Overfitting

A method is **overfitting** the data when it has a small training MSE but a large test MSE.

Bias-variance tradeoff

Interpretation:

- $Var(\widehat{f})$ is the amount of variability in our predictor with respect to the training data. Increases with increasing model flexibility.
- $Bias(\widehat{f})$ is the systematic error introduced by model approximation. Decreases with increasing model flexibility.
- σ^2 is *irreducible error*, inherent in the error term ϵ. Cannot get rid of this!

If we have a family of flexible regression methods, we should try to balance squared bias and variance.

Summary from today

- Least squares coefficients correspond to minimum of a quadratic surface
- Confidence intervals computed using standard errors of coefficients
- R^2 is a scale-invariant accuracy measure — proportion of variance in Y explained by the model
- Multiple linear regression (many predictors) estimated by solving a linear system — normal equations

Two flavors of classifiers

Generative models model both the input X and the output Y.

Discriminative models model only the output Y given X.

Which one is logistic regression? Which do you think is better?

Generative models

$$p(x_i, y_i) = p(x_i \mid y_i)p(y_i) = p(y_i|x_i)p(x_i).$$

In the generative case we typically estimate the joint distribution by maximizing the *joint likelihood*:

$$\prod_{i=1}^{n} p(x_i, y_i) = \underbrace{\prod_{i=1}^{n} p(x_i \mid y_i)}_{\text{parametric model}} \underbrace{\prod_{i=1}^{n} p(y_i)}_{\text{Bernoulli}}.$$

Discriminative models

$$p(x_i, y_i) = p(x_i \mid y_i)p(y_i) = p(y_i|x_i)p(x_i).$$

In the generative case we typically estimate the joint distribution by maximizing the *conditional likelihood*:

$$\prod_{i=1}^{n} p(x_i, y_i) = \underbrace{\prod_{i=1}^{n} p(y_i \mid x_i)}_{\text{parametric model}} \underbrace{\prod_{i=1}^{n} p(x_i)}_{\text{ignored}}.$$

What did we learn today?

- Classifiers come in two flavors: generative & discriminitive.
- Linear Gaussian discriminant analysis is a simple generative classifier.
- Logistic regression is the discriminative version. Default method.
- Can be fit with iterative, weighted least squared regression.

Improving Upon Gradient Descent

Each step of (batch) gradient descent requires a calculation involving all of the data points.

Stochastic gradient descent, in contrast, only computes based on a smaller subset of the data points (e.g. 1 observation) at each step.

Approaches to feature selection

- Subset selection – use a “good subset” of the p predictors
- Shrinkage – use all p predictors but encourage more coefficients to be near 0
- Dimension reduction – condense the set of predictors by projecting to a lower subspace

Subset selection: Stepwise selection

1. Forward stepwise selection

Starting from the null model, build an increasing sequence of *nested models*.

- Start with $\mathcal{M}_0$.
- For $k = 1, \ldots, p$, pick the best **one** of the remaining unused predictors to add to $\mathcal{M}_{k-1}$ to form $\mathcal{M}_k$.
- Select the best model among $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_p$ on basis of estimated prediction error.

Subset selection: Stepwise selection

2. Backward stepwise selection

Starting from the full model, build a decreasing sequence of *nested models*.

- Start with $\mathcal{M}_p$.
- For $k = p-1, p-2, \ldots, 0$, pick the worst **one** of the existing predictors to remove from $\mathcal{M}_{k+1}$ to form $\mathcal{M}_k$.
- Select the best model among $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_p$ on basis of estimated prediction error.

Backward and forward stepwise selection are more computationally feasible than best subsets, but no guarantee they'll find the best subset of the p predictors to use.

Leave-One-Out Cross-Validation

How do we use more data to train with?

- Use a tiny validation set (e.g. (x_1, y_1))
- Train with the rest (e.g. $\{(x_2, y_2), \ldots, (x_n, y_n)\}$)

How do we feel about error rate evaluated on a single observation?

