

IRDS: Evaluation, Debugging, and Diagnostics

Charles Sutton
University of Edinburgh

Evaluation

Automatic evaluation is key to applied AI.

- Evaluation data
 - Validation set
 - Cross-validation
 - But what to do for time series data, network data, etc?
- Evaluation metrics
 - Accuracy
 - Precision, recall, F-measure
 - ROC / AUC curves
 - More complex measures for “partial credit”
- Mostly, we won't discuss this (because other courses should)

Debugging ML

- You build a classifier (e.g., a spam filter using logistic regression) and the error is too high.
- What do you do to fix it? There are lots of things you could try:
 - Collect more training data.
 - Add different features (e.g., from the email header)
 - Try fewer features (e.g., exclude rare words from the classifier)
 - Try an SVM instead of logistic regression
 - Fix a bug in your stochastic gradient descent procedure
- You could do trial and error, but better is to think of **diagnostics**

Bias-Variance Tradeoff

Let $\theta \in \mathbb{R}$ be some quantity we estimate by a random variable $\hat{\theta}$
Example:

$$\theta = \int xp(x)dx \quad \hat{\theta} = \frac{1}{N} \sum_{i=1}^N x_i \quad x_1 \dots x_N \sim p$$

Define the *bias* and *variance*

$$\text{Bias}(\hat{\theta}) = E(\theta - \hat{\theta}) = \theta - \mu$$

$$\text{Var}(\hat{\theta}) = E \left[\hat{\theta} - \mu \right]^2$$

$$\text{where } \mu = \iiint \hat{\theta} p(x_1, \dots, x_N) dx_1 \dots dx_N$$

Both are averages across all data sets that we might have seen.

FUN to look up: Bias-variance decomposition

Useful concepts more generally. These trade off...

