# Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data

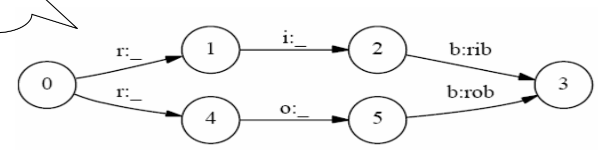
John Lafferty, Andrew McCallum, Fernando Pereira

Presenter: Yejin Choi

## Label Bias Problem

arcs: observations (X)

nodes: outputs (Y)



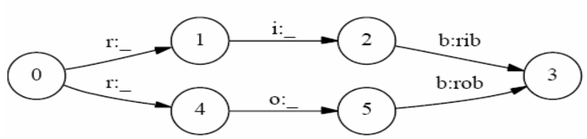

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| S1 | Herbivores | ( - ) | like | ( r ) | merino | ( i ) | sheep | ( b ) |
| | NNS | ( 0 ) | IN | ( 1 ) | JJ | ( 2 ) | NN | ( 3 ) |
| S2 | Carnivores | ( - ) | like | ( r ) | eating | ( o ) | sheep | ( b ) |
| | NNS | ( 0 ) | VB | ( 4 ) | VBG | ( 5 ) | NN | ( 3 ) |

- Suppose [NNS => VB] transition more frequent than [NNS => IN]
- Suppose from [VB], only [VB => VBG] transition is possible

➔ now, What is $P(Y_i = VBG \mid Y_{i-1} = VB, X_i = merino)$ ???

- Recall MEMM models $P(Y_i \mid Y_{i-1}, X_i)$

## Label Bias Problem




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Herbivores | ( h ) | like | ( r ) | merino | ( i ) | sheep | ( b ) |
| NNS | ( 0 ) | IN | ( 1 ) | JJ | ( 2 ) | NN | ( 3 ) |
| Carnivores | ( c ) | like | ( r ) | eating | ( o ) | sheep | ( b ) |
| NNS | ( 0 ) | VB | ( 4 ) | VBG | ( 5 ) | NN | ( 3 ) |

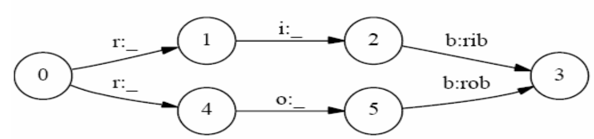
**So, how do we fix this nonsense ?**

$P(Y_i = VBG \mid Y_{i-1} = VB, X_i = ???) = 1$ for any $X_i$

➔ Do not normalize on each node!
Instead, normalize over the entire sequence.
This motivates the "global normalization" scheme of CRFs.

Other approaches: Cohen and Carvalho (2005), Sutton and McCallum (2005)

## Label Bias Problem




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Herbivores | ( h ) | like | ( r ) | merino | ( i ) | sheep | ( b ) |
| NNS | ( 0 ) | IN | ( 1 ) | JJ | ( 2 ) | NN | ( 3 ) |
| Carnivores | ( c ) | like | ( r ) | eating | ( o ) | sheep | ( b ) |
| NNS | ( 0 ) | VB | ( 4 ) | VBG | ( 5 ) | NN | ( 3 ) |

**Wait, what about HMMs?**

- Recall HMM models $P(X_i \mid Y_i)$ and $P(Y_i \mid Y_{i-1})$

(so that $P(X_i \mid Y_i, Y_{i-1}) \bullet P(Y_i \mid Y_{i-1}) = P(X_i, Y_i \mid Y_{i-1})$)

then we can get $P(merino \mid VBG) = 0$ !

# Label Bias v.s. Observation Bias

| Label Bias (Bottou, 1991; Lafferty et. al. 2001) | Observation Bias (Klein and Manning 2002) |
|---|---|
| Previous state explains current state so well (hence ignoring observation) | Observation explains current state so well (hence ignoring state transition) |

e.g.

| | All the indexes above | HMM | MEMM |
|---|---|---|---|
| Correct | PDT DT NNS VBD | -0.0 | -1.3 |
| Incorrect | DT DT NNS VBD | -5.4 | -0.3 |

➔ Both are to do with local conditional normalization.

