

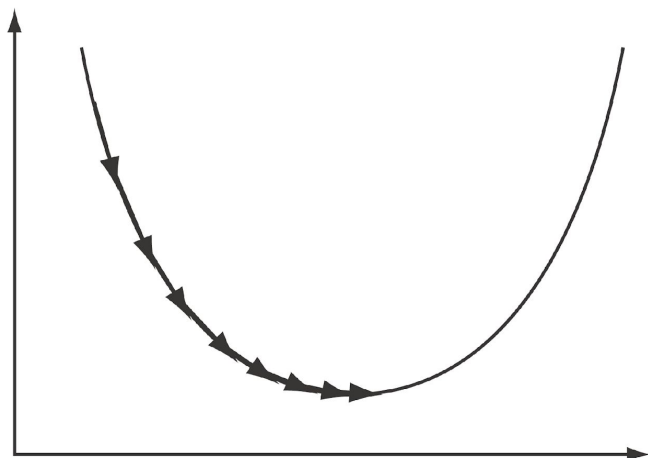
MLE, MAP, AND NAIVE BAYES

Loose Ends from HW

- How to select r ? (The learning rate)

$$\theta_j \Leftarrow \theta_j + r \sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i) x_i^{(j)}$$

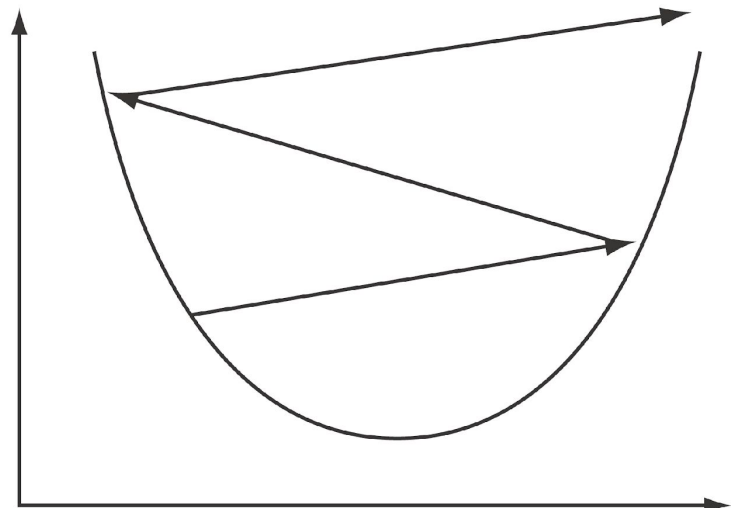
Small learning rate



<https://www.packtpub.com/books/content/big-data>

r too small and the model converges slowly

Large learning rate



r too large and the model diverges

Learning rate issues

- Typically r is normalized with the amount of training examples in a mini-batch. (Divide by m)

$$\theta_j \Leftarrow \theta_j + r \sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i) x_i^{(j)}$$

- Typical values are 0.1-0.001
- Usually have a decay over time

Scaling the input data

- We use age, passenger class, gender, and embark as our input.
- Age has a lot more variance (0.42 – 80) than the other data.
- This makes parameter initialization hard, and makes the learning rate selection hard.
- $h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$

Scaling the input data

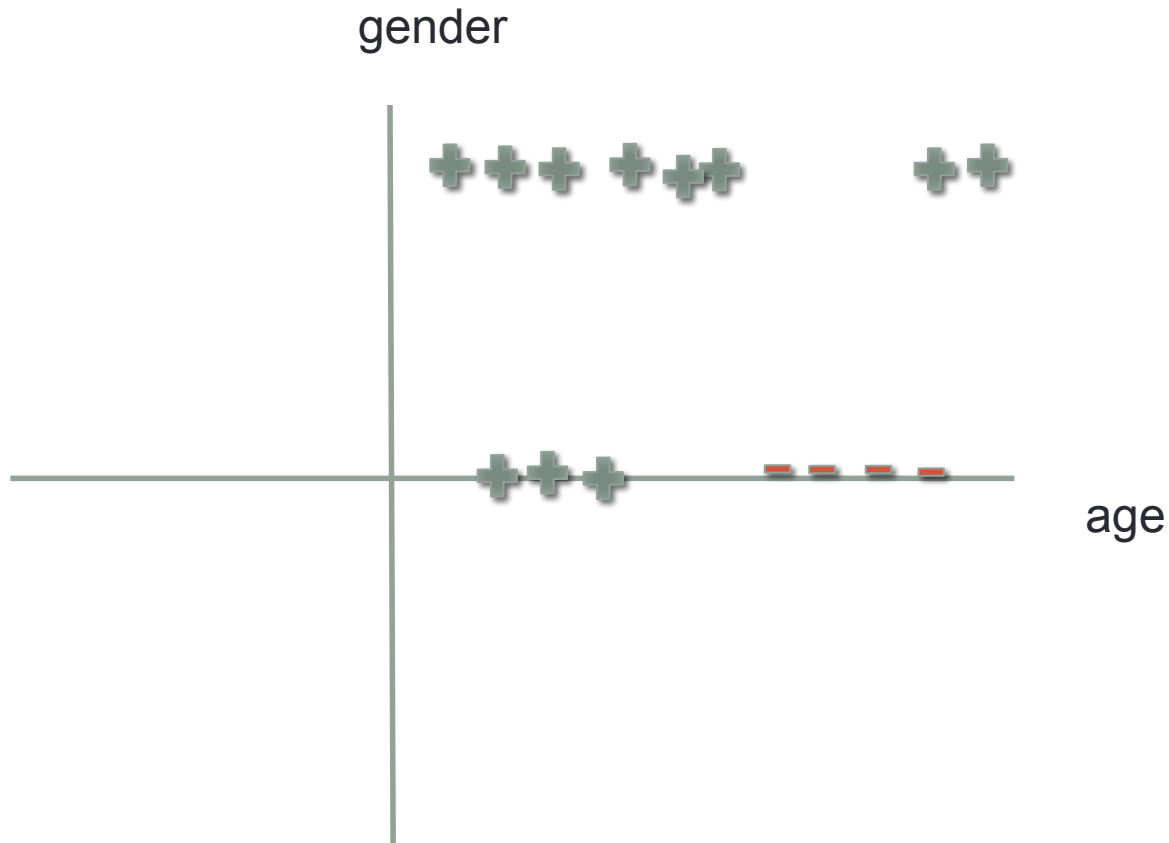
- Scale all input data to be in the same range
- Using statistics from training data
 - Scale to $[-1, 1]$
 - Scale to $[0, 1]$
 - Scale to standard normal
- Don't forget to apply the same scaling to the test data

Feature selection

- If you actually try the third optional question, most likely you will get better results with just two features.
- This is the importance of feature selection.
- Knowing what good features to select is not trivial
- Approaches for feature selection (or for not having to do feature selection)
 - Cross validation
 - Random forest
 - Boosting
 - Model averaging