Bias-Variance Tradeoff

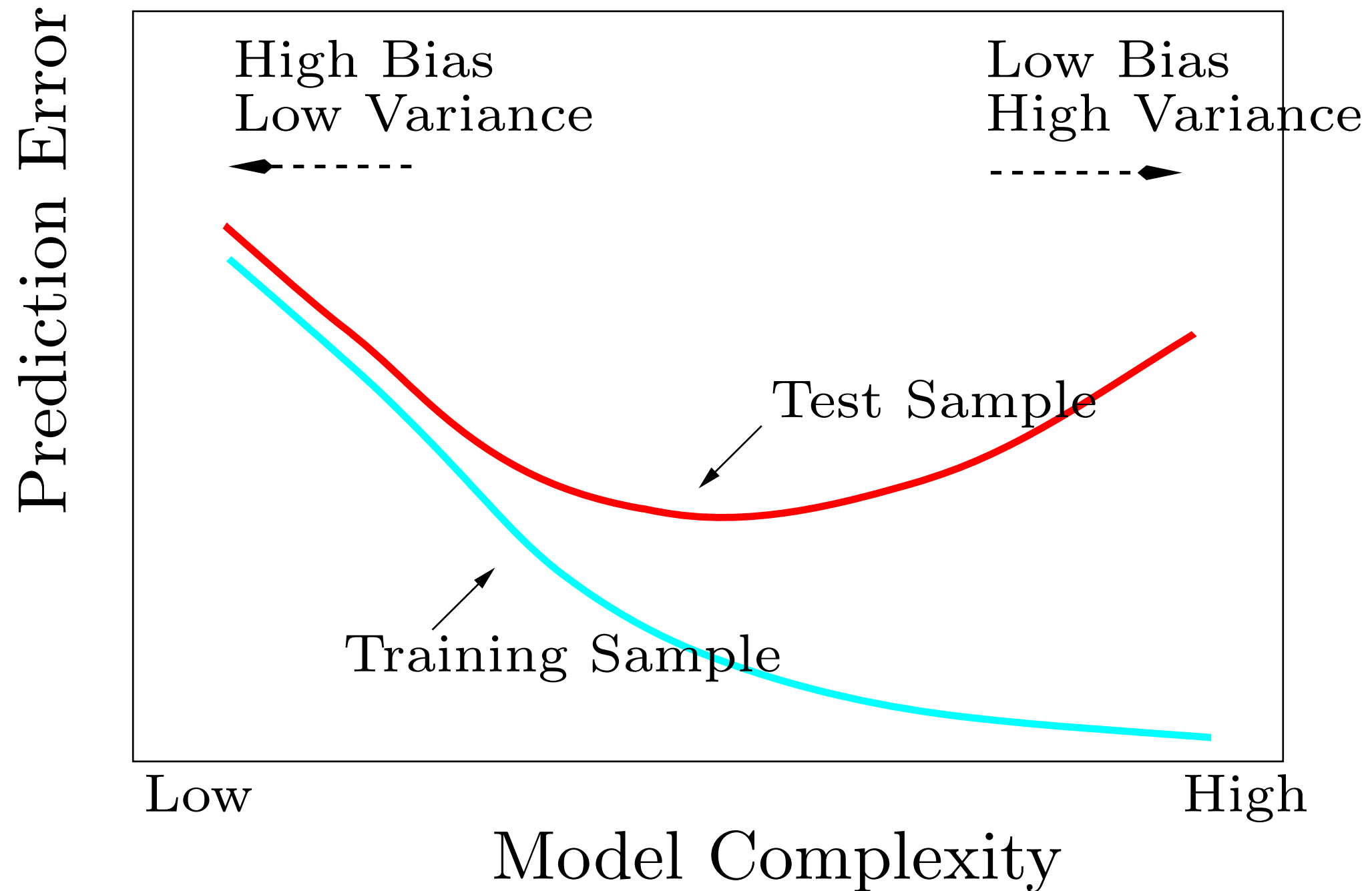
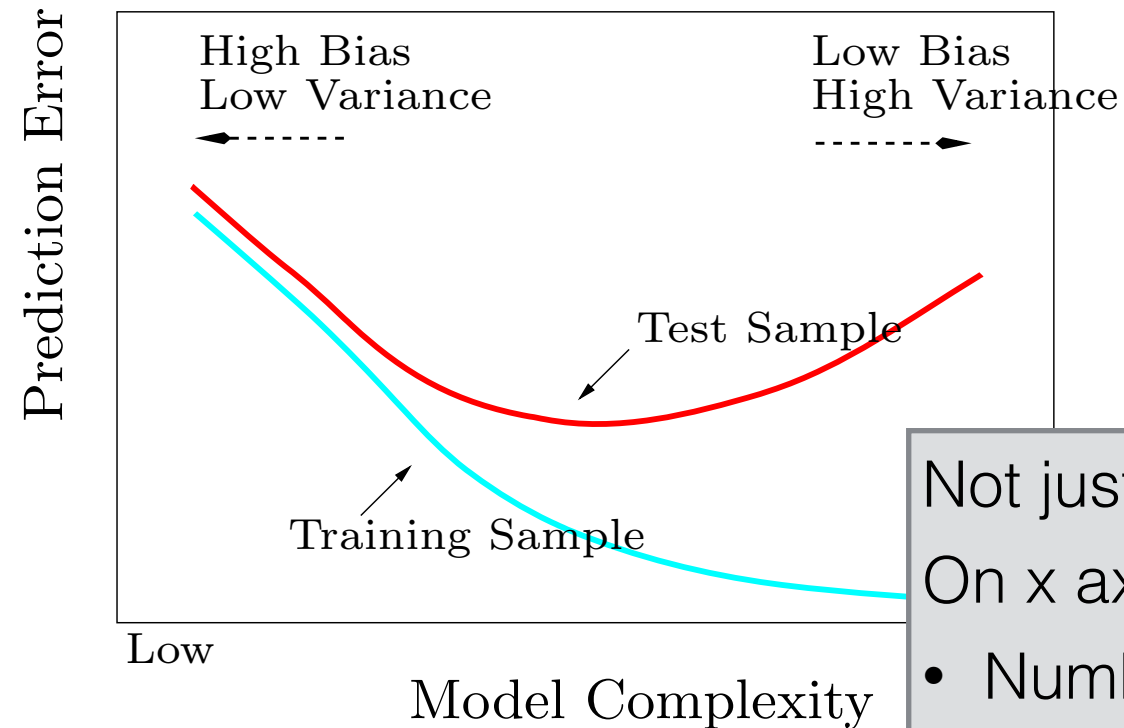


