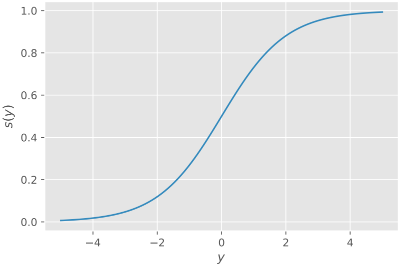
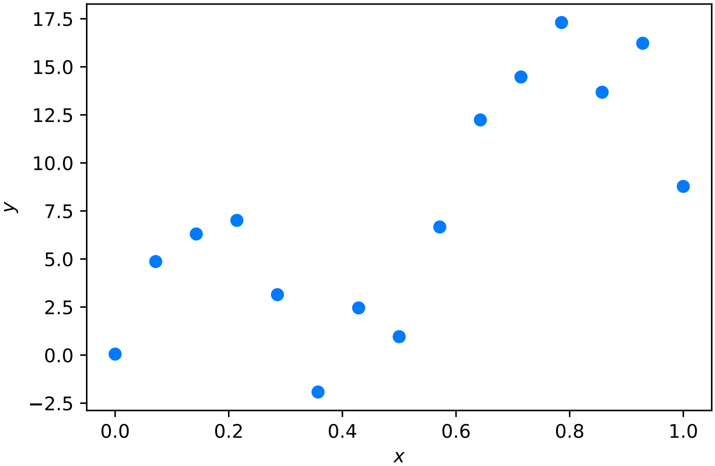


CSE 151A

Intro to Machine Learning

Lecture 08 – Part 01

Model Complexity


17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
y
0.0
0.2
0.4
0.6
0.8
1.0
x

Empirical Risk Minimization

1. Pick a **model**.
   - E.g., linear prediction rules.

2. Pick a loss.
   - E.g., mean squared error.

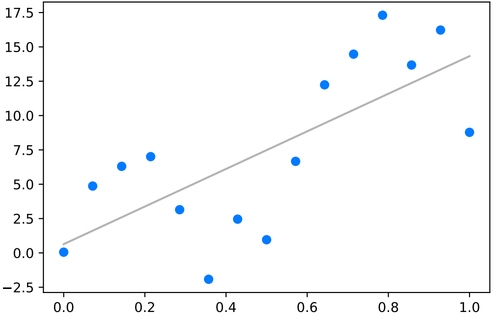
3. Find a prediction rule minimizing the **risk**.

Big Decision

- Pick a model.
- Picking the wrong model causes problems.

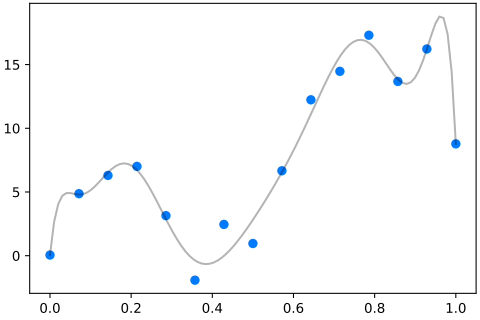
Underfitting

- Fit $H(x) = w_0 + w_1 x$?
  - We have **underfit** the data.



Overfitting

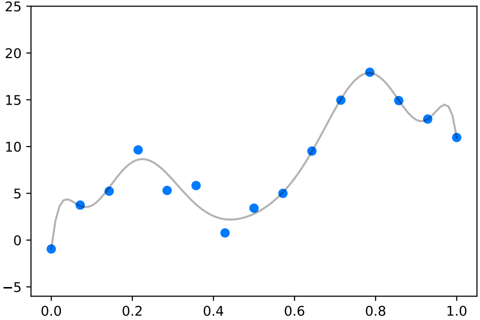
- Fit $H(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_{10} x^{10}$?
  - We have **overfit** the data.



