ECE 5984: Introduction to Machine Learning

Topics:

- (Finish) Regression
- Model selection, Cross-validation
- Error decomposition
- Bias-Variance Tradeoff

Readings: Barber 17.1, 17.2

Dhruv Batra

Virginia Tech

Administrativia

- HW1
  - Solutions available

- Project Proposal
  - Due: Tue 02/24, 11:55 pm
  - <=2pages, NIPS format
  - Show Igor’s proposal

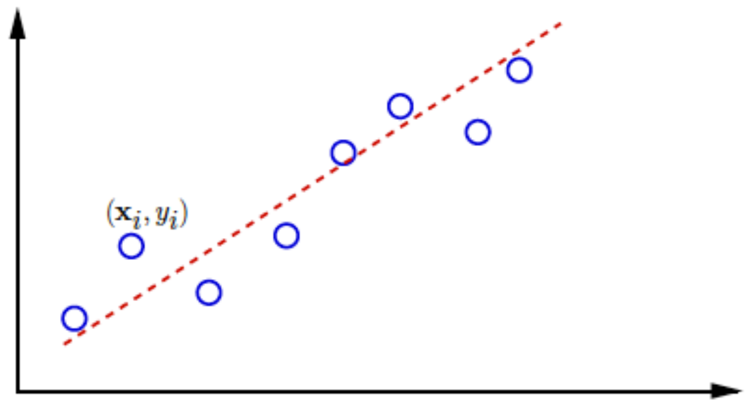
- HW2
  - Due: Friday 03/06, 11:55pm
  - Implement linear regression, Naïve Bayes, Logistic Regression

Recap of last time

Regression

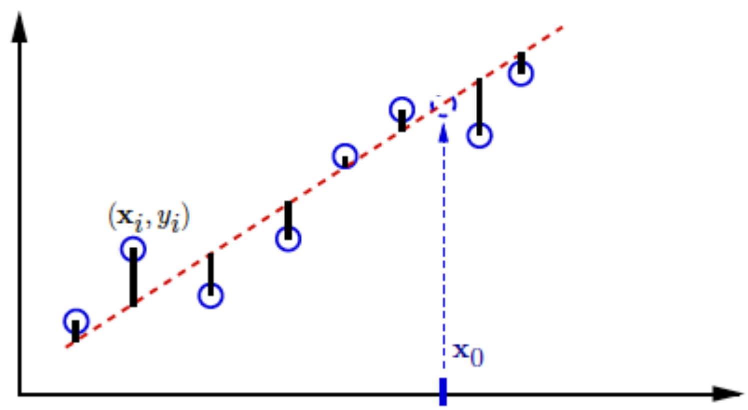
Linear fitting to data

- We want to fit a linear function to an observed set of points $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ 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 $\mathbf{x}$.



Linear fitting to data

- We want to fit a linear function to an observed set of points $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ 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 $\mathbf{x}$.
- Least squares (LSQ) fitting criterion: find the function that minimizes sum (or average) of square distances between actual ys in the training set and predicted ones.



The fitted line is used as a predictor

Least squares in matrix form

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ \vdots & & \vdots & \\ 1 & x_{N1} & \cdots & x_{Nd} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_d \end{bmatrix}.$$

- Predictions: $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$, errors: $\mathbf{y} - \mathbf{X}\mathbf{w}$, empirical loss:

$$L(\mathbf{w}, \mathbf{X}) = \frac{1}{N} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$$

Least squares solution

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

$$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\mathbf{w} \Rightarrow \mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

- $\mathbf{X}^\dagger \triangleq \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$ is called the *Moore-Penrose pseudoinverse* of $\mathbf{X}$.
- Linear regression in Matlab:

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

- Prediction:

$$\hat{y} = \mathbf{w}^{*T}\begin{bmatrix} 1 \\ \mathbf{x}_0 \end{bmatrix} = \mathbf{y}^T\mathbf{X}^{\dagger T}\begin{bmatrix} 1 \\ \mathbf{x}_0 \end{bmatrix}$$

But, why?

