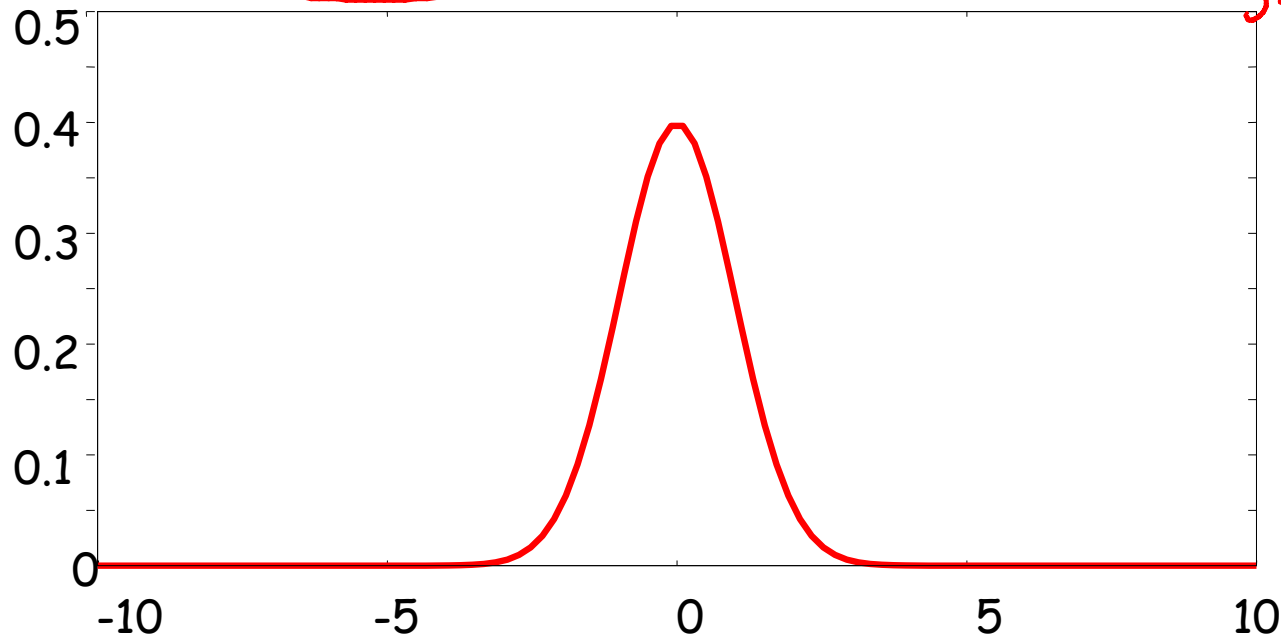
---

MAP  
Estimation for  
MRFs, CRFs

# Gaussian Parameter Prior

$$P(\theta : \sigma^2) = \prod_{i=1}^k \frac{1}{\sqrt{2\pi}\sigma} \exp \left\{ -\frac{\theta_i^2}{2\sigma^2} \right\}$$