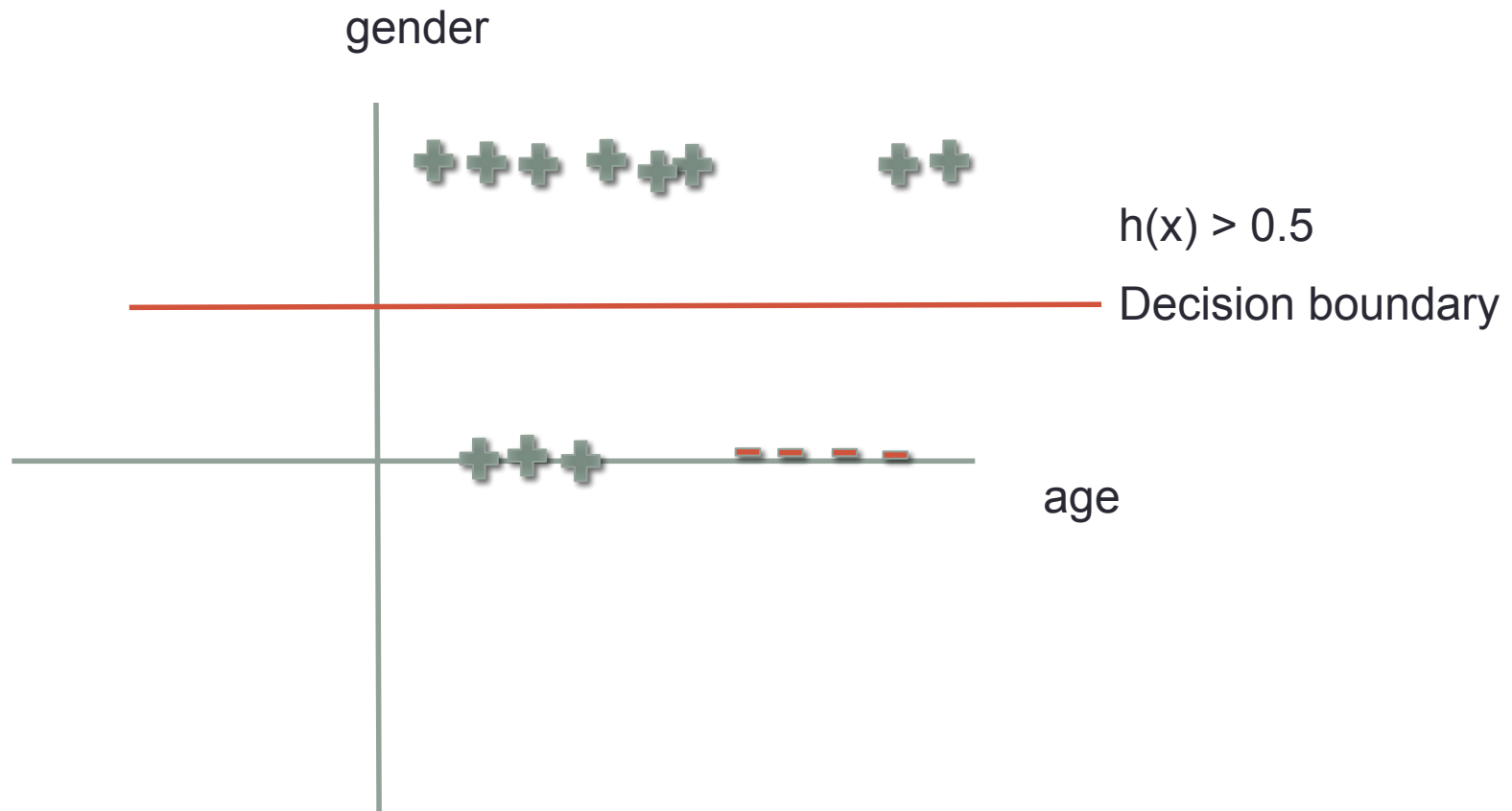
Feature engineering

- Logistic regression is a linear classification



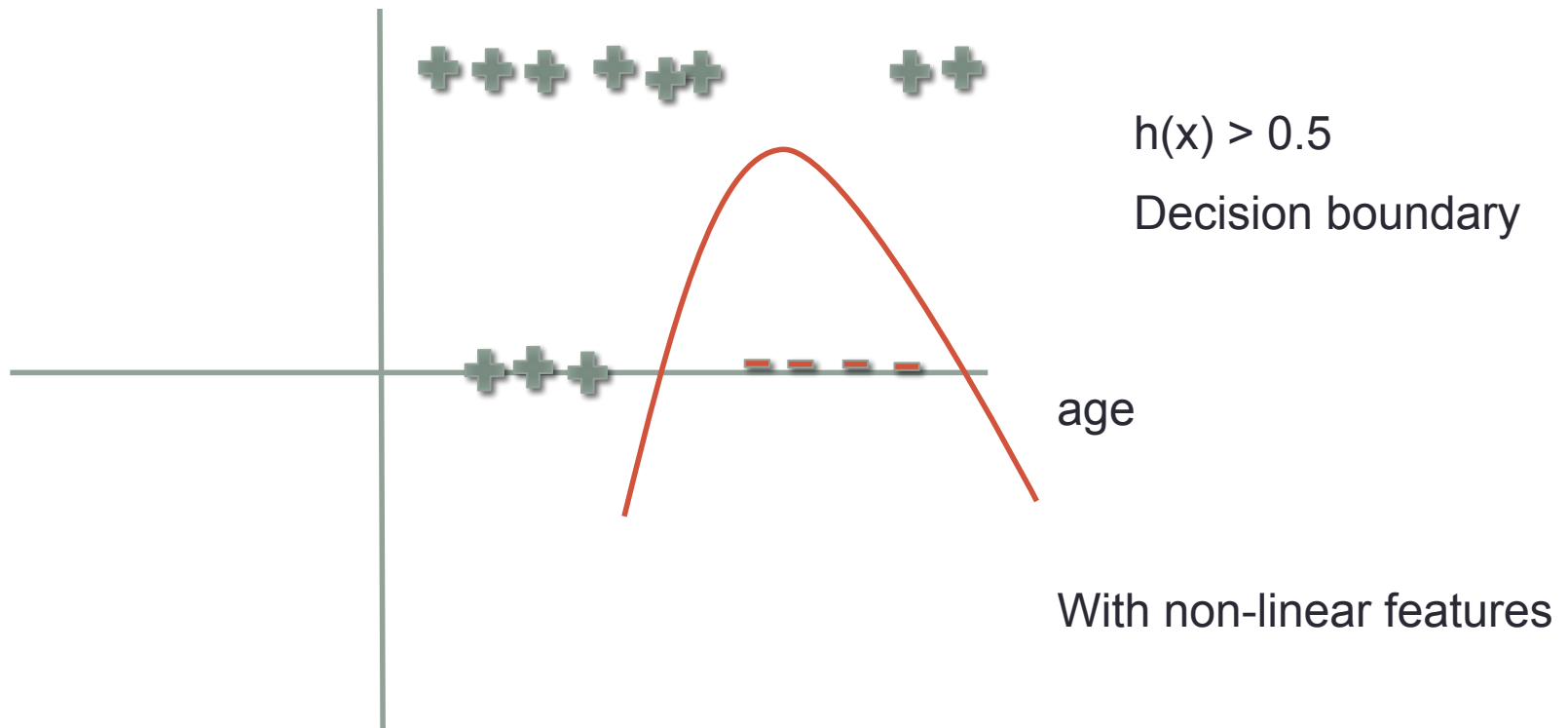
Feature engineering

- Logistic regression is a linear classification



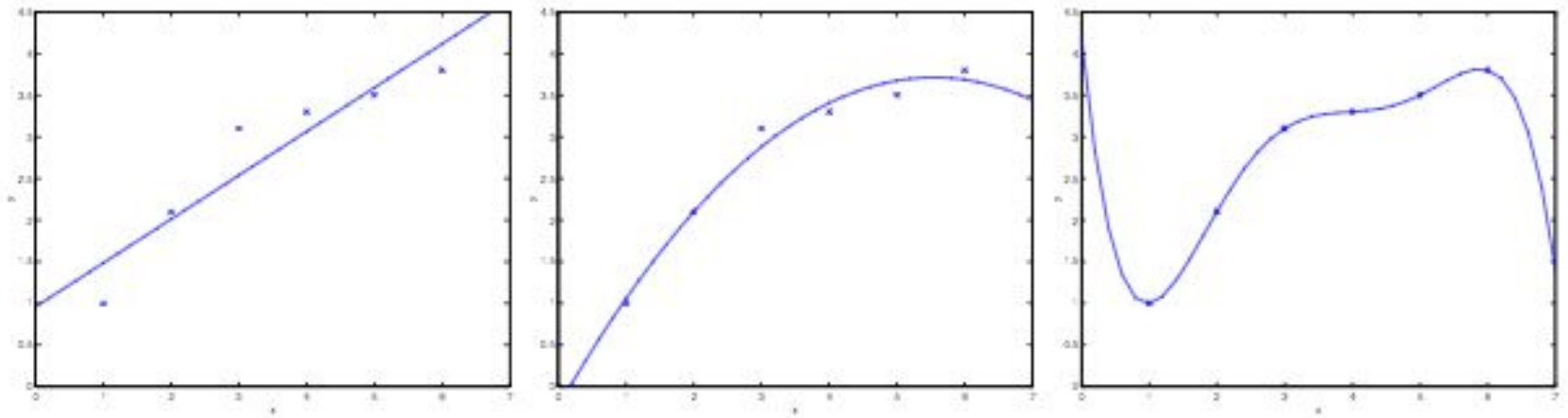
Feature engineering

- Add non-linear features to get non-linear decision boundaries



This is also a form of feature selection (more specifically feature engineering)

Overfitting Underfitting (from Lecture3)



Adding more non-linear features makes the line more curvy
(Adding more features also means more model parameters)

The curve can go directly to the outliers with enough parameters.