- Why sum squared error???
- Gaussians, Watson, Gaussians…

Gaussian noise model

$$y = f(\mathbf{x}; \mathbf{w}) + \nu, \quad \nu \sim \mathcal{N}\left(\nu; 0, \sigma^2\right)$$

- Given the input $\mathbf{x}$, the label y is a random variable

$$p(y|\mathbf{x}; \mathbf{w}, \sigma) \; = \; \mathcal{N}\left(y; \; f(\mathbf{x}; \mathbf{w}), \sigma^2\right)$$

that is,

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

- This is an explicit model of y that allows us, for instance, to *sample* y for a given $\mathbf{x}$.

Is OLS Robust?

- Demo
  - http://www.calpoly.edu/~srein/StatDemo/All.html

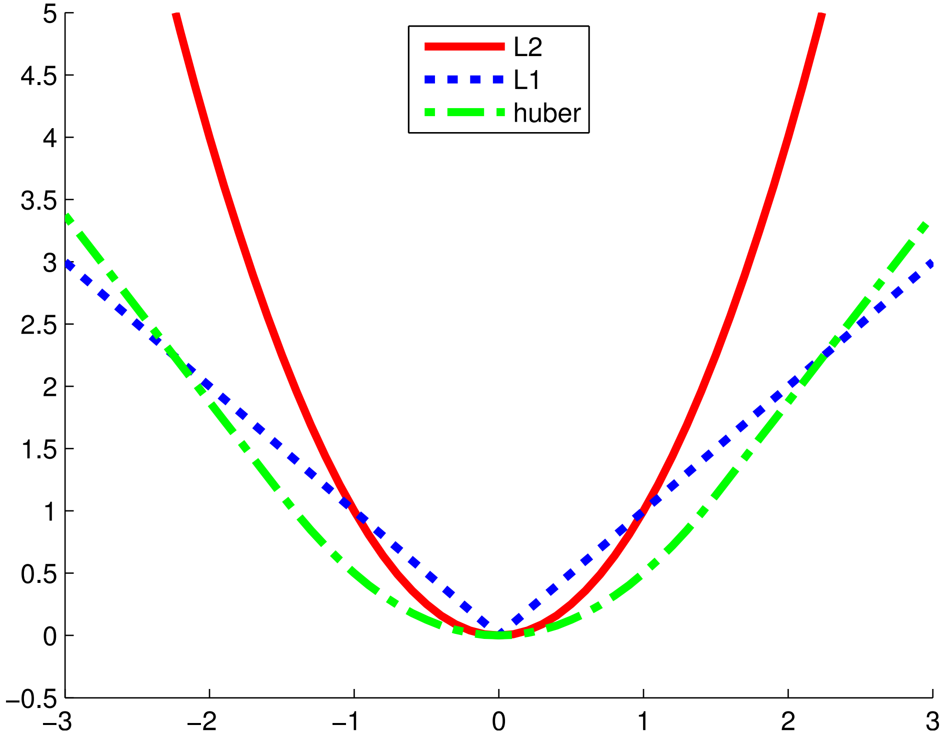
- Bad things happen when the data does not come from your model!