Not good, but we can iterate through the dataset, each time using a different (x_i, y_i) as the validation set and obtaining an error MSE_i.

k-fold Cross-Validation

A potentially faster approach:

- Randomly divide the dataset into k *folds*.
- For $b = 1, \ldots, k$:
  - Use b-th fold ("batch") as validation set.
  - Use everything else as training set.
  - Compute validation error on b-th fold.
- Estimate test error using:

$$CV_{(k)} = \sum_b \frac{n_b}{n} MSE_b,$$

where n_b is the total # observations in the b-th fold, and n is the total # observations in the entire dataset.

Classification and Regression Trees (CART)

Trees provide alternative ways of modeling nonlinear relationships, and give a **nonparametric** approach that does not require any assumptions about the underlying data.

- Can be used for either classification or regression.
- Feature variables can be categorical or quantitative.
- Yields a set of interpretable decision rules (popular in medicine).
- Predictive ability is often mediocre, *but* can be improved with ideas of resampling (will be covered on Thursday).

Bias vs Variance

- As tree is grown deeper, bias decreases
- But the variance increases
- How to choose the right size of tree?

Option 1: Change the stopping criterion.

What did we learn today?

- Trees are a nonparametric method
- Gives interpretable decision rules
- Shallow trees have high bias and low variance, deep trees have low bias, high variance
- Trees are grown greedily to the full, then pruned back

Ensemble methods

Ensemble methods pool together multiple different models to arrive at more reliable predictions.

To ensure the models are different, we will train each one slightly differently:

- **b**ootstrap **agg**regation (bagging): randomizes training data
- random forests (feature bagging): randomizes training data + randomizes features
- boosting: changes/weights training data

These techniques are **general** and can be applied to other models – today we focus on trees.

Ensemble methods: Pros & Cons

1. Bagging:
   - Advantages: Reduces variance / avoids overfitting. Can be parallelized.
   - Disadvantages: Can suffer from bias. Training base models may be computationally expensive. Not be desirable when base models still highly correlated.
2. Feature bagging:
   - Advantages: Reduces variance / avoids overfitting. Better at de-correlating base models. Can be parallelized.
   - Disadvantages: Can suffer from bias. Training base models may be computationally expensive.
3. Boosting:
   - Advantages: Reduces bias. Training base models is fast.
   - Disadvantages: Not as effective against overfitting. Has to be done sequentially (may be more costly overall).

Unsupervised Learning

Supervised learning is about being able to predict a Y using a series of predictors $X_1, X_2, \ldots, X_p$.

Unsupervised learning deals with data that do not have labels Y.

We are not trying to predict anything. So what else might we hope to do?

Consider:

- Are there interesting ways to visualize/summarize the data?
- Are there natural subgroups in the data?