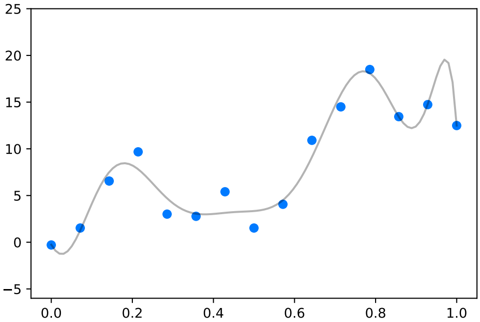
Model Complexity

- Difference? **Complexity**.
- Complex models are highly flexible.
  - They tend to **overfit**.
- Simple models are stiff.
  - They tend to **underfit**.

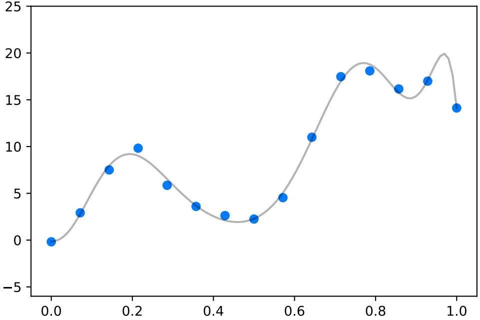
Degree 10 Polynomial



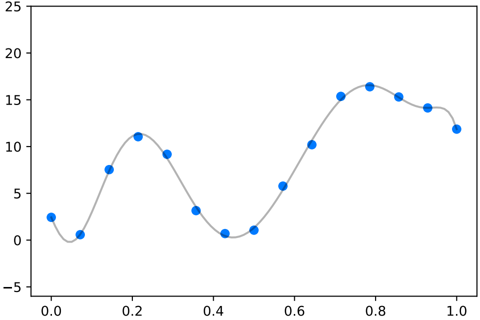
Degree 10 Polynomial



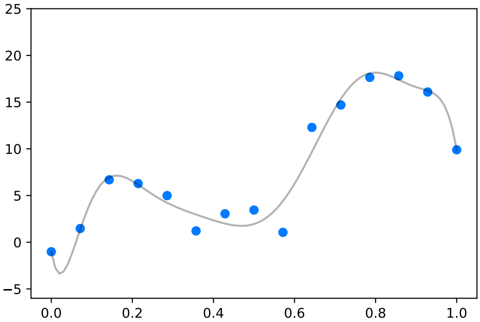
Degree 10 Polynomial



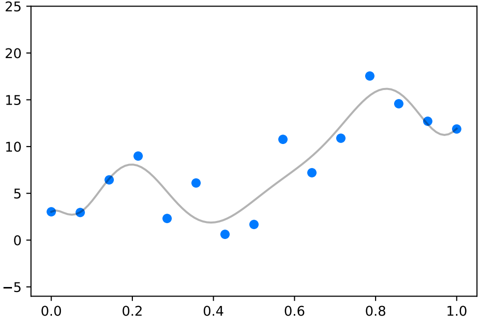
Degree 10 Polynomial



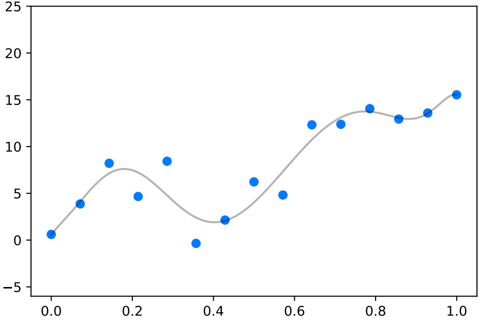
Degree 10 Polynomial



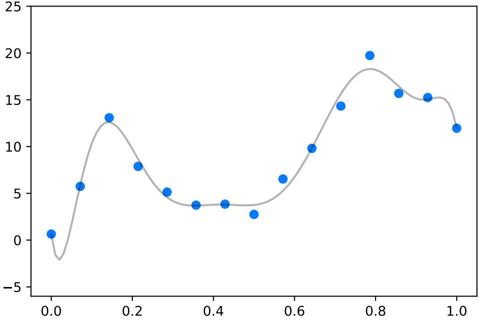
Degree 10 Polynomial



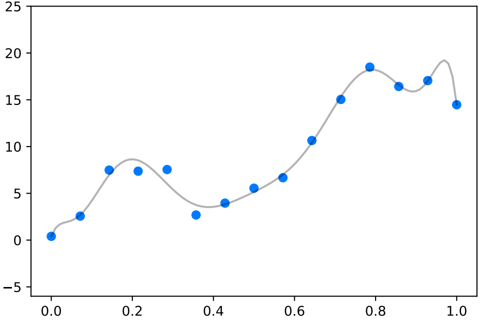
Degree 10 Polynomial



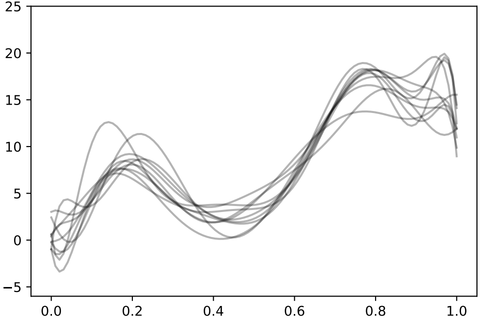
Degree 10 Polynomial



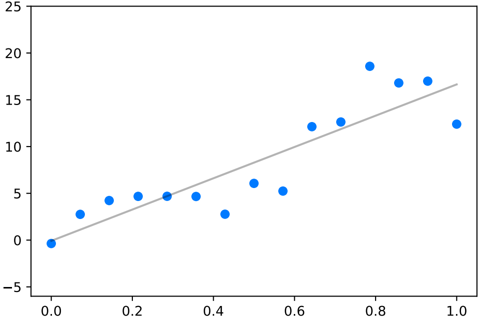
Degree 10 Polynomial