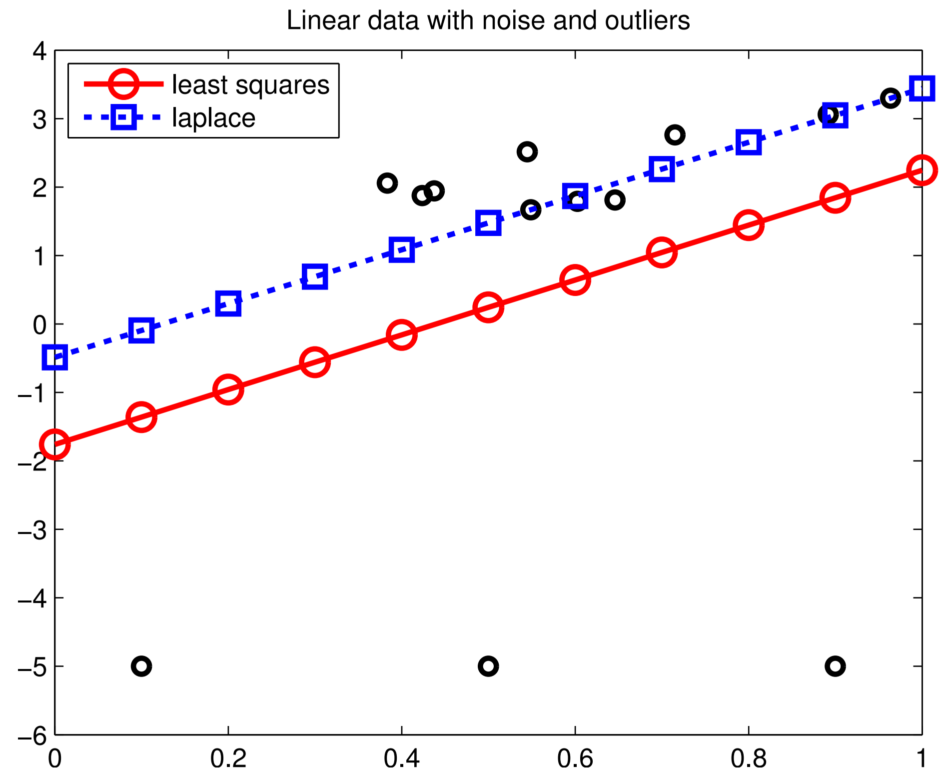
- How do we fix this?

Robust Linear Regression

- y ~ Lap(w'x, b)

- On paper





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

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

- $P(w \mid x,y) =$

Summary

Likelihood	Prior	Name
Gaussian	Uniform	Least Squares
Gaussian	Gaussian	Ridge Regression
Gaussian	Laplace	Lasso
Laplace	Uniform	Robust Regression
Student	Uniform	Robust Regression

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 $\mathbf{w}$!

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 $\phi(\mathbf{x}) = [1, x, x^2, \ldots, x^m]^T$
- Then, $f(x;\mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x})$ and we are back to the familiar simple linear regression. The least squares solution:

$$\hat{\mathbf{w}} \ = \ \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}, \text{ where } \mathbf{X} \ = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^m \\ 1 & x_2 & x_2^2 & \ldots & x_2^m \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_N & x_N^2 & \ldots & x_N^m \end{bmatrix}$$

General additive regression models

- A general extension of the linear regression model:

$$f(\mathbf{x}; \mathbf{w}) \;=\; w_0 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \ldots + w_m\phi_m(\mathbf{x}),$$

  where $\phi_j(\mathbf{x}) : \mathcal{X} \to \mathbb{R}, j = 1, \ldots, m$ are the *basis functions*.
- This is still linear in $\mathbf{w}$,

$$f(\mathbf{x}; \mathbf{w}) \;=\; \mathbf{w}^T\phi(\mathbf{x})$$

  even when ϕ is non-linear in the inputs $\mathbf{x}$.

General additive regression models

$$f(\mathbf{x}; \mathbf{w}) \;=\; w_0 + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \ldots + w_m\phi_m(\mathbf{x}),$$

- Still the same ML estimation technique applies:

$$\hat{\mathbf{w}} \;=\; \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

where $\mathbf{X}$ is the *design matrix*

$$\begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \ldots & \phi_m(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \ldots & \phi_m(\mathbf{x}_2) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \ldots & \phi_m(\mathbf{x}_N) \end{bmatrix}$$

(for convenience we will denote $\phi_0(\mathbf{x}) \equiv 1$)