PCA: Summary

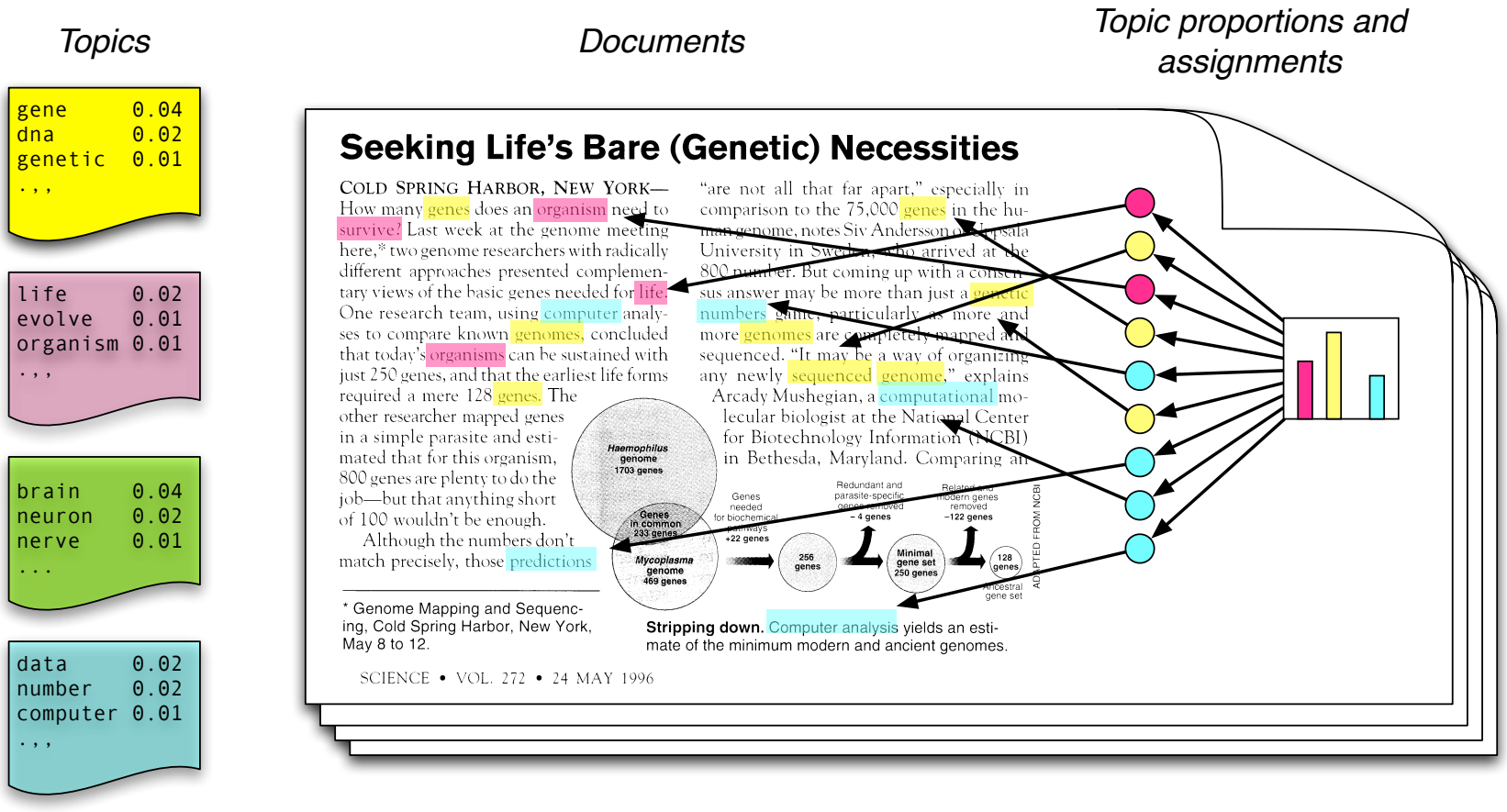
- PCA is an unsupervised method
- Finds directions of greatest variation in the data
- The directions are called the *principal vectors*; the weightings on the vectors are called the *principal components*
- The first few vectors may be interpretable
- Orthogonality makes interpretation difficult for the higher components
- Can be used for visualization or dimensionality reduction
- Let's go to the notebook!

Bayesian Inference

The parameter θ of a model is viewed as a random variable. Inference usually carried out as follows:

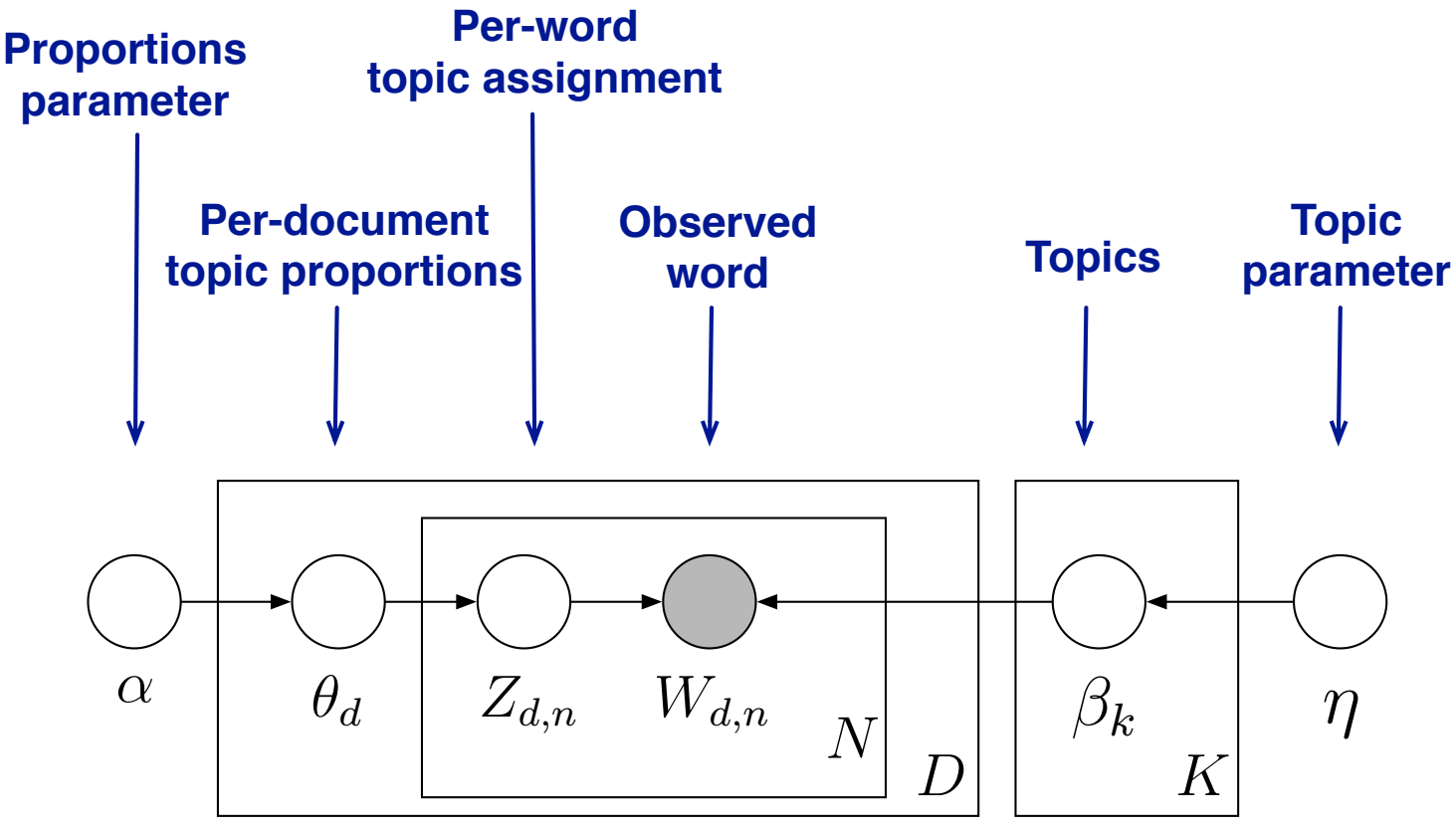
- Choose a *generative model* $p(x \mid \theta)$ for the data.
- Choose a *prior distribution* $\pi(\theta)$ that expresses beliefs about the parameter before seeing any data.
- After observing data $\mathcal{D}_n = \{x_1, \ldots, x_n\}$, update beliefs and calculate the *posterior distribution* $p(\theta \mid \mathcal{D}_n)$.

