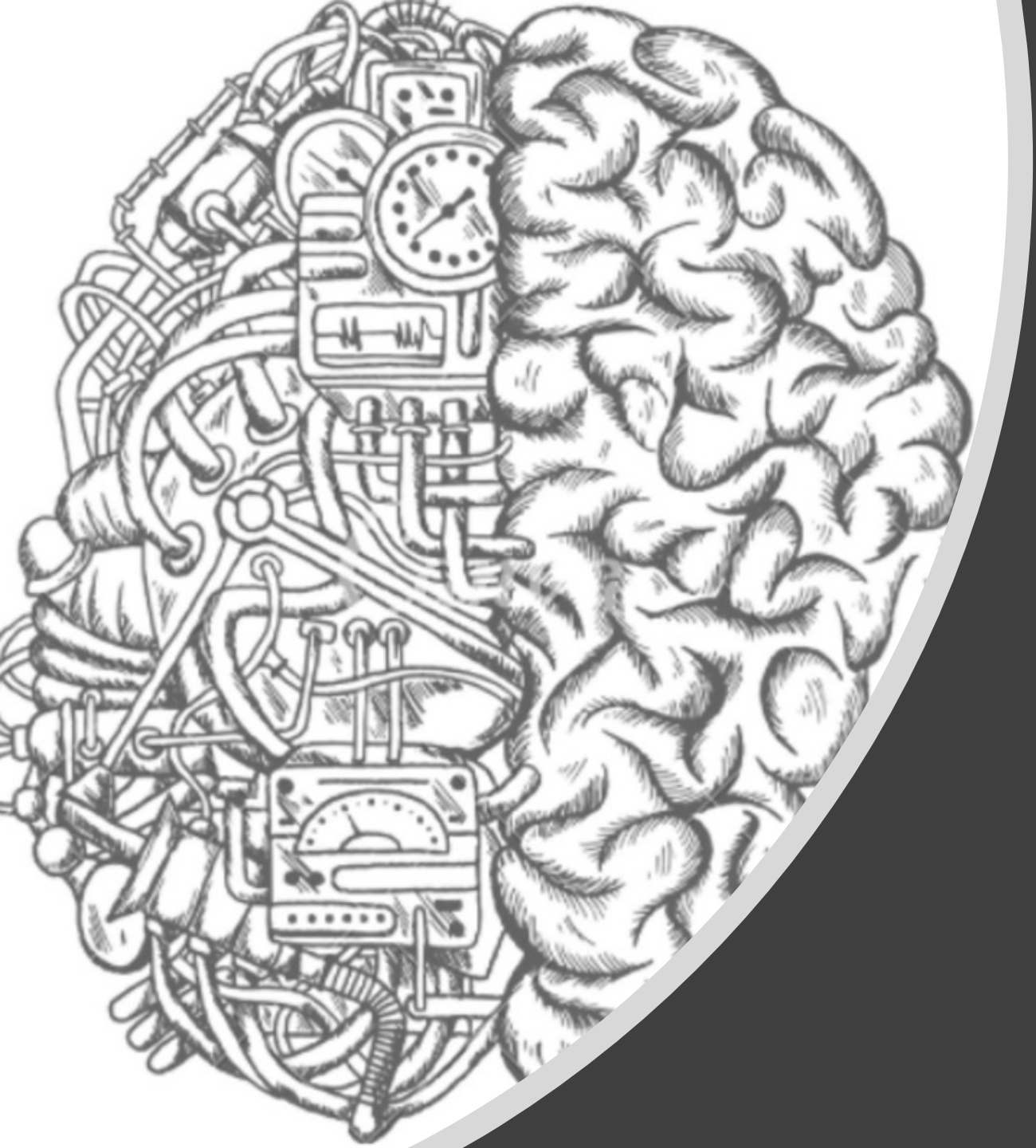


Machine Learning: Models and Applications

Lecture 4

Lecturer: Xinchao Wang

xinchao@nus.edu.sg

National University of Singapore

Overview

- Parameter Estimation
  - Frequentist or Maximum Likelihood approach (cont.)
  - Bayesian approach (Barber Ch. 8 and DHS Ch. 3)
- Cross-validation
- Overfitting
- Non-parametric Techniques

MLE Classifier Example

Data

- Pima Indians Diabetes Database
  - http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes
  - Number of Instances: 768
  - Number of Attributes: 8 plus class
  - Class Distribution: (class value 1 is interpreted as "tested positive for diabetes")
  - Class Value Number of instances

    0 500

    1 268

Data

Attributes: (all numeric-valued)

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

Simple MLE Classifier

```
data = dlmread('pima-indians-diabetes.data');

data = reshape(data,[],9);

% use randperm to re-order data.
% ignore if not using Matlab
rp = randperm(length(data));
data=data(rp,:);

train_data = data(1:length(data)/2,:);
test_data = data(length(data)/2+1:end,:);
```

```
% pick a feature
active_feat = 3;

% training
mean1 =
  mean(train_data(train_data(:,9)==0,active_feat))
mean2 =
  mean(train_data(train_data(:,9)==1,active_feat))

var1 = var(train_data(train_data(:,9)==0,active_feat))
var2 = var(train_data(train_data(:,9)==1,active_feat))

prior1tmp = length(train_data(train_data(:,9)==0));
prior2tmp = length(train_data(train_data(:,9)==1));

prior1 = prior1tmp/(prior1tmp+prior2tmp)
prior2 = prior2tmp/(prior1tmp+prior2tmp)
```

```
% testing
correct=0;
wrong=0;


for i=1:length(test_data)
    lklhood1 = exp(-(test_data(i,active_feat)-mean1)^2/(2*var1))
  /sqrt(var1);
    lklhood2 = exp(-(test_data(i,active_feat)-mean2)^2/(2*var2));
  /sqrt(var2);


    post1 = lklhood1*prior1;
    post2 = lklhood2*prior2;


    if(post1 > post2 && test_data(i,9) == 0)
        correct = correct+1;
    elseif(post1 < post2 && test_data(i,9) == 1)
        correct = correct+1;
    else
        wrong = wrong+1;
    end
end
```