We call this effect **overfitting**

For the opposite case, having not enough parameters to model the data is called **underfitting**

Bias-Variance trade-off

- We will formulate overfitting and underfitting mathematically
- Using regression model

Regression with Gaussian noise

- $y = h(\mathbf{x}) + \varepsilon$
 - Where ε is normally distributed with mean zero and variance σ^2
 - The training data $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3) \dots\}$ is drawn from some distribution $P(\mathbf{x}, y)$ governing our universe!
 - Assume (\mathbf{x}_i, y_i) is iid
- Given D we can train a regressor $h_D(\mathbf{x})$
- We calculate the expected error (squared error) on new (\mathbf{x}, y) data with the regressor
 - $$E_{(\mathbf{x}, y)}[(h_D(\mathbf{x}) - y)^2] = \int \int_{\mathbf{x} \ y} (h_D(\mathbf{x}) - y)^2 \text{Pr}(\mathbf{x}, y) \partial y \partial \mathbf{x}$$
- But D is actually a random variable too!

Regression with Gaussian noise

- We calculate the expected error (squared error) on new (\mathbf{x}, y) data with the regressor

$$\bullet E_{(\mathbf{x}, y)}[(h_D(\mathbf{x}) - y)^2] = \int \int_{\mathbf{x} \ y} (h_D(\mathbf{x}) - y)^2 \text{Pr}(\mathbf{x}, y) \partial y \partial \mathbf{x}$$

- Consider parallel worlds, we can receive different training data D which yields different regression $h_D(\mathbf{x})$
- The expectation of error over all possible new test data point (\mathbf{x}, y) and different possible training data D is

$$E_{\substack{(\mathbf{x}, y) \sim P \\ D \sim P^n}} [(h_D(\mathbf{x}) - y)^2] = \int_D \int_{\mathbf{x}} \int_y (h_D(\mathbf{x}) - y)^2 P(\mathbf{x}, y) P(D) \partial \mathbf{x} \partial y \partial D$$

Regression with Gaussian noise

- This expression tells the expected quality of our model with random training data and a random test data

$$E_{\substack{(\mathbf{x}, y) \sim P \\ D \sim P^n}} \left[(h_D(\mathbf{x}) - y)^2 \right] = \int_D \int_{\mathbf{x}} \int_y (h_D(\mathbf{x}) - y)^2 P(\mathbf{x}, y) P(D) \partial \mathbf{x} \partial y \partial D$$

$$\underbrace{E_{\mathbf{x}, y, D} \left[(h_D(\mathbf{x}) - y)^2 \right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x}, D} \left[(h_D(\mathbf{x}) - \bar{h}(\mathbf{x}))^2 \right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x}, y} \left[(\bar{y}(\mathbf{x}) - y)^2 \right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}} \left[(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}))^2 \right]}_{\text{Bias}^2}$$

Variance, Bias, and noise

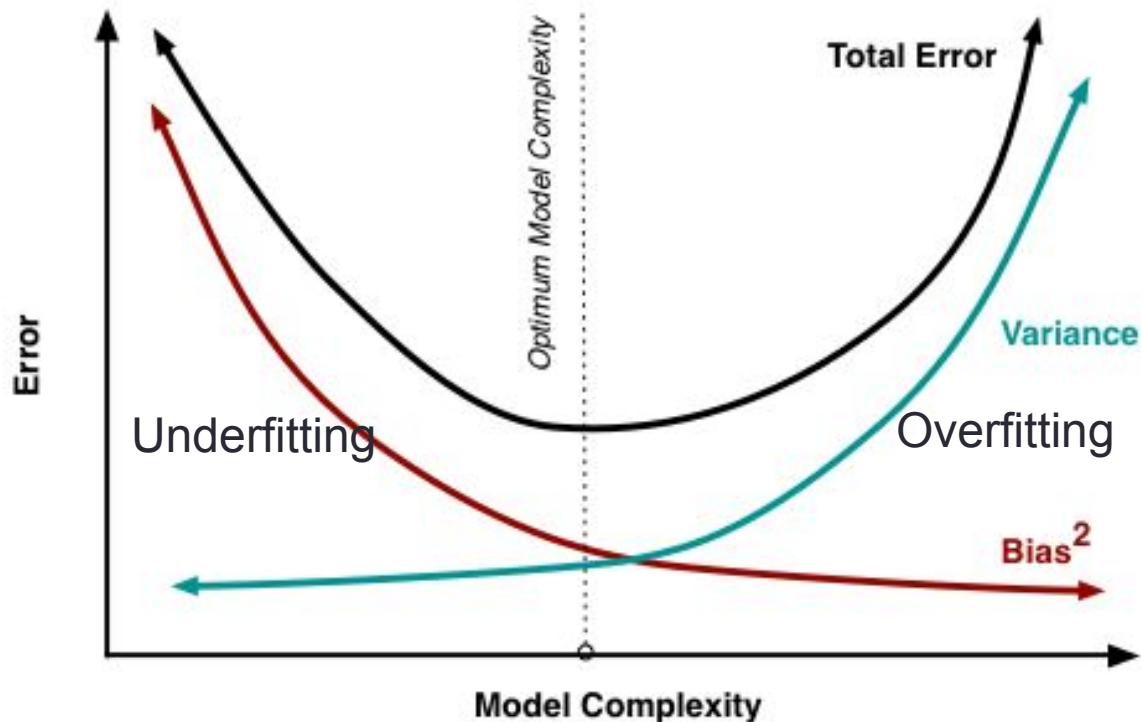
$$\underbrace{E_{\mathbf{x},y,D} \left[(h_D(\mathbf{x}) - y)^2 \right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D} \left[(h_D(\mathbf{x}) - \bar{h}(\mathbf{x}))^2 \right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y} \left[(\bar{y}(\mathbf{x}) - y)^2 \right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}} \left[(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}))^2 \right]}_{\text{Bias}^2}$$

- Variance: how your classifier changes if the training data changes. Measures **generalizability**.
- Bias: The model's inherent error. If you have **infinite training** data, you will have the average classifier \bar{h} and still left with this error.
 - For example, even with infinite training data, a linear classifier will still have errors if the distribution is non-linear.
- Noise: data-intrinsic noise. Noise from measurement, noise from feature extraction, etc. Regardless of your model this remains.

Bias-Variance

Underfitting-Overfitting

- Usually if you try to reduce the bias of your model, the variance will increase, and vice versa.
- Called the bias-variance trade-off

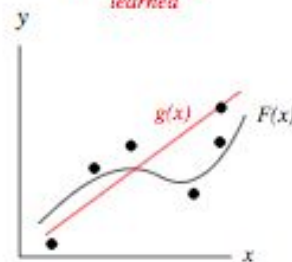
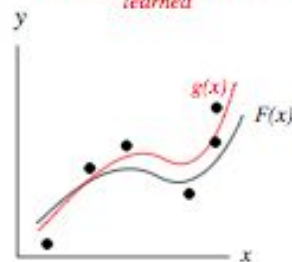
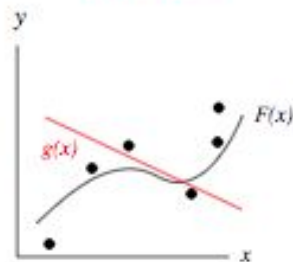
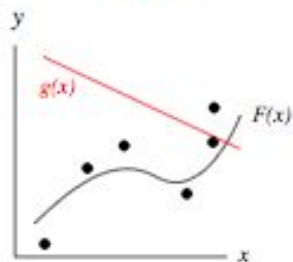
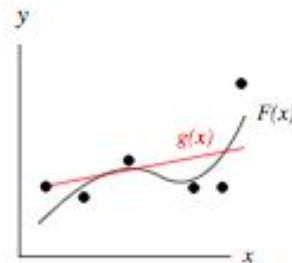
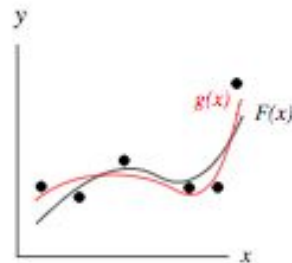
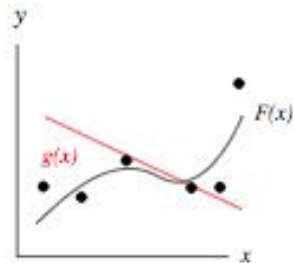
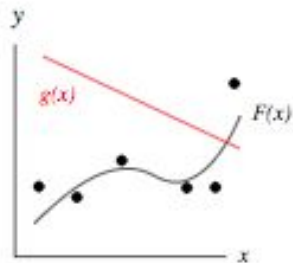
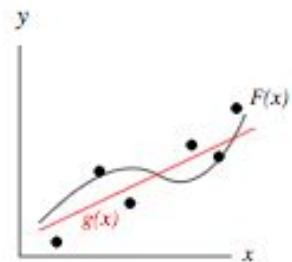
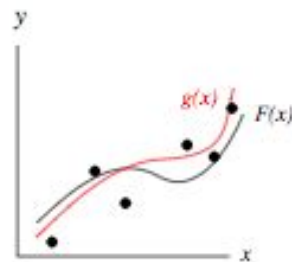
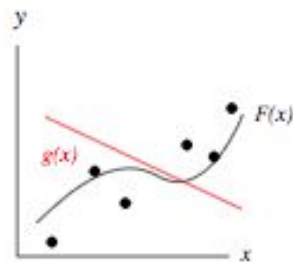
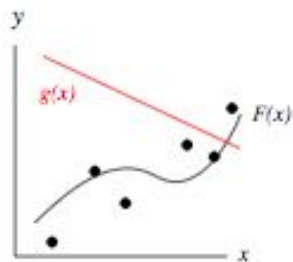


a)

b)

c)

d)