# So, Let's normalize globally! (But, how?)

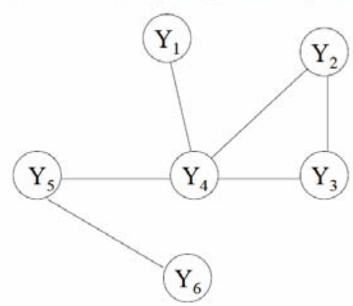
- What we need is the global joint distribution. i.e., **p(y | x)**
  - where $y = (y_1, \ldots, y_n)$ and $x = (x_1, \ldots, x_n)$
  - we do not want distributions on individual node. i.e., $\mathbf{p(y_i \mid x_i)}$
  - Instead, we want non-probabilistic ***potential*** function. i.e., $\boldsymbol{\phi(y_i, x_i)}$
    - $p(y \mid x) \approx g(\phi(y_1, x_1), \ldots, \phi(y_n, x_n))$ ????????
- Problem with directed graphs (like Bayesian Network)
  - a probability distribution should be given for each node
  - then, the joint probability $\mathbf{p(y) = \prod_i p(y_i \mid parents(y_i))}$
    ➔ btw, what is **parents($y_i$)** for MEMM ?
- Markov Random Field ! ( = Markov Network, Random Field )

# Markov Random Field

*Just for now, forget about conditioning on X…*

Let G = (Y, E) be a graph where each vertex $Y_v$ is a random variable
Suppose $P(Y_v \mid \text{all other } Y) = P(Y_v \mid \text{neighbors}(Y_v))$ then Y is a random field

Example:




- $P(Y_5 \mid \text{all other } Y) = P(Y_5 \mid Y_4, Y_6)$

- But, how do we compute the global joint distribution P (Y) out of this ?
- Besides, We don't want to compute $P(Y_i \mid \text{neighbors}(Y_i))$ !!!

# Hammersley-Clifford theorem (1971)

$$P(\vec{Y}) = \frac{1}{Z} \prod_{\text{clique } C} \phi(\vec{Y}_C)$$

where $Z = \sum_Y \prod_c \phi(Y_c)$

1. Given MRF G=(Y,E) such that $P(Y_i \mid Y \setminus Y_i) = P(Y_i \mid nbr(Y_i))$
2. Given $\phi(Y_c)$ for $\forall$ clique C in G, such that $\phi(Y_c)$ **>= 0**
   - cliques may overlap.
   - cliques may not be maximal.

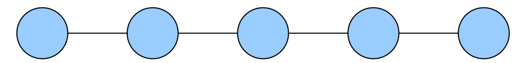
➔ this implies we don't need to compute $P(Y_i \mid nbr(Y_i))$ to get P(Y) !

## Definition of CRFs

**Definition.** *Let $G = (V, E)$ be a graph such that $\mathbf{Y} = (\mathbf{Y}_v)_{v \in V}$, so that $\mathbf{Y}$ is indexed by the vertices of $G$. Then $(\mathbf{X}, \mathbf{Y})$ is a conditional random field in case, when conditioned on $\mathbf{X}$, the random variables $\mathbf{Y}_v$ obey the Markov property with respect to the graph: $p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_w, w \neq v) = p(\mathbf{Y}_v \mid \mathbf{X}, \mathbf{Y}_w, w \sim v)$, where $w \sim v$ means that $w$ and $v$ are neighbors in $G$.*

- Almost identical to the definition of MRFs, except CRFs condition on X. and notice that X is not part of V.
- Hammersley-Clifford theorem can be extended to the conditional cases (on X) as well!

## P(y | x) for linear-chain CRFs



Let $\phi(y_i, x) = \exp \sum_k \lambda_k f_k(y_i, x)$

$\phi(y_i, y_{i+1}, x) = \exp \sum_k \lambda_k f_k(y_i, y_{i+1}, x)$

Can use any other non negative function, but exp function will give us a nice continuous convex objective function later.

