Outline

- Introduce **basis function** to go beyond linear regression
- Understanding the tradeoff between **bias** and **variance**
- Overfitting: What happens when we make models too complex
- Regularization as a means to control model complexity
- Cross-validation to perform model selection
Basis Expansion
Basis Expansion
Basis Expansion

\[ \phi(x) = [1, x, x^2, x^3, x^4] \]

\[ w_1 + w_2 x + w_3 x^2 + w_4 x^3 + w_5 x^5 = \phi(x) \cdot [w_1, w_2, w_3, w_4, w_5] \]
Basis Expansion

\[ \phi(x) = [1, x, x^2, x^3, x^4] \]

\[ w_1 + w_2x + w_3x^2 + w_4x^3 + w_4x^5 = \phi(x) \cdot [w_1, w_2, w_3, w_4, w_5] \]
Basis Expansion

\[ \phi(x) = [1, x, x^2] \]
Basis Expansion

\[ \phi(x) = [1, x, x^2, x^3, \ldots, x^9] \]
Basis Expansion

\[ \phi(x) = [1, x, x^2, x^3, \ldots, x^d] \]

How can we avoid overfitting?
Basis Expansion

\[ \phi(x) = [1, x, x^2, x^3, \ldots, x^d] \]

How can we avoid overfitting?

Does more data help?
Basis Expansion

\[ \phi(x) = [1, x, x^2, x^3, \ldots, x^d] \]

How can we avoid overfitting?

Does more data help?
Basis Expansion

\[ \phi(x) = [1, x, x^2, x^3, \ldots, x^d] \]

How can we avoid overfitting?

Does more data help?
Bias Variance Tradeoff

- For linear model, more data would make little difference

- Bias results from model being simpler than the “truth”

- High bias results in underfitting
What happens when we fit model on different (randomly drawn) training datasets?

- Variance arises when the (complex) model is sensitive to fluctuations in training dataset
- Variance results in overfitting
Bias Variance Tradeoff

- When does more data help?
- Error = Bias^2 + Variance + Noise (Exercise for linear regression)

- For more complex models, difficult to visually overfitting and underfitting
- Keep aside some points as “test set”
Learning Curves

- Suppose we have a training set and test set
- Train on increasing sizes of the training set, and plot the errors

- Once training and test error approach each other, then more data won’t help!
Basis Expansion Using Kernels

We can use kernels as features, e.g., radial basis functions (RBFs)

Feature expansion:

\[ \phi(x) = [1, \kappa(x, \mu_1, \sigma), \ldots, \kappa(x, \mu_d, \sigma)] \]

\[ \text{e.g., } \kappa(x, \mu_i, \sigma) = e^{-\frac{||x-\mu_i||^2}{2\sigma^2}} \]

Model: \[ y = \phi(x)^T w + \text{noise} \]
Basis Expansion Using Kernels

As in the case of polynomials, the width $\sigma$ can cause overfitting or underfitting.

Image Source: K. Murphy (2012)
Polynomial Basis Expansion in Higher Dimensions

We are basically fitting linear models (at the cost of increasing dimensions)

\[ y = \phi(x) + \text{noise} \]

Linear Model: \( \phi(x) = [1, x_1, x_2] \)

Quadratic Model: \( \phi(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2] \)

Image Source: K. Murphy (2012)

How many dimensions do you get for degree \( d \) polynomials over \( n \) variables?
Overfitting!

In high dimensions, we can have many many parameters!

With $100$ variables and degree $10$ we have $\sim 10^{20}$ parameters!

Enrico Fermi to Freeman Dyson

“I remember my friend Johnny von Neumann used to say, with four parameters I can fit an elephant, and with five I can make him wiggle his trunk.” [video]

How do we prevent overfitting?
How does overfitting occur?

Suppose $X_{100 \times 100}$ with every entry $\mathcal{N}(0, 1)$

And let $y_i = x_{i, 1} + \mathcal{N}(0, \sigma^2)$, for $\sigma = 0.2$
Ridge Regression

Say our data is $\langle (x_i, y_i) \rangle_{i=1}^{m}$, where $x \in \mathbb{R}^N$ where $N$ is really really large!

We used the squared loss

$$L(w) = (Xw - y)^T (Xw - y)$$

and obtained the estimate

$$w = (X^TX)^{-1}X^Ty$$

Suppose we want $w$ to be "small" (weight decay)

$$L(w) = (Xw - y)^T (Xw - y) + \lambda w^Tw - \lambda w_1^2$$

We will not regularize the "bias" (or constant) term

Exercise: If all $x_i$ (except $x_1$) have mean 0 and $y$ has mean 0, $w_1 = 0$
Deriving Estimate for Ridge Regression

\[ L(w) = (Xw - y)^T (Xw - y) + \lambda w^T w \]
Ridge Regression

Objective/Loss: \[ L(w) = (Xw - y)^T(Xw - y) + \lambda w^T w \]