Figure from [Hastie, Tibshirani, and Friedman, 2009]

Bias-Variance Tradeoff

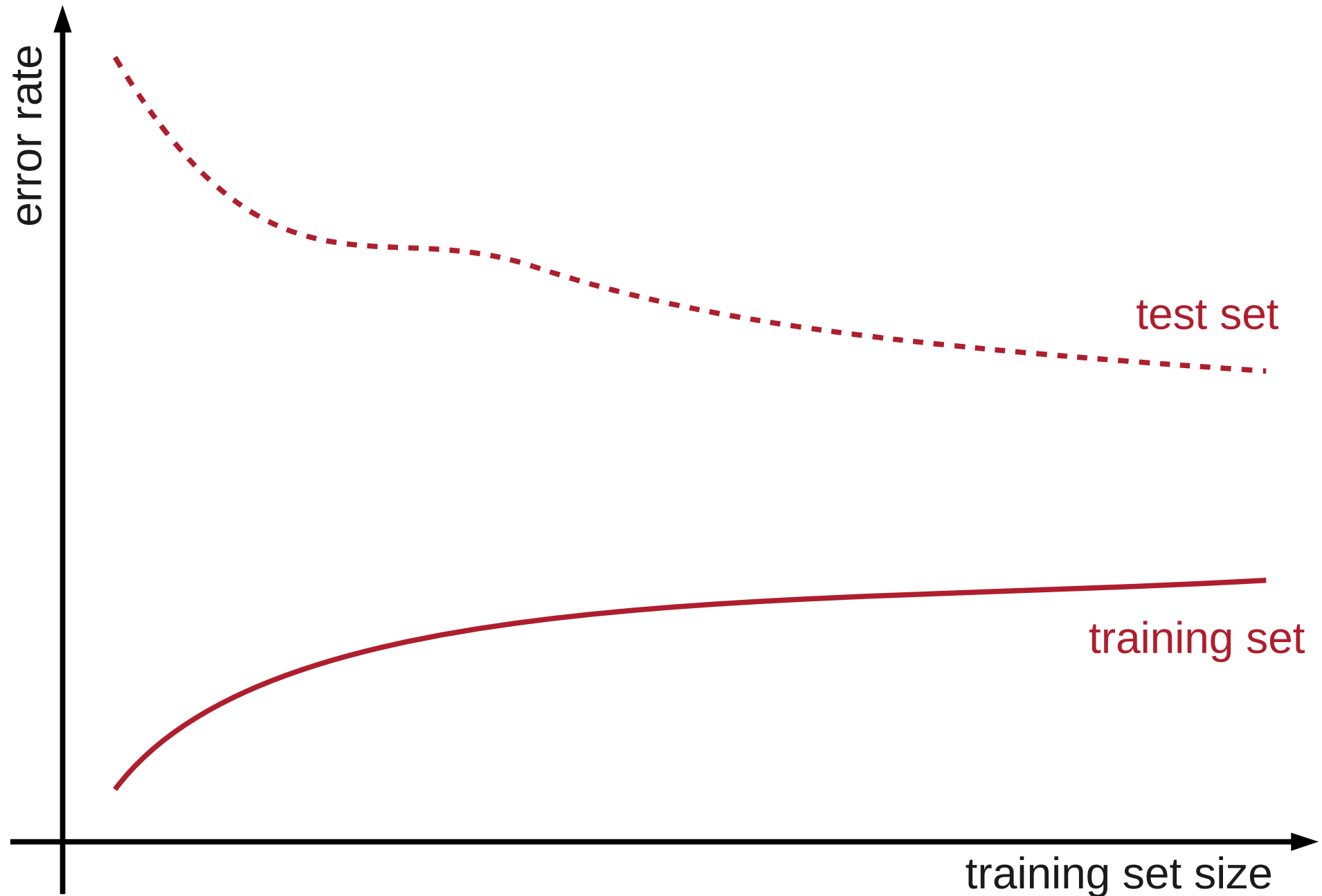


Not just a cartoon. Can use as a diagnostic.

