ECE 5424: Introduction to Machine Learning

Topics:
- (Finish) Regression
- Model selection, Cross-validation
- Error decomposition

Readings: Barber 17.1, 17.2

Stefan Lee
Virginia Tech
Administrative

• Project Proposal
  – Due: Fri 09/23, 11:55 pm  
    NOTE: DEADLINE SHIFTED
  – <=2pages, NIPS format

• HW2
  – Due: Wed 09/28, 11:55pm
  – Implement linear regression, Naïve Bayes, Logistic Regression

• Reminder:
  – Participation on Scholar forum is part of your grade
    • Ask questions if you have them!
Recap of last time
Regression
Linear fitting to data

- We want to fit a linear function to an observed set of points $X = [x_1, \ldots, x_N]$ with associated labels $Y = [y_1, \ldots, y_N]$.
- Once we fit the function, we want to use it to predict the $y$ for new $x$. 

Slide Credit: Greg Shakhnarovich
Linear fitting to data

- We want to fit a linear function to an observed set of points 
  \[ X = [x_1, \ldots, x_N] \] with associated labels \( Y = [y_1, \ldots, y_N] \).
  - Once we fit the function, we want to use it to \textit{predict} the \( y \) for new \( x \).

- Least squares (LSQ) fitting criterion: find the function that minimizes sum (or average) of square distances between actual \( y \)s in the training set and predicted ones.

The fitted line is used as a predictor.
Least squares in matrix form

\[ X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \cdots & x_{Nd} \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \quad w = \begin{bmatrix} w_0 \\ \vdots \\ w_d \end{bmatrix}. \]

- Predictions: \( \hat{y} = Xw \), errors: \( y - Xw \), empirical loss:

\[
L(w, X) = \frac{1}{N} (y - Xw)^T (y - Xw)
\]
Least squares solution

\[
\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}) = -\frac{2}{N} \left( \mathbf{X}^T \mathbf{y} - \mathbf{X}^T \mathbf{X} \mathbf{w} \right) = 0
\]

\[
\mathbf{X}^T \mathbf{y} = \mathbf{X}^T \mathbf{X} \mathbf{w} \Rightarrow \mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}
\]

- \( \mathbf{X}^\dagger \triangleq (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \) is called the Moore-Penrose pseudoinverse of \( \mathbf{X} \).

- Linear regression in Matlab:
  
  ```matlab
  \% X(i,:) is i-th example, y(i) is i-th label
  wLSQ = pinv([ones(size(X,1),1) X])*y;
  ```

- Prediction:
  
  \[
  \hat{y} = \mathbf{w}^* \mathbf{T} \begin{bmatrix} 1 \\ \mathbf{x}_0 \end{bmatrix} = \mathbf{y}^T \mathbf{X}^\dagger \mathbf{T} \begin{bmatrix} 1 \\ \mathbf{x}_0 \end{bmatrix}
  \]

(C) Dhruv Batra

Slide Credit: Greg Shakhnarovich
But, why?

• Why sum squared error???
• Gaussians, Watson, Gaussians…
Gaussian noise model

\[ y = f(x; w) + \nu, \quad \nu \sim \mathcal{N}(\nu; 0, \sigma^2) \]

- Given the input \( x \), the label \( y \) is a random variable

\[ p(y|x; w, \sigma) = \mathcal{N}(y; f(x; w), \sigma^2) \]

that is,

\[ p(y|x; w, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{(y - f(x; w))^2}{2\sigma^2} \right) \]

- This is an explicit model of \( y \) that allows us, for instance, to sample \( y \) for a given \( x \).
Is OLS Robust?

• Demo
  – [http://www.calpoly.edu/~srein/StatDemo/All.html](http://www.calpoly.edu/~srein/StatDemo/All.html)

• Bad things happen when the data does not come from your model!

• How do we fix this?
Robust Linear Regression

- $y \sim \text{Lap}(w'x, b)$
- On paper

![Diagram showing comparison of L2, L1, and Huber norms in robust linear regression. The left plot illustrates the norms for linear data with noise and outliers, while the right plot compares least squares and Laplace distributions.]
Plan for Today

• (Finish) Regression
  – Bayesian Regression
  – Different prior vs likelihood combination
  – Polynomial Regression

• Error Decomposition
  – Bias-Variance
  – Cross-validation
Robustify via Prior

- Ridge Regression

- \( y \sim N(w'x, \sigma^2) \)
- \( w \sim N(0, t^2I) \)

- \( P(w \mid x,y) = \)
## Summary

<table>
<thead>
<tr>
<th>Likelihood</th>
<th>Prior</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>Uniform</td>
<td>Least Squares</td>
</tr>
<tr>
<td>Gaussian</td>
<td>Gaussian</td>
<td>Ridge Regression</td>
</tr>
<tr>
<td>Gaussian</td>
<td>Laplace</td>
<td>Lasso</td>
</tr>
<tr>
<td>Laplace</td>
<td>Uniform</td>
<td>Robust Regression</td>
</tr>
<tr>
<td>Student</td>
<td>Uniform</td>
<td>Robust Regression</td>
</tr>
</tbody>
</table>
Polynomial regression

- Consider 1D for simplicity:

\[ f(x; w) = w_0 + w_1 x + w_2 x^2 + \ldots + w_m x^m. \]

- No longer linear in \( x \) – but still linear in \( w \)!
Polynomial regression