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Training/Test Split

- Randomly split dataset into two parts:
  - Training data
  - Test data
- Use training data to optimize parameters
- Evaluate error using test data

Training/Test Split

- How many points in each set?
- Very hard question
  - Too few points in training set, learned classifier is bad
  - Too few points in test set, classifier evaluation is insufficient
- Cross-validation
- Leave-one-out cross-validation
- Bootstrapping

Cross Validation

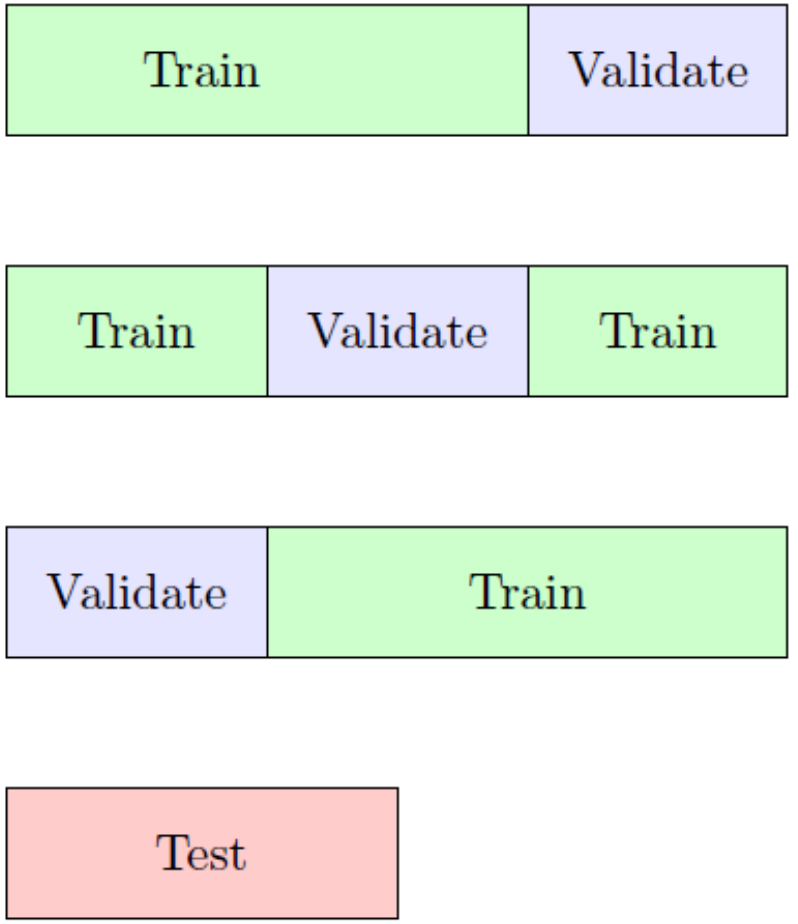
- In practice
- Available data => training and validation
- Train on the training data
- Test on the validation data
- k-fold cross validation:
  - Data randomly separated into k groups
  - Each time k−1 groups used for training and one as testing

Cross Validation and Test Accuracy

- If we select parameters so that CV is highest:
  - Does CV represent future test accuracy?
  - Slightly different
- So split available data with class labels, into:
  - training
  - validation
  - testing

Cross Validation and Test Accuracy

- Using CV on training + validation
- Classify test data with the best parameters from CV



Overfitting

- Prediction error: probability of test pattern not in class with max posterior (true)
- Training error: probability of test pattern not in class with max posterior (estimated)
- Classifier optimized w.r.t. training error
  - Training error: optimistically biased estimate of prediction error

Overfitting

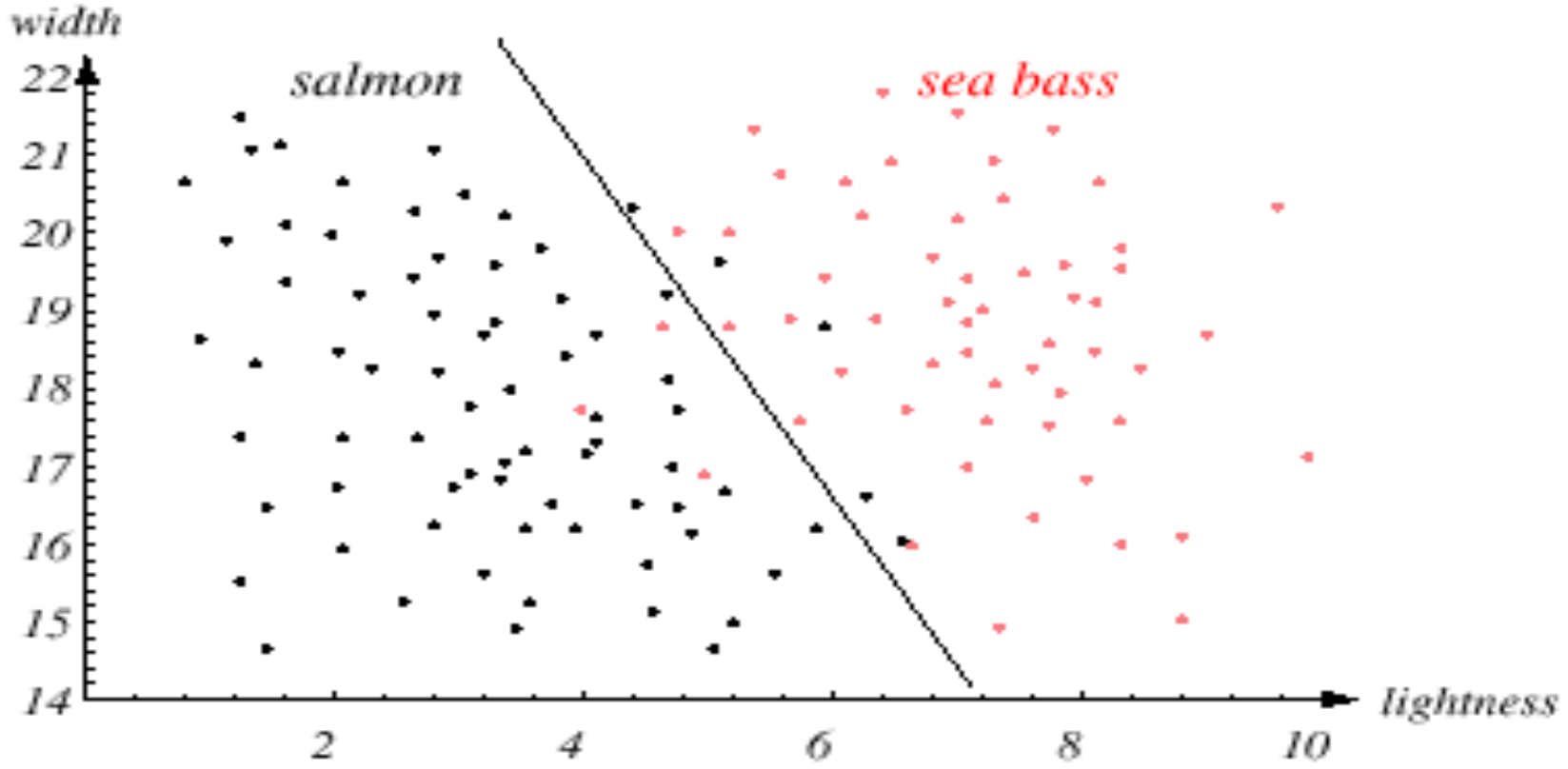
Overfitting: a learning algorithm overfits the ***training data*** if it outputs a solution **w** when another solution **w'** exists such that:

$$\text{error}_{\text{train}}(w) < \text{error}_{\text{train}}(w')$$

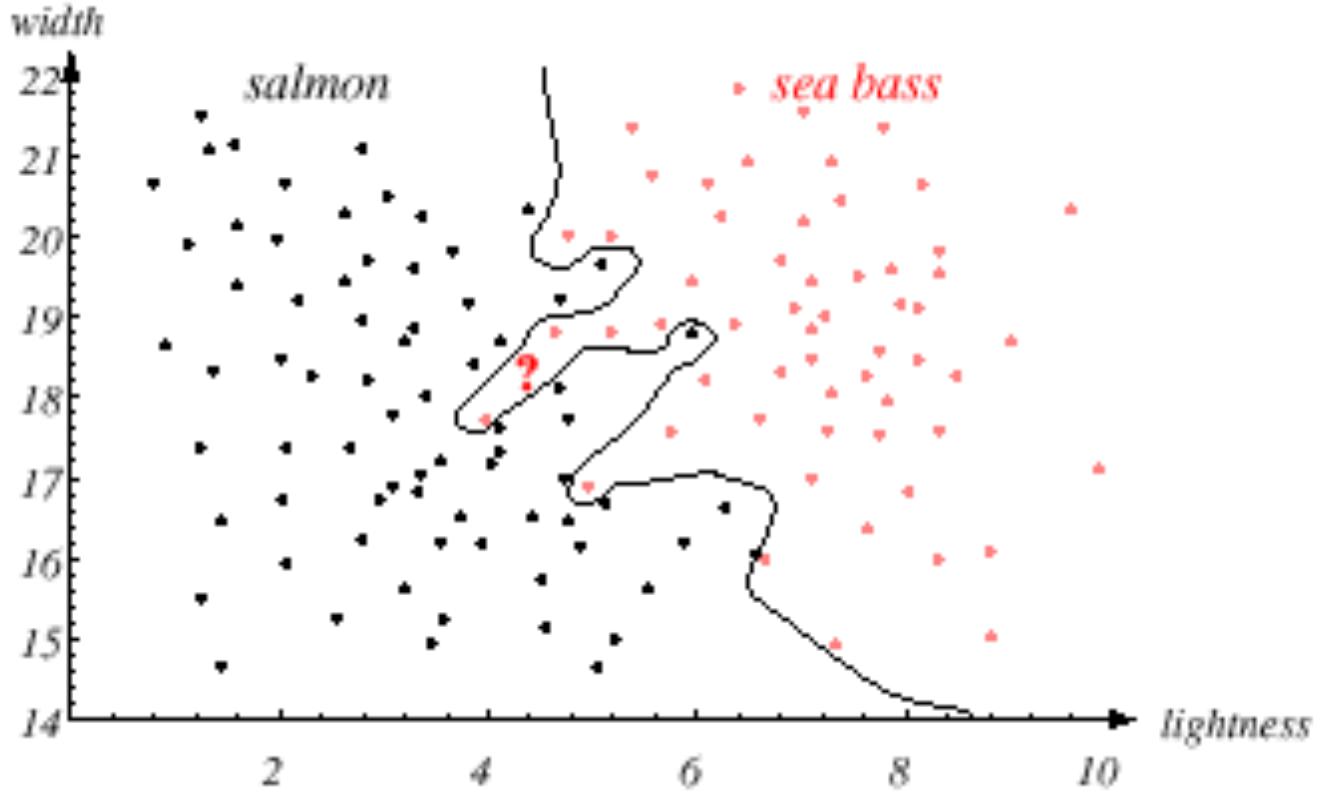
AND

$$\text{error}_{\text{true}}(w') < \text{error}_{\text{true}}(w)$$

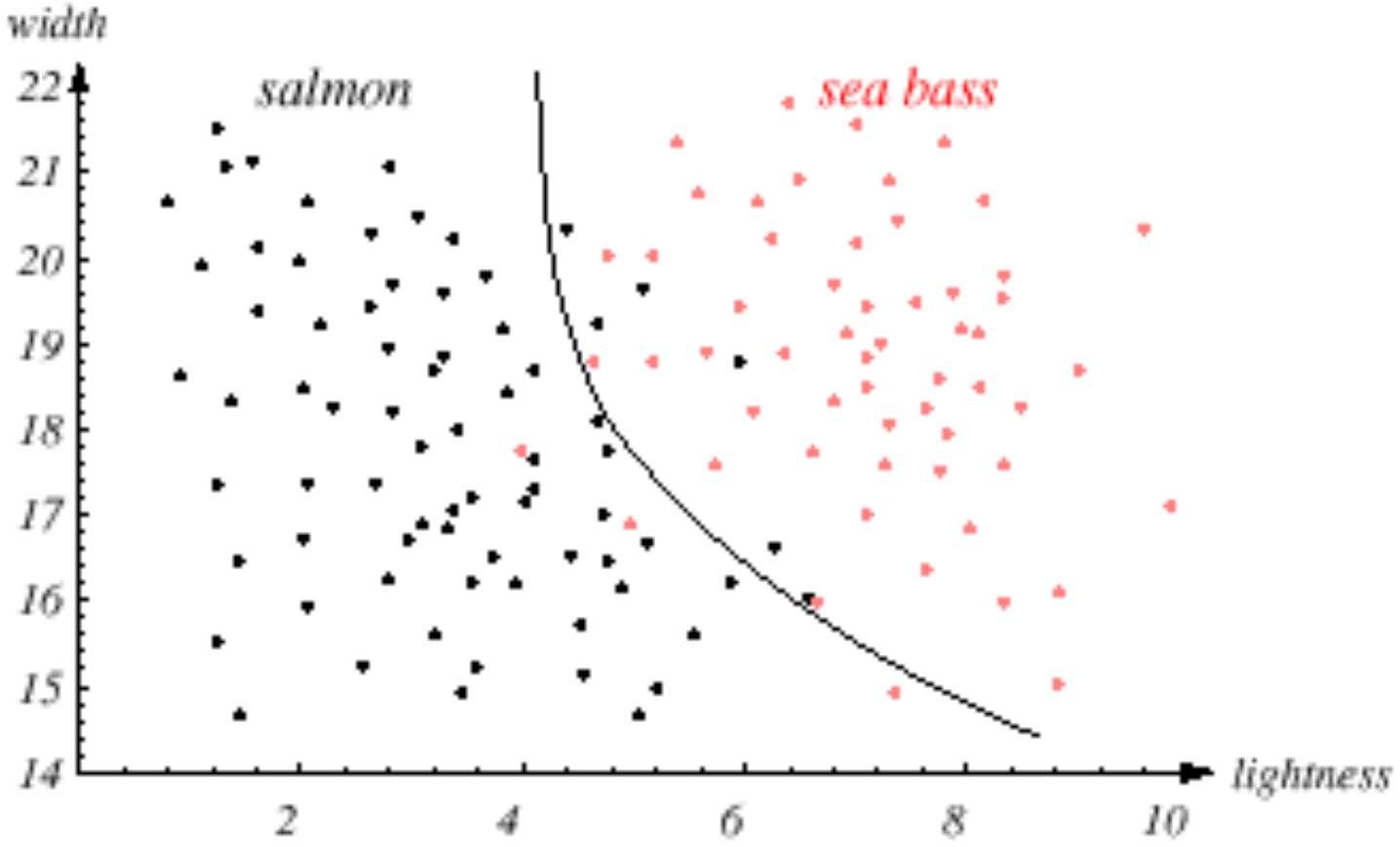
Fish Classifier from DHS Ch. 1