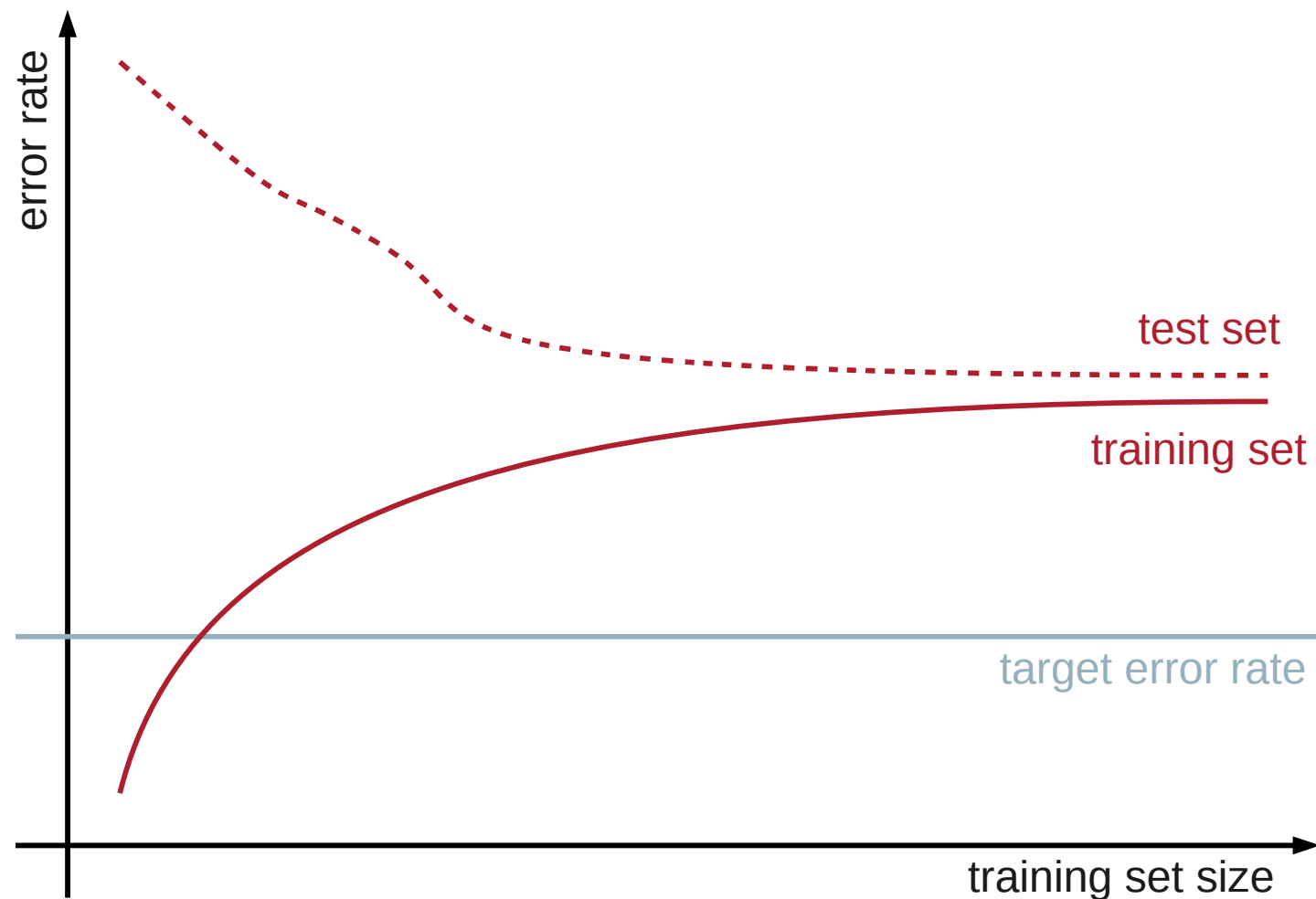
On x axis could put

- Number of features
 - (sort in some meaningful way)
- Model parameter that controls complexity
 - k in k-nearest neighbour
 - number of trees in boosting, random forests
 - regularization parameters
- Or perhaps you have access to more complex models
 - e.g., naive Bayes versus HMM

Learning Curves



Learning Curve Example 1



(Q: Why is error going up?)