 $g(x) = \text{fixed}$ $g(x) = \text{fixed}$ $g(x) = a_0 + a_1x + a_2x^2 + a_3x^3$
learned $g(x) = a_0 + a_1x$
learned D_1  D_2  D_3 

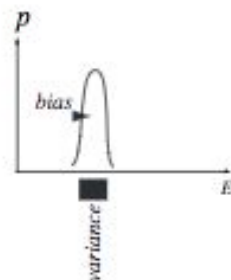
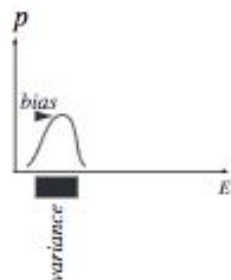
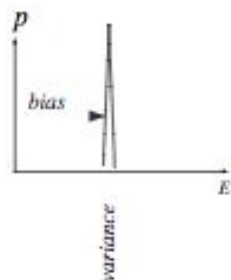
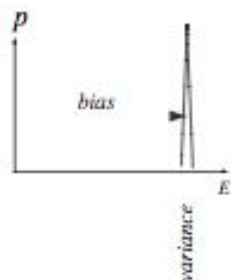
⋮

⋮

⋮

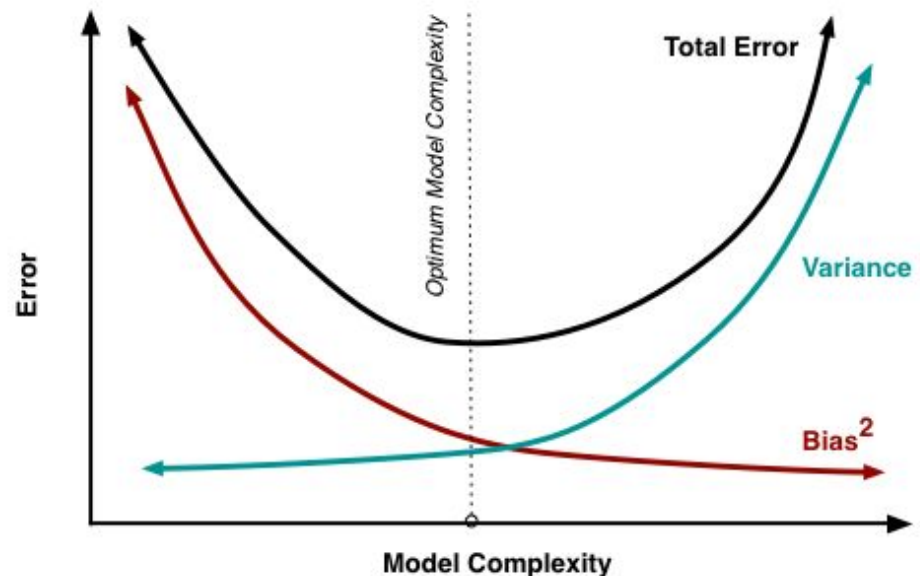
⋮

⋮



When to stop the update?

- Consider the updates of Logistic regression as trying to reduce the bias of the model
 - As we keep updating, the model overfits more to the training data
- We want to stop when the error on the validation set increases*
 - More on this later
- Validation test: a separate set that is use to measure overfitting
 - Training set
 - Validation set
 - Test set








Best submissions

- Accuracy 77.990
- Use scaling to $[0,1]$
- Learning rate decay
- Use scaling to unit normal for age, $[0,1]$ for the rest

More tricks?

- <http://ahmedbesbes.com/how-to-score-08134-in-titanic-kaggle-challenge.html>
- Feature Engineering/selection
- Parameter tuning
- Try different models

449	▲ 62...	Kaustubh Kulkarni 2		0.81340	6	6h
450	new	AshishDoshi		0.81340	1	5h
451	new	SouravKarwa		0.81340	2	31m
452	▲ 18...	Ahmed Besbes		0.81340	15	now
Your Best Entry ↑ Your submission scored 0.81340, which is not an improvement of your best score. Keep trying!						
453	▼ 7	Clement Sengelen		0.80861	11	2mo

The Bayes Lecture

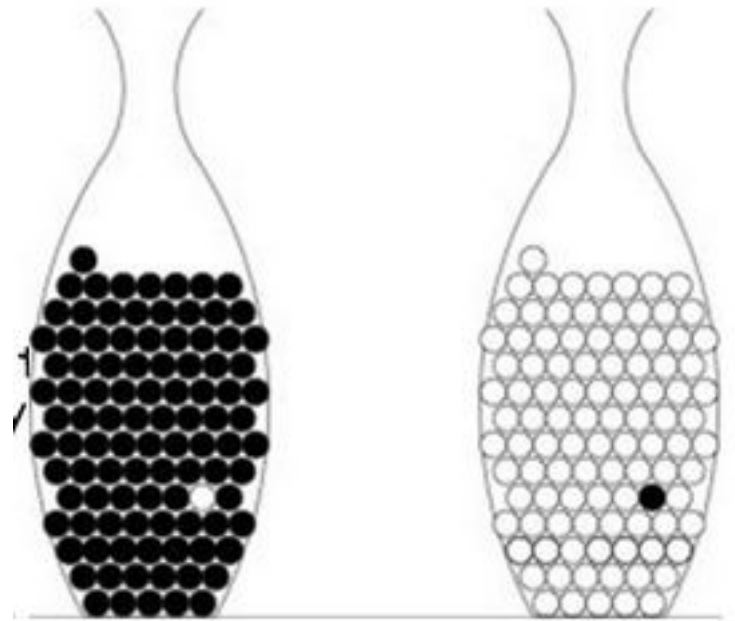
- MLE vs MAP
- Bayes Decision Rule
- Naïve Bayes

Bayes Rule & Learning theory

- x – observed data
- w_i – probability that x comes from class i

$$p(w_i|x) = \frac{p(x|w_i)p(w_i)}{p(x)}$$

$$\text{Posterior} = \frac{\text{likelihood} * \text{prior}}{\text{evidence}}$$

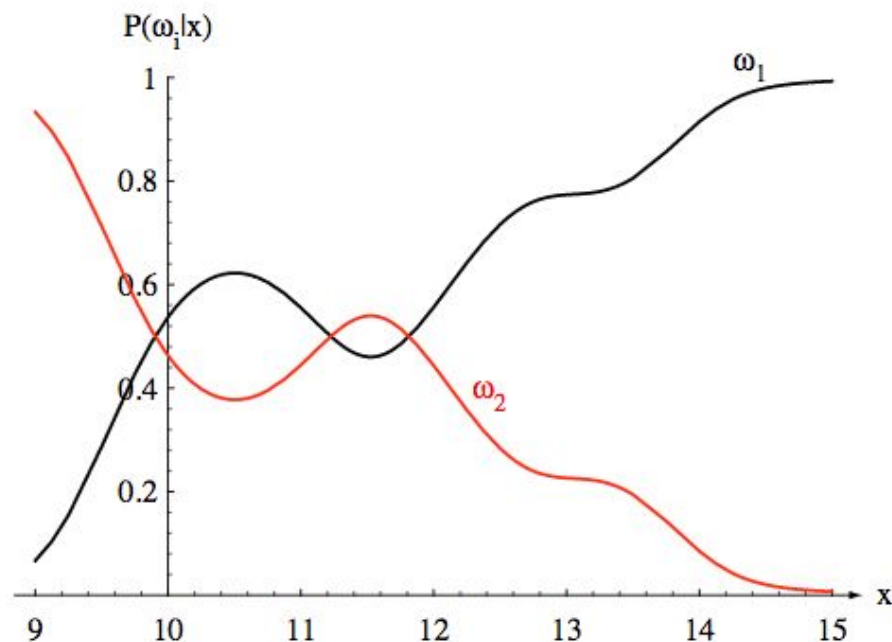
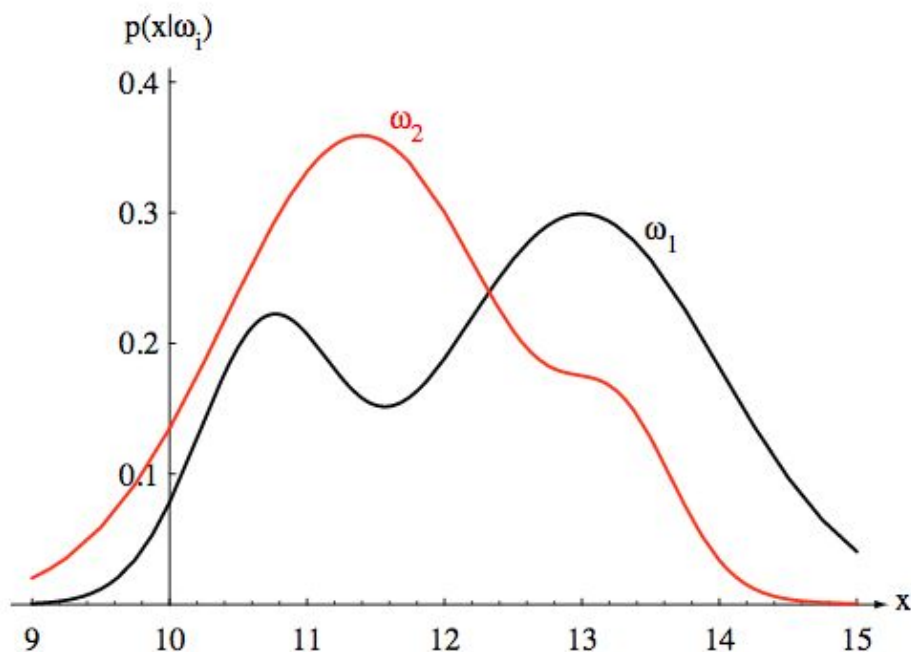


