Maximum entropy

- We construct as set of constraints from the training data (*sufficient statistics*)y:

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

- Satisfying these constraints while maximizing the entropy gives us the conditional maximum entropy model:

$$p(x|w;\lambda) = \frac{\exp \sum_i \lambda_i f_i(x,w)}{\sum_z \exp \sum_i \lambda_i f_i(z,w)}$$

Parameter estimation

- To find particular values of λ given particular constraints, we need to maximize the log likelihood of the training data:

$$L(\lambda) = \sum_{x,w} \tilde{p}(x,w) \log p(x|w;\lambda)$$

- We can use the gradient of the log likelihood to iteratively update our best guess for λ:

$$\frac{\partial L(\lambda)}{\partial \lambda_i} = \mathsf{E}_{\tilde{p}}[f_i] - \mathsf{E}_p[f_i]$$

- We continue updating λ until the gradient stops getting smaller

Gaussian prior

- Another option is to use MAP estimation:

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

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

Gaussian prior

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

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

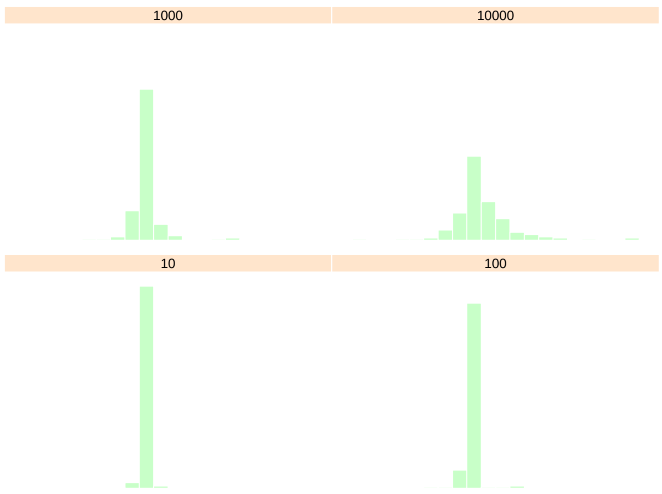
- And the gradient G' is:

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

Gaussian prior

Gaussian prior

Gaussian prior



MaxEnt models

- MaxEnt models can be applied to any problem with dubious independence assumptions
- For example, PCFGs:

$$p(t) = \prod_i p(r_i(t))$$

could be recast as:

$$p(t) = \frac{\exp \sum_i \lambda_i r_i(t)}{\sum_{t'} \exp \sum_i \lambda_i r_i(t')}$$

- The requires summing over all trees, so for disambiguation a conditional model would be more practical
- This can be extended to grammatical formalisms beyond CFGs (e.g., DCGs, HPSG)

MaxEnt models

- Tagging:

$$\begin{aligned} p(t_1 \ldots t_n | w_1 \ldots w_n) &= \frac{p(w_1 \ldots w_n | t_1 \ldots t_n) p(t_1 \ldots t_n)}{p(w_1 \ldots w_n)} \\ &\propto p(w_1 \ldots w_n | t_1 \ldots t_n)\, p(t_1 \ldots t_n) \\ &\approx \prod_i p(w_i | t_i)\, p(t_i | t_{i-1}) \end{aligned}$$

- A Maximum Entropy Markov Model reduces the independence assumptions:

$$p(t_i | w_i, t_{i-1}) = \frac{\exp \sum_i \lambda_i f_i(w_i, t_{i-1}, t_i)}{\sum_{t'} \exp \sum_i \lambda_i f_i(w_i, t_{i-1}, t')}$$

Conditional Random Fields

- MEMM's still make the *Markov assumption*:

$$p(t_1 \ldots t_n | w_1 \ldots w_n) = \prod_i p(t_i | w_i, t_{i-1})$$

- Label bias problem: all probability going into state is passed on to successors, and states with fewer outgoing transitions will be preferred to those with more

- Lafferty, McCallum, and Pereira (2001) eliminate that too: Conditional Random Fields

$$p(t_1 \ldots t_n | w_1 \ldots w_n) = \frac{1}{Z(w_1 \ldots w_n)} \exp \sum_i \lambda_i f_i(w_1 \ldots w_n, t_1 \ldots t_n)$$

Conditional Random Fields

- Notice the partition function: $Z(w_1 \ldots w_n)$

- If our features are like those from typical HMMs, we can use a variant of the forward-backward algorithm to compute feature expectations during training

- Since features don't need to be independent, we can add morphological features for unknown words

- Some tagging results:

	error	OOV error
HMM	5.69	45.99
MEMM	6.37	54.61
CRF	5.55	48.05
MEMM+	4.81	26.99
CRF+	4.27	23.76

Minimum Divergence models

- The maximum entropy principle says that a uniform prior best represents total ignorance

- What if we're not totally ignorant?