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

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

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

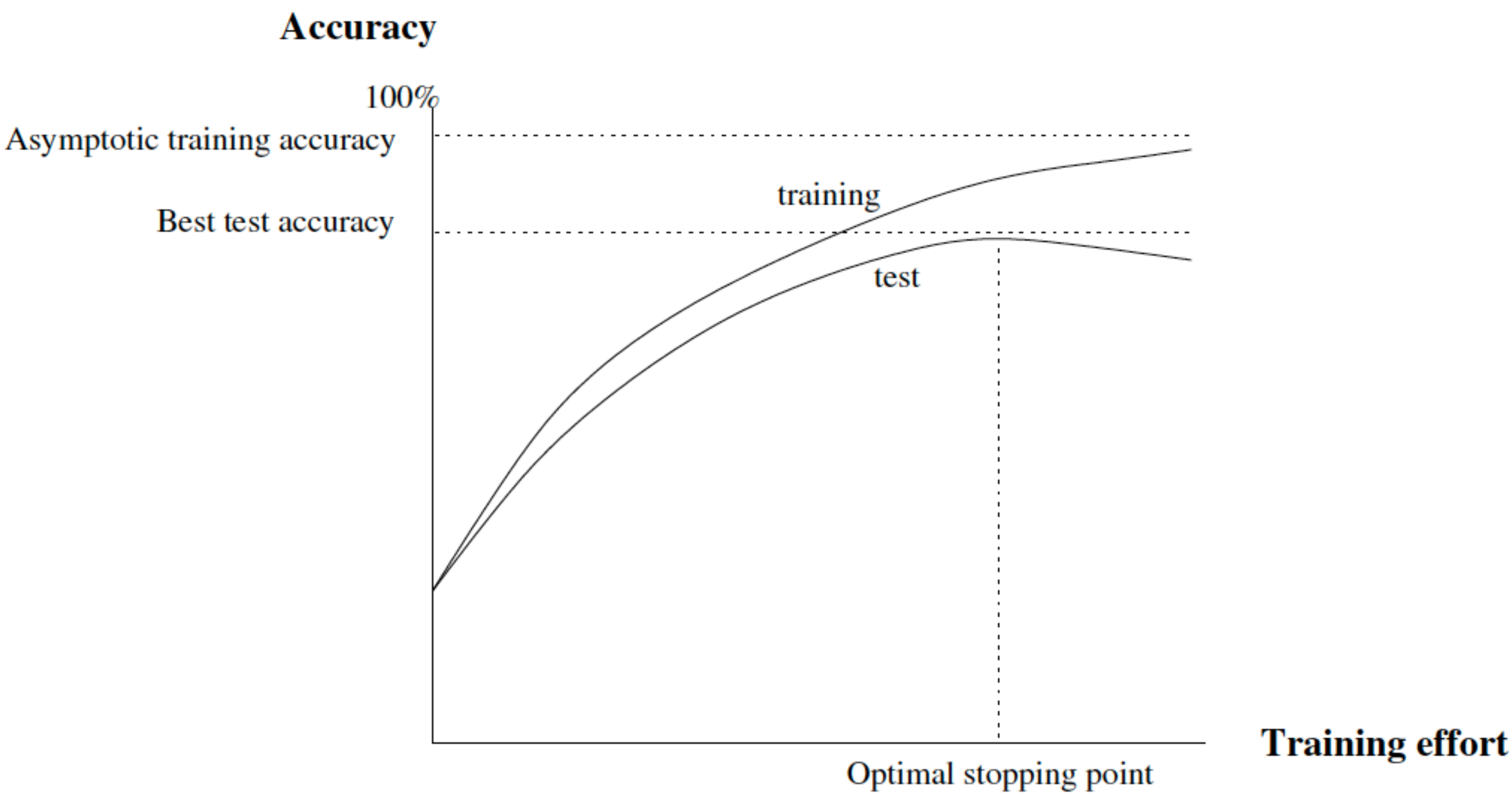
- The improved holdout method: k-fold *cross-validation*
  - Partition data into k roughly equal parts;
  - Train on all but j-th part, test on j-th part

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

$$\hat{L}_{\text{CV}} = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i; \hat{\mathbf{w}}_{-i}))^2$$