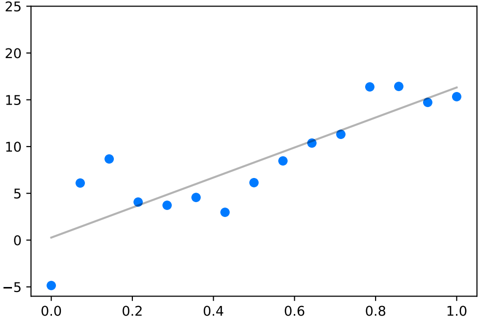
Degree 10 Polynomial



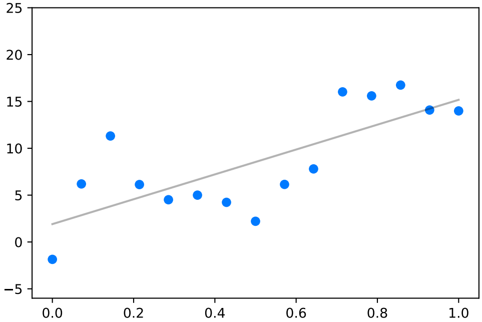
Degree 1 Polynomial



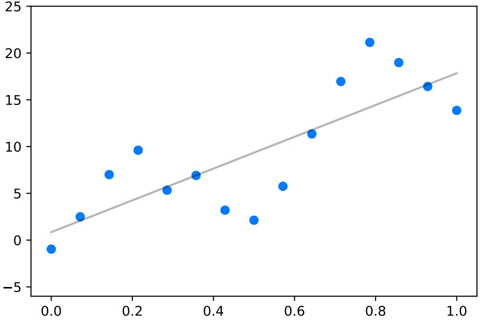
Degree 1 Polynomial



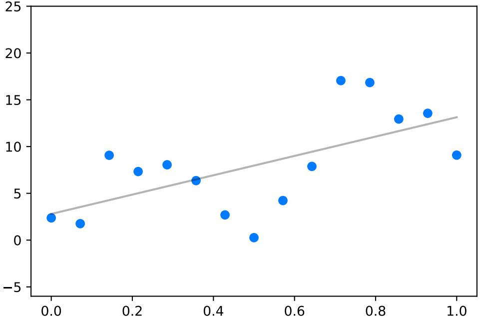
Degree 1 Polynomial



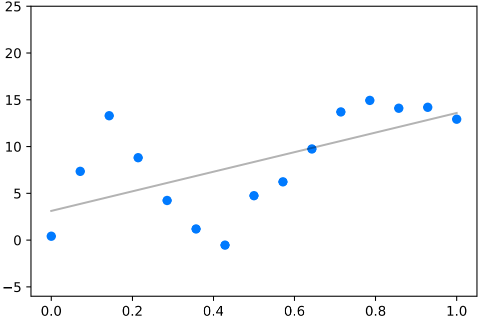
Degree 1 Polynomial



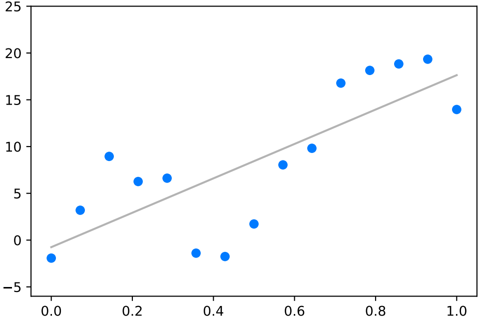
Degree 1 Polynomial



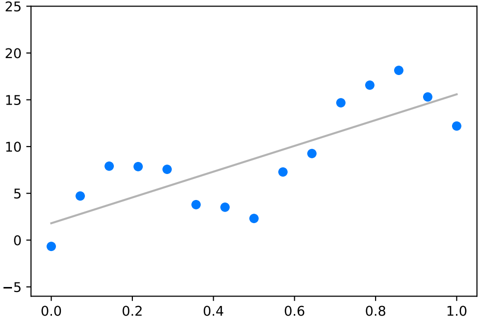
Degree 1 Polynomial



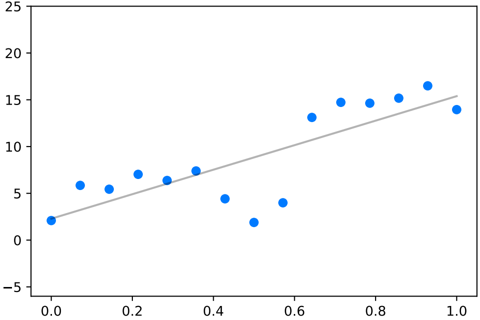
Degree 1 Polynomial



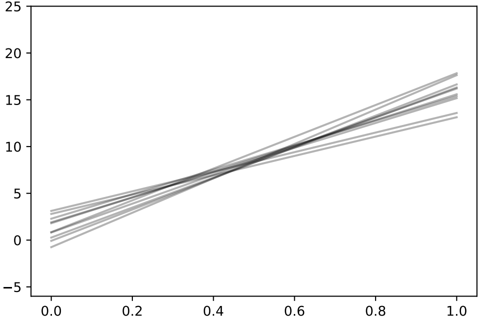
Degree 1 Polynomial



Degree 1 Polynomial



Degree 1 Polynomial



Example: kNN

- 1NN: complex model, likely to overfit.
- 20NN: less complex model, likely to underfit.

Choosing Model Complexity

- How do we choose between two models?
  - Between degree 10 and degree 1?
  - Between 1NN and 20NN?

- Not always obvious.

Bad Idea: Use training MSE

- Which has smaller MSE on training data?

Bad Idea: Use training MSE

- Which has smaller MSE on training data?
- **Problem**: Best 10-degree polynomial will **always** have smaller MSE on training data.

Good Idea: Use validation MSE

- We care about **generalization**.
- So keep a small amount of data hidden in a **validation set**.
- Fit model on training data, compute MSE on **validation set**.
- Pick whichever model has smaller validation error.

What do you expect?

- You fit a complex model on training data.
- Test it on a validation set.
- Likely: validation MSE > training MSE.

What do you expect?

- You fit a very simple model on training data.
- Test it on a validation set.
- Likely: validation MSE ≈ training MSE.

