Homework

- Read section 16.2 (Check errata on web page!)
- Do exercise 16.7, 16.8, and 16.9

Naive Bayes

- Bayes Decision Rule minimizes expected error:

$$\begin{aligned}\hat{c} &= \operatorname*{argmax}_{c \in C} p(c|x) \\ &= \operatorname*{argmax}_{c \in C} p(x, c)\end{aligned}$$

- We can split $p(x, c)$ into two parts: the class prior $p(c)$, and $p(x|c)$, where:

$$p(x|c) = \prod_i p(x_i|c)$$

- Or, we can try other ways to get from $p(x_i|c)$ to $p(x|c)$

Maximum Entropy

- We often need to build probability models without having access to all the required information
- In general, our probability estimates should reflect what we know *and what we don't know*: ignorance is preferable to error
- Shannon's entropy is a measure of ignorance
- Jaynes (1957): "The least informative probability distribution maximizes the entropy S subject to known constraints."
- Wallis derivation

Maximum entropy

- A bit of terminology: let's say $p(x, w)$ is the 'real' probability of event x in context w, and our predicted probability is $q(x, w)$
- We suppose we can get reasonable estimates of $\mathsf{E}_p[f_i]$ for each feature f_i from our training data
- These are our constraints:

$$\begin{aligned} \mathsf{E}_p[f_i] &= \mathsf{E}_q[f_i] \\ \sum_{x,w} p(x, w)\, f_i(x, w) &= \sum_{x,w} q(x, w)\, f_i(x, w) \end{aligned}$$

Maximum entropy

- We used the method of Lagrange multipliers to derive a general solution for the distribution which satisfies these constraints (what we know) while maximizing the entropy (what we don't know)

- The parametric form of the distribution is:

$$q(x;\lambda) = \frac{\exp \sum_i \lambda_i f_i(x)}{\sum_x \exp \sum_i \lambda_i f_i(x)}$$

- But, evaluating the partition function requires summing over all possible configurations, which is often impractical or impossible

Maximum entropy

- One way to avoid this problem is to limit ourselves to just those configurations which actually occur in the training data

- We use these constraints instead:

$$\begin{aligned} \mathsf{E}_p[f_i] &= \mathsf{E}_q[f_i] \\ \sum_{x,w} p(x,w)\, f_i(x,w) &= \sum_{x,w} p(w)\, q(x|w)\, f_i(x,w) \end{aligned}$$

- This gives us the conditional maximum entropy model:

$$q(x|w;\lambda) = \frac{\exp \sum_i \lambda_i f_i(x,w)}{\sum_x \exp \sum_i \lambda_i f_i(x,w)}$$

Parameter estimation

- Given this general form for the distribution, we still need to find λ for any given set of training data
- The form of the distribution maximizes the entropy
- What's left to do is satisfy the constraints: we need to select values for λ which accurately predict our feature expectations

Parameter estimation

- That means, we want to minimize the KL divergence:

$$
\begin{aligned}
D(p||q) &= \sum_{x,w} p(x,w) \log \frac{p(x,w)}{q(x,w;\lambda)} \\
&= \sum_{x,w} p(x,w) \log \frac{p(w)\, p(x|w)}{p(w)\, q(x|w;\lambda)} \\
&= \sum_{x,w} p(x,w) \log \frac{p(x|w)}{q(x|w;\lambda)} \\
&= \sum_{x,w} p(x,w) \left(\log p(x|w) - \log q(x|w;\lambda)\right) \\
&= \sum_{x,w} p(x,w) \log p(x|w) - \sum_{x,w} p(x,w) \log q(x|w;\lambda)
\end{aligned}
$$

Parameter estimation

- Or, in other words, we want to maximize the log-likelihood:

$$
\begin{aligned}
L(\lambda) &= \sum_{x,w} p(x,w) \log q(x|w;\lambda) \\
&= \sum_{x,w} p(x,w) \log \frac{\exp \sum_i \lambda_i f_i(x,w)}{\sum_z \exp \sum_i \lambda_i f_i(z,w)} \\
&= \sum_{x,w} p(x,w) \sum_i \lambda_i f_i(x,w) - \sum_{x,w} p(x,w) \log \sum_z \exp \sum_i \lambda_i f_i(z,w)
\end{aligned}
$$

Parameter estimation

- So, we we need to find the *gradient* of the log likelihood $G(\lambda) = \nabla L(\lambda)$ and find a stationary point.
- Some reminders:

$$
\begin{aligned}
\frac{d}{dx}\left[f(x)\,g(x)\right] &= f(x)\,g'(x) + g(x)\,f'(x) \\
\frac{d}{dx}\left[\log f(x)\right] &= \frac{1}{f(x)}\,f'(x) \\
\frac{d}{dx}\left[\exp f(x)\right] &= f'(x)\,\exp f(x)
\end{aligned}
$$

