Performance Prediction and Shrinking Language Models

Stanley F. Chen†

IBM T.J. Watson Research Center
Yorktown Heights, New York, USA

27 June 2011

† Joint work with Stephen Chu, Ahmad Emami, Lidia Mangu, Bhuvana Ramabhadran, Ruhi Sarikaya, and Abhinav Sethy.

What Does a Good Model Look Like?

$$\textit{(test error)} \equiv \textit{(training error)} + \textit{(overfit)}$$

Overfitting: Theory

- *e.g.*, Akaike Information Criterion (1973)

$$-\textit{(test LL)} \approx -\textit{(train LL)} + \textit{(\# params)}$$

- *e.g.*, structural risk minimization (Vapnik, 1974)

$$\textit{(test err)} \leq \textit{(train err)} + f(\textit{VC dimension})$$

- Down with big models!?

The Big Idea

- Maybe *overfit* doesn't act like we think it does.
- Let's try to fit *overfit* empirically.

What This Talk Is About

- An empirical estimate of the overfit in log likelihood of . . .
  - Exponential language models . . .
  - That is really simple and works really well.
- Why it works.
- What you can do with it.

Outline

Exponential N-Gram Language Models

- Language model: predict next word given previous, say, two words.

$$P(y = \textit{ate} \mid x = \textit{the cat})$$

- Log-linear model: features $f_i(\cdot)$; parameters λ_i.

$$P(y|x) = \frac{\exp(\sum_i \lambda_i f_i(x, y))}{Z_\Lambda(x)}$$

- A binary feature $f_i(\cdot)$ for each n-gram in training set.
- An alternative parameterization of back-off n-gram models.

Details: Regression

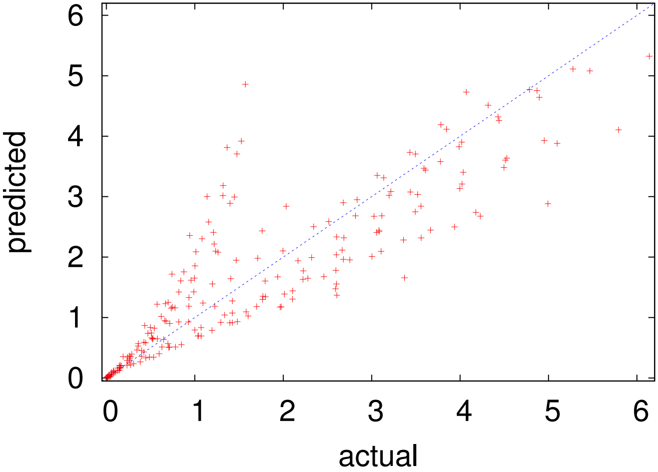
- Build hundreds of (regularized!) language models.
- Compute actual overfit: log likelihood (LL) per event = log PP.
- Calculate lots of statistics for each model.
  - F = # parameters; D = # training events.

$$\frac{F}{D};\ \frac{F\log D}{D};\ \frac{1}{D}\sum\lambda_i;\ \frac{1}{D}\sum\lambda_i^2;\ \frac{1}{D}\sum|\lambda_i|^{\frac{4}{3}};\ \ldots$$

- Do linear regression!

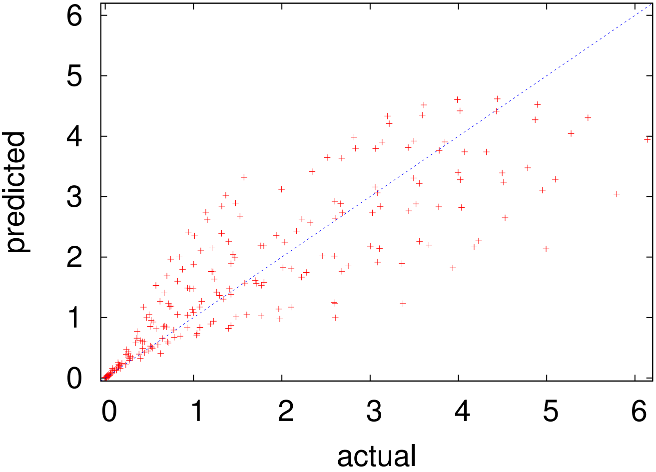
What Doesn't Work? AIC-like Prediction

$$\textit{(overfit)} \equiv LL_{\text{test}} - LL_{\text{train}} \approx \gamma \frac{\textit{(\# params)}}{\textit{(\# train evs)}}$$