Cross-Validation

- We want all the training data we can get.
- Reserving some of it is wasteful.
- Idea: **split** data into pieces, each takes turn as validation set.

k-Fold Cross Validation

1. Split data set into k pieces, $S_1, \ldots, S_k$.

2. Loop k times; on iteration i:
   - Use S_i as validation set; rest as training.
   - Compute validation error ϵ_i

3. Overall error: $\frac{1}{k} \sum \epsilon_i$

Leave-One-Out Cross Validation

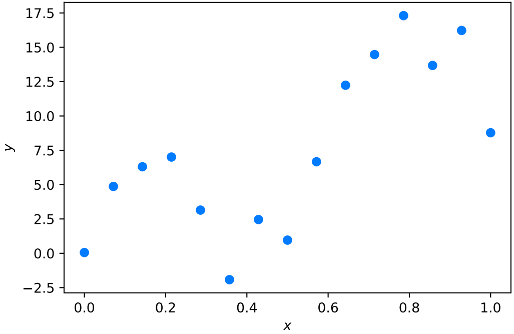
- Suppose we have n labeled data points.
- LOOCV: k-fold CV with $k = n$.

Another Approach

- We can control complexity by choosing model.
- Also: via **regularization**.

Regularization

- Let's fit a complex model: $w_0 + w_x x + \ldots + w_{10} x^{10}$.
- But impose a budget on weights, $w_0, \ldots, w_d$.



Budgeting Weights

- One way to budget: ask that $\|\vec{w}\|^2$ is small.

- Before: minimize

$$R_{\text{sq}}(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} (\vec{w} \cdot \vec{x}^{(i)} - y_i)^2$$

- Now: minimize

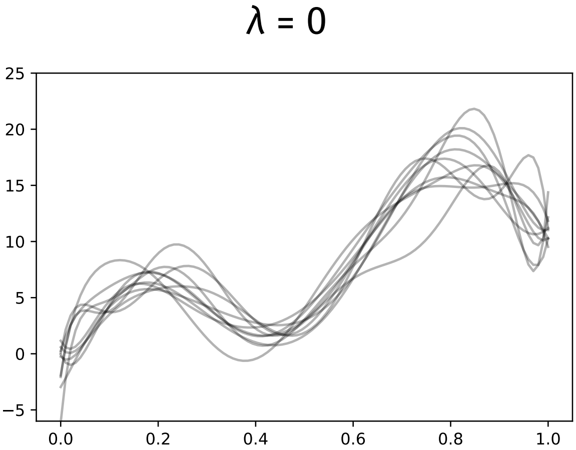
$$\tilde{R}_{\text{sq}}(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} (\vec{w} \cdot \vec{x}^{(i)} - y_i)^2 + \lambda \|\vec{w}\|^2$$

Solution

- The **regularized** Normal Equations:

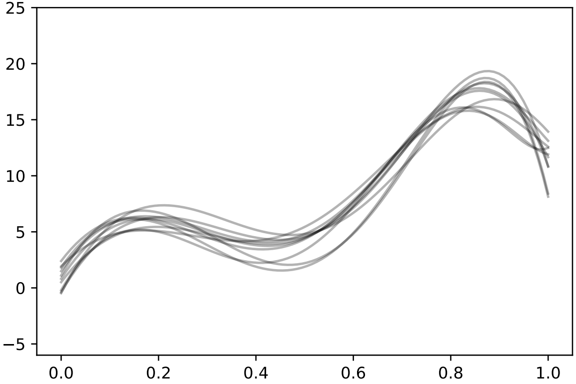
$$(X^T X + \lambda I)\vec{w} = X^T \vec{y}$$