Parameter estimation

- So, for the gradient we get:

$$\begin{aligned}
\frac{\partial L(\lambda)}{\partial \lambda_i} &= \sum_{x,w} p(x,w) f_i(x,w) - \\
&\quad \sum_{x,w} p(x,w) \sum_z \frac{\exp \sum_k \lambda_k f_k(z,w)}{\sum_y \exp \sum_k \lambda_k f_k(y,w)} f_i(z,w) \\
&= \sum_{x,w} p(x,w) f_i(x,w) - \sum_w \left(\sum_x p(x,w) \right) \sum_z q(z|w;\lambda) f_i(z,w) \\
&= \sum_{x,w} p(x,w) f_i(x,w) - \sum_{w,z} p(w) q(z|w;\lambda) f_i(z,w) \\
&= \mathsf{E}_p[f_i] - \mathsf{E}_q[f_i]
\end{aligned}$$

which should be reassuring

Parameter estimation

- The log-likelihood function L is *convex*
- That means that its value is maximized at λ^* where $G(\lambda^*) = 0$.
- The partial derivative of $L(\lambda)$ for any λ_i depends on all the other λ's, so there is no closed form solution
- Instead we proceed iteratively.

Parameter estimation

ESTIMATE(p)

$\lambda^0 \leftarrow 0$
$k \leftarrow 0$
repeat
    compute $q^{(k)}$ from $\lambda^{(k)}$
    compute update $\delta^{(k)}$
    $\lambda^{(k+1)} \leftarrow \lambda^{(k)} + \delta^{(k)}$
    $k \leftarrow k + 1$
  until converged
return $\lambda^{(k)}$

Iterative scaling

- Generalized Iterative Scaling (Darroch and Ratcliff 1972):

$$\delta^{(k)} = \log \left(\frac{\mathsf{E}_p[f]}{\mathsf{E}_{q^{(k)}}[f]} \right)^{\frac{1}{C}}$$

- Descended from Iterative Proportional Fitting (Deming and Stephan 1940)

- Learning rate C is the maximum sum of the values of all the features:

$$C = \max_{x,w} \sum_i f_i(x, w)$$

- Easy to compute, doesn't require evaluating gradient, or even probabilities

Iterative scaling

- Improved Iterative Scaling (Della Pietra, Della Pietra, Lafferty 1997) relaxes requirement for constant C
- Perform iterative scaling in each dimension in parallel, to find $\delta_i^{(k)}$ such that:

$$\mathsf{E}_p[f_i] = \sum_{x,w} p(x,w) q^{(k)}(x|w) f_i(x,w) \exp(C(x,w)\delta_i^{(k)})$$

- This one-dimensional optimization problem can itself be solved iteratively
- Improved Iterative Scaling also only requires computation of expectations.
- But, for this problem, iterative scaling updates are as expensive to compute as the gradient.

First order methods

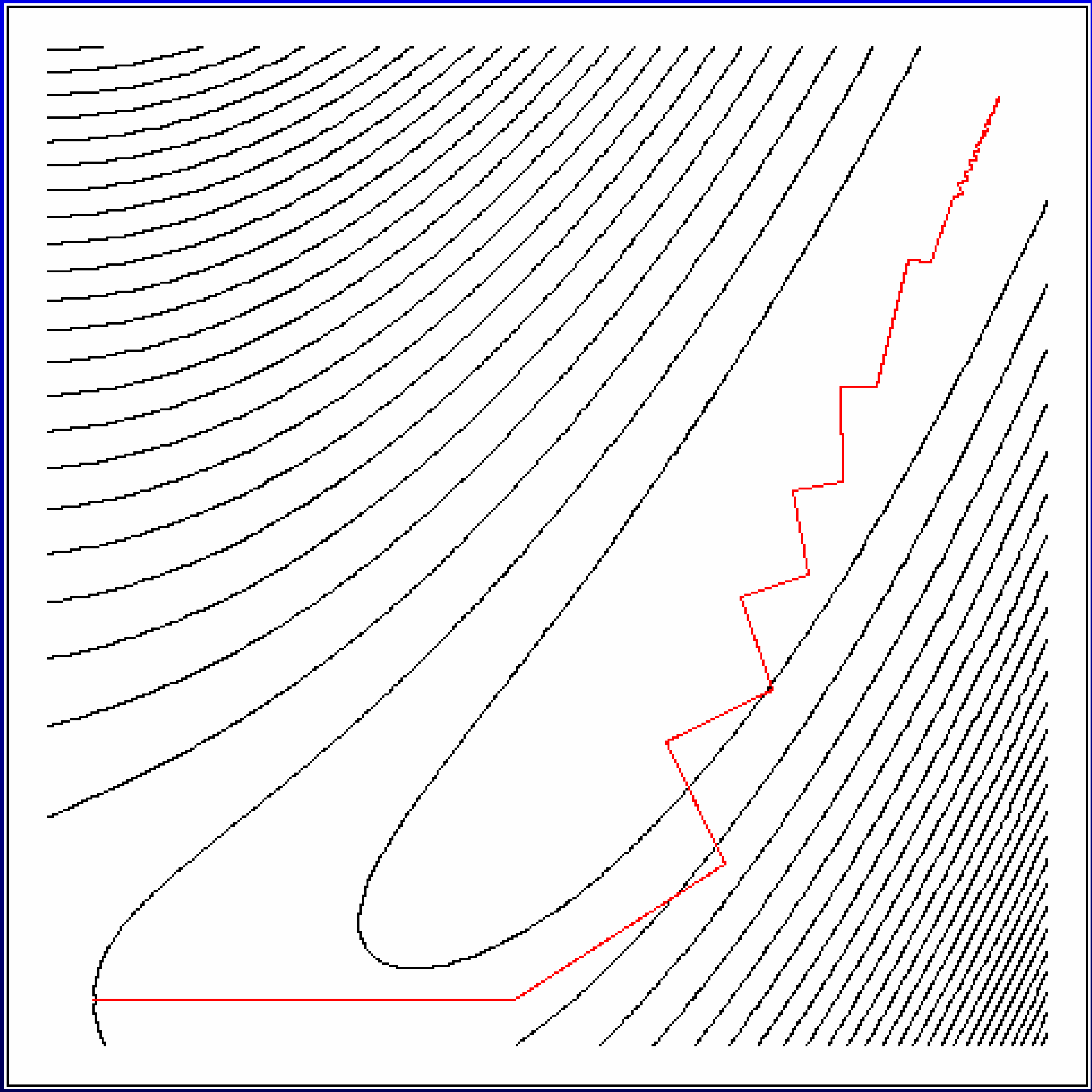
- The simplest first-order method follows the gradient to find the direction of *steepest ascent,* with the step size $\alpha^{(k)}$ selected by line search:

$$\delta^{(k)} = \alpha^{(k)} G(\lambda^{(k)})$$

- Steepest ascent is locally optimal, in a narrow sense

- Steepest ascent considers the same search directions repeatedly, leading to slow convergence.

First order methods



First order methods

- Conjugate gradient methods such as the *Fletcher-Reeves* or *Polak-Ribière* algorithms avoid this.
- Search direction p is a function of the previous search direction and the steepest ascent direction:

$$\begin{aligned} \beta^{(k)} &= \frac{G(\lambda^{(k)})^T G(\lambda^{(k)})}{G(\lambda^{(k-1)})^T G(\lambda^{(k-1)})} \\ p^{(k)} &= G(\lambda^{(k)}) + \beta^{(k)} p^{(k-1)} \end{aligned}$$

- As with steepest ascent, optimal step size is found by a line search:

$$\delta^{(k)} = \alpha^{(k)} p^{(k)}$$

Second order methods

- We can improve on first-order methods by taking the second derivative into account

- If we locally model our log likelihood as a quadratic function, then the Taylor series approximation gives us:

$$L(\lambda + \delta) \approx L(\lambda) + \delta^T G(\lambda) + \frac{1}{2}\delta^T H(\lambda)\delta$$

- We want to find the δ which maximizes this, so:

$$\begin{aligned} 0 + G(\lambda) + \delta^T H(\lambda) &= 0 \\ \delta^T H(\lambda) &= -G(\lambda) \\ \delta^T &= -\frac{G(\lambda)}{H(\lambda)} \end{aligned}$$