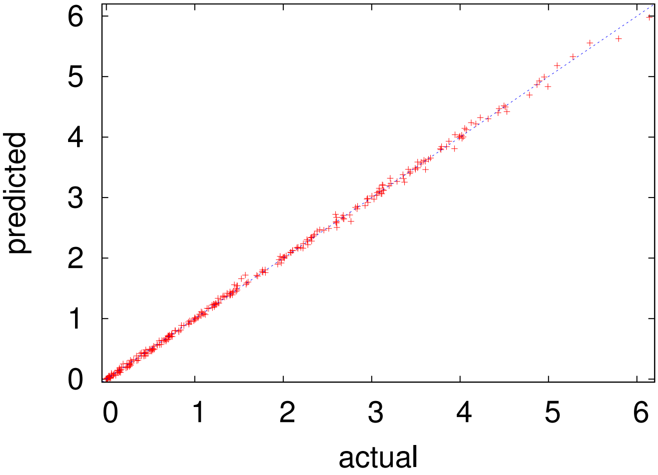
What Doesn't Work? BIC-like Prediction

$$LL_{\text{test}} - LL_{\text{train}} \approx \gamma \frac{\textit{(\# params)} \log \textit{(\# train evs)}}{\textit{(\# train evs)}}$$



What Does Work? (r = 0.9996)

$$LL_{\text{test}} - LL_{\text{train}} \approx \frac{\gamma}{\textit{(\# train evs)}} \sum_{i=1}^{F} |\lambda_i|$$

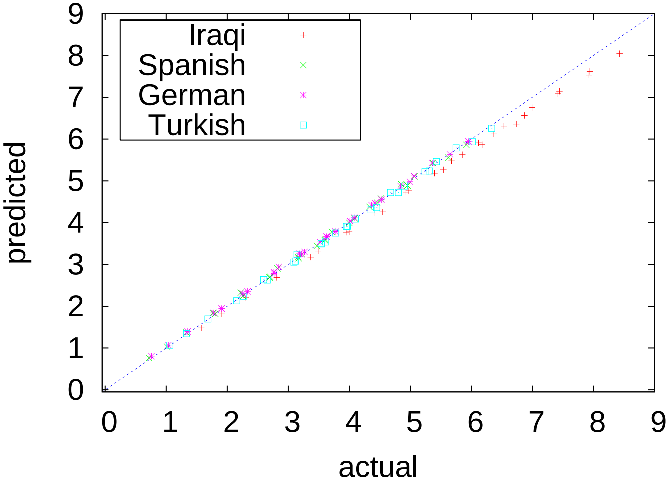


$\gamma = 0.938$

- Holds for many different types of data.
  - Different domains (*e.g.*, Wall Street Journal, ...)
  - Different token types (letters, parts-of-speech, words).
  - Different vocabulary sizes (27–84,000 words).
  - Different training set sizes (100–100,000 sentences).
  - Different *n*-gram orders (2–7).
- Holds for many different types of exponential models.
  - Word *n*-gram models; class-based *n*-gram models; minimum discrimination information models.

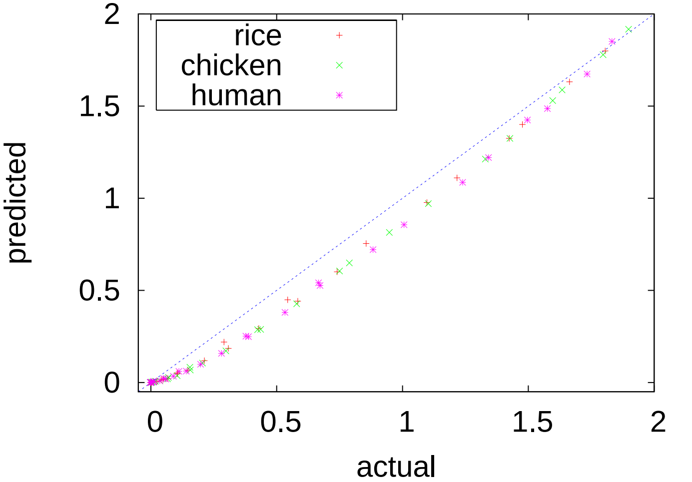
What About Other Languages?

$$LL_{\text{test}} - LL_{\text{train}} \approx \frac{0.938}{\textit{(\# train evs)}} \sum_{i=1}^{F} |\lambda_i|$$



What About Genetic Data?

$$LL_{\text{test}} - LL_{\text{train}} \approx \frac{0.938}{\textit{(\# train evs)}} \sum_{i=1}^{F} |\lambda_i|$$



Outline

Regularization

- Improves test set performance.
- ℓ_1, ℓ_2^2, $\ell_1 + \ell_2^2$ regularization: choose λ_i to minimize

$$\textit{(obj fn)} \equiv LL_{\text{train}} + \alpha \sum_{i=1}^{F} |\lambda_i| + \frac{1}{2\sigma^2} \sum_{i=1}^{F} \lambda_i^2$$

- The problem: γ depends on α, σ!