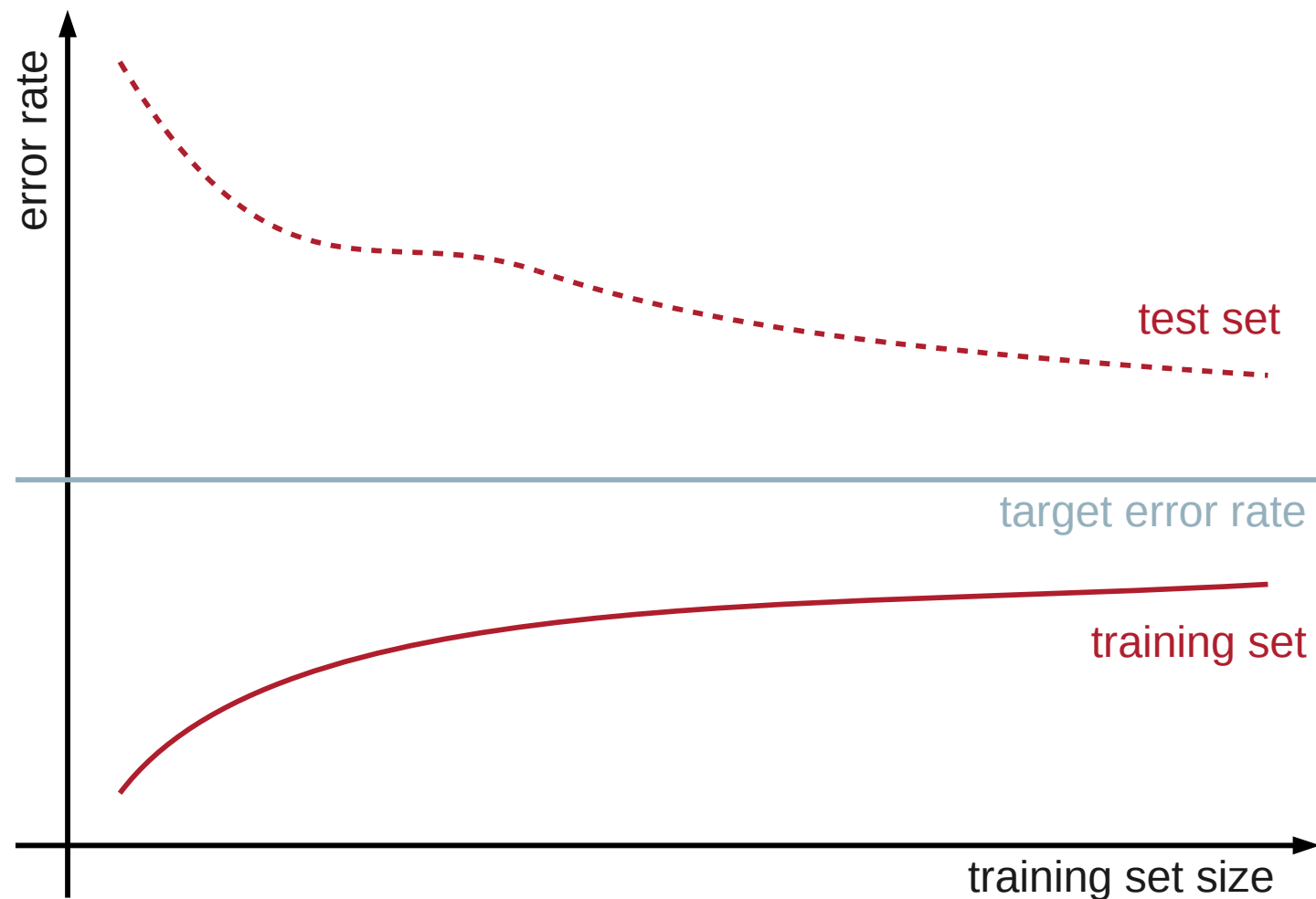
Test error no longer decreasing

Even training error is too high

Not much difference between training and test error

high bias

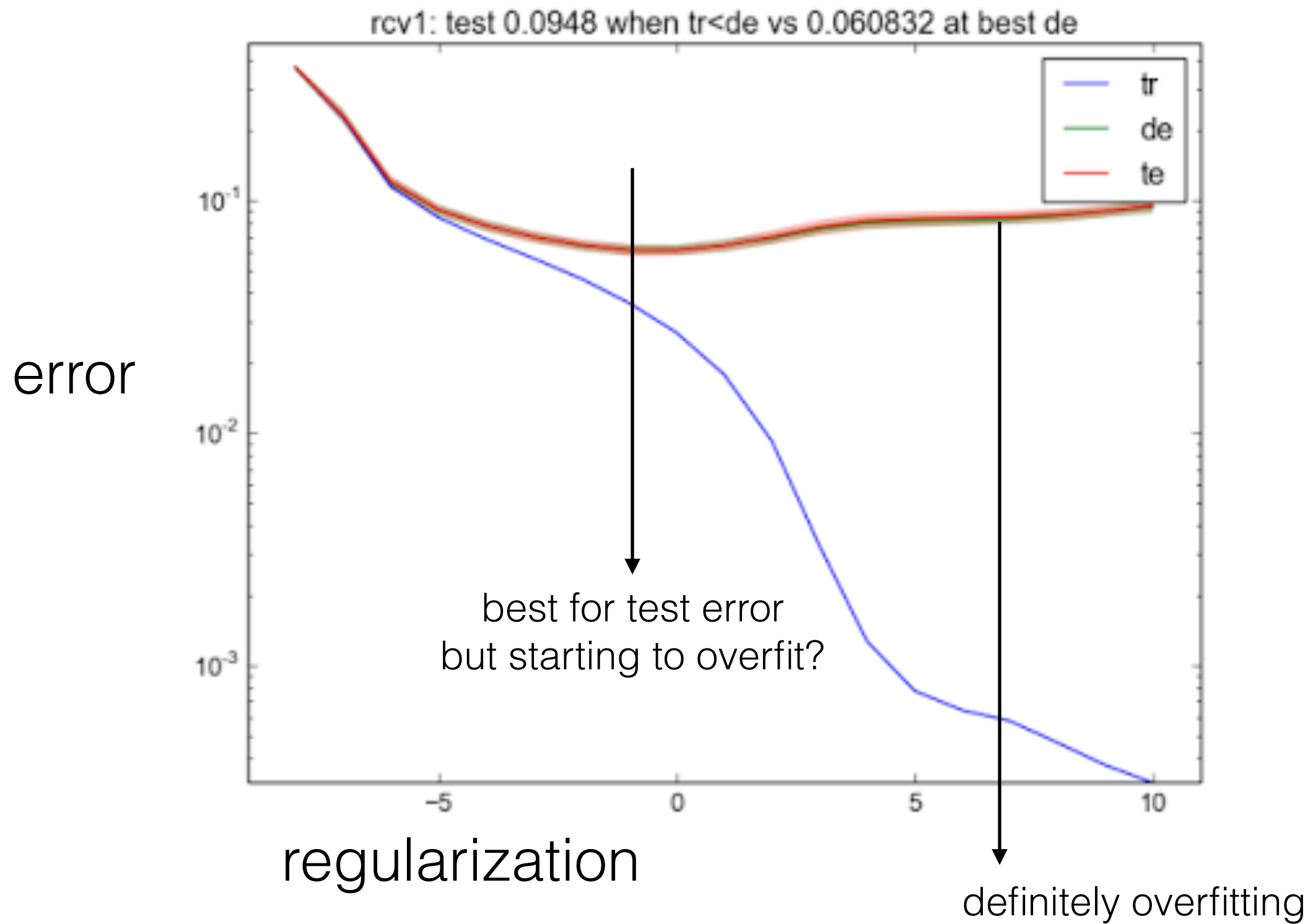
Learning Curve Example 2



Test error still decreasing
Big gap training and test error

high variance

Zero Overfitting Not Desirable



Optimization in the Loop

- Often learning methods work by optimizing some objective function.
- For example, recall logistic regression