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

Typical Behavior

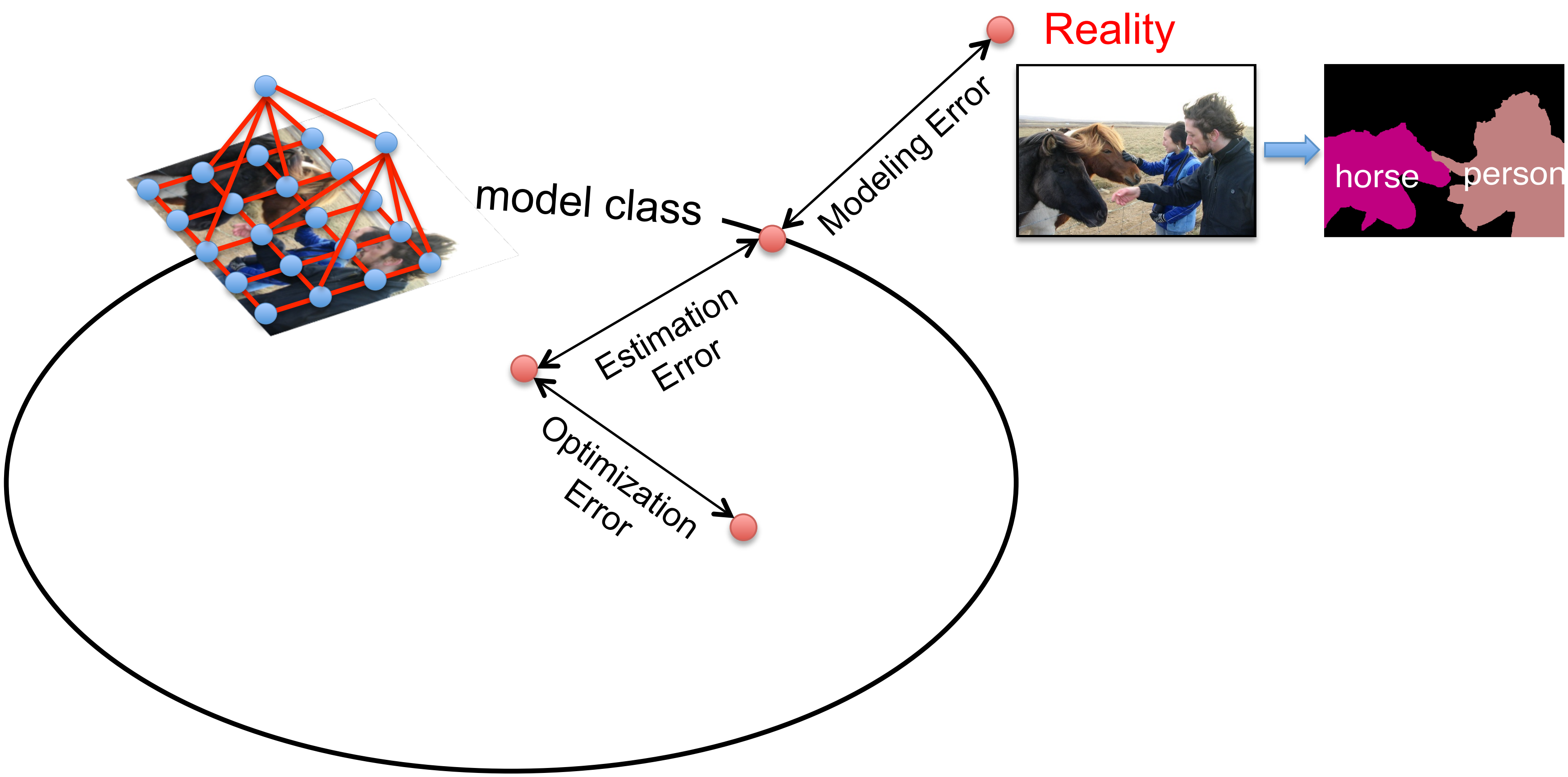


Overfitting

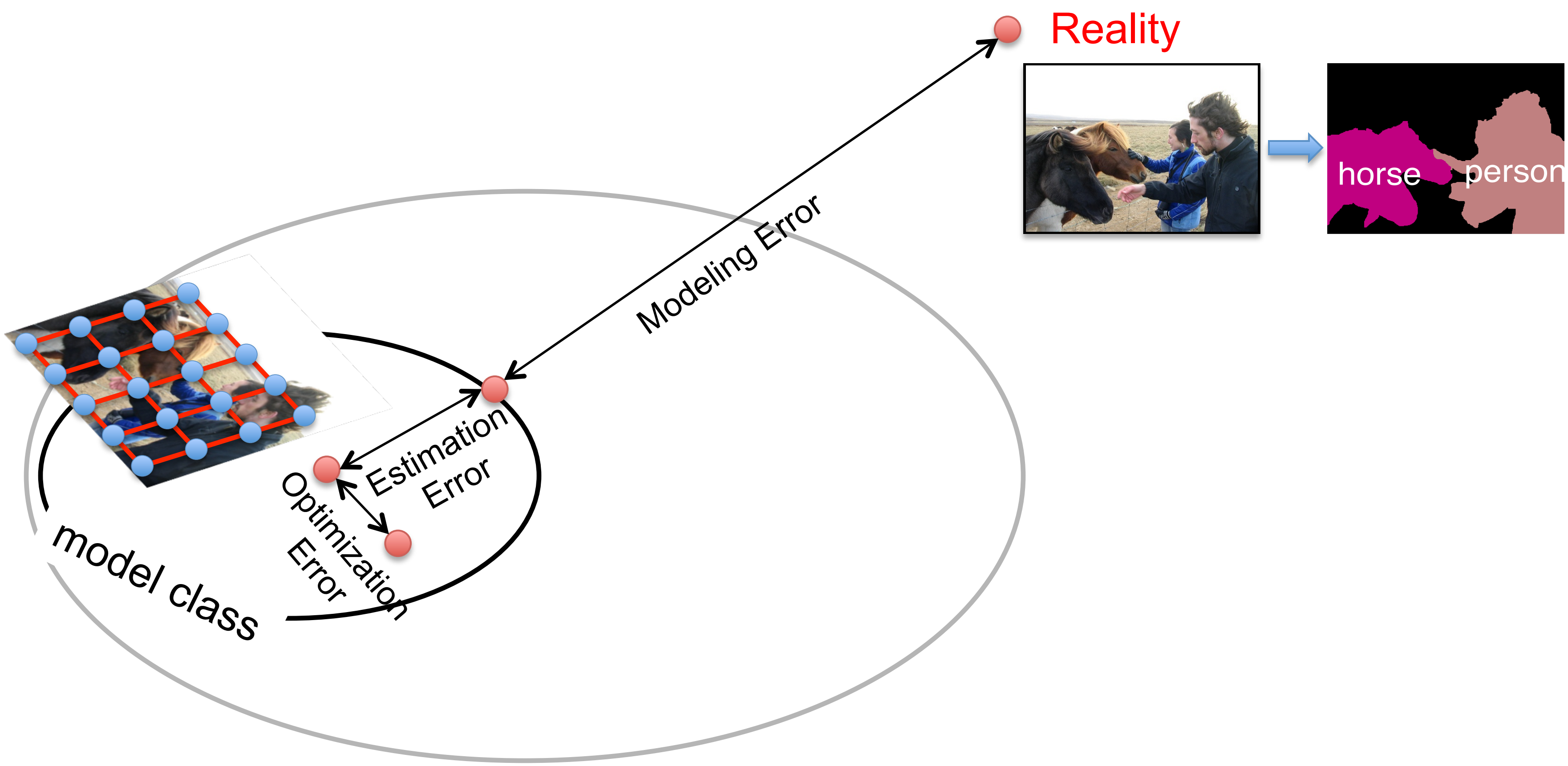
- **Overfitting:** a learning algorithm overfits the training data if it outputs a solution **w** when there exists another solution **w'** such that:

$$[error_{train}(\mathbf{w}) < error_{train}(\mathbf{w}')] \wedge [error_{true}(\mathbf{w}') < error_{true}(\mathbf{w})]$$