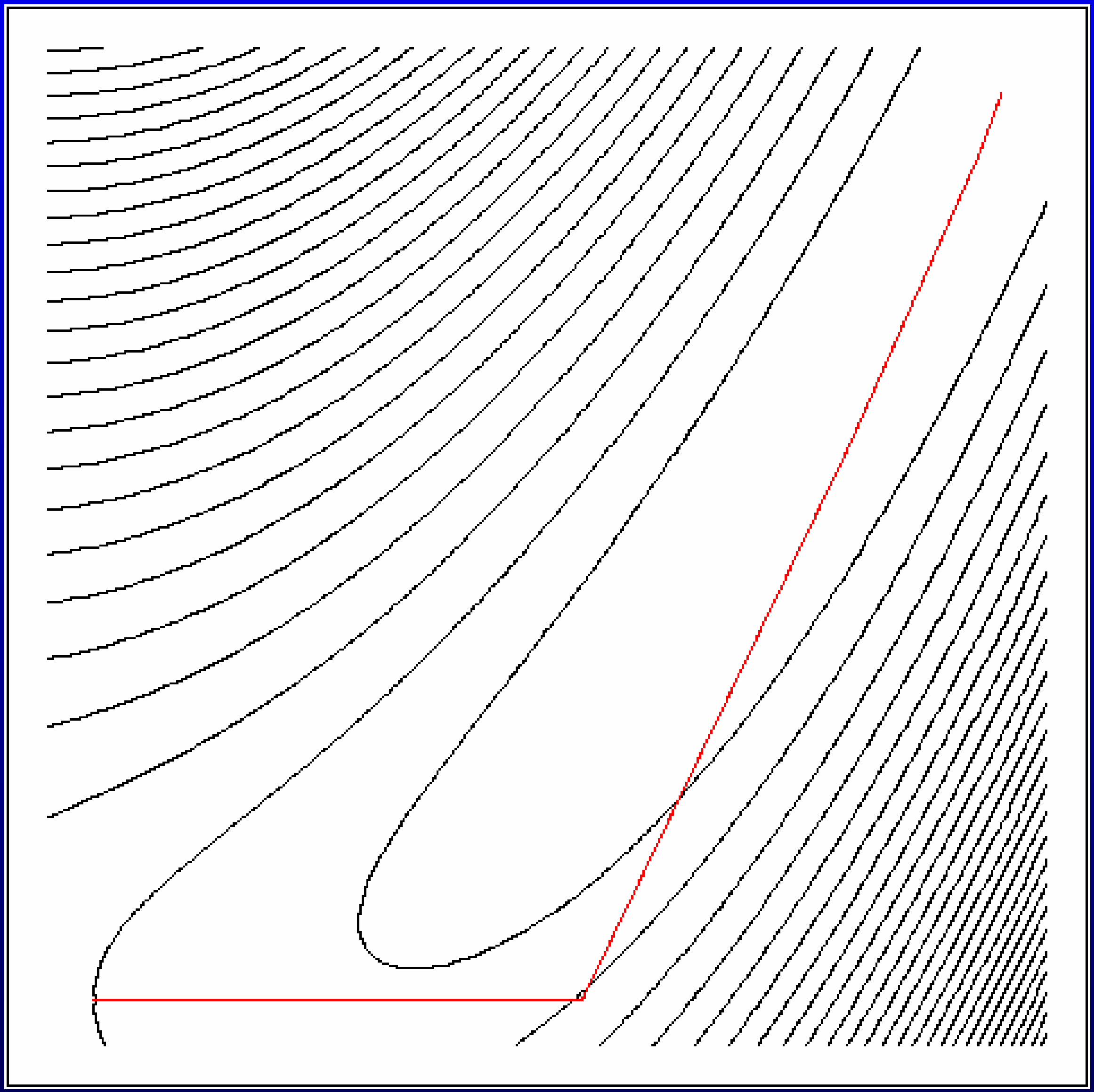
Second order methods

- This yields *Newton's method*:

$$\delta^{(k)} = -H^{-1}(\lambda^{(k)})G(\lambda^{(k)})$$

- This update rule provides both a direction and a step size, so a line search is generally unnecessary
- Under certain conditions, $\delta^{(k)}$ will not be an ascent direction, so to guarantee convergence a line search is sometimes required
- Newton's method converges quickly (in one step, for a quadratic objective function)

Second order methods



Second order methods

- Our log likelihood is twice differentiable, with the *Hessian* matrix:

$$H_{ij}(\lambda) = \mathsf{E}_{q_\lambda}[f_i f_j] - \mathsf{E}_{q_\lambda}[f_i]\mathsf{E}_{q_\lambda}[f_j]$$

(This is the variance-covariance matrix for f.)

- A variant of this (Fisher scoring) is used to fit log-linear models for statistical analysis

- For models with lots of parameters, H is too expensive to compute and invert on each iteration

Second order methods

- As we get close to a solution, we will be computing the gradient G at lots of closely space points
- We can use these gradients to estimate H (analogous to finite differencing)
- Quasi-Newton methods replace inverse Hessian with:

$$\delta^{(k)} = B^{(k)} G(\lambda^{(k)})$$

where $B^{(k)}$ is a symmetric, positive definite matrix which satisfies the equation:

$$B^{(k)} y^{(k)} = \delta^{(k-1)}$$

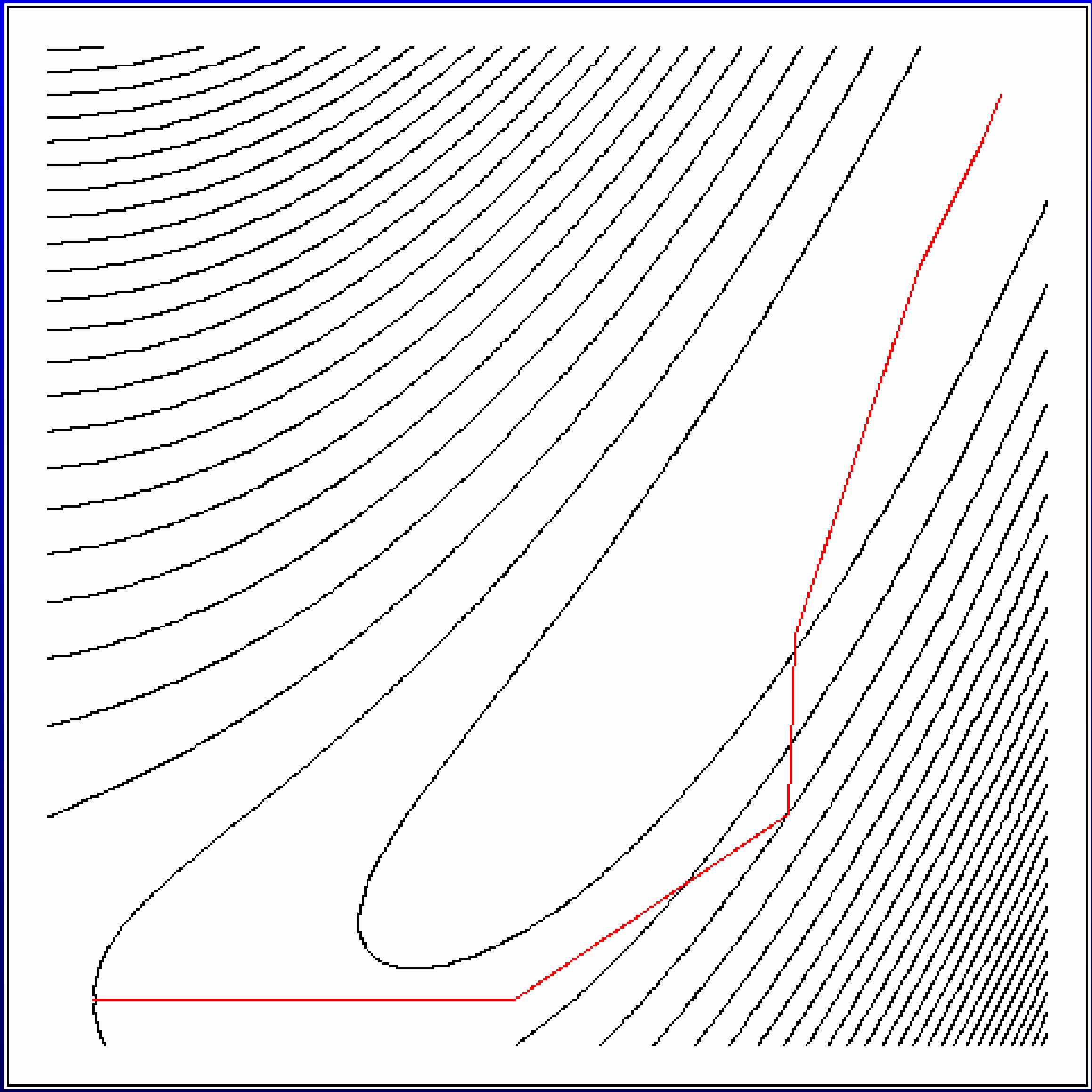
with

$$y^{(k)} = G(\lambda^{(k)}) - G(\lambda^{(k-1)})$$

Second order methods

- *Quasi-Newton methods* update an approximation of H^{-1} on each iteration, saving the cost of recomputing it
- But, we still need to store B: for 100,000 features, this would require more than 74gb!
- *Limited memory variable metric* methods store $B^{(k)}$ in a compact form, using the previous m values of $y^{(k)}$ and $\delta^{(k)}$.
- In practice, $m \leq 5$ works well, converging almost as fast as Newton's method with much more modest computational requirements