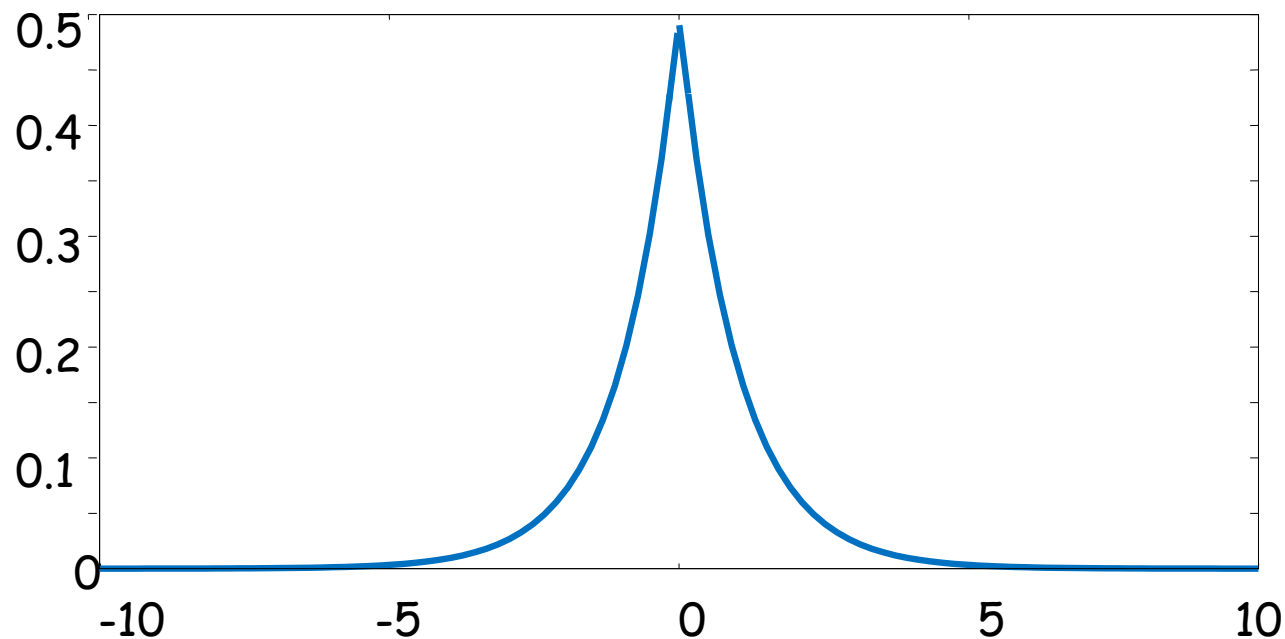
*σ mean univariate Gaussian*  
*hyperparameter*



# Laplacian Parameter Prior

$$P(\theta : \beta) = \prod_{i=1}^k \frac{1}{2\beta} \exp \left\{ -\frac{|\theta_i|}{\beta} \right\}$$

*Annotations:* A blue circle highlights  $\beta$  in the notation  $P(\theta : \beta)$ . A blue arrow points from the handwritten word "hyperparameter" to  $\beta$ . A red arrow points from the same word to the denominator  $\beta$  in the exponent of the product term.



# MAP Estimation & Regularization

$$P(\boldsymbol{\theta} : \sigma^2) = \prod_{i=1}^k \frac{1}{\sqrt{2\pi}\sigma} \exp \left\{ -\frac{\theta_i^2}{2\sigma^2} \right\}$$

$$P(\boldsymbol{\theta} : \beta) = \prod_{i=1}^k \frac{1}{2\beta} \exp \left\{ -\frac{|\theta_i|}{\beta} \right\}$$

$$\text{argmax}_{\boldsymbol{\theta}} P(\mathcal{D}, \boldsymbol{\theta}) = \text{argmax}_{\boldsymbol{\theta}} P(\mathcal{D} | \boldsymbol{\theta}) P(\boldsymbol{\theta})$$

likelihood prior  
← log likelihood ← log prior

$$= \text{argmax}_{\boldsymbol{\theta}} (\ell(\boldsymbol{\theta} : \mathcal{D}) + \log P(\boldsymbol{\theta}))$$

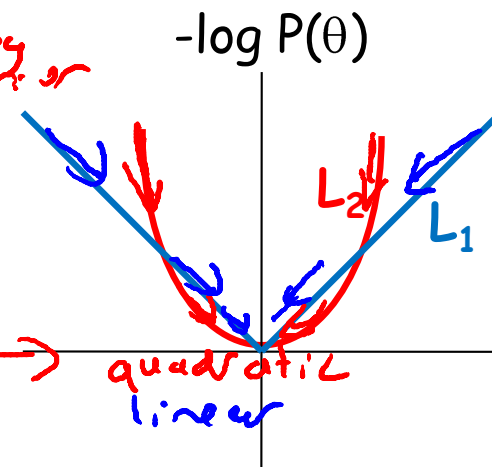
many  $\theta_i \neq 0$

dense

sparse

→  $L_2$ -regularization

→  $L_1$ -regularization



# Summary

- In undirected models, parameter coupling prevents efficient Bayesian estimation
- However, can still use parameter priors to avoid overfitting of MLE *MAP*
- Typical priors are  $L_1, L_2$ 
  - Drive parameters toward zero
- $L_1$  provably induces sparse solutions
  - Performs feature selection / structure learning