

Supervised Learning (COMP0078)

1. Introduction to supervised learning

Mark Herbster

30 September, 2019

University College London
Department of Computer Science

SLIntro-19v3

Useful References

- *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, T. Hastie, R. Tibshirani, & J. Friedman, Springer (2009)
- *Understanding Machine Learning from Theory to Algorithms*, S. Shalev-Shwartz and S. Ben-David, Cambridge University Press (2014)
- *Kernel Methods for Pattern Analysis*, J. Shawe-Taylor, and N. Cristianini, Cambridge University Press (2004)
- *Pattern Recognition and Machine Learning*, C. Bishop, Springer (2006)

1. When: Mondays, 14:00–15:00, Wednesdays, 14:00–15:00, Fridays 9:00–10:00
2. Moodle enrolment key : SL
3. Questions :
 - 3.1 Moodle forum
 - 3.2 `sl-support@cs.ucl.ac.uk`

1. Homework (25%) and Exam (75%)
2. 2 courseworks assignments
(deliver them on-time, penalty otherwise)
3. To pass the course, you must obtain an average of at least 50% when the homework and exam components are weighted together.

Prerequisites

- Calculus (real-valued functions, limits, derivatives, Taylor series, integrals,...)
- Elements of probability theory (random variables, expectation, variance, conditional probabilities, Bayes rule,...)
- Fundamentals of linear algebra (vectors, angles, matrices, eigenvectors/eigenvalues,...),
- A bit of optimization theory (convex functions, Lagrange multipliers)

Provisional course outline

- (Week 1) Key concepts (probabilistic formulation of learning from examples, loss function, learning algorithm, overfitting and underfitting, model selection, cross validation); two basic learning algorithms linear regression and k -NN;
- (Week 2) Regularisation, Kernels
- (Week 3) Support Vector Machines
- (Week 4) Decision Trees and Ensemble Learning
- (Week 5) Online Learning I

Provisional course outline

- (Week 6) Online Learning II
- (Week 7) Multi-task and Learning to Learn
- (Week 8) Learning Theory
- (Week 9) Graphs in Machine Learning
- (Week 10) TBA

This week's plan

- The supervised learning problem
- Two learning algorithms: least squares and k -NN
- Probabilistic model, error function, optimal solutions
- Bias-Variance tradeoff, NFL theorems
- Asymptotic Optimality of k -NN
- Hypothesis space, overfitting and underfitting
- Choice of the learning algorithm (Model selection)

Supervised Learning Problem

Given a set of **input/output** pairs (**training set**) we wish to compute the functional relationship between the input and the output

$$\mathbf{x} \longrightarrow \boxed{f} \longrightarrow y$$

- **Example 1:** (people detection) given an image we wish to say if it depicts a person or not. The output is one of 2 possible categories
- **Example 2:** (pose estimation) we wish to predict the pose of a face image The output is a continuous number (here a real number describing the face rotation angle)

In both problems the input is a high dimensional vector \mathbf{x} representing pixel intensity/color

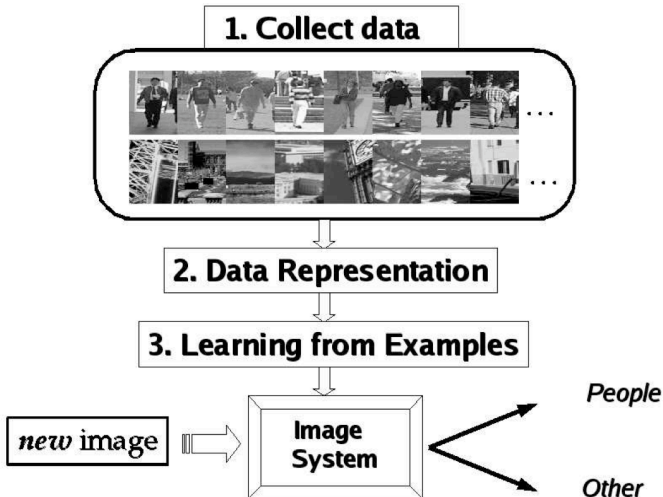
Why and when learning?

- The aim of learning is to develop software systems able to perform particular tasks such as people detection.
- Standard software engineering approach would be to specify the problem, develop an algorithm to compute the solution, and then implement efficiently.
- Problem with this approach is developing the algorithm:
 - No known criterion for distinguishing the images;
 - In many cases humans have no difficulty;
 - Typically problem is to specify the problem in logical terms.

