Avoiding Overfitting
&
Exact Bayesian Models

CMPUT 366: Intelligent Systems

P&M §10.4

Assignment #2

- **Assignment #2 was released on Friday**
  See eClass
- Due **Friday, February 28 at 11:59pm**
- This week's lab: **Thursday, 5:00pm to 8:00pm**, BS M 149 (this room!)
  - Not mandatory
  - You can get help from the TAs on your assignment in labs

Recap: Overfitting

- **Overfitting** is when a learned model fails to **generalize** due to **overconfidence** and/or learning **spurious regularities**
- **Bias-variance tradeoff**: More **complex** models can be more **accurate**, but also require more **data** to train

Lecture Outline

1. Recap & Logistics
2. Avoiding Overfitting
3. Model Probabilities
4. Using Model Probabilities
5. Prior Distributions as Bias

Avoiding Overfitting

There are multiple approaches to avoiding overfitting:

1. **Pseudocounts**: Explicitly account for **regression to the mean**
2. **Regularization**: Explicitly **trade off** between fitting the data and model complexity
3. **Cross-validation**: **Detect** overfitting using some of the training data

Pseudocounts

- When we have not observed all the **values** of a variable, those variables should not be assigned **probability zero**
- If we don't have very much **data**, we should not be making very extreme predictions (**why?**)
- **Solution:** artificially add some "pretend" observations for each value of a variable (**pseudocounts**)
  - When there is not much data, predictions will tend to be less extreme (**why**?)
  - When there is more data, the pseudocounts will have less effect on the predictions

Regularization

- We shouldn't choose a **complicated** model unless there is **clear evidence** for it
- Instead of optimizing directly for training error, optimize training error plus a **penalty** for complexity:

$$\arg\min_{h \in \mathcal{H}} \sum_{e} error(e, h) + \lambda \times regularizer(h)$$

- *regularizer* measures the **complexity** of the hypothesis
- λ is the **regularization parameter**: indicates how important hypothesis complexity is compared to fit
  - Larger λ means complexity is more important

Types of Regularizer

- Number of **parameters**
- **Degree** of polynomial
- **L2** regularizer ("ridge regularizer"): sum of squares of weights
  - Prefers models with **smaller** weights
- **L1** regularizer ("lasso regularizer"): sum of absolute values of weights
  - Prefers models with **fewer nonzero** weights
  - Often used for **feature selection**: only features with nonzero weights are used

Cross-Validation

- Previous methods require us to already know how simple a model "should" be:
  - How many **pseudocounts** to add?
  - What should **regularization parameter** be?
- Ideally we would like to be able to answer these questions **from the data**
- **Question:** Can we use the **test data** to see which of these work best?
- **Idea:** Use some of the **training data** as an **estimate** of the test data

Cross-Validation Procedure

Cross-validation can be used to estimate most bias-control parameters (**hyperparameters**)

1. **Randomly remove** some datapoints from the training set; these examples are the **validation set**
2. **Train** the model on the training set using some values of hyperparameters (pseudocounts, polynomial degree, regression parameter, etc.)
3. **Evaluate** the results on the validation set
4. Update values of **hyperparameters**
5. Repeat

k-Fold Cross-Validation

- We want our **training set** to be as large as possible, so we get better models
- We want our **validation set** to be as large as possible, so that it is an accurate estimation of test performance
- When one is larger, the other must be **smaller**
- ***k*-fold cross-validation** lets us use every one of our examples for both validation and training

k-Fold Cross-Validation Procedure

1. **Randomly partition** training data into k approximately equal-sized sets (**folds**)
2. Train k times, each time using all the folds but one; **remaining fold** is used for **validation**
3. Optimize hyperparameters based on **validation errors**

- Each example is used exactly **once** for **validation** and $k - 1$ **times** for **training**
- **Extreme case**: $k = n$ is called **leave-one-out** cross-validation

Overfitting Summary

- **Overfitting** is when a learned model fails to **generalize** due to **overconfidence** and/or learning **spurious regularities**
- **Bias-variance tradeoff**: More **complex** models can be more **accurate**, but also require more **data** to train
- Techniques for avoiding overfitting:
  1. **Pseudocounts**: Add **imaginary** observations
  2. **Regularization**: **Penalize** model complexity
  3. **Cross-validation**: Reserve **validation data** to estimate test error