Regularization: Two Criteria

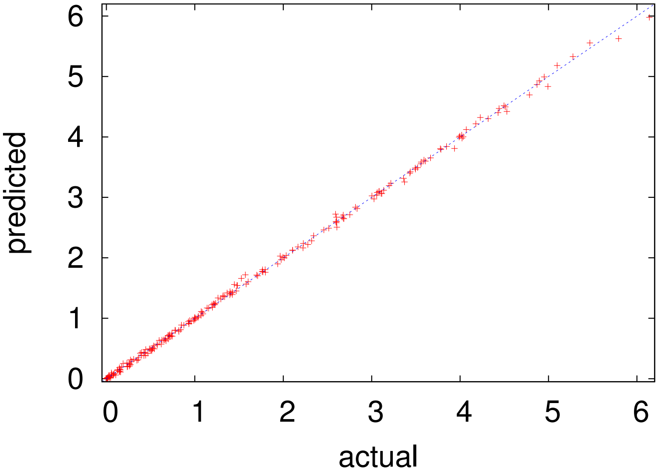
- Here: pick single α, σ across all models.
  - Usual way: pick α, σ per model for good performance.
- Good performance and good overfit prediction?

	performance	overfit prediction
ℓ_1		$\surd$
ℓ_2^2	$\surd$	
$\ell_1 + \ell_2^2$	$\surd$	$\surd$

- $(\alpha = 0.5, \sigma^2 = 6)$ as good as best *n*-gram smoothing.

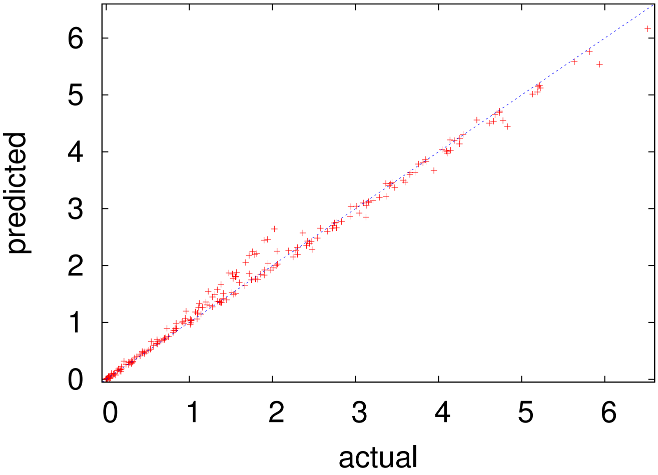
The Law and $\ell_1 + \ell_2^2$ Regularization

$$LL_{\text{test}} - LL_{\text{train}} \approx \frac{0.938}{\textit{(\# train evs)}} \sum_{i=1}^{F} |\lambda_i|$$



The Law and ℓ_2^2 Regularization

$$LL_{\text{test}} - LL_{\text{train}} \approx \frac{0.882}{\text{(\# train evs)}} \sum_{i=1}^{F} |\lambda_i|$$



Outline

Why Exponential Models Are Special

- Do some math (and include normalization features):

$$LL_{\text{test}} - LL_{\text{train}} = \frac{1}{\textit{(\# train evs)}} \sum_{i=1}^{F'} \lambda_i \times \textit{(discount of } f_i(\cdot)\textit{)}$$

- Compare this to The Law:

$$LL_{\text{test}} - LL_{\text{train}} \approx \frac{1}{\textit{(\# train evs)}} \sum_{i=1}^{F} |\lambda_i| \times 0.938$$

- If only . . .

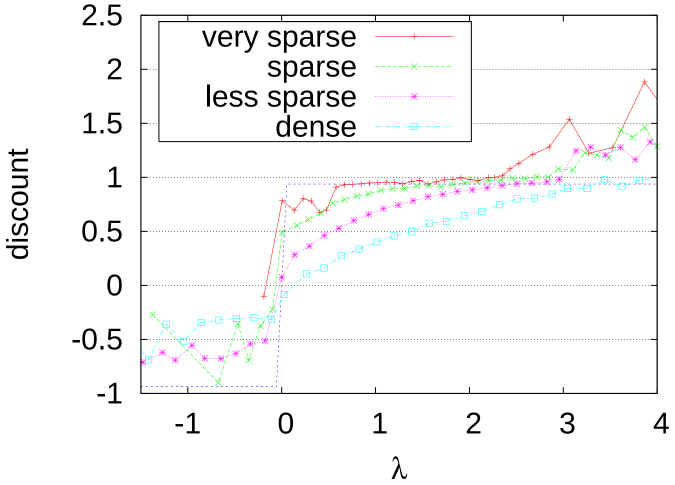
$$\textit{(discount of } f_i(\cdot)\textit{)} \approx 0.938 \times \operatorname{sgn} \lambda_i$$

What Are Discounts?