Then,

$$p(y \mid x) = \frac{1}{Z(x)} \prod_i \left( \phi(y_i, x)\, \phi(y_i, y_{i+1}, x) \right)$$
$$= \frac{1}{Z(x)} \exp\left( \sum_{i,k} \lambda_k f_k(y_i, x) + \sum_{i,k} \lambda_k f_k(y_i, y_{i+1}, x) \right)$$
$$= \frac{1}{Z(x)} \exp\left( \lambda \bullet \mathbf{F}(y, x) \right)$$

where $Z(x) = \sum_y \exp\left( \lambda \bullet \mathbf{F}(y, x) \right)$

$$p_\theta(y \mid x) = \frac{1}{Z(x)} \exp\left( \sum_{e \in E, k} \lambda_k f_k(e, y|_e, x) + \sum_{v \in V, k} \mu_k g_k(v, y|_v, x) \right)$$

## Objective function for training

Given the training data $D = \{x^{(j)}, y^{(j)}\}^N_{j=1}$

and $p(y \mid x) = \frac{1}{Z(x)} \exp \lambda \bullet \mathbf{F}(y, x)$

Objective function :

conditional likelihood $L(\lambda) = L(\lambda \mid D) = P(D \mid \lambda) = \prod_j p(y^{(j)} \mid x^{(j)})$

equiv. to optimize $l(\lambda) = \log L(\lambda) = \sum_j \log p(y^{(j)} \mid x^{(j)})$

$$l(\lambda) = \sum_j \log p(y^{(j)} \mid x^{(j)}) = \sum_j \log \frac{1}{Z(x)} \exp \lambda \bullet \mathbf{F}(y^{(j)}, x^{(j)})$$
$$= \sum_j \lambda \bullet \mathbf{F}(y^{(j)}, x^{(j)}) - \log Z(x^{(j)})$$
$$= \sum_j \left( \lambda \bullet \mathbf{F}(y^{(j)}, x^{(j)}) - \log \sum_y \exp \lambda \bullet \mathbf{F}(y^{(j)}, x^{(j)}) \right)$$

## Objective function for training

Given the training data $D = \{x^{(j)}, y^{(j)}\}^N_{j=1}$

and $p(y \mid x) = \frac{1}{Z(x)} \exp \lambda \bullet \mathbf{F}(y, x)$

Objective function :

conditional likelihood $L(\lambda) = L(\lambda \mid D) = P(D \mid \lambda) = \prod_j p(y^{(j)} \mid x^{(j)})$

equiv. to optimize $l(\lambda) = \log L(\lambda) = \sum_j \log p(y^{(j)} \mid x^{(j)})$

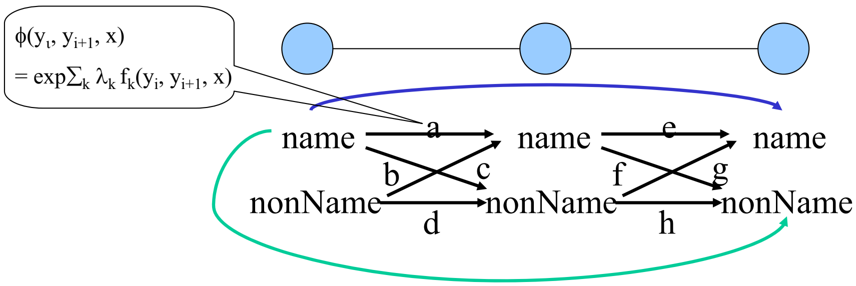
$$l(\lambda) = \sum_j \log p(y^{(j)} \mid x^{(j)}) = \sum_j \log \frac{1}{Z(x)} \exp \lambda \bullet \mathbf{F}(y^{(j)}, x^{(j)})$$
$$= \sum_j \lambda \bullet \mathbf{F}(y^{(j)}, x^{(j)}) - \log Z(x^{(j)})$$
$$= \sum_j \left( \lambda \bullet \mathbf{F}(y^{(j)}, x^{(j)}) - \log \sum_y \exp \lambda \bullet \mathbf{F}(y^{(j)}, x^{(j)}) \right)$$

nasty!