Estimate: \[ \hat{w} = (X^T X + \lambda I)^{-1} X^T y \]

Estimate depends on the scaling of the inputs

Common practice: Normalise all input dimensions
Ridge Regression and MAP Estimation

Objective/Loss: $L(w) = (Xw - y)^T(Xw - y) + \lambda w^T w$
Ridge Regression and MAP Estimation

Objective/Loss: \( L(w) = (Xw - y)^T(Xw - y) + \lambda w^T w \)

\[
\exp \left( -\frac{1}{2\sigma^2} L(w) \right) = \exp \left( -\frac{1}{2} (y - Xw)^T \Sigma^{-1} (y - Xw) \right) \cdot \exp \left( -\frac{1}{2} w^T \Lambda^{-1} w \right)
\]
Bayesian View of Machine Learning

General Formulation

Prior \( p(w) \) on \( w \)

Model \( p(y \mid x, w) \)

Linear Regression

\[ p(w) = \mathcal{N}(w \mid 0, \tau^2 I_n) \]

\[ p(y \mid x, w) = \mathcal{N}(y \mid x^T w, \sigma^2) \]

Compute posterior given data \( \mathcal{D} = \langle (x_i, y_i) \rangle_{i=1}^m \)

\[ p(w \mid \mathcal{D}) \propto \]
Bayesian View of Regression

Making a prediction on a new point $x_{\text{new}}$

\[
p(y \mid D, x_{\text{new}}) = \int_{w} p(y \mid w, x_{\text{new}}) p(w \mid D) dw
\]
Prediction MAP vs Fully Bayesian

**MAP Approach**

\[ y_{\text{new}} \sim \mathcal{N}(x_{\text{new}}^T w_{\text{map}}, \sigma^2) \]

**Bayesian Approach**

\[ y_{\text{new}} \sim \mathcal{N}(x_{\text{new}}^T w_{\text{map}}, \sigma_X^2) \]

\[ \sigma_X^2 = \sigma^2 \left(1 + x_{\text{new}}^T (X^T X + \frac{\sigma^2}{\tau^2} I)^{-1} x_{\text{new}}\right) \]
Ridge Regression

Minimize: 

$$(Xw - y)^T (Xw - y)^T + \lambda w^T w$$

Minimize $$(Xw - y)^T (Xw - y)$$

such that $w^T w \leq R$$
Ridge Regression

Image Source: Hastie, Tibshirani, Friedman (2013)
Lasso Regression

Minimize:

$$(Xw - y)^T (Xw - y)^T + \lambda \sum_{i=1}^{N} |w_i|$$

Minimize $$(Xw - y)^T (Xw - y)$$

such that $$\sum_{i=1}^{N} |w_i| \leq R$$
Lasso Regression

Image Source: Hastie, Tibshirani, Friedman (2013)
Regularization: Ridge or Lasso

No regularization

Ridge

Lasso
Choosing Parameters

Before we were just trying to find \( w \)

Now, we have to worry about how to choose \( \lambda \) for ridge or Lasso

If we use kernels, we also have to pick the width \( \sigma \)

If we use higher degree polynomials, we have to pick \( d \)
Cross Validation

- Keep a part of data as “validation” or “test” set
- Look for error on “training” and “validation” sets

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>Train Error(%)</th>
<th>Test Error(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>25</td>
<td>27</td>
</tr>
<tr>
<td>0.1</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>43</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>89</td>
</tr>
</tbody>
</table>
$k$-Fold Cross Validation

What do we do when data is scarce?

- Divide data into $k$ parts
- Use $k - 1$ parts for training and 1 part as validation
- When $k = m$ (the number of datapoints), we get LOOCV (Leave one out cross validation)
Overfitting on the Validation Set

Suppose you do all the right things

- Train on the training set
- Choose hyperparameters using proper validation
- Test on the test set, and your error is high!

What would you do?
Winning Kaggle without reading the data!

Suppose the task is to predict $m$ binary labels
Winning Kaggle without reading the data!

Suppose the task is to predict $m$ binary labels

Algorithm (Wacky Boosting):

1. Choose $y^1, \ldots, y^k \in \{0, 1\}^m$ uniformly
2. Set $I = \{i \mid \text{accuracy}(y^i) > 51\%\}$
3. Output $\hat{y}_j = \text{majority}\{y^i_j \mid i \in I\}$

Source: blog.mrtz.org
Suppose the task is to predict $m$ binary labels

**Algorithm (Wacky Boosting):**

1. Choose $y^1, \ldots, y^k \in \{0, 1\}^m$ uniformly
2. Set $I = \{i \mid \text{accuracy}(y^i) > 51\%\}$
3. Output $\hat{y}_j = \text{majority}\{y^i_j \mid i \in I\}$

Source: blog.mrtz.org
Next Time

- Optimization Methods

- Read up on gradients, multivariate calculus