- How many times fewer an *n*-gram occurs in test set . . .
  - Compared to training set (of equal length).
- Studied extensively in language model smoothing.
- Let's look at the data.

Smoothed Discount Per Feature

$$(\textit{discount of } f_i(\cdot)) \overset{?}{\approx} 0.938 \times \operatorname{sgn} \lambda_i$$



Why The Law Holds More Than It Should

- Sparse models all act alike.
- Dense models don't overfit much.

$$LL_{\text{test}} - LL_{\text{train}} \approx \frac{0.938}{(\#\ \textit{train evs})} \sum_{i=1}^{F} |\lambda_i|$$

Outline

Explain Things

- Why backoff features help.
- Why word class features help.
- Why domain adaptation helps.
- Why increasing n doesn't hurt.
- Why relative performance differences shrink with more data.

Make Models Better

$$\textit{(test error)} \approx \textit{(training error)} + \textit{(overfit)}$$

- Decrease overfit $\Rightarrow$ decrease test error.

Reducing Overfitting

$$\textit{(overfit)} \approx \frac{0.938}{\textit{(\# train evs)}} \sum_{i=1}^{F} |\lambda_i|$$

- In practice, the number of features matters not!
- More features lead to less overfitting . . .
  - If sum of parameters decreases!

A Method for Reducing Overfitting

- Before: $\lambda_1 = \lambda_2 = 2$.

$$P_{\text{before}}(y|x) = \frac{\exp(2 \cdot f_1(x,y) + 2 \cdot f_2(x,y))}{Z_\Lambda(x)}$$

- After: $\lambda_1 = \lambda_2 = 0$, $\lambda_3 = 2$, $f_3(x,y) = f_1(x,y) + f_2(x,y)$.

$$\begin{aligned} P_{\text{after}}(y|x) &= \frac{\exp(2 \cdot f_3(x,y))}{Z_\Lambda(x)} \\ &= \frac{\exp(2 \cdot f_1(x,y) + 2 \cdot f_2(x,y))}{Z_\Lambda(x)} \end{aligned}$$

What's the Catch? (Part I)

- Same test set performance?
- Re-regularize model: improves performance more!

$$\textit{(obj fn)} \equiv LL_{\text{train}} + \alpha \sum_{i=1}^{F} |\lambda_i| + \frac{1}{2\sigma^2} \sum_{i=1}^{F} \lambda_i^2$$

What's the Catch? (Part II)

- Select features to sum in hindsight?
- When sum features, sums discounts!

$$LL_{\text{test}} - LL_{\text{train}} = \frac{1}{\textit{(\# train evs)}} \sum_{i=1}^{F'} \lambda_i \times \textit{(discount of } f_i(\cdot)\textit{)}$$

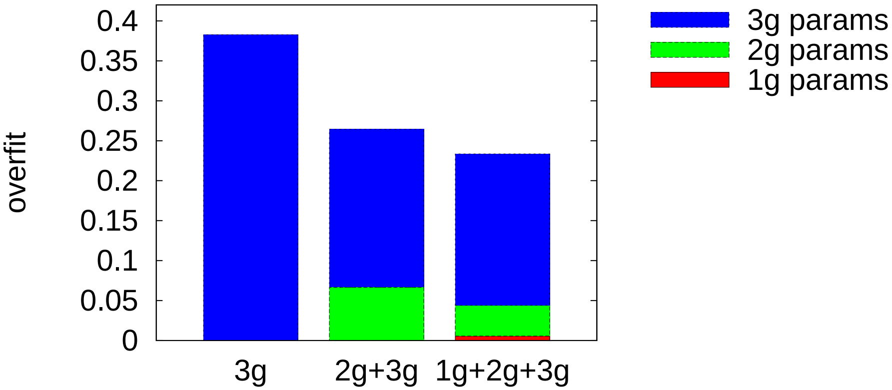
- Need to pick features to sum a priori!

Heuristic 1: Improving Model Performance

- Identify features a priori with similar λ_i.
- Create new feature that is sum of original features.