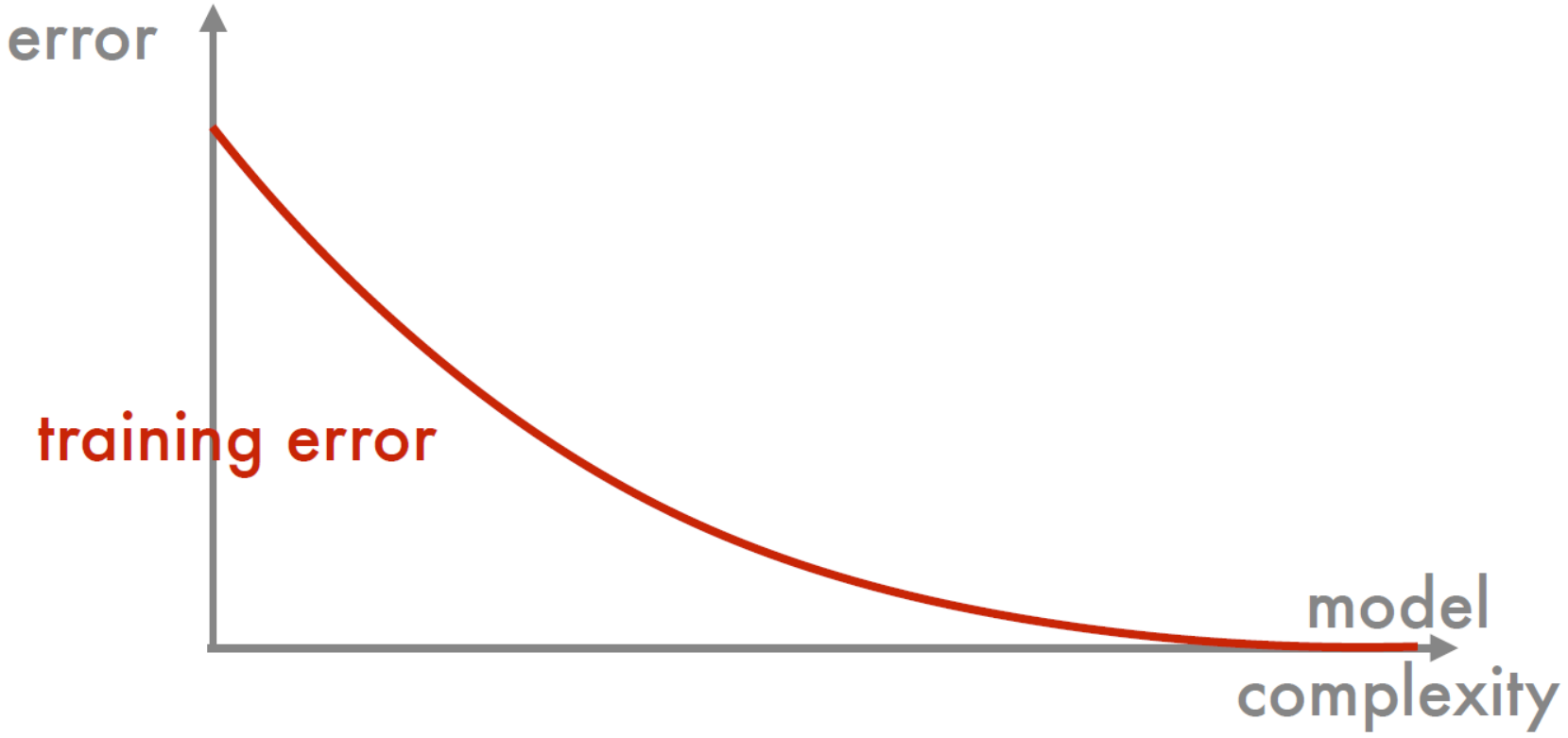
Minimum Training Error



Final Decision Boundary

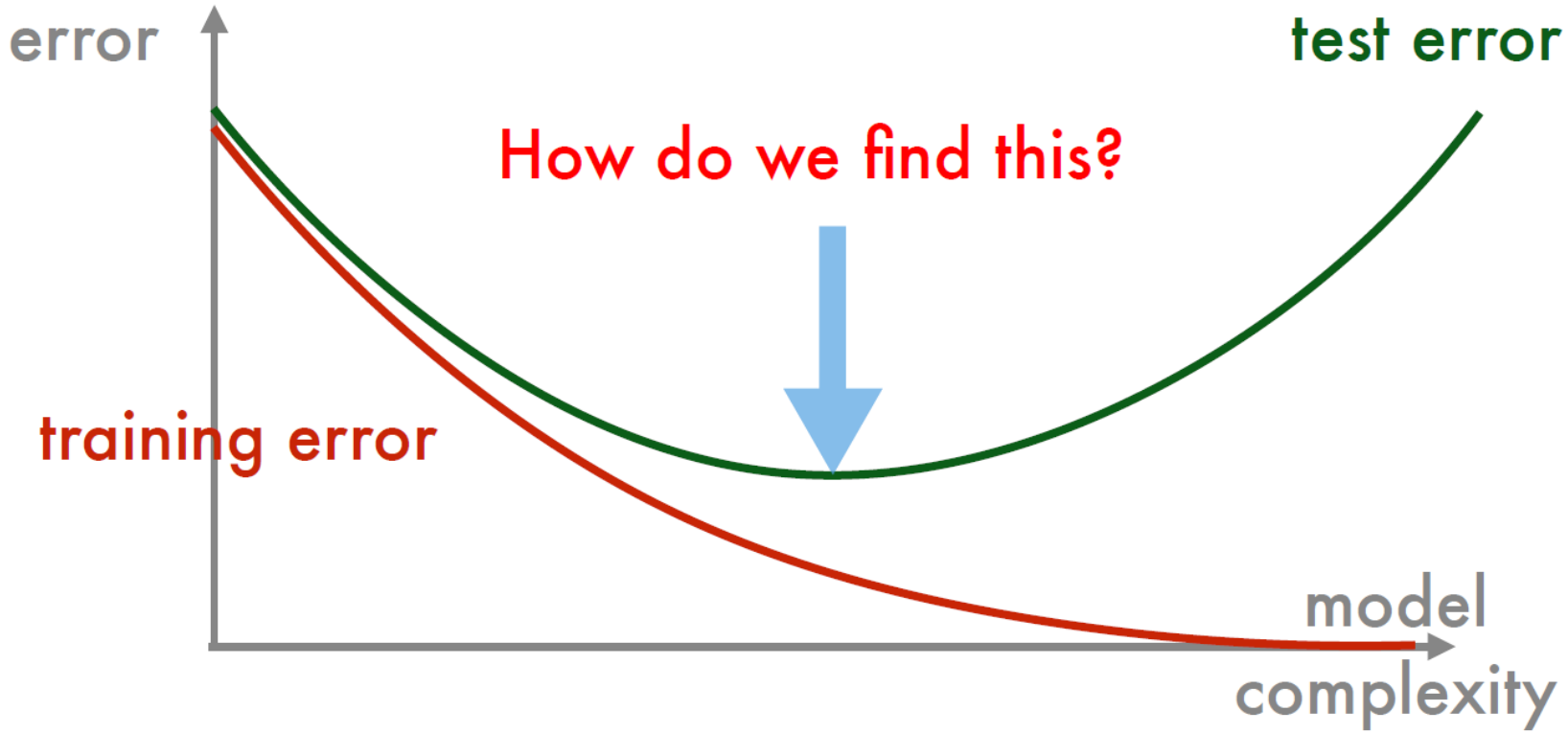


Typical Behavior



Slide credit: A. Smola

Typical Behavior



Slide credit: A. Smola

Maximum a posteriori (MAP) Estimation

MLE

$$\theta_{MLE} = \arg\max_{\theta} P\,(D|\theta)$$
$$= \arg\max_{\theta} \log P(D|\theta)$$
$$= \arg\max_{\theta} \log \Pi_i P(x_i|\theta)$$
$$= \arg\max_{\theta} \Sigma_i \log P(x_i|\theta)$$

MAP

- Recall Bayesian

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \propto P(D|\theta)P(\theta)$$