<http://slideplayer.com/slide/8845876/>

This relationship between the likelihood and the posterior will lead to two different approaches we can do parameter estimation

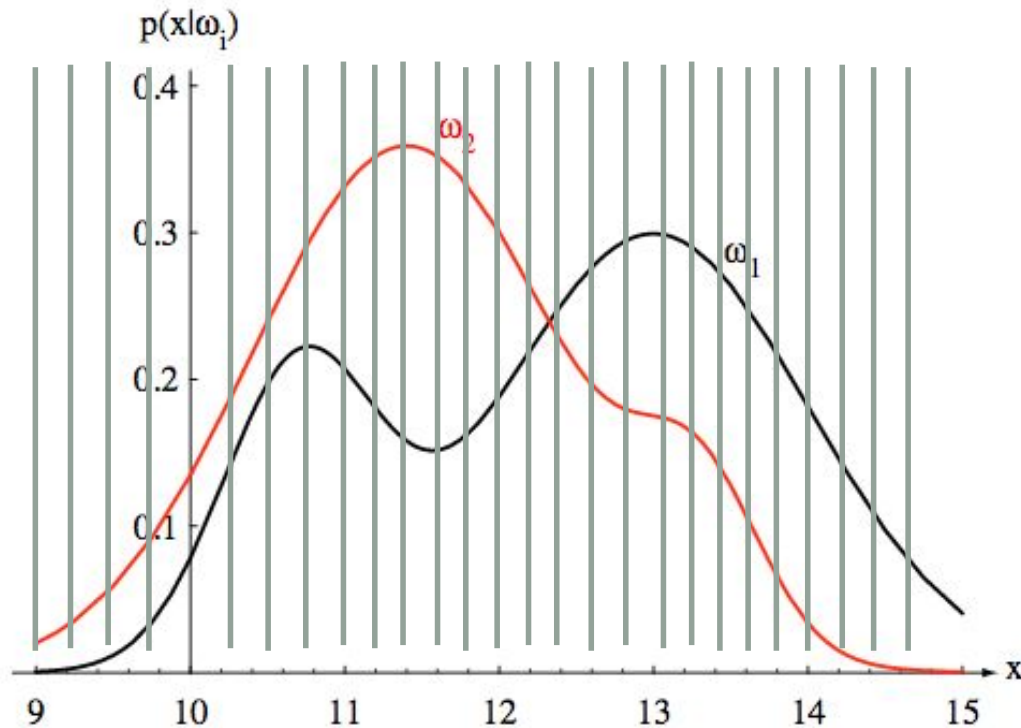
A simple decision rule

- If we can know either $p(x|w)$ or $p(w|x)$ we can make a classification guess



Goal: Find $p(x|w)$ or $p(w|x)$

A simple way to estimate $p(x|w)$

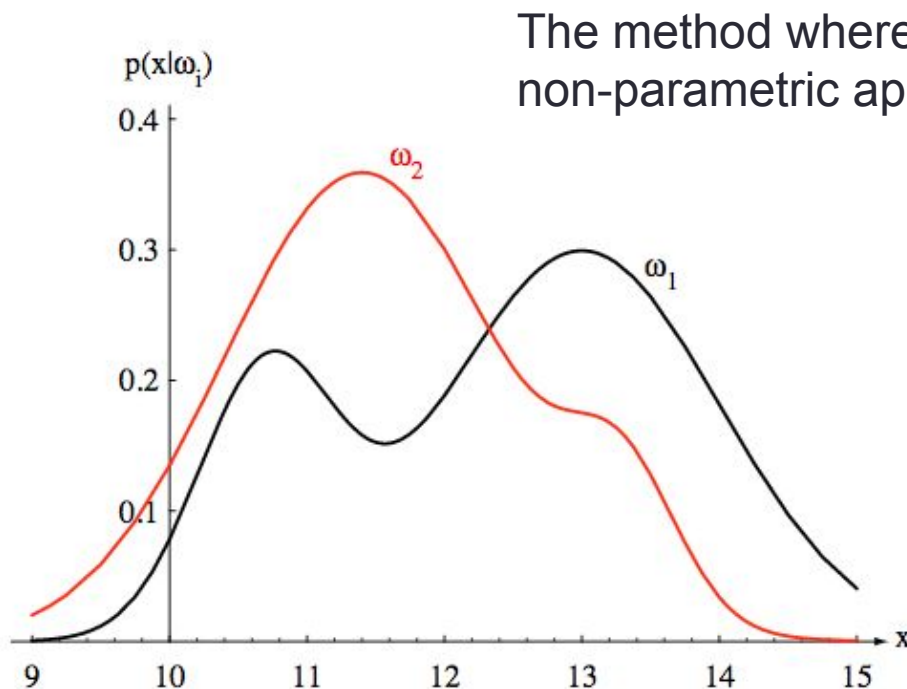


Make a histogram!

What happens if there is no data in a bin?

The parametric approach

- We **assume** $p(x|w)$ or $p(w|x)$ follow some distributions with parameter θ



$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$

Goal: Find θ so that we can estimate $p(x|w)$ or $p(w|x)$

Maximum Likelihood Estimate (MLE)

$$p(w_i|x) = \frac{p(x|w_i)p(w_i)}{p(x)}$$

$$\text{Posterior} = \frac{\text{likelihood} * \text{prior}}{\text{evidence}}$$

- Maximizing the likelihood (probability of data given model parameters)

$$p(\mathbf{x}|\theta) = L(\theta) \leftarrow \text{This assumes the data is fixed}$$

- Usually done on log likelihood
- Take the partial derivative wrt to θ and solve for the θ that maximizes the likelihood

MLE of binomial trials

- A coin with bias is tossed N times. k times are heads. Find θ , the probability of the coin landing head.

MLE of Gaussian

Maximum Likelihood Estimate (MLE)

$$p(w_i|x) = \frac{p(x|w_i)p(w_i)}{p(x)}$$

$$\text{Posterior} = \frac{\text{likelihood} * \text{prior}}{\text{evidence}}$$

- Maximizing the likelihood (probability of data given model parameters)

$$p(\mathbf{x}|\theta) = L(\theta) \leftarrow \text{This assumes the data is fixed}$$

- Usually done on log likelihood
- Take the partial derivative wrt to θ and solve for the θ that maximizes the likelihood

Maximum A Posteriori (MAP) Estimate

MLE

- Maximizing the likelihood (probability of data given model parameters)

$$\operatorname{argmax}_{\theta} p(\mathbf{x}|\theta)$$

$$p(\mathbf{x}|\theta) \\ = L(\theta)$$

- Usually done on log likelihood
- Take the partial derivative wrt to θ and solve for the θ that maximizes the likelihood