This diagram is from William Cohen's slides.

## Computing Z(x)! … for linear-chain CRFs

$$l(\lambda) = \sum_j \log p(y^{(j)} | x^{(j)}) = \sum_j \left( \lambda \bullet \mathbf{F}(y^{(j)}, x^{(j)}) - \log \sum_y \exp \lambda \bullet \mathbf{F}(y, x^{(j)}) \right)$$




$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ \dots & \dots \end{bmatrix}$$

## Parameter Estimation for CRFs

➔ how to compute $\text{argmax}_\lambda\, l(\lambda)$ ?

- Iterative scaling algorithms
  - Generalized iterative scaling (GIS)
  - Improved iterative scaling (IIS)
  - easy to implement
  - both really slow to converge
- Gradient descent methods
  - CG : conjugate gradient
  - L-BFGS : limited memory Newton's method
  - much harder to implement, but lots of code available

    ➔ you only need to provide $l(\lambda)$ and $l`(\lambda)$
  - much faster to converge

## Parameter Estimation for CRFs

- Iterative scaling algorithms
  - Generalized iterative scaling (GIS)
  - Improved iterative scaling (IIS)
- Gradient descent methods
  - CG : conjugate gradient
  - L-BFGS : limited memory Newton's method

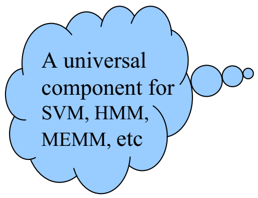
•All of these maximize the (conditional) likelihood.

•Possible to train discriminatively with **voted perceptron**. (Collins 2002)

## Sha and Pereira (2003)

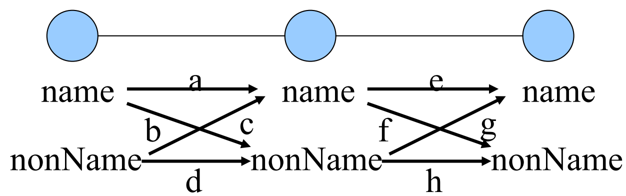
| training method | time | F score | $\mathcal{L}'_\lambda$ |
|---|---|---|---|
| Precond. CG | 130 | 94.19% | -2968 |
| Mixed CG | 540 | 94.20% | -2990 |
| Plain CG | 648 | 94.04% | -2967 |
| L-BFGS | 84 | 94.19% | -2948 |
| GIS | 3700 | 93.55% | -5668 |

Table 3: Runtime for various training methods in minutes, 375k examples



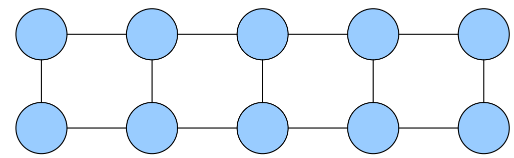

# $\text{argmax}_y\ P(y \mid x)$ … for linear-chain CRFs

$$\begin{aligned}\text{argmax}_y\ p(y \mid x) &= \text{argmax}_y \log p(y \mid x) \\ &= \text{argmax}_y \left( \lambda \bullet \mathbf{F}(y, x) - \log Z(x) \right) \\ &= \text{argmax}_y\ \lambda \bullet \mathbf{F}(y, x)\end{aligned}$$

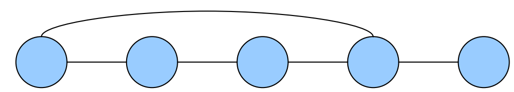



# Other CRFs

Factorial CRFs (Sutton et.al., 2004)

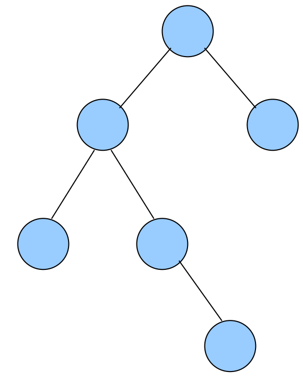


Skip-chain CRFs (Sutton and McCallum, 2004)