Example: *N*-Gram Models and Backoff

- $\lambda_{w_{j-2}w_{j-1}w_j}$, $\lambda_{w'_{j-2}w_{j-1}w_j}$ tend to be alike $\Rightarrow$ create $\lambda_{w_{j-1}w_j}$!?
- Bigram features reduce overfitting for trigram features.



Example: *N*-Gram Models and Word Classes

- Group related words into classes, e.g., $\{Monday, Tuesday, \ldots\}$
  - Add class *n*-gram features to address sparsity.
- Problem: space of word/class *n*-gram features is large.

$$c_{j-2}c_{j-1}c_j;\ \ w_{j-2}w_{j-1}c_j;\ \ w_{j-1}c_jw_j;\ \ \ldots$$

- Apply Heuristic 1 to word *n*-gram model!

Goldilocks and the Three Class-Based LM's

- Model S

$$p(c_j \mid c_{j-2} c_{j-1})$$
$$p(w_j \mid c_j)$$

- Model M (Heuristic 1)

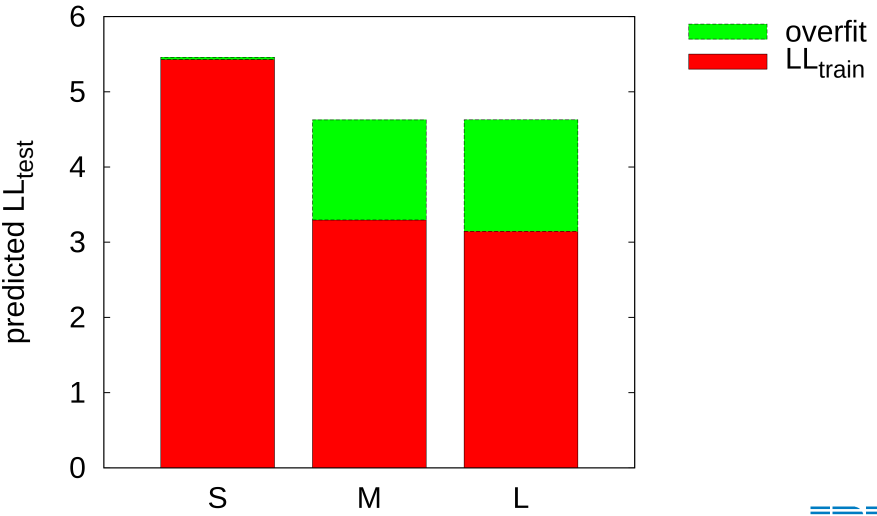
$$p(c_j \mid c_{j-2} c_{j-1}) \times p(c_j \mid w_{j-2} w_{j-1})$$
$$p(w_j \mid w_{j-2} w_{j-1} c_j)$$

- Model L

$$p(c_j \mid w_{j-2} c_{j-2} w_{j-1} c_{j-1})$$
$$p(w_j \mid w_{j-2} c_{j-2} w_{j-1} c_{j-1} c_j)$$

IBM

This One Is Just Right!



Model M

- Best class-based model results for speech recognition . . .
  - Over a wide range of data sets; training set sizes.
- Gains up to 3% absolute in error rate over word *n*-gram.

Outline

Long Live Big Models!

$$\textit{(test error)} \equiv \textit{(training error)} + \textit{(overfit)}$$

$$\textit{(overfit)} \approx \frac{0.938}{\textit{(\# train evs)}} \sum_{i=1}^{F} |\lambda_i|$$

- Despite theory, models with lots of parameters perform well!
- Adding the right parameters can lower overfitting!
  - Heuristic 1.

Applicability to Other Domains

- Log likelihood *vs.* error rate.
- Log-linear models

$$LL_{\text{test}} - LL_{\text{train}} = \frac{1}{\textit{(\# train evs)}} \sum_{i=1}^{F'} \lambda_i \times \textit{(discount of } f_i(\cdot)\textit{)}$$

  - It's not the number of parameters ...
  - It's the size of the parameters!
- Explain and/or enhance existing practice?
  - *e.g.*, backoff features; class-based features.
  - Sometimes the space of feature types is large.

For More Details

- Stanley F. Chen.
  2008.
  Performance prediction for exponential language models.
  Tech. Report RC 24671, IBM Research Division, October.
- Stanley F. Chen, Lidia Mangu, Bhuvana Ramabhadran, Ruhi Sarikaya, Abhinav Sethy.
  2009.
  Scaling shrinkage-based language models.
  In *Proceedings of ASRU*.
- Stanley F. Chen, Stephen M. Chu.
  2010.
  Enhanced Word Classing for Model M.
  Submitted to *Proceedings of Interspeech*.