Generative model for LDA



- Each **topic** is a distribution over words
- Each **document** is a mixture of corpus-wide topics
- Each **word** is drawn from one of those topics

LDA as a graphical model



- Nodes are random variables; edges indicate dependence.
- Shaded nodes are observed.
- Plates indicate replicated variables.

Language models

- A language model is a way of assigning a probability to any sequence of words (or string of text)

$$p(w_1, \ldots, w_n)$$

- By the basic rules of conditional probability we can factor this as

$$p(w_1, \ldots, w_n) = p(w_1)p(w_2 \mid w_1) \ldots p(w_n \mid w_1, \ldots, w_{n-1})$$

- The number of *histories* grows as V^{n-1}. Number of parameters in model grows as V^n, where V is number of words in vocabulary.
- What are some ways of reducing the number of parameters?

Class-based bigram model

- Model takes form

$$\begin{aligned} p(w_2 \mid w_1) &= p(\text{class}(w_2) \mid \text{class}(w_1))\, p(w_2 \mid \text{class}(w_2)) \\ &= p(c_2 \mid c_1)\, p(w_2 \mid c_2) \end{aligned}$$

- Use bottom-up agglomerative clustering to group the words.
- In each step, merge the pair of classes that gives the smallest reduction in likelihood of the data. (The MLE bigram model has the greatest likelihood.)
- $O(V^5)$ complexity to go down to $O(1)$ classes.
- V is number of words in vocabulary

Brown et al., "Class-based *n*-gram models of natural language"

Perplexity

"perplexity" is evaluated as

$$\text{Perplexity}(\theta) = \left(\prod_{i=1}^{N} p_\theta(w_i \mid w_{1:i-1}) \right)^{-1/N}$$

If perplexity is 100, then the model predicts, on average, as if there were 100 equally likely words to follow.

Pointwise mutual information (PMI)

Related statistic is "pointwise mutual information" (PMI)

$$\log\left(\frac{p_{\text{near}}(w_1, w_2)}{p(w_1)p(w_2)}\right)$$

- How likely are specific words/clusters to co-occur together within some window, compared to if they were independent?

Core idea of embeddings

- Form a language model but replace classes by vectors, one for each word
- Use PMI-like scores to fit the vectors
- Can be applied whenever have cooccurrence data.

Analogies

These heuristics enable training on very large text collections. Leads to vector representations of words with interesting properties.

For example, analogies:

$\texttt{king}$ is to $\texttt{man}$ as ? is to $\texttt{woman}$

$\texttt{Paris}$ is to $\texttt{France}$ as ? is to $\texttt{Germany}$

$$\phi(\texttt{king}) - \phi(\texttt{man}) \stackrel{?}{\approx} \phi(\texttt{queen}) - \phi(\texttt{woman})$$

$$\widehat{w} = \arg\min_{w} \|\phi(\texttt{king}) - \phi(\texttt{man}) + \phi(\texttt{woman}) - \phi(w)\|^2$$

Does $\widehat{w} = \texttt{queen}$?

Embedding embeddings: t-SNE

- How can we visualize the embeddings?
- We're in a very high dimensional space
- Can do PCA, but this will introduce an additional projection/approximation step
- Many visualization techniques exist. A currently popular one is t-SNE: "Student-t Stochastic Neighborhood Embedding"

Summary: Word embeddings

- Word embeddings are vector representations of words, learned from cooccurrence statistics
- The models can be viewed in terms of logistic regression and class-based bigram models
- Surprising semantic relations are encoded in linear relations
- Various heuristics have been introduced to get scalability
- Embeddings improve with more data
- t-SNE is an algorithm for visualizing embeddings

Summary: What did we learn today?

- Autoencoders compress the input and then reconstruct it
- Bottleneck forces extraction of useful features
- Will overfit and “memorize” the data
- Overfitting mitigated by denoising autoencoders
- More fundamental view: Latent variable generative models and posterior inference
- Parameterize variational parameters in terms of neural networks
- Reparameterization trick allows simultaneous training of both networks