$$p(y = 1|\mathbf{x}) = \frac{1}{1 + \exp\{-\mathbf{w}^\top \mathbf{x}\}}$$

- To learn the weights, we solve

$$\max_{\mathbf{w}} L(\mathbf{w}) = \sum_{i=1}^N \log p(y = y^{(i)} | \mathbf{x} = \mathbf{x}^{(i)})$$

data points $(\mathbf{x}^{(i)}, y^{(i)})$ for i in $1 \dots N$

- Maybe optimise this using gradient descent
- When this performs poorly, now have two questions
 - Is my numerical optimization algorithm performing poorly?
 - Or is objective function L not doing what I want?
 - (Simple ex: spam filtering with cost-sensitive error)
- Comes up especially often during **research** in data science
 - Often we introduce new models (== new objective function)
 - Which might be harder to optimize

Optimization Example

Example: To optimize

$$\max_{\mathbf{w}} L(\mathbf{w}) = \sum_{i=1}^N \log p(y = y^{(i)} | \mathbf{x} = \mathbf{x}^{(i)})$$

Simple choice is batch gradient descent:

$$\nabla_{\mathbf{w}} L(\mathbf{w}) = \sum_{i=1}^N \nabla_{\mathbf{w}} [\log p(y = y^{(i)} | \mathbf{x} = \mathbf{x}^{(i)})]$$

This will be slow if N is big.

Alternative: *stochastic gradient descent*.

Simplest version: Sample $i \sim \text{Uniform}(\{1, 2, \dots, N\})$

Compute $\nabla_{\mathbf{w}} \log p(y = y^{(i)} | \mathbf{x} = \mathbf{x}^{(i)})$ (single instance!)

Update using this gradient.

(this is standard in deep learning, e.g.)

Optimization Diagnostic

- You run a logistic regression spam filter on 100,000 training instances. \mathbf{w}_{GD}^*
 - Using batch gradient descent, you get an accuracy of 85%
- Not good enough, so you get a larger set of 100,000,000 examples $\mathbf{w}_{\text{SGD}}^*$
 - Batch gradient is too slow, so you switch to SGD
 - Now you only get 80% accuracy (!?!?)

Diagnostic: Check the batch **training** objective

$$L(\mathbf{w}) = \sum_{i=1}^{100\,000\,000} \log p(y = y^{(i)} | \mathbf{x} = \mathbf{x}^{(i)})$$

Compute this for final result of batch GD \mathbf{w}_{GD}^* and SGD $\mathbf{w}_{\text{SGD}}^*$

$$\text{If } L(\mathbf{w}_{\text{SGD}}^*) \leq L(\mathbf{w}_{\text{GD}}^*)$$

then your SGD procedure is screwed up
(maybe try a different step size?)

This kind of thing happens far more generally.

The Numerical Gradient Check

- Often optimization packages require you to implement functions for both

$$\mathbf{w} \mapsto L(\mathbf{w}) \qquad \mathbf{w} \mapsto \nabla_{\mathbf{w}} L$$

- (although automatic differentiation is becoming more popular)
- In that case, check whether

$$\epsilon^{-1} L(\mathbf{w} + \epsilon) - L(\mathbf{w}) \approx \nabla_{\mathbf{w}} L$$

- Easy to have a bug in one function but not the other.
- Do this for different settings of \mathbf{w}
- MATLAB does this automatically if you ask it to...