- MAP

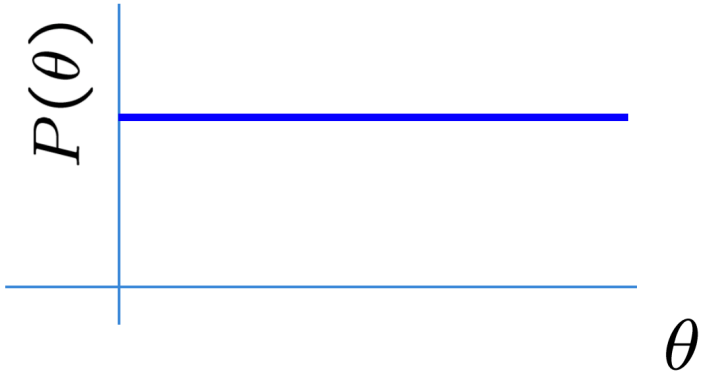
$$\theta_{MAP} = \arg\max_{\theta} P\,(\theta|D)$$
$$= \arg\max_{\theta} \log P(D|\theta)\,P(\theta)$$
$$= \arg\max_{\theta} \log \Pi_i P(x_i|\theta)\,P(\theta)$$
$$= \arg\max_{\theta} \Sigma_i \log P(x_i|\theta)\,P(\theta)$$

MLE and MAP

$$\theta_{MLE} = \arg\max_{\theta} P\,(D|\theta)$$
$$= \arg\max_{\theta} \log P(D|\theta)$$
$$= \arg\max_{\theta} \log \Pi_i P(x_i|\theta)$$
$$= \arg\max_{\theta} \Sigma_i \log P(x_i|\theta)$$

$$\theta_{MAP} = \arg\max_{\theta} P\,(\theta|D)$$
$$= \arg\max_{\theta} \log P(D|\theta)\,P(\theta)$$
$$= \arg\max_{\theta} \log \Pi_i P(x_i|\theta)\,P(\theta)$$
$$= \arg\max_{\theta} \Sigma_i \log P(x_i|\theta)\,P(\theta)$$

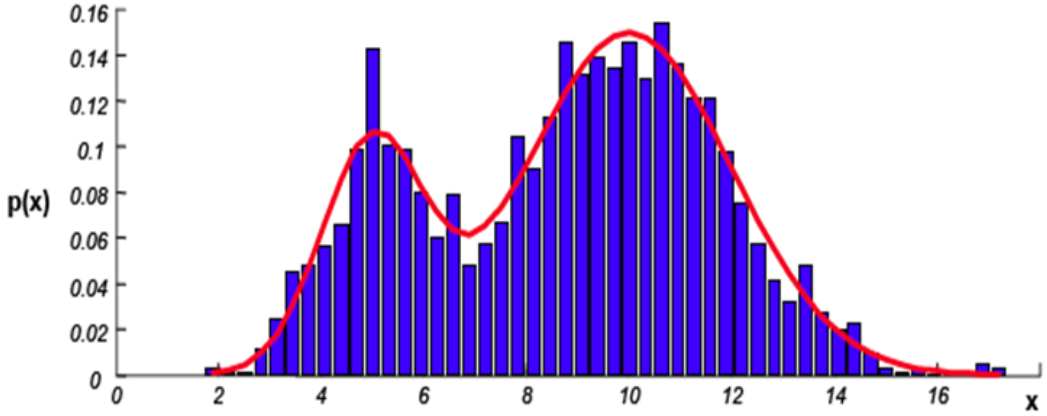
MLE can be thought as an uniform-prior version of MAP!



Non-parametric Classification

The Histogram

- The simplest form of non-parametric density estimation is the histogram
  - Divide sample space in number of bins
  - Approximate the density at the center of each bin by the fraction of points that fall into the bin
  - Two parameters: bin width and starting position of first bin (or other equivalent pairs)
- Drawbacks:
  - Depends on position of bin centers
    - Often compute two histograms, offset by ½ bin width
  - Discontinuities as an artifact of bin boundaries
  - Curse of dimensionality



Non-parametric Methods

- Non-parametric procedures can be used with arbitrary distributions and without the assumption that the forms of the underlying densities are known

Density Estimation

- Probability that a vector x will fall in region R is:

$$P = \int_{\Re} p(x')dx' \qquad (1)$$

- P is a smoothed (or averaged) version of the density function p(x) if we have a sample of size **n**; therefore, the probability that k points fall in R is:

$$P_k = \binom{n}{k} P^k (1-P)^{n-k} \qquad (2)$$

and the expected value for ***k*** is:

$$\mathbf{E(k) = nP} \qquad (3)$$

Answer the question: What is k, given n and P?

ML Estimate

ML estimation of P = θ

$\underset{\theta}{Max}(P_k \mid \theta)$ is reached for $\hat{\theta} = \frac{k}{n} \cong P$

Therefore, the ratio **k/n** is a good estimate for the probability **P** and hence for the density function **p(x)** (for large **n**)

Answer the question: What is P given k and n?

The k–Nearest-Neighbor Rule

- Goal: Classify x by assigning it the label most frequently represented among the k nearest samples
- Use a voting scheme