- A generalization of MaxEnt models (MEMD) minimizes the KL divergence $D(p||q)$ between the empirical distribution and some prior distribution q:

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

- For example, this can be an efficient way of combining local and non-local features in language modeling

Committee machines

- Committe machines (or ensemble machines) combine the predictions of more than one classifier (Nilsson 1965)
- Divide and conquer algorithms
- Like real committees, committee machines depend on the members being:
  - reasonably accurate (better than guessing)
  - diverse (errors are uncorrelated)

Majority vote

- Suppose we have L classifiers, each with an error rate of $p < 0.5$
- If the errors are uncorrelated, then the error rate of the committee is given by the c.d.f. of the binomial distribution:

$$p(X > \frac{L}{2}) = \sum_{x=L/2}^{L} \binom{n}{x} p^x (1-p)^{n-x}$$

- For example, if $L = 21$ and $p = 0.3$, the overall error rate is 0.026
- If errors are not independent, then the overall error rate may be worse than predicted
- If $p > 0.5$, then the committee will be worse than any individual member

Majority vote

CoNLL 2003 shared task results

	precision	recall	F
*[FIJZ03]	88.99%	88.54%	88.76±0.7
*[CN03]	88.12%	88.51%	88.31±0.7
*[KSNM03]	85.93%	86.21%	86.07±0.8
[ZJ03]	86.13%	84.88%	85.50±0.9
[CMP03b]	84.05%	85.96%	85.00±0.8
[CC03]	84.29%	85.50%	84.89±0.9
[MMP03]	84.45%	84.90%	84.67±1.0
[CMP03a]	85.81%	82.84%	84.30±0.9
*[ML03]	84.52%	83.55%	84.04±0.9
[BON03]	84.68%	83.18%	83.92±1.0
[MLP03]	80.87%	84.21%	82.50±1.0
[WNC03]	82.02%	81.39%	81.70±0.9
*[WP03]	81.60%	78.05%	79.78±1.0
[HV03]	76.33%	80.17%	78.20±1.0
[DD03]	75.84%	78.13%	76.97±1.2
[Ham03]	69.09%	53.26%	60.15±1.3
baseline	71.91%	50.90%	59.61±1.2

Majority vote

- Sets of five individual classifiers were combined, with output selected by majority vote
- Best committee: [FIJZ03]+[CN03]+[KSNM03]+[ML03]+[WP03]
- Majority vote gives $F = 90.3$ (vs. $F = 88.76$ for best individual system)
- Notice that the best combination doesn't use the highest-scoring individual classifiers

Majority vote

- Weighted majority scales votes by classifier' s accuracy
- Weights can be set by one pass through training data, decreasing weights of classifiers when they make a mistake
- Bayesian voting averages all hypothesis weighted by their posterior probability:
  $$p(c|x,T) = \sum h(x)\, p(h|T)$$
  where:
  $$p(h|T) \propto p(T|h)\, p(h)$$
- A meta-classifier can be used to map a set of individual answers and/or the test instance into a single aggregate classifier

Committee machines

- Why do committee machines work?
- Statistical error: more than one hypothesis fits the training data equally well
- Computational error: learning algorithm finds a locally optimal hypothesis
- Representational error: hypothesis space does not include the target concept
- Committee machines can reduce or eliminate these problems

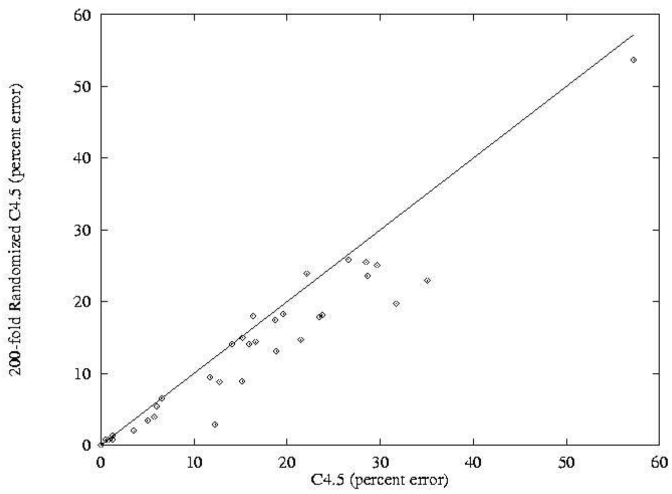
Committee machines

- Take C4.5 as an example
- Statistical error: Large decision trees require a lot of training data
- Computational error: C4.5 uses a *greedy* feature selection strategy
- Representational error: decision trees divide space into rectangular regions

Generating diversity

- Given training data and a learning method, how to we generate a diverse committee of classifiers?
- We can inject randomness into the learning procedure by varying initial conditions or hyperparameters
- Randomized C4.5 (Dietterich 2000) randomly selects one out of the features with the n highest gain ratios
- Normal C4.5 is unstable, since small changes in the training data can make a large difference in the resulting decision tree (and the classification accuracy)

Randomized C4.5



Generating diversity

- We can also add randomness to the training data (noise)
- n-fold cross-validation produces n semi-independent training sets and n semi-independent classifiers
- *Bagging* (Breiman 1996): randomly generate lots (25–200) of training sets by sampling the original training data with replacement (63.2% overlap on average)
- Construct classifiers for each training set$\rightarrow$ majority vote
- Reduces variance, so most effective for high variance classifiers
- If variance is low, bagging can actually make things worse

Generating diversity

- Another strategy is to select disjoint subsets of features
- Reduces dimensionality for each classifier (important for classifiers like neural nets)
- Each resulting classifier must still be fairly accurate, so only works if features can be broken down into independent subsets
- Volcano identification (Cherkauer 1996) – 32 neural nets using 8 subsets of 119 features $\times$ 4 network sizes
- Not especially useful for NLP (but: cotraining)