Example

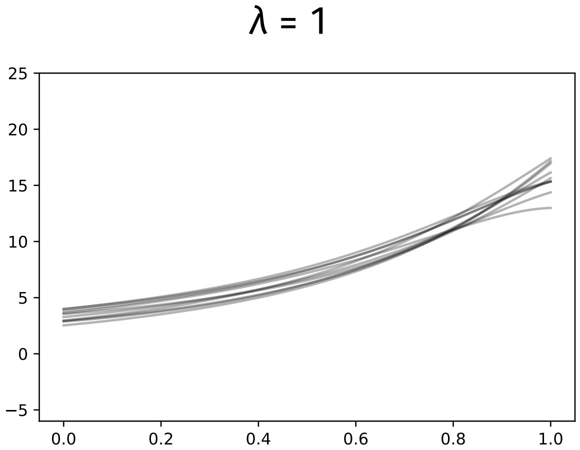


Example

$\lambda = 1 \times 10^{-4}$



Example

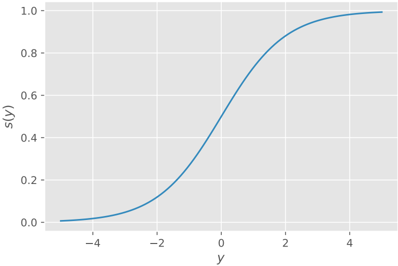


Regularization

- As λ increases, simpler models preferred.
- Pick λ using cross-validation.

Other Penalizations

- $\|\vec{w}\|_2^2$ is ℓ_2 regularization (explicit solution)
  - a.k.a., **ridge regression**
- $\|\vec{w}\|_1$ is ℓ_1 regularization (no explicit solution)
  - a.k.a., **the LASSO**
  - encourages **sparse** $\vec{w}$



CSE 151A
Intro to Machine Learning

Lecture 08 – Part 02
Logistic Regression

Note

- The midterm will cover everything up to right now.
- Regularization: **yes**.
- Logistic regression: **no**.

Predicting Heart Disease

- **Classification problem**: Does a patient have heart disease?
- **Features**: blood pressure, cholesterol level, exercise amount, maximum heart rate, sex

Better idea…

- Instead of predicting **yes/no**…
- Give a **probability** that they have heart disease.
  - 1 = definitely yes
  - 0 = definitely no
  - 0.75 = probably, yes
  - …

Associations

- If cholesterol is high, increased likelihood.
  - Positive association.

- If exercise is low, increased likelihood.
  - Negative association.

The Model

- Measure cholesterol (x_1), exercise (x_2), etc.
- **Idea**: weighted[1] "vote" for heart disease:

$$w_1 x_1 + w_2 x_2 + \ldots + w_d x_d$$

- Convention:
  - A positive number = vote for yes
  - A negative number = vote for no

[1]We'll learn weights later.

The Model

- Add a “bias” term:

$$
\begin{aligned}
&w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d \\
&= \vec{w} \cdot \text{Aug}(\vec{x})
\end{aligned}
$$

- The more positive $\vec{w} \cdot \text{Aug}(\vec{x})$, the more likely.

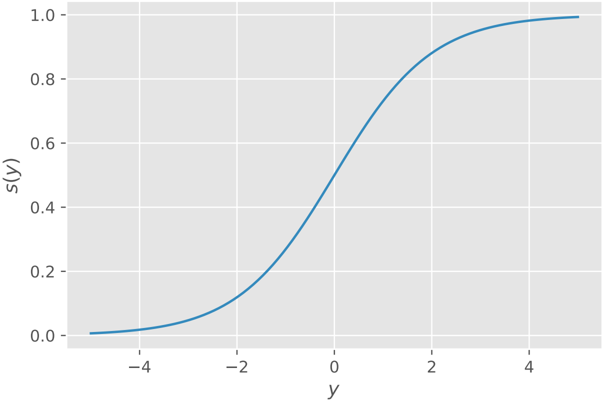
- The more negative $\vec{w} \cdot \text{Aug}(\vec{x})$, the less likely.

Converting to a Probability

- Probabilities are between 0 and 1.
- **Problem**: $\vec{w} \cdot \text{Aug}(\vec{x})$ can be anything in $(-\infty, \infty)$
- We need to convert it to a probability.

The Logistic Function

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$



The Model

- Our simplified model for probability of heart disease:
$$H(\vec{x}) = \sigma(\vec{w} \cdot \text{Aug}(\vec{x}))$$

- What should $\vec{w}$ be?

- To find $\vec{w}$, use principle of **maximum likelihood**.

Maximum Likelihood

- Suppose you have an unfair coin.
- Probability of heads is p, unknown.
- Flip 8 times and see: H, H, T, H, H, H, H, T
- Which is more **likely**: $p = 0.5$ or $p = 0.75$?