Nested Models

- Often complicated models *contain* simpler models as a special case. For logistic regression:

$$p(y = 1|\mathbf{x}) = \frac{1}{1 + \exp\{-\mathbf{w}^\top \mathbf{x}\}}$$

- so if $\mathbf{w} = 0$, the distribution over y is be uniform. Is that what happens in your code? If not, bug.
- Another example: a hidden Markov model and a mixture model

$$p(\mathbf{x}, \mathbf{z}) = \prod_t p(x_t|z_t)p(z_t|z_{t-1})$$

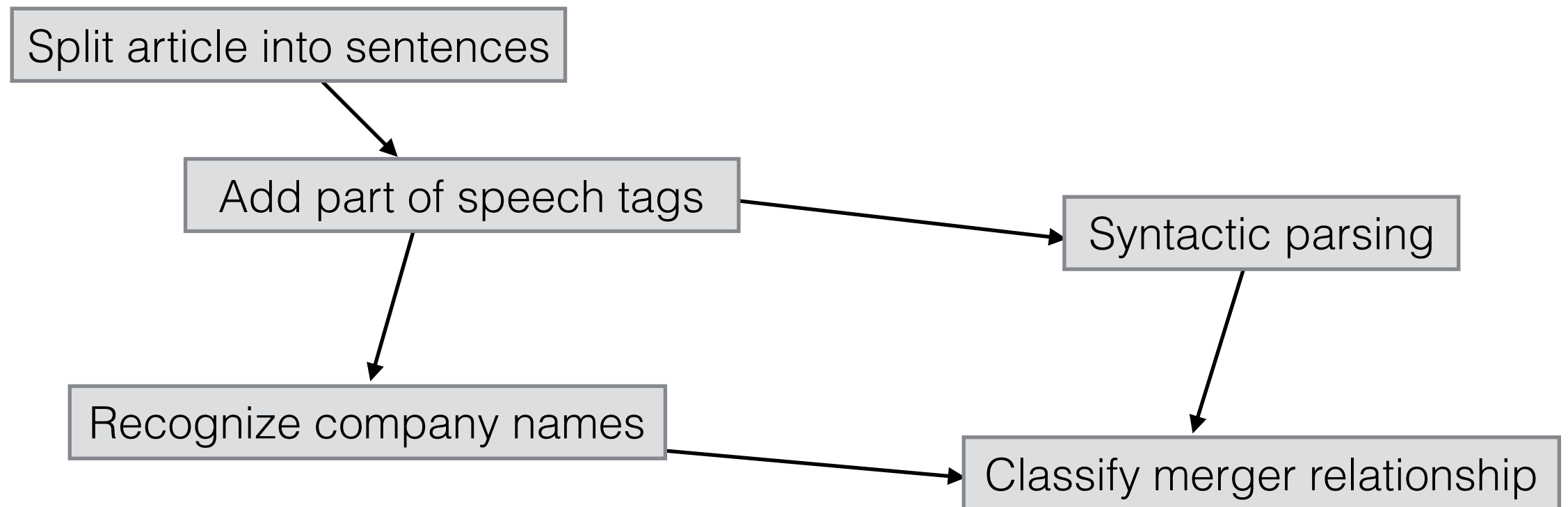
if $p(z_t|z_{t-1})$ ignores z_{t-1} , then all x_t independent ... mixture model

- Lots of ways to get diagnostics from this:
 - Training error of HMM should be strictly better
 - Force your HMM code to fit observation distributions only.
 - Do you get the same distribution as mixture model
 - Logistic regression: Numerical gradient check
 - Try it first at $\mathbf{w}=0$. It will be easier to debug there.

Pipelines of Predictions

Practical systems use predictors at multiple points

e.g., Finding company mergers from newswire text



Many steps rely on learning, will make errors

Is one step a weak link? Or are errors slowly propagating?

Debug by replacing intermediate predictions with gold standard (human annotations)

Overall Advice

- For practical work: Try quick and dirty first. Iterate quickly
- Different diagnostics
 - Learning curves
 - As function of size of training set
 - As function of model complexity
 - Additionally: number of iterations of learning algorithm
 - Optimization diagnostics
 - Diagnostics using model nesting
 - Breaking chains of predictions
- Sometimes diagnostics require a bit of ingenuity.
- “Trust no one”
 - Just because something is true in the maths doesn't mean it is in your code
 - Imagine how you think the method is probably behaving and check whether that happens!
 - (this holds for research too!)