Second order methods

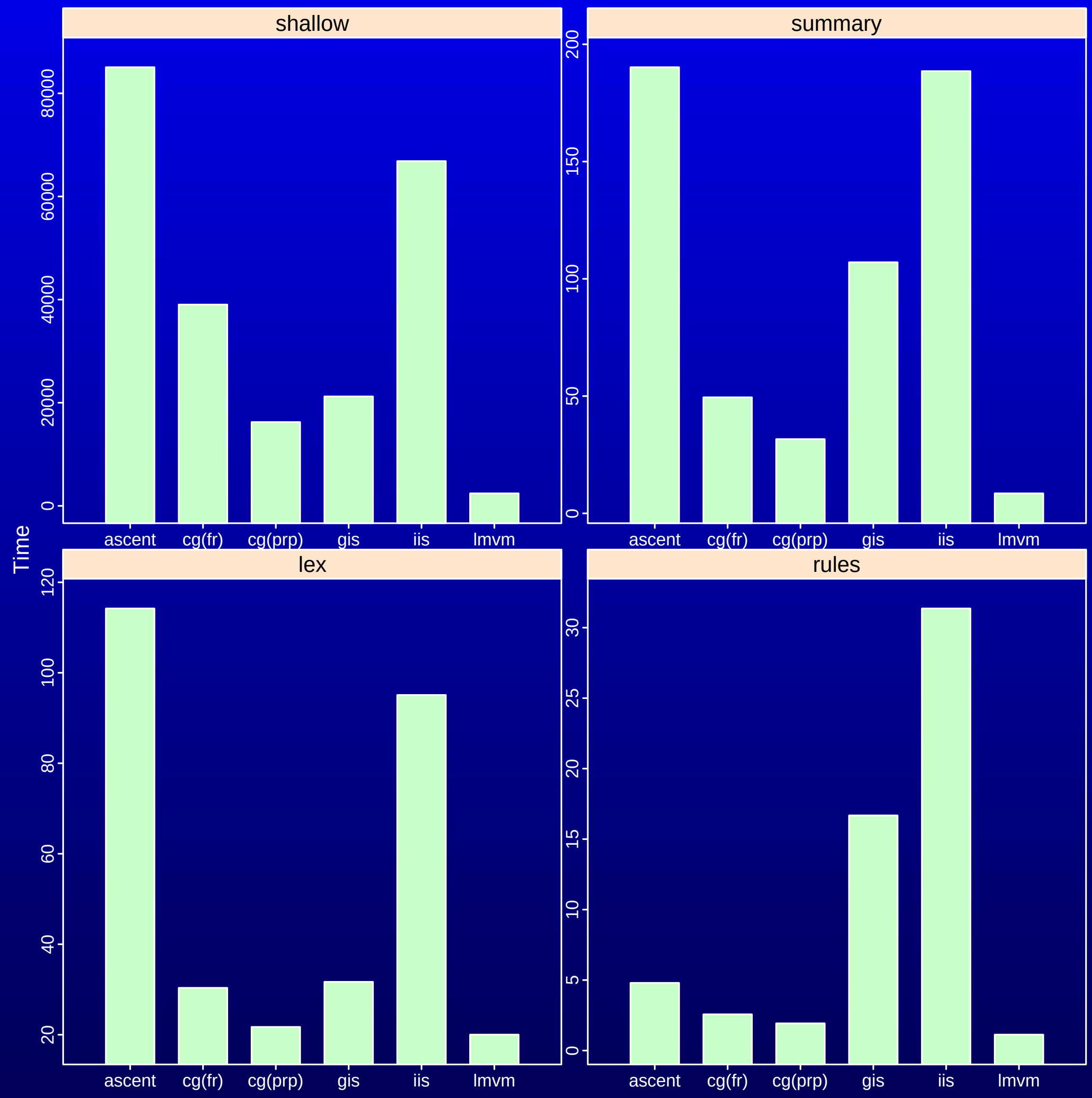


Evaluation

- Reduces parameter estimation to well-known problems (non-linear optimization, sparse matrix-vector products).
- PETSc and TAO (part of DoE's ACTS Toolkit) provide the basis for efficient, highly scalable parameter estimation software, optimized for workstations, clusters, and parallel supercomputers.
- Data sets used for evaluation:

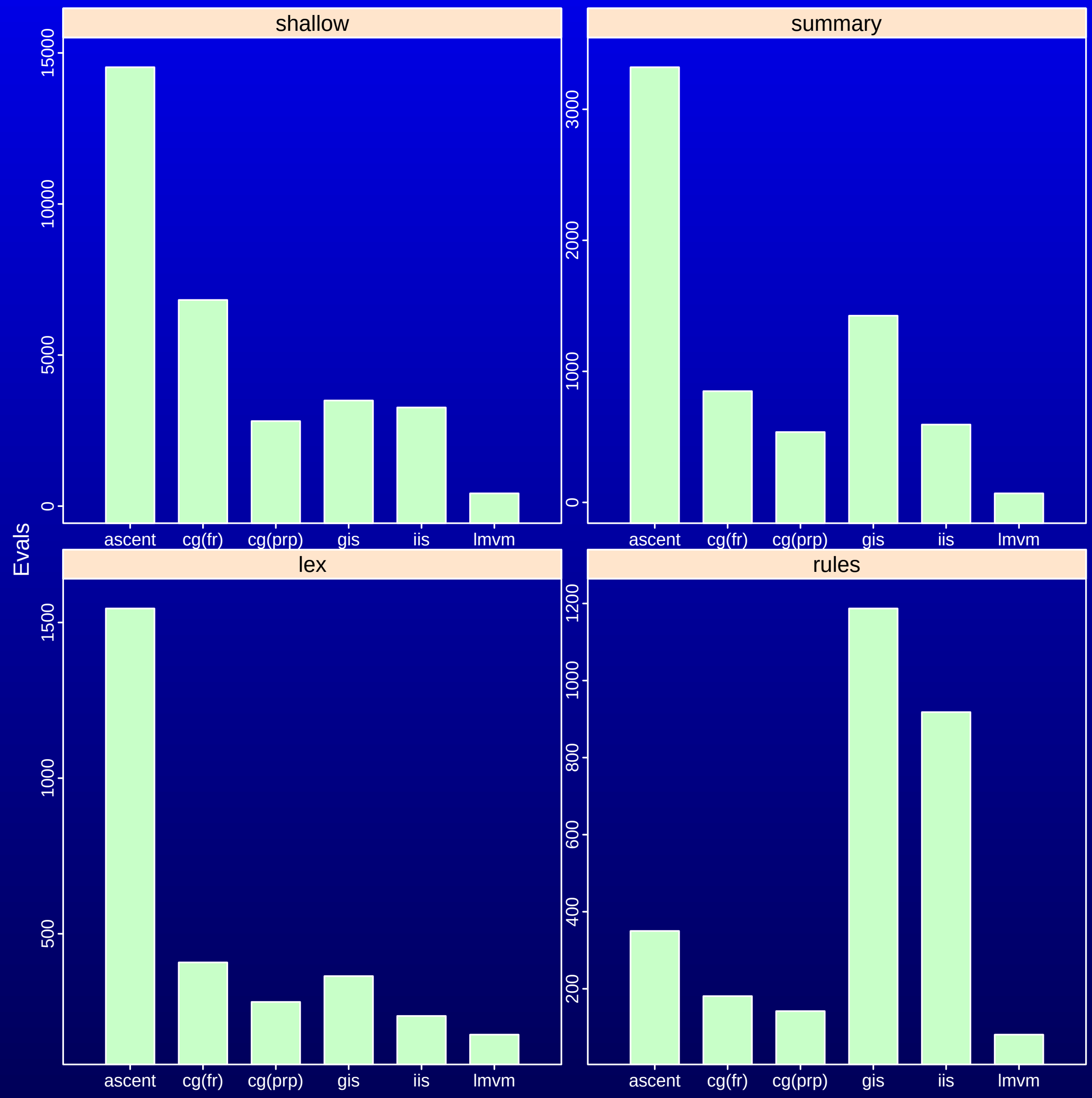
dataset	classes	contexts	features	non-zeros
rules	29,602	2,525	246	732,384
lex	42,509	2,547	135,182	3,930,406
summary	24,044	12,022	198,467	396,626
shallow	8,625,782	375,034	264,142	55,192,723