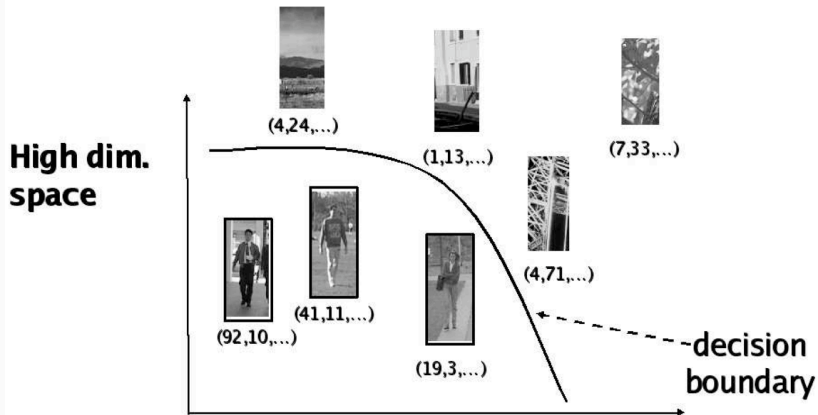
Learning approach

- Learning attempts to infer the algorithm for a set of (labelled) examples in much the same way that children learn by being shown a set of examples (eg sports/non sports car).
- Attempts to isolate underlying structure from a set of examples. Approach should be
 - stable: finds something that is not chance part of set of examples
 - efficient: infers solution in time polynomial in the size of the data
 - robust: should not be too sensitive to mislabelled/noisy examples

People Detection Example



People detection example (cont.)



Data are sparse! Risk for overfitting!

Supervised Learning Model

- Goal: Given training data (pattern,target) pairs

$$S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$$

infer a function f_S such that

$$f_S(\mathbf{x}_i) \approx y_i$$

for the **future** data

$$S' = \{(\mathbf{x}_{m+1}, y_{m+1}), (\mathbf{x}_{m+2}, y_{m+2}), \dots\}.$$

- Classification : $y \in \{-1, +1\}$; Regression : $y \in \mathbb{R}$
- \mathcal{X} : input space (eg, $\mathcal{X} \subseteq \mathbb{R}^d$), with elements $\mathbf{x}, \mathbf{x}', \mathbf{x}_i, \dots$
- \mathcal{Y} : output space, with elements y, y', y_i, \dots

Supervised learning problem: compute a function which “best describes” I/O relationship

Learning algorithm

- Training set: $S = \{(\mathbf{x}_i, y_i)_{i=1}^m\} \subseteq \mathcal{X} \times \mathcal{Y}$
- A **learning algorithm** is a mapping $S \mapsto f_S$
- A new input \mathbf{x} is predicted as $f_S(\mathbf{x})$

Example Algorithms

- Linear Regression
 - Neural Networks
 - Decision Trees
 - Support Vector Machines
-
- In the course we mainly deal with deterministic algorithms but we'll also comment on some randomized ones
 - Today: we describe two simple learning algorithms: linear regression and k -nearest neighbours

Some Questions

- How is the data **collected**? (need assumptions!)
- How do we **represent** the inputs? (may require preprocessing step)
- How **accurate** is f_S on new data (study of **generalization error**)?
- Many algorithms may exist for a task. How do we choose?
- How “**complex**” is a learning task? (computational complexity, sample complexity)

Some difficulties/aspects of the learning process

- New inputs **differ** from the ones in the training set (look up tables do not work!)
- Inputs are measured with **noise**
- Output is **not deterministically** obtained by the input
- Input is often **high dimensional** but some components/variables may be irrelevant
- How can we incorporate **prior knowledge**?

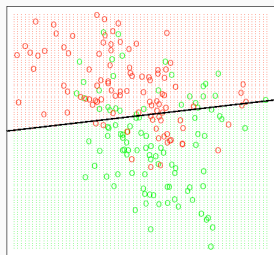
Binary classification: an example

We describe two basic learning algorithms/models for classification which can be easily adapted to regression as well.

We choose: $\mathcal{X} = \mathbb{R}^2$, $\mathbf{x} = (x_1, x_2)$ and $\mathcal{Y} = \{\text{green}, \text{red}\}$

Our first learning algorithm computes a **linear function**, $\mathbf{w}^\top \mathbf{x} + b$ and classifies an input \mathbf{x} as

$$f(\mathbf{x}) = \begin{cases} \text{red} & \mathbf{w}^\top \mathbf{x} + b > 0 \\ \text{green} & \mathbf{w}^\top \mathbf{x} + b \leq 0 \end{cases}$$



Linear Regression (Least Squares)

- Emerged in response to problems in Astronomy and Navigation
- Motivated by the need to combine multiple noisy measurements
- Method first described by Gauss in 1794



A Simple Problem – 1

Given the data set

$$S = \{((1, 1), 3), ((2, 3), 7)\}$$

Then with the new input $\mathbf{x}_3 = (4, 2)$

how should we predict y_3 ?