MAP

- Maximizing the posterior (model parameters given data)

$$\operatorname{argmax}_{\theta} p(\theta|\mathbf{x})$$

- But we don't know $p(\theta|\mathbf{x})$

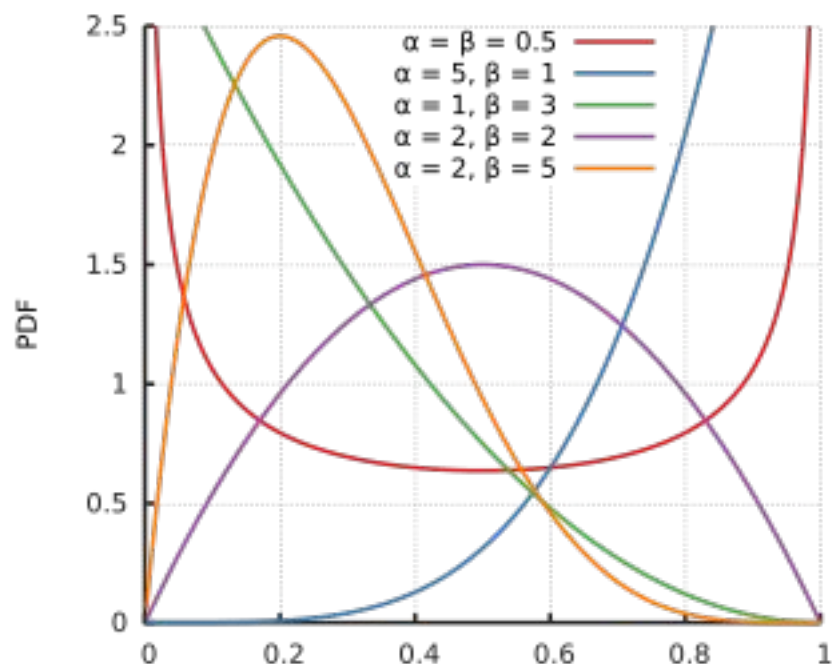
- Use Bayes rule

$$p(\theta|\mathbf{x}) = \frac{p(\mathbf{x}|\theta)p(\theta)}{p(\mathbf{x})}$$

- Taking the argmax for θ we can ignore $p(\mathbf{x})$
- $\operatorname{argmax}_{\theta} p(\mathbf{x}|\theta) p(\theta)$

MAP on binomial trials

- A coin with bias is tossed N times. k times are heads. Find θ , the probability of the coin landing head.
- We assume θ comes from a Beta distribution



$$p(x) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1}$$

The parameters α, β are assumed given and constant throughout the derivation (Called **hyperparameters**)
 α, β represents our knowledge of the world

MAP on Gaussian

- We know x is Gaussian with unknown mean μ that we need to estimate and known variance σ^2
- Assume the prior of μ is $N(\mu_0, \sigma_0^2)$
- MAP estimate of μ is

$$\mu_n = \left(\frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2} \right) \left[\frac{1}{n} \sum_{i=1}^n x_i \right] + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2} \mu_0$$

Notes of MAP estimate

- Usually harder to estimate than MLE
- If we use a uniform prior distribution for θ
 - MAP estimate = MLE
- Given infinite data
 - MAP estimate converges to MLE
- MAP is useful when you have less data, so you need additional knowledge about the domain
 - MAP estimate tends to converge faster than MLE even with an arbitrary distribution
 - Can help prevent overfitting
- **Useful for model adaptation** (MAP adaptation)
 - Learn MLE on larger dataset, use this as your prior distribution
 - Learn MAP estimate on your dataset

Notes on MAP estimate

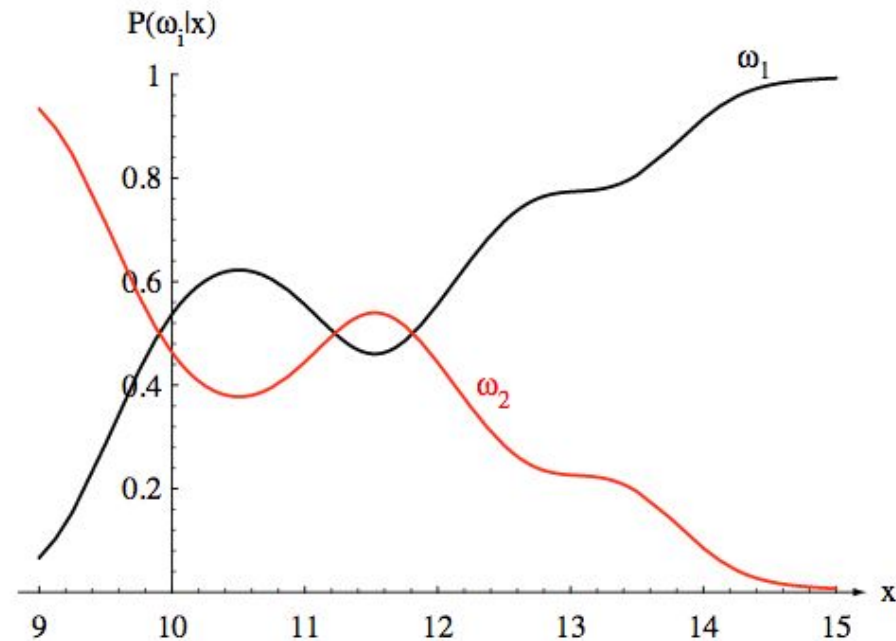
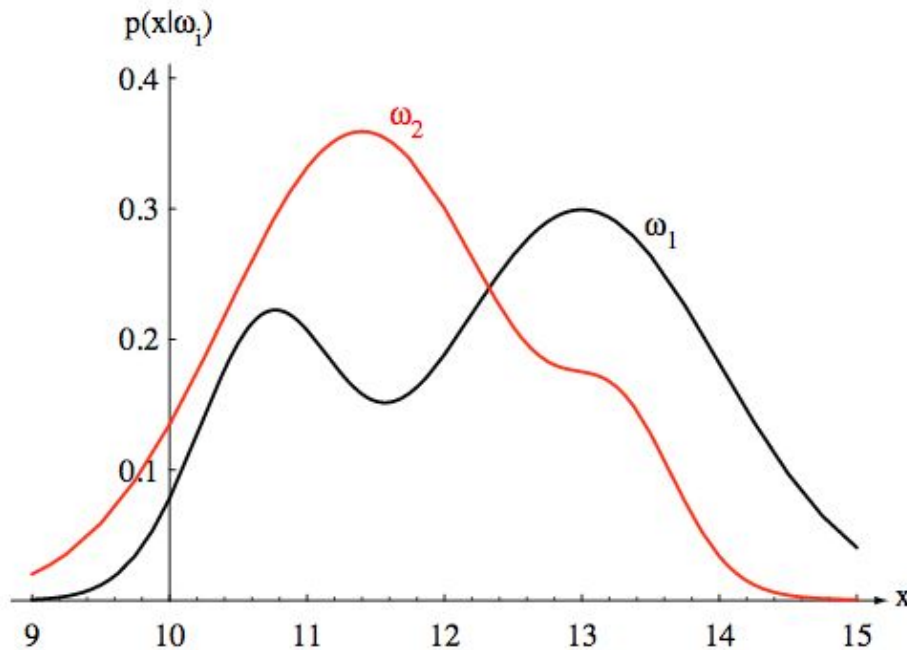
- What is the choice of distribution for the prior?
 - Pick the conjugate priors
 - Makes the math simple without loss of generalizability

https://en.wikipedia.org/wiki/Conjugate_prior

Likelihood	Model parameters	Conjugate prior distribution	Prior hyperparameters	Posterior hyperparameters	h
Bernoulli	p (probability)	Beta	α, β	$\alpha + \sum_{i=1}^n x_i, \beta + n - \sum_{i=1}^n x_i$	α β
Binomial	p (probability)	Beta	α, β	$\alpha + \sum_{i=1}^n x_i, \beta + \sum_{i=1}^n N_i - \sum_{i=1}^n x_i$	α β
Negative binomial with known failure number, r	p (probability)	Beta	α, β	$\alpha + \sum_{i=1}^n x_i, \beta + rn$	α β (i e: r
Poisson	λ (rate)	Gamma	k, θ	$k + \sum_{i=1}^n x_i, \frac{\theta}{n\theta + 1}$	k $\frac{1}{\theta}$
			α, β ^[note 3]	$\alpha + \sum_{i=1}^n x_i, \beta + n$	α in
	p				

A simple decision rule

- If we can know either $p(x|w)$ or $p(w|x)$ we can make a classification guess



Goal: Find $p(x|w)$ or $p(w|x)$ by finding the parameter of the distribution

Likelihood ratio test

- If $P(w_1|x) > P(w_2|x)$, that x is more likely to be class w_1
- Again we know $P(x|w_1)$ is more intuitive and easier to calculate than $P(w_1|x)$

- Our classifier becomes

- $$P(x|w_1)P(w_1) \quad ? \quad P(x|w_2)P(w_2)$$

- $$\frac{P(x|w_1)}{P(x|w_2)}$$

?