Evaluation

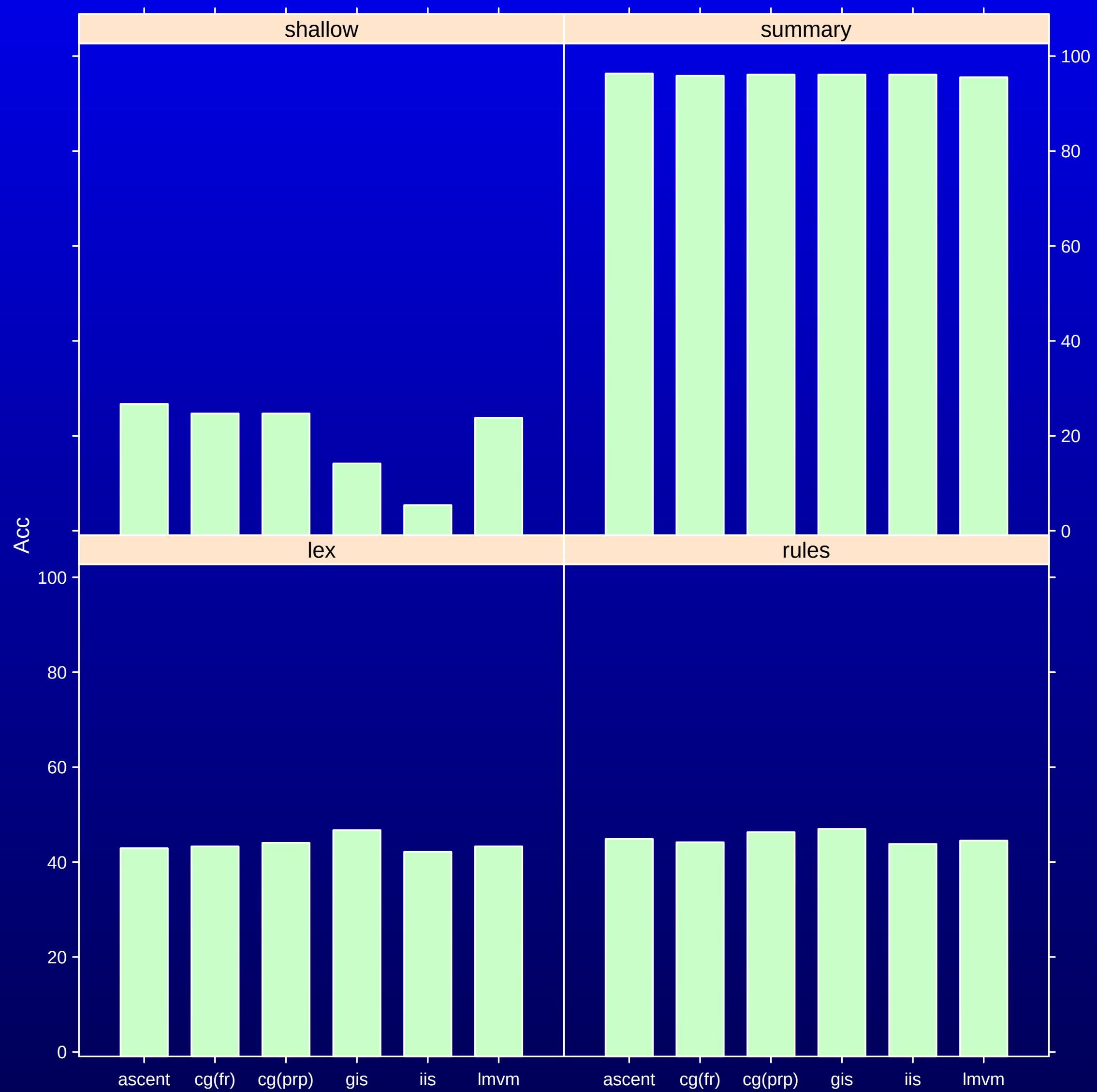


shallow
summary
lex
rules
Time
0
20000
40000
60000
80000
50
100
150
200
20
40
60
80
100
120
5
10
15
20
25
30
ascent
cg(fr)
cg(prp)
gis
iis
lmvm