Exact Bayesian Models

CMPUT 366: Intelligent Systems

P&M §10.4

Learning Point Estimates

- So far, we have considered how to find the best **single** model, e.g.,
  - learn ***a*** decision tree
  - optimize ***the*** weights of a linear or logistic regression
- The **predictions** might be a probability distribution, but they are coming out of a single **model**:

$$P(Y \mid X)$$ Probability of target Y given observation X

- We have been learning **point estimates** of our model

Learning Model Probabilities

- Instead, we could learn a distribution over **models**:
  - $\Pr(X, Y \mid \theta)$ Probability of target Y and features X given model θ
  - $\Pr(\theta \mid D)$ Probability of model θ given dataset D
- This is called **Bayesian learning**: we never discard any model, we only weight them differently depending upon their **posterior probability**
- **Question:** Why would we want to do that?

What is a Model?

- $\Pr(X, Y \mid \theta)$ Probability of target Y and features X given model θ
- $\Pr(\theta \mid D)$ Probability of model θ given dataset D

- We can do Bayesian learning over **finite** sets of models:
  - e.g., { rank by feature $\theta \mid \theta \in$ {height, weight, age} }
- We can do Bayesian learning over **parametric families** of models:
  - e.g., { regression with weights $w_0=\theta_1$, $w_1=\theta_2 \mid \theta \in \mathbb{R}^2$ }
- We can mix the two!
  - θ can encode choice of **model family and parameters**

What is the Dataset?

- $\Pr(X, Y \mid \theta)$ Probability of target Y and features X given model θ
- $\Pr(\theta \mid D)$ Probability of model θ given dataset D

- We have an expression for the probability of a single example given a model: $\Pr(X, Y \mid \theta)$
- **Question:** What is the expression for the probability of a dataset of observations $D = \{(X_1, Y_1), \ldots, (X_m, Y_m)\}$ given a model?
  - Easiest approach: Assume that the dataset independent, identically distributed observations: $(X_i, Y_i) \sim P(X, Y \mid \theta)$

$$\Pr(D \mid \theta) = \Pr(X_1, Y_1 \mid \theta) \times \ldots \times \Pr(X_m, Y_m \mid \theta)$$

$$= \prod_{i=1}^{m} \Pr(X_i, Y_i \mid \theta)$$

What is the Posterior Model Probability?

- $\Pr(X, Y \mid \theta)$ Probability of target Y and features X given model θ
- $\Pr(\theta \mid D)$ Probability of model θ given dataset D

Now we can use **Bayes' Rule** to compute the posterior probability of a model θ:

Prior probability of model θ

$$\Pr(\theta \mid D) = \frac{\Pr(D \mid \theta)\Pr(\theta)}{\Pr(D)}$$

Likelihood of data D given model θ

$$= \frac{\prod_i \Pr(X_i, Y_i \mid \theta)\Pr(\theta)}{\Pr(D)}$$

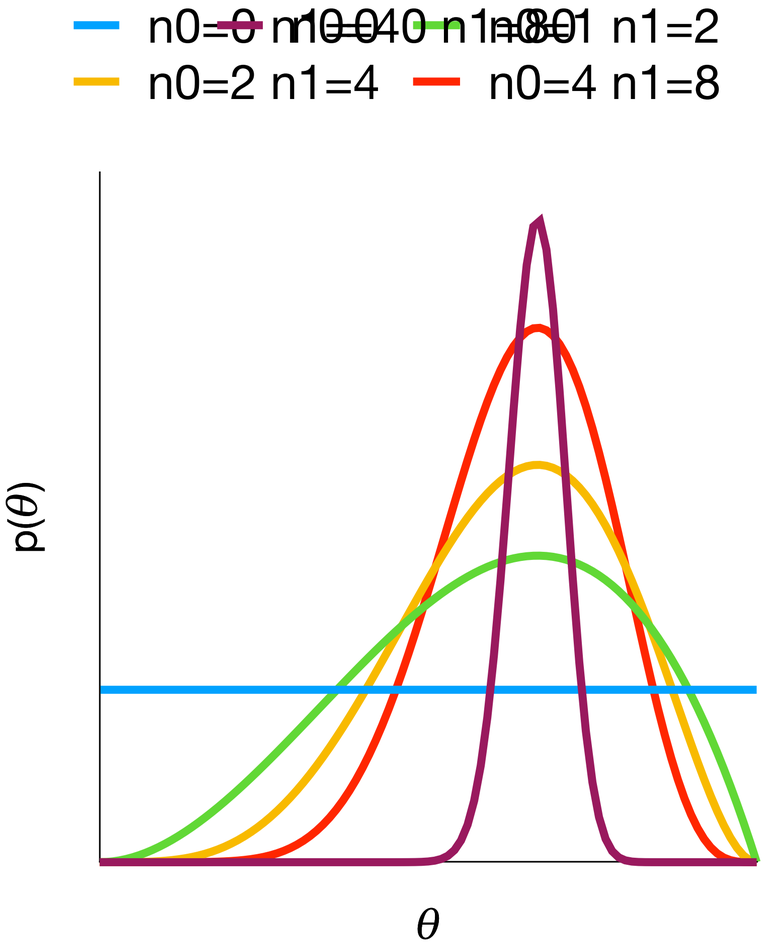
$$= \frac{\prod_i \Pr(X_i, Y_i \mid \theta)\Pr(\theta)}{\sum_{\theta'} \Pr(D \mid \theta')\Pr(\theta')}$$

Example: Biased Coin

- Back to coin flipping! We can flip a coin and observe heads or tails, but we don't know the coin's bias
- Model: **Binomial observations**
  - Observations: $Y \in \{h, t\}$
  - Bias: $\theta \in [0,1]$
  - Likelihood: $\Pr(H \mid \theta) = \theta$
  - **Question:** What should the **prior** $\Pr(\theta)$ be?

Biased Coin: Posterior Probabilities

- Before we see any flips, all biases are equally probable (according to our prior)
- After more and more flips, we become more confident in θ
- θ with **highest probability** is 2/3
  - **Expected** value of θ is less! (**why**?)
  - But with more observations, mode and expected value get **closer**



Beta-Binomial Models

- Likelihood: $P(h \mid \theta) = \theta$
  - aka Bernoulli$(h \mid \theta)$
  - Dataset likelihood: $\theta^{n_1} \times (1 - \theta)^{n_0}$
  - aka Binomial(n_1, n_0)
- Prior: $P(\theta) \propto 1$
  - aka Beta$(1,1)$
- Models of this kind are called **Beta-Binomial models**
- They can be solved analytically: $Pr(\theta \mid D) =$ Beta$(1 + n_1, 1 + n_0)$

Conjugate Priors

- The beta distribution is a **conjgate prior** for the binomial distribution:
  - Updating a beta prior with a binomial likelihood gives a **beta posterior**
- Other distributions have this property:
  - Gaussian-Gaussian (for means)
  - Dirichlet-Multinomial (generalization of Beta-Binomial for multiple values)

Using Model Probabilities

So we can estimate Pr(θ | D). What can we do with it?

1. Parameter estimates
2. Target predictions (model averaging)
3. Target predictions (point estimates)

1. Parameter Estimates

- Sometimes, we really want to know the **parameters** of a model itself
- E.g., maybe I don't care about predicting the next coin flip, but I do want to know whether the coin is fair
- Can use $\Pr(\theta \mid D)$ to make statements like

$$\Pr(0.49 \leq \theta \leq 0.51) > 0.9$$

2. Model Averaging

- Sometimes we do want to make **predictions**:

$$\Pr(Y \mid D) = \sum_{\theta} \Pr(Y \mid \theta) \Pr(\theta \mid D)$$

- This is called the **posterior predictive distribution**
- **Question:** How is this different from just learning a point estimate of a model, and then predicting with that model?