- Consider 1D for simplicity:

\[ f(x; \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_m x^m. \]

- No longer linear in \( x \) – but still linear in \( \mathbf{w} \)!
- Define \( \phi(x) = [1, x, x^2, \ldots, x^m]^T \)
- Then, \( f(x; \mathbf{w}) = \mathbf{w}^T \phi(x) \) and we are back to the familiar simple linear regression. The least squares solution:

\[ \hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}, \text{ where } \mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^m \\ 1 & x_2 & x_2^2 & \ldots & x_2^m \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_N & x_N^2 & \ldots & x_N^m \end{bmatrix} \]
General additive regression models

- A general extension of the linear regression model:

\[ f(x; w) = w_0 + w_1\phi_1(x) + w_2\phi_2(x) + \ldots + w_m\phi_m(x), \]

where \( \phi_j(x) : \mathcal{X} \rightarrow \mathbb{R}, j = 1, \ldots, m \) are the basis functions.

- This is still linear in \( w \),

\[ f(x; w) = w^T \phi(x) \]

even when \( \phi \) is non-linear in the inputs \( x \).
General additive regression models

\[ f(x; w) = w_0 + w_1 \phi_1(x) + w_2 \phi_2(x) + \ldots + w_m \phi_m(x), \]

- Still the same ML estimation technique applies:
  \[ \hat{w} = (X^T X)^{-1} X^T y \]

where \( X \) is the \textit{design matrix}

\[
\begin{bmatrix}
\phi_0(x_1) & \phi_1(x_1) & \phi_2(x_1) & \ldots & \phi_m(x_1) \\
\phi_0(x_2) & \phi_1(x_2) & \phi_2(x_2) & \ldots & \phi_m(x_2) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\phi_0(x_N) & \phi_1(x_N) & \phi_2(x_N) & \ldots & \phi_m(x_N)
\end{bmatrix}
\]

(for convenience we will denote \( \phi_0(x) \equiv 1 \))
Example

• Demo
  – http://www.princeton.edu/~rkatzwer/PolynomialRegression/
What you need to know

• Linear Regression
  – Model
  – Least Squares Objective
  – Connections to Max Likelihood with Gaussian Conditional
  – Robust regression with Laplacian Likelihood
  – Ridge Regression with priors
  – Polynomial and General Additive Regression
New Topic: Model Selection and Error Decomposition
Example for Regression

• Demo
  – http://www.princeton.edu/~rkatzwer/PolynomialRegression/

• How do we pick the hypothesis class?
Model Selection

• How do we pick the right model class?

• Similar questions
  – How do I pick magic hyper-parameters?
  – How do I do feature selection?
Errors

• Expected Loss/Error
• Training Loss/Error
• Validation Loss/Error
• Test Loss/Error

• Reporting Training Error (instead of Test) is CHEATING

• Optimizing parameters on Test Error is CHEATING
Cross-validation

- The improved holdout method: \( k \)-fold cross-validation
  - Partition data into \( k \) roughly equal parts;
  - Train on all but \( j \)-th part, test on \( j \)-th part

\[
\begin{array}{cccc}
\cdots & x_1 & \cdots & x_N \\
\end{array}
\]
Cross-validation

- The improved holdout method: \( k \)-fold cross-validation
  - Partition data into \( k \) roughly equal parts;
  - Train on all but \( j \)-th part, test on \( j \)-th part

\[
\begin{align*}
&x_1 &\ldots &x_N
\end{align*}
\]
Cross-validation

- The improved holdout method: $k$-fold *cross-validation*
  - Partition data into $k$ roughly equal parts;
  - Train on all but $j$-th part, test on $j$-th part
Cross-validation

- The improved holdout method: *k*-fold cross-validation
  - Partition data into *k* roughly equal parts;
  - Train on all but *j*-th part, test on *j*-th part

\[ x_1 \quad \ldots \quad x_N \]
Cross-validation

• The improved holdout method: $k$-fold cross-validation
  • Partition data into $k$ roughly equal parts;
  • Train on all but $j$-th part, test on $j$-th part

\[ x_1 \quad \ldots \quad x_N \]

• An extreme case: leave-one-out cross-validation

\[
\hat{L}_{cv} = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i; \hat{w}_{-i}))^2
\]

where $\hat{w}_{-i}$ is fit to all the data but the $i$-th example.
Typical Behavior

Accuracy

100%

Asymptotic training accuracy

Best test accuracy

training

test

Optimal stopping point

Training effort
Overfitting

- **Overfitting**: a learning algorithm overfits the training data if it outputs a solution \( w \) when there exists another solution \( w' \) such that:

\[
[error_{train}(w) < error_{train}(w')] \land [error_{true}(w') < error_{true}(w)]
\]
Error Decomposition

model class

Optimization Error

Estimation Error

Modeling Error

Reality

(C) Dhruv Batra
Error Decomposition

- Model Class
- Optimization Error
- Estimation Error
- Modeling Error

Reality

(C) Dhruv Batra
Error Decomposition

Higher-Order Potentials

model class

Reality

Optimization Error

Estimation Error

Modeling Error
Error Decomposition

• Approximation/Modeling Error
  – You approximated reality with model

• Estimation Error
  – You tried to learn model with finite data

• Optimization Error
  – You were lazy and couldn’t/didn’t optimize to completion

• (Next time) Bayes Error
  – Reality just sucks