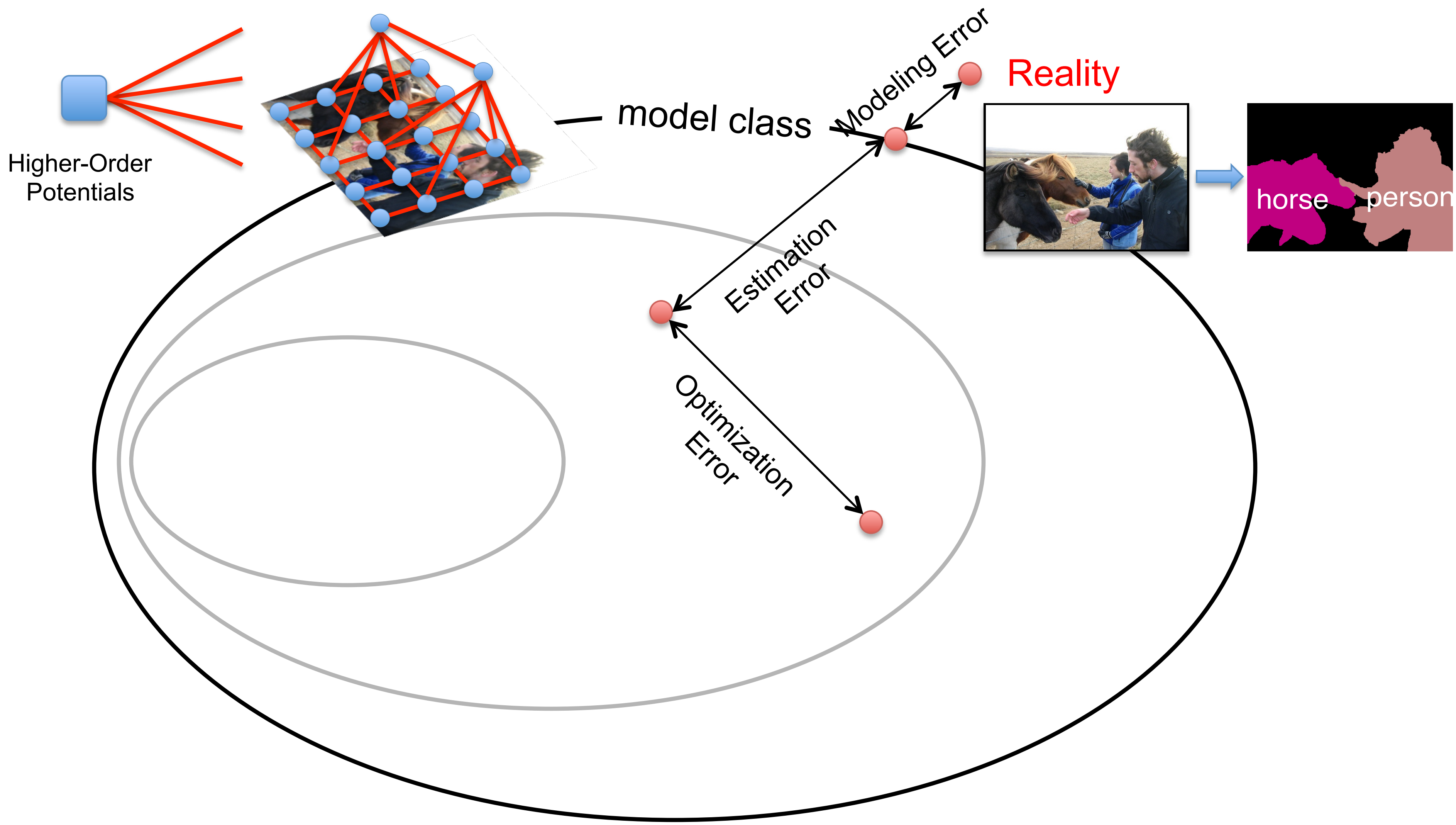
Error Decomposition



Error Decomposition



Error Decomposition



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