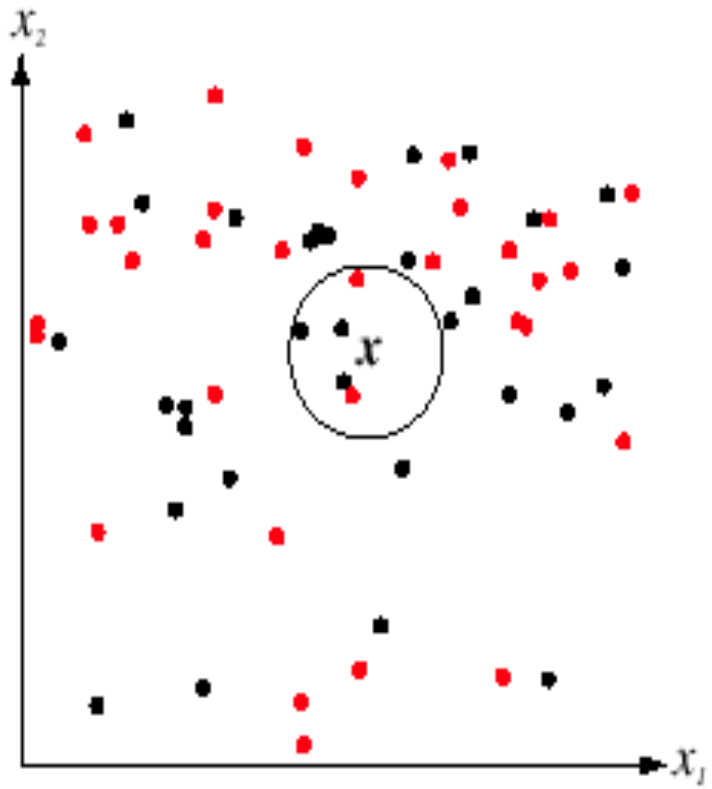


FIGURE 4.15. The k-nearest-neighbor query starts at the test point **x** and grows a spherical region until it encloses k training samples, and it labels the test point by a majority vote of these samples. In this $k = 5$ case, the test point **x** would be labeled the category of the black points. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Matlab Example

```
data = dlmread('pima-indians-diabetes.data');

data = reshape(data,[],9);

% use randperm to re-order data. ignore if not using Matlab
rp = randperm(length(data));
data=data(rp,:);

%split = length(data)/2;
split = 300;

train_data = data(1:split,:);
test_data = data(split+1:end,:);
```

```
% pick features
active_feat = [1:3];

% training
% NOT NEEDED

% testing
correct=0;
wrong=0;
```

```
for i=1:length(test_data)

   sample=test_data(i,active_feat);

   dist = train_data(:,active_feat)-repmat(sample,length(train_data),1);
   dist = dist*dist';

   % we are only interested in the diagonal elements
   % DON'T USE QUADRATIC DISTANCE COMPUTATION IN PRACTICE
   fin_dist = diag(dist);
   [min_d index] = min(fin_dist);

   if(test_data(i,9) == train_data(index,9))
      correct = correct+1;
   else
      wrong = wrong+1;
   end
end
```

Supplement

Bayesian Parameter Estimation

- Gaussian Case
- General Estimation

Bayesian Estimation

- In MLE θ was assumed fixed
- In BE θ is a random variable
- Suppose we have some idea of the range where the parameters θ should be
  - Shouldn't we utilize this prior knowledge in hope that it will lead to better parameter estimation?

Bayesian Estimation

- Let θ be a random variable with prior distribution P(θ)
  - This is the key difference between ML and Bayesian parameter estimation
  - This allows us to use a prior to express the uncertainty present before seeing the data
  - Frequentist approach does not account for uncertainty in θ

Motivation

- As in MLE, suppose $p(x|\theta)$ is completely specified if θ is given
- But now θ is a random variable with prior $p(\theta)$
  - Unlike MLE case, $p(x|\theta)$ is a conditional density
- After we observe the data D, using Bayes rule we can compute the posterior $p(\theta|D)$

Motivation

- Recall that for the MAP classifier we find the class ω_i that maximizes the posterior $p(\omega|D)$
- By analogy, a reasonable estimate of θ is the one that maximizes the posterior $p(\theta|D)$
- But θ is not our final goal, our final goal is the unknown $p(x)$
- Therefore a better thing to do is to maximize $p(x|D)$, this is as close as we can come to the unknown $p(x)$!

Parameter Distribution

- Assumptions:
  - p(x) is unknown, but has known parametric form
  - Parameter vector θ is unknown
  - p(x| θ) is completely known
  - Prior density p(θ) is known
- Observation of samples provides posterior density p(θ|D)
  - Hopefully peaked around true value of θ
- Treat each class separately and drop subscripts

- Converted problem of learning ***probability density*** function to learning ***parameter vector***
- Goal: compute p(x|D) as best possible estimate of p(x)

$$p(\mathrm{x} \mid \mathrm{D}) = \int p(\mathrm{x}, \theta \mid D) d\theta$$

$$p(\mathrm{x} \mid \mathrm{D}) = \int p(\mathrm{x} \mid \theta, D)\, p(\theta \mid D) d\theta = \int p(\mathrm{x} \mid \theta)\, p(\theta \mid D) d\theta$$

p(x) is completely known given θ, independent of samples in D

$$p(x \mid D) = \int p(x \mid \theta, D)\, p(\theta \mid D) d\theta = \int p(x \mid \theta)\, p(\theta \mid D) d\theta$$

- Links class-conditional density p(x|D) to posterior density p(θ|D)

Bayesian Parameter Estimation: Gaussian Case

Goal: Estimate θ using the a-posteriori density P(θ | D)

- The univariate case: p(μ | D)
  μ is the only unknown parameter

$$p(x \mid \mu) \sim N(\mu, \sigma^2)$$

$$p(\mu) \sim N(\mu_0, \sigma_0^2)$$

$$p(\mu \mid \mathsf{D}) = \frac{p(\mathsf{D} \mid \mu)\, p(\mu)}{\int p(\mathsf{D} \mid \mu)\, p(\mu) d\mu}$$

$$= \alpha \prod_{k=1}^{k=n} p(x_k \mid \mu)\, p(\mu)$$

- α depends on D, not μ
- Shows how training samples affect our idea about the true value of μ

$$p(\mu \mid \mathsf{D}) = \frac{p(\mathsf{D} \mid \mu)\, p(\mu)}{\int p(\mathsf{D} \mid \mu)\, p(\mu)\, d\mu} \qquad (1)$$

$$= \alpha \prod_{k=1}^{k=n} p(x_k \mid \mu)\, p(\mu)$$

Reproducing density (remains Gaussian)

$$p(\mu \mid \mathsf{D}) \sim N(\mu_n, \sigma_n^2) \qquad (2)$$

(1) and (2) yield:

$$\mu_n = \left(\frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2} \right) \hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2}\, \mu_0$$