Maximum Likelihood

- Assume coin flips are independent.
- The **likelihood** of H, H, T, H, H, H, H, T is:

$$\begin{aligned}\mathcal{L}(p) &= p \cdot p \cdot (1 - p) \cdot p \cdot p \cdot p \cdot p \cdot (1 - p) \\ &= p^6 (1 - p)^2\end{aligned}$$

- Idea: find p maximizing $\mathcal{L}(p)$
  - Equivalently, find p maximizing $\log \mathcal{L}(p)$

Maximum Likelihood

Find p maximizing $\log \mathcal{L}(p) = \log p^6(1 - p)^2$:

Maximum Likelihood

- In general, given n_1 observed heads, n_2 observed tails, maximize:

$$\log\left[P(F_1 = f_1) \cdot P(F_2 = f_2) \cdots \cdot P(F_n = f_n)\right]$$

$$= \sum_{i=1}^{n} \log P(F_1 = f_1)$$

Back to Logistic Regression

- The probability that person i has heart disease:

$$H(\vec{x}^{(i)}) = \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})$$

- Gather a data set, $(\vec{x}^{(1)}, y_1), \ldots, (\vec{x}^{(n)}, y_n)$.
- What is the most **likely** $\vec{w}$?

Maximum Likelihood

- Suppose 3 people, (+,-,+).
- Likelihood:

$$
\begin{aligned}
&H(\vec{x}^{(1)}) \cdot (1 - H(\vec{x}^{(2)})) \cdot H(\vec{x}^{(3)}) \\
&\quad = \sigma(\vec{w} \cdot \operatorname{Aug}(\vec{x}^{(1)})) \cdot \left(1 - \sigma(\vec{w} \cdot \operatorname{Aug}(\vec{x}^{(2)}))\right) \cdot \sigma(\vec{w} \cdot \operatorname{Aug}(\vec{x}^{(3)})) \\
&\quad = \frac{1}{1 + e^{-\vec{w} \cdot \operatorname{Aug}(\vec{x}^{(1)})}} \cdot \left(1 - \frac{1}{1 + e^{-\vec{w} \cdot \operatorname{Aug}(\vec{x}^{(2)})}}\right) \cdot \frac{1}{1 + e^{-\vec{w} \cdot \operatorname{Aug}(\vec{x}^{(3)})}}
\end{aligned}
$$

Observation

- Note:

$$1 - \frac{1}{1 + e^{-t}} = \frac{1}{1 + e^{t}}$$

Maximum Likelihood

- The likelihood:

$$\frac{1}{1 + e^{-\vec{w} \cdot \text{Aug}(\vec{x}^{(1)})}} \cdot \frac{1}{1 + e^{\vec{w} \cdot \text{Aug}(\vec{x}^{(2)})}} \cdot \frac{1}{1 + e^{-\vec{w} \cdot \text{Aug}(\vec{x}^{(3)})}}$$

- Suppose $y_i = 1$ if positive, $y_i = -1$ if negative:

$$\frac{1}{1 + e^{-y_1 \vec{w} \cdot \text{Aug}(\vec{x}^{(1)})}} \cdot \frac{1}{1 + e^{-y_2 \vec{w} \cdot \text{Aug}(\vec{x}^{(2)})}} \cdot \frac{1}{1 + e^{-y_3 \vec{w} \cdot \text{Aug}(\vec{x}^{(3)})}}$$

Maximum Likelihood

- In general, the likelihood is:

$$\mathcal{L}(\vec{w}) = \prod_{i=1}^{n} \frac{1}{1 + e^{-y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})}}$$

- The **log likelihood** is:

$$\log \mathcal{L}(\vec{w}) = -\sum_{i=1}^{n} \log\left[1 + e^{-y_i \vec{w} \cdot \text{Aug}(\vec{x}^{(i)})}\right]$$

Maximizing Likelihood

- **Goal**: find $\vec{w}$ maximizing $\log \mathcal{L}$
- Take gradient, set to zero, solve?
- **Problem**: try it, you'll get stuck.
- Unlike least-squares regression, there is no explicit solution.

Next Time

How to maximize the log loss with gradient descent.