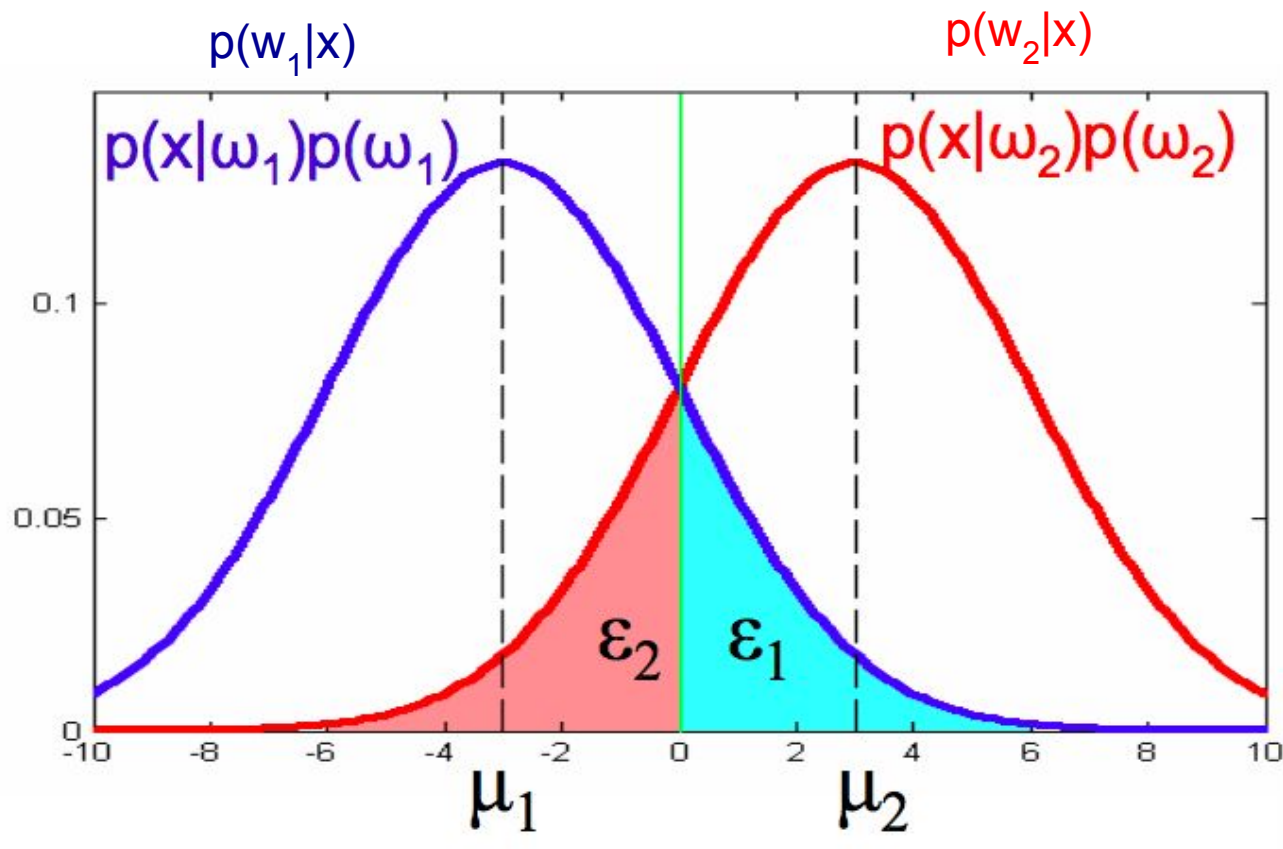
$$\frac{P(w_2)}{P(w_1)}$$

Ratio of priors

Likelihood ratio

Notes on likelihood ratio test (LRT)

- LRT minimizes the classification error (all errors are equally bad)



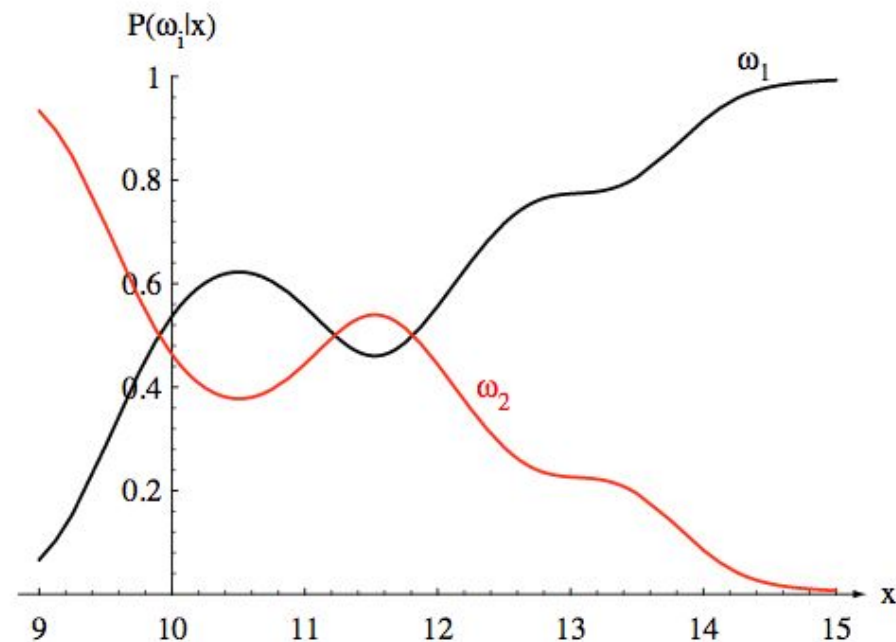
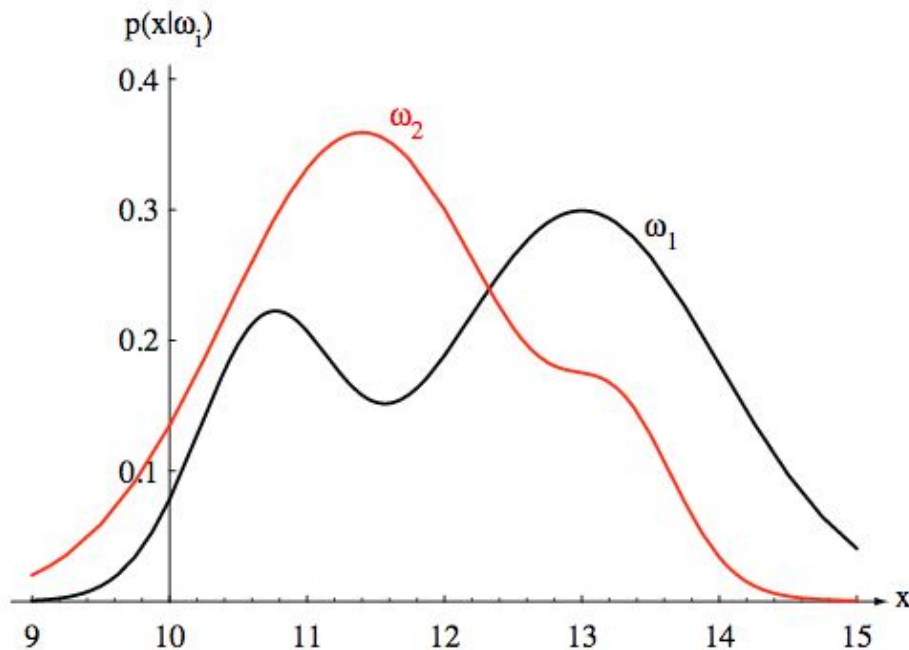
Notes on LRT

- If $P(w_1|x) > P(w_2|x)$, that x is more likely to be class w_1
 - Also known as MAP decision rule
 - The classifier is sometimes called the **Bayes classifier**
- If we do not want to treat all error equally, we can assign different loss to each error, and minimize the expected loss. This is called **Bayes loss/risk classifier**
- $$\frac{P(x|w_1)}{P(x|w_2)} \quad ? \quad \frac{P(w_2)(L_{1|2} - L_{2|2})}{P(w_1)(L_{2|1} - L_{1|1})}$$
- When we treat errors equally we refer to the **zero-one loss**
- $L_{1|2} = 1, L_{2|2} = 0, L_{2|1} = 1, L_{1|1} = 0$