Evaluation

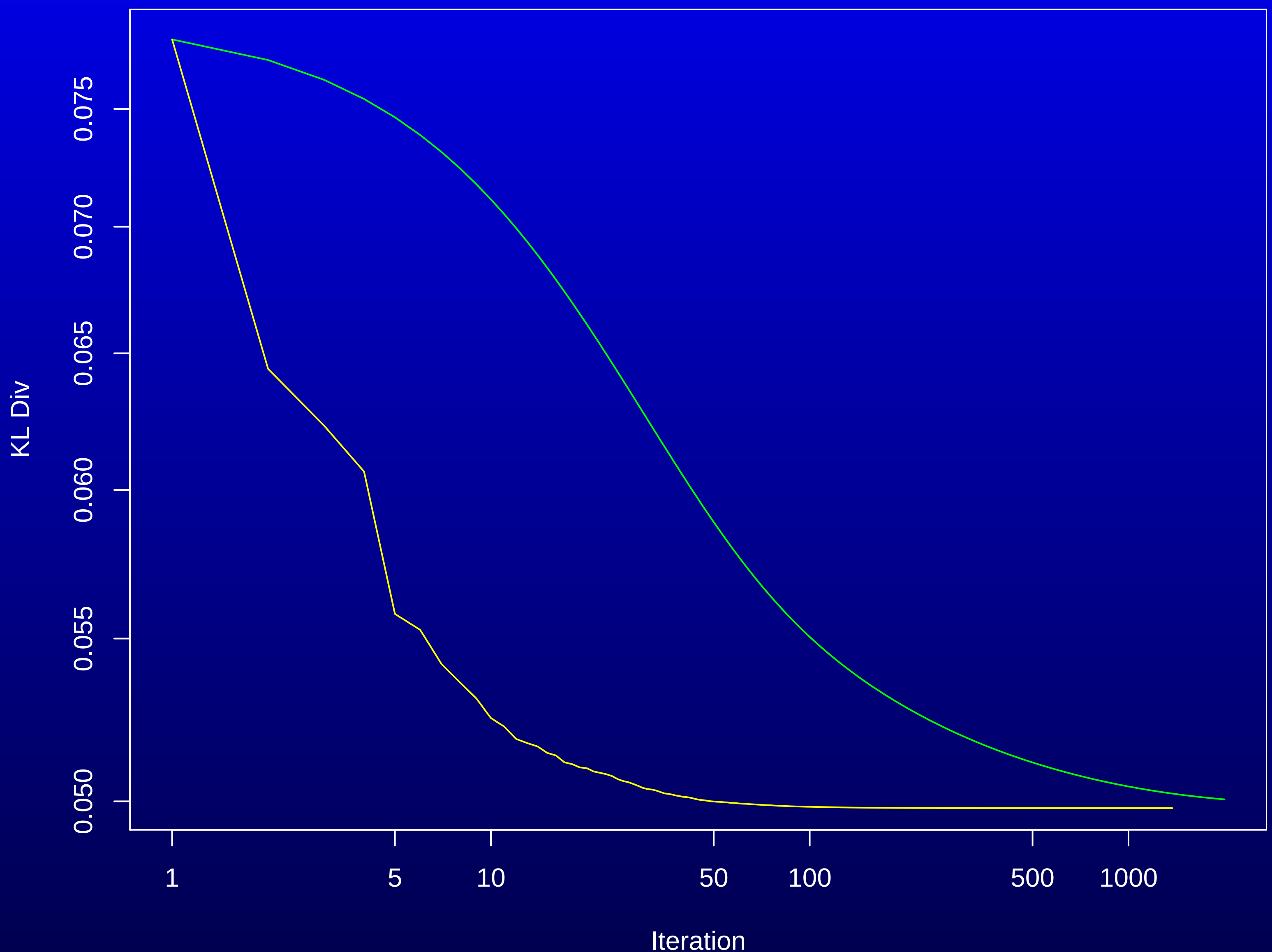

Evals
shallow
summary
lex
rules
0
5000
10000
15000
1000
2000
3000
500
1500
200
400
600
800
1200
ascent
cg(fr)
cg(prp)
gis
iis
lmvm

Evaluation



shallow
summary
lex
rules
Acc
0
20
40
60
80
100
ascent
cg(fr)
cg(prp)
gis
iis
lmvm

Convergence



Results

- Advantages of IIS over GIS are slim.
- CG and LMVM show similar convergence properties, but LMVM tends to take less time per iteration.
- Both methods converge substantially faster than iterative scaling.
- Some algorithms are more robust than others to problems with the training data.
- High-quality numerical libraries offer many advantages for NLP
- Software available: `estimate` and `classify`

Smoothing

- As described, this is will find the maximum likelihood estimate, and runs into all the usual problems
- In fact, it's worse, since MaxEnt models can't represent probabilities of 0 or 1 with finite feature values
- Smoothing is just as important with MaxEnt models as any other probabilistic models
- All the usual smoothing methods can be applied in computing the empirical expectation $\mathrm{E}_p[f_i]$

Gaussian prior

- Another option is to use MAP estimation:

$$\lambda^* = \operatorname*{argmax}_{\lambda} q(x|w;\lambda)\, p(\lambda)$$

- The parameter prior $p(\lambda)$ is the probability of a particular parameter vector independent from the training data
- MLE implicitly assumes a uniform prior over parameters
- A Gaussian prior with $\mu = 0$ will tend to prefer uniform models

Gaussian prior

- If $L(\lambda)$ is the log likelihood we use for ML estimation, we can construct a penalized likelihood:

$$
\begin{aligned}
L'(\lambda) &= L(\lambda) + \sum_i \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-\lambda_i}{2\sigma^2}\right) \\
&= L(\lambda) - \sum_i \frac{\lambda_i^2}{2\sigma^2} + C
\end{aligned}
$$

- And the gradient G' is:

$$
G'(\lambda) = G(\lambda) - \sum_i \frac{\lambda_i}{\sigma^2}
$$