Tree CRFs

(Cohn and Blunsom, 2005)



# Graphical representation

HMM MEMM CRF

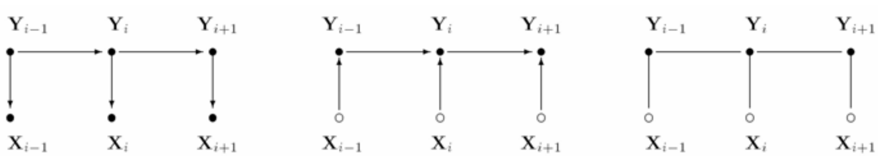
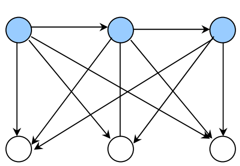
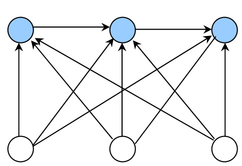


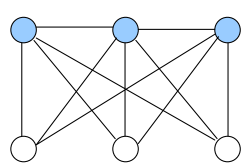

*Figure 2.* Graphical structures of simple HMMs (left), MEMMs (center), and the chain-structured case of CRFs (right) for sequences. An open circle indicates that the variable is not generated by the model.



?



?



?

# Graphical representation

HMM MEMM CRF

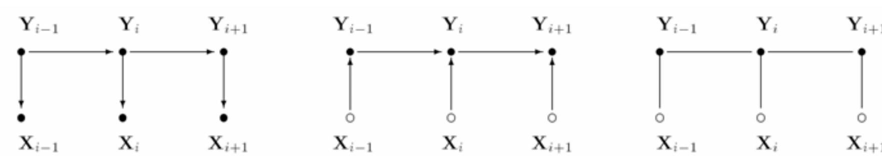
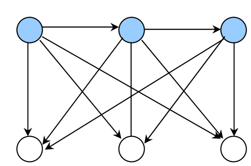
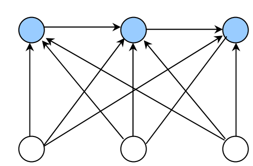


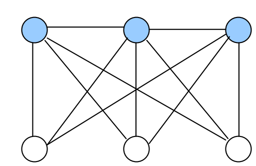

*Figure 2.* Graphical structures of simple HMMs (left), MEMMs (center), and the chain-structured case of CRFs (right) for sequences. An open circle indicates that the variable is not generated by the model.



NO!



YES



YES

From Klein and Taskar's slides.

# Maximum Entropy II

- Also: regularization (smoothing)

$$\max_{\mathbf{w}} \sum_i \left( \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right) - k||w||^2$$

- Maximize likelihood = Minimize "log-loss"

$$\min_{\mathbf{w}} \; k||w||^2 - \sum_i \left( \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

- Motivation
  - Connection to maximum entropy principle
  - Might want to do a good job of being uncertain on noisy cases…
  - … in practice, though, posteriors are pretty peaked

From Klein and Taskar's slides.

# Max vs "Soft-Max" Margin

- SVMs:

$$\min_{\mathbf{w}} k||\mathbf{w}||^2 - \sum_i \Bigg( \underbrace{\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y}} \left( \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}) \right)}_{\text{Hard (Penalized) Margin}} \Bigg)$$

- Maxent:

$$\min_{\mathbf{w}} \; k||w||^2 - \sum_i \Bigg( \underbrace{\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\right)}_{\text{Soft Margin}} \Bigg)$$

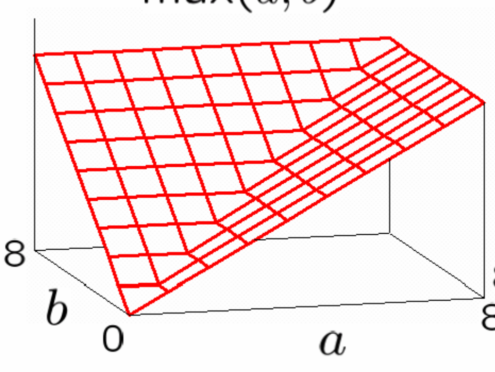
- Very similar! Both try to make the true score better than a function of the other scores.
  - The SVM tries to beat the augmented runner-up
  - The maxent classifier tries to beat the "soft-max"

From Klein and Taskar's slides.