Notes on LRT

- If we treat the priors as equal, we get the **maximum likelihood criterion**

- $\frac{P(x|w_1)}{P(x|w_2)} \quad ? \quad 1$



Naïve Bayes

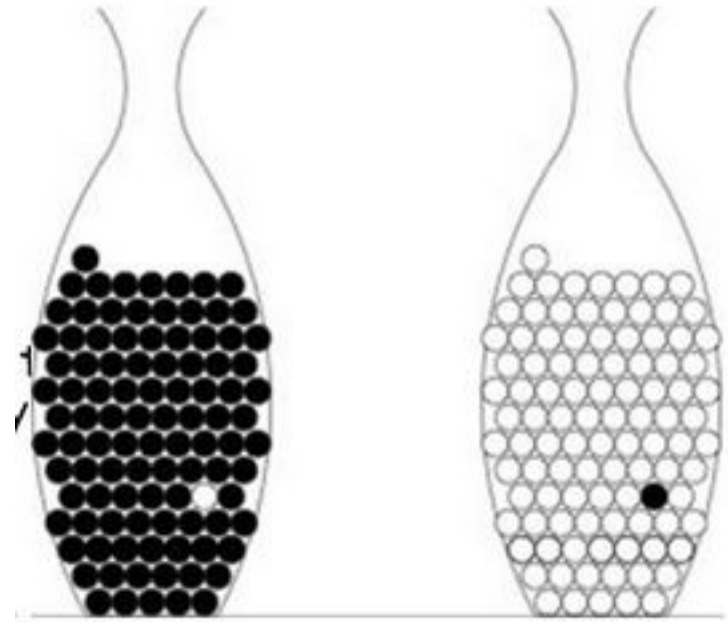
- Below is the LRT or the Bayes classifier

$$P(x|w_1)P(w_1) \quad ? \quad P(x|w_2)P(w_2)$$

- What about Naïve Bayes?
- Here x is a vector with m features $[x_1, x_2, \dots, x_m]$
- $P(x|w_i)$ is $m+1$ dimensional
 - Sometimes too hard to model, not enough data, overfit, *curse of dimensionality*, etc.
- Assumes x_1, x_2, \dots, x_m independent given w_i (conditional independence)
 - What does this mean?

Conditional independence

- x_1 x_2 are two draws from the same urn
- w represents the urn being drawn



Naïve Bayes

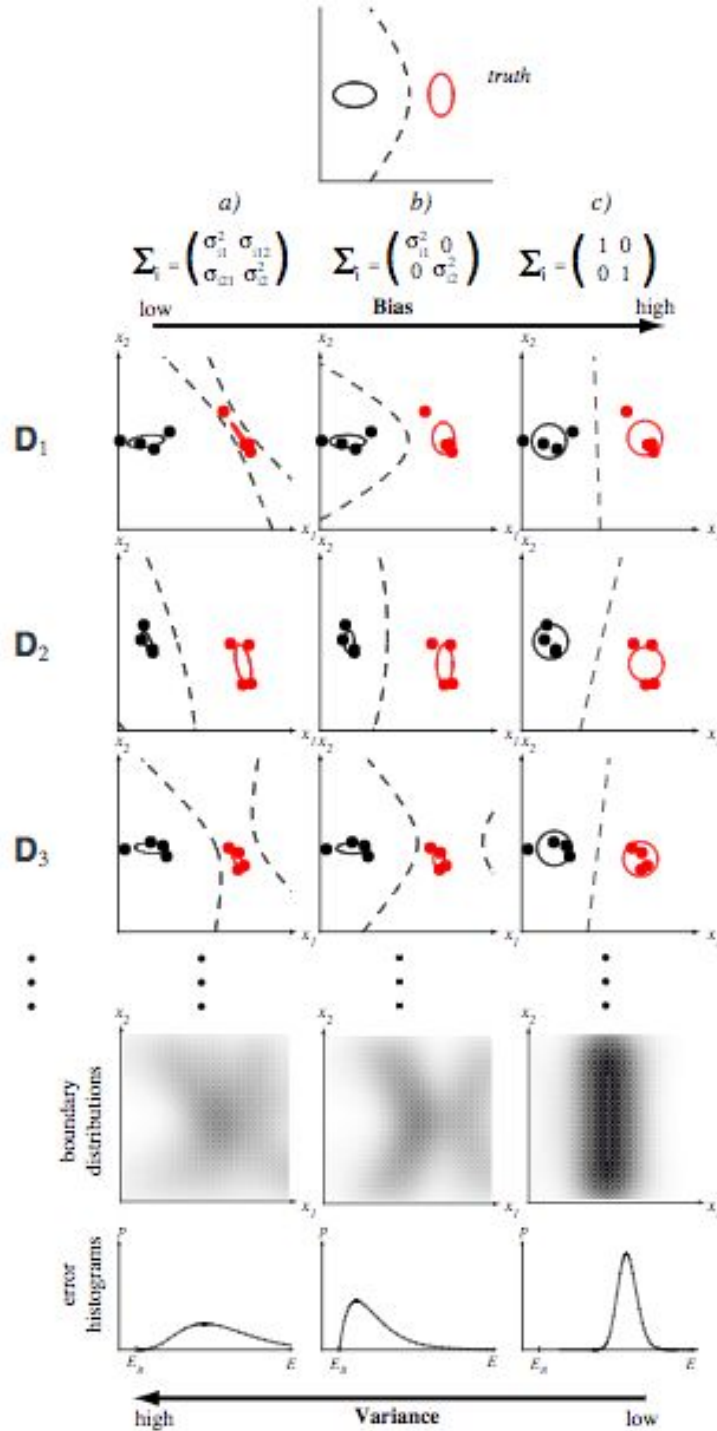
- Below is the LRT or the Bayes classifier

$$P(x|w_1)P(w_1) \quad ? \quad P(x|w_2)P(w_2)$$

- What about Naïve Bayes?
- Here x is a vector with m features $[x_1, x_2, \dots, x_m]$
- $P(x|w_i)$ is $m+1$ dimensional
 - Sometimes too hard to model, not enough data, overfit, *curse of dimensionality*, etc.
- Assumes x_1, x_2, \dots, x_m independent given w_i (conditional independence)

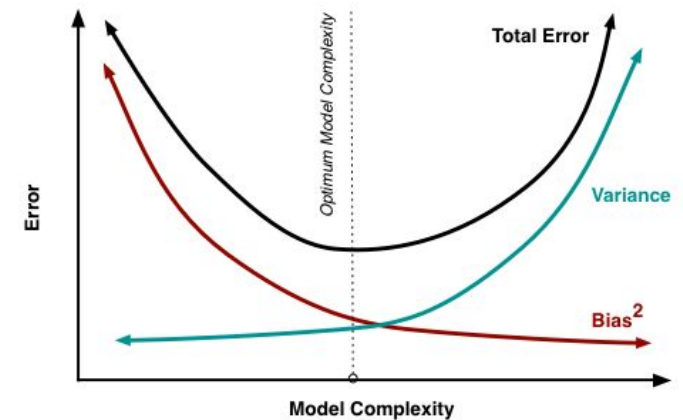
Naïve Bayes

- $P(\mathbf{x}|w_i)P(w_i) = P(w_i) \prod_j P(x_j|w_i)$
- This assumption simplifies the calculation
- Note that we do not say anything about what kind of distribution $P(x_j|w_i)$ is.
 - In the homework you will play with this
 - Clean data
 - Estimate $P(x_j|w_i)$ using MLE
 - Do prediction



Summary

- Normalization
- Bias-Variance trade-off
 - Overfitting and underfitting
- MLE vs MAP estimate
 - How to use the prior
- LRT (Bayes Classifier)
 - Naïve Bayes



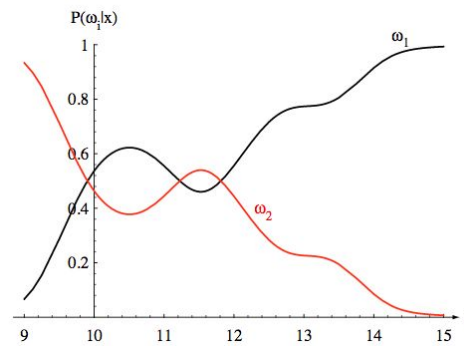
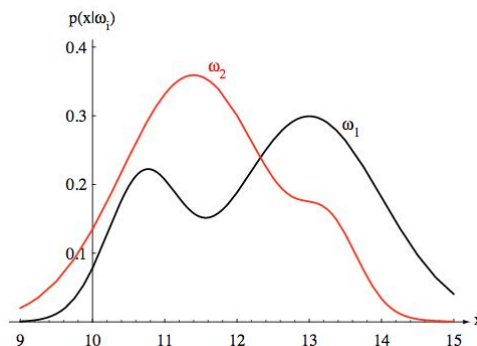
$$\frac{P(x|w_1)}{P(x|w_2)}$$

Likelihood ratio

?

$$\frac{P(w_2)}{P(w_1)}$$

Ratio of priors



Next homework