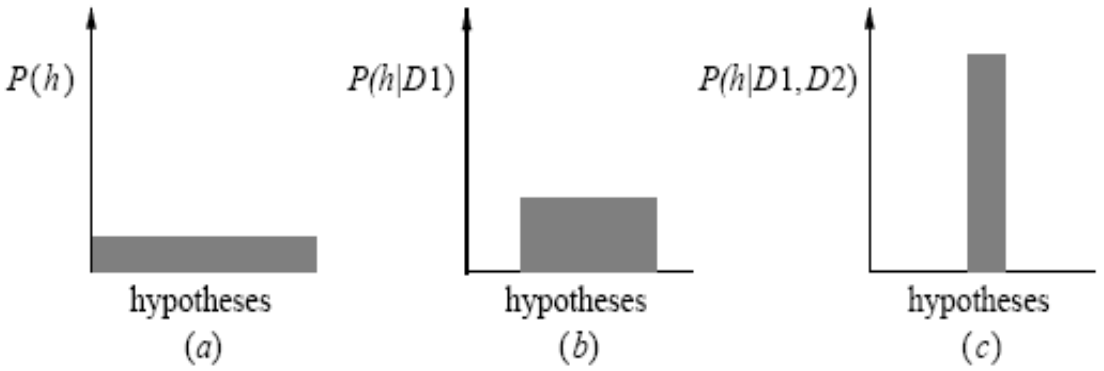
$$and \quad \sigma_n^2 = \frac{\sigma_0^2 \sigma^2}{n\sigma_0^2 + \sigma^2}$$

Empirical (sample) mean

$$\mu_n = \left(\frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2} \right) \hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2} \mu_0$$

$$and \quad \sigma_n^2 = \frac{\sigma_0^2 \sigma^2}{n\sigma_0^2 + \sigma^2}$$

- μ is linear combination of empirical and prior information
- Each additional observation decreases uncertainty about μ



- The univariate case p(x | D)
  - p(μ | D) computed
  - p(x | D) remains to be computed*

$$p(x \mid \mathsf{D}) = \int p(x \mid \mu)\, p(\mu \mid \mathsf{D})\, d\mu \text{ is Gaussian}$$

It provides: $p(x \mid \mathsf{D}) \sim N(\mu_n, \sigma^2 + \sigma_n^2)$

$$p(x \mid \mathsf{D}) \sim N(\mu_n, \sigma^2 + \sigma_n^2)$$

- We have:
  - Replaced mean with conditional mean
  - Increased variance to account for additional uncertainty in x due to inexact knowledge of mean

Bayesian Parameter Estimation: General Theory

- p(x | D) computation can be applied to any situation in which the unknown density can be parameterized. The basic assumptions are:
  - The form of p(x | θ) is assumed known, but the value of θ is not known exactly
  - Our knowledge about θ is assumed to be contained in a known prior density p(θ)
  - The rest of our knowledge θ is contained in a set D of n random variables x_1, x_2, …, x_n that follows p(x)

Recursive Bayes Learning

- Assume that training samples become available one by one

$$p(\mathsf{D}^n \mid \theta) = p(x_n \mid \theta)\, p(\mathsf{D}^{n-1} \mid \theta)$$

- Due to independence, result is independent of order:

$$p(\mathsf{D} \mid \theta) = \prod_{k=1}^{k=n} p(x_k \mid \theta)$$

The basic problem is:

"Compute the posterior density p(θ | D)"

then "Derive p(x | D)"

Using Bayes formula, we have:

$$p(\theta \mid \mathsf{D}) = \frac{p(\mathsf{D} \mid \theta)\, p(\theta)}{\int p(\mathsf{D} \mid \theta)\, p(\theta) d\theta}$$

And by the independence assumption:

$$p(\mathsf{D} \mid \theta) = \prod_{k=1}^{k=n} p(x_k \mid \theta)$$

Estimation of p(x|D)

- The basic problem is: Compute p(x | D)

$$p(x \mid D) = \int \underbrace{p(x \mid \theta)}_{\text{known}} \underbrace{p(\theta \mid D)}_{\text{unknown}} d\theta$$

- Compute the posterior density p(θ | D)

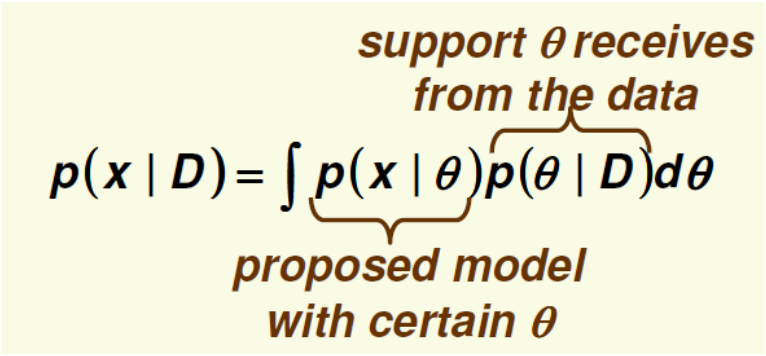
$$p(\theta \mid \mathsf{D}) = \frac{p(\mathsf{D} \mid \theta)\, p(\theta)}{\int p(\mathsf{D} \mid \theta)\, p(\theta)\, d\theta}$$

- Then derive p(x | D)

ML vs. Bayesian Parameter Estimation: Summary

BE vs. MLE

- BE: p(x|D) can be thought of as the weighted average of the proposed model for all possible values of θ



- Contrast this with the MLE solution which always gives us a single model:

$$p(x|\hat{\theta})$$

- When we have many possible solutions, taking their sum averaged by their probabilities seems better than pick just one solution

BE vs. MLE

- In practice, it may be hard to do integration analytically and we may have to resort to numerical methods
- The MLE solution requires differentiation, instead of integration, to get

$$p(x|\hat{\theta})$$

  - Differentiation is easy and can always be done analytically

When do Maximum-Likelihood and Bayes Methods Differ?

- Equivalent asymptotically (for infinite training data)
  - For reasonable prior distributions
  - When prior $p(\theta)$ is uninformative and $p(\theta|D)$ is peaked
- MLE computationally cheaper, simpler solutions
- BE uses more information (more general model)

Naïve Bayes Classifier (not BE)

- Simple classifier that applies Bayes' rule with strong (naive) independence assumptions
- A.k.a. the "independent feature model”
- Often performs reasonably well despite simplicity