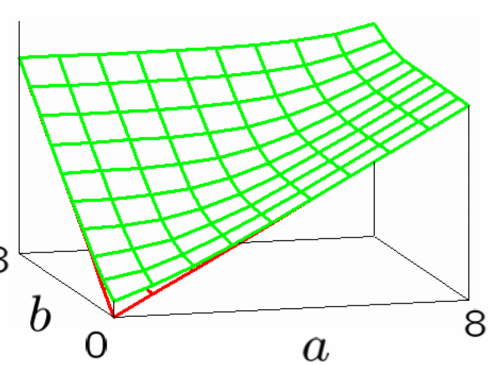
# "Soft-Max"

$$\max(a,b) \approx \log\left(\exp(a) + \exp(b)\right)$$

$\max(a,b)$




$\log\left(\exp(a) + \exp(b)\right)$




From Klein and Taskar's slides.

# Log-Loss

- If we view maxent as a minimization problem:

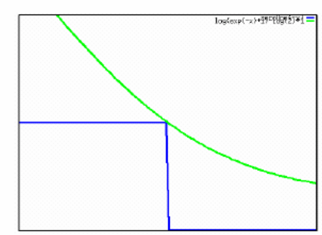
$$\min_{\mathbf{w}} \; k||w||^2 - \sum_i \left( \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right)$$

- This minimizes the "log-loss" on each example

$$-\left[ \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})) \right]$$

$$-\log\left( \frac{\exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i))}{\sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}))} \right) = -\log \mathsf{P}(\mathbf{y}^i | \mathbf{x}^i, \mathbf{w})$$

- Log-loss bounds zero-one loss



$\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y} \neq \mathbf{y}^i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$

From Klein and Taskar's slides.

# Loss Functions: I

- Zero-One Loss

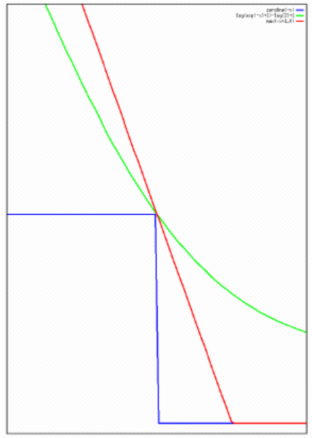
$$\sum_i step\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y} \neq \mathbf{y}^i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\right)$$

- Hinge

$$\sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y}} \left[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(y)\right]\right)$$

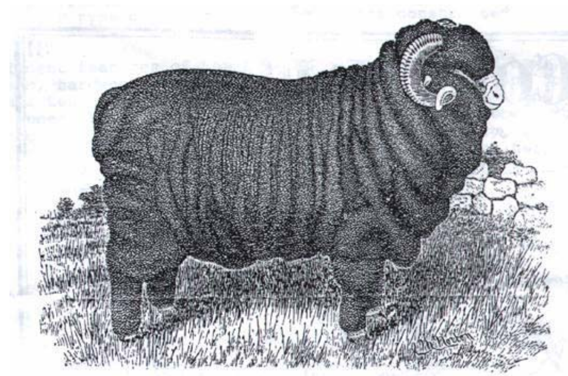
- Log

$$\sum_i \left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \log \sum_{\mathbf{y}} \exp\left(\mathbf{w}^\top \mathbf{f}_i(\mathbf{y})\right)\right)$$



$\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^i) - \max_{\mathbf{y} \neq \mathbf{y}^i} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$

# Merino Sheep



CHERBINO & WILLIAMSON 589.

Bred by Cherbino & Williamson, Middlebury, Vt.; owned by I. L. Eells, Middlebury, Vt.