3. Maximum A Posteriori

- Sometimes we do want to make predictions, **but...**

$$\Pr(Y \mid D) = \int_0^1 \Pr(Y \mid \theta) \Pr(\theta \mid D) d\theta$$

- the posterior predictive distribution may be **expensive** to compute (or even **intractable**)

- One possible solution is to use the **maximum a posterior** model as a point estimate:

$$\Pr(Y \mid D) \simeq \Pr(Y \mid \hat{\theta}) \qquad \text{where } \hat{\theta} = \arg\max_{\theta} \Pr(\theta \mid D)$$

- **Question:** Why would you do this instead of just using a point estimate that was computed in the usual way?

Prior Distributions as Bias

- Suppose I'm comparing two models, θ_1 and θ_2 such that

$$\Pr(D \mid \theta_1) = Pr(D \mid \theta_2)$$

- **Question:** Which model has higher **posterior probability**?
- Priors are a way of encoding **bias**: the tell use which models to prefer when the data doesn't

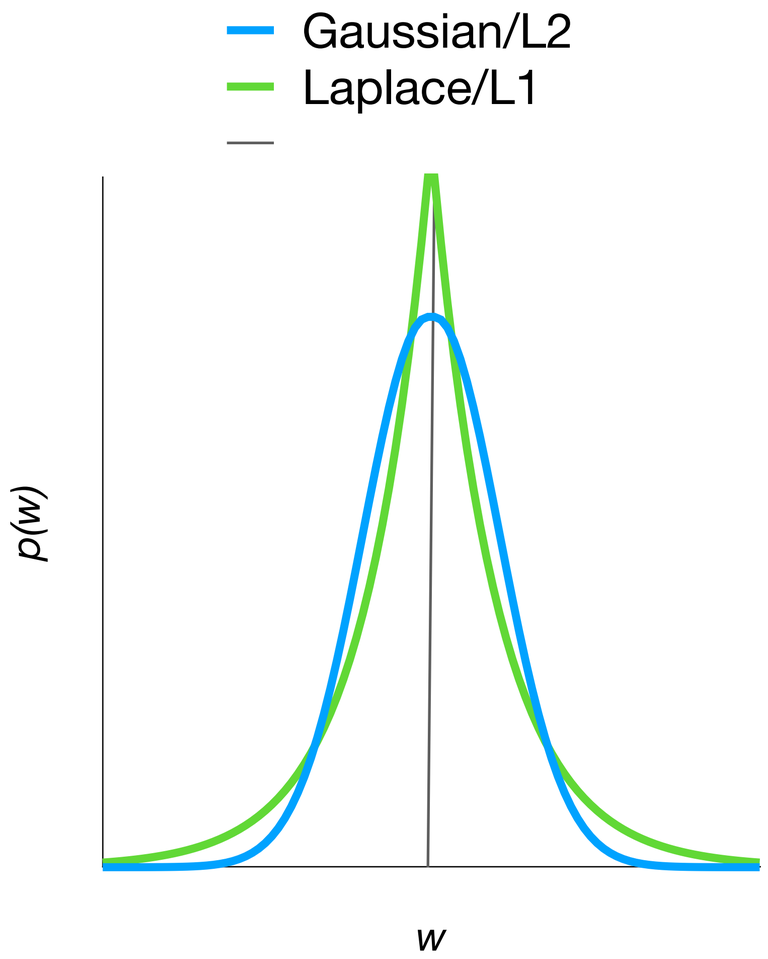
Priors for Pseudocounts

- We can straightforwardly encode pseudocounts as prior information in beta-binomial and dirichlet-multinomial models
- E.g., for pseudocounts k_1 and k_0,

$$p(\theta) = \text{Beta}(1+k_1, 1+k_0)$$

Priors for Regularization

- Some **regularizers** can be encoded as priors also
- **L2 regularization** is equivalent to a **Gaussian** prior on the weights: p(w) = N(w|m,s)
- **L1 regularization** is equivalent to a **Laplacian** prior on the weights: p(w) = exp(|w|)/2



Summary

- In Bayesian Learning, we learn a **distribution** over models instead of a **single model**
- When the model is **conjugate**, posterior probabilities can be computed **analytically**
  - See next lecture for non-conjugate models
- We can make predictions by **model averaging** to compute the **posterior predictive distribution**
- The **prior** can encode **bias over models**, much the same as **regularization**
  - In fact, it can exactly encode certain kinds of regularization