Why? What Assumptions?

A Simple Problem – 2

Model as a system of equations

$$w_1 + w_2 = 3$$

$$2w_1 + 3w_2 = 7$$

or more directly as

$$X\mathbf{w} = \mathbf{y}$$

with

$$X = \begin{pmatrix} 1 & 1 \\ 2 & 3 \end{pmatrix}; \quad \mathbf{y} = \begin{pmatrix} 3 \\ 7 \end{pmatrix}$$

A Simple Problem – 3

Solving in matlab

$$\mathbf{w} = \mathbf{X}^{-1}\mathbf{y}$$

```
>> y = [3 ; 7]
y = 3
      7
>> X = [1,1 ; 2, 3]
X = 1      1
      2      3
>> XI = X^(-1)
XI = 3      -1
      -2      1
>> w= XI * y
w = 2
      1
>> w= X \ y %% More efficient than calculating inverse use in practice see help mldivide
w = 2
      1
```

We now have the linear predictor

$$\hat{y} = \mathbf{w} \cdot \mathbf{x}$$

Thus predict $\hat{y}_3 = \mathbf{w} \cdot \mathbf{x}_3 = w_1 x_{3,1} + w_2 x_{3,2} = 4 \times 2 + 1 \times 2 = 10$. 22

A Simple Problem – 4

What if?

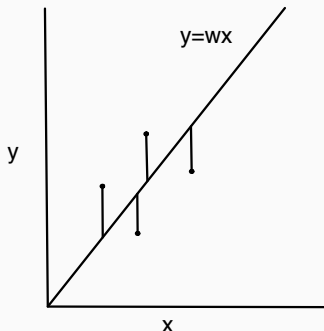
- Overdetermined:

$$S = \{((1, 1), 3), ((2, 3), 7), ((2, 1), 3)\}$$

- Underdetermined:

$$S = \{((1, 1, 2), 3), ((2, 4, 3), 7)\}$$

Minimize square error – 1



Find a linear predictor $\hat{y} = \mathbf{w} \cdot \mathbf{x}$ to minimize the square error over the data $S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$ thus

$$\text{Minimize: } \sum_{i=1}^m (y_i - \hat{y}_i)^2 = \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$$

Minimize square error – 2

Thus given,

$$\mathcal{S} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$$

with $\mathbf{x} \in \mathbb{R}^n$ we may represent the pattern and target vectors with the matrices

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \dots & x_{m,n} \end{pmatrix} \text{ and } \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

MSE in Matrix Notation

Thus in matrix notation the *empirical* mean (square) error of the linear predictor $\hat{y} = \mathbf{w} \cdot \mathbf{x}$ on the data sequence \mathcal{S} is

$$\begin{aligned}\mathcal{E}_{\text{emp}}(\mathcal{S}, \mathbf{w}) &= \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2 \\ &= \frac{1}{m} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 \\ &= \frac{1}{m} \sum_{i=1}^m (y_i - \sum_{j=1}^n w_j x_{i,j})^2 \\ &= \frac{1}{m} (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y})\end{aligned}$$

MSE minimization. General Case

To compute the minimum we solve for

$$\nabla_{\mathbf{w}} \mathcal{E}_{\text{emp}}(\mathcal{S}, \mathbf{w}) = \mathbf{0}.$$

recalling that

$$\nabla_{\mathbf{w}} = \begin{pmatrix} \frac{\partial}{\partial w_1} \\ \vdots \\ \frac{\partial}{\partial w_n} \end{pmatrix}$$

Thus we need to solve,

$$\begin{aligned} \nabla_{\mathbf{w}} \left[(\mathbf{X}\mathbf{w} - \mathbf{y})^T (\mathbf{X}\mathbf{w} - \mathbf{y}) \right] &= \mathbf{0} \\ \left(\sum_{i=1}^m \frac{\partial}{\partial w_1} \left(\sum_{j=1}^n X_{ij} w_j - y_i \right)^2, \dots, \sum_{i=1}^m \frac{\partial}{\partial w_n} \left(\sum_{j=1}^n X_{ij} w_j - y_i \right)^2 \right)^T &= \mathbf{0} \end{aligned}$$

Normal equations

Consider the 2-d case ($n = 2$)

$$\mathcal{E}_{\text{emp}}(\mathcal{S}, \mathbf{w}) = \frac{1}{m} \sum_{i=1}^m (y_i - \mathbf{w}^\top \mathbf{x}_i)^2$$

Note that:

$$\frac{\partial \mathcal{E}_{\text{emp}}(\mathcal{S}, \mathbf{w})}{\partial w_k} = \frac{2}{m} \sum_{i=1}^m (\mathbf{w}^\top \mathbf{x}_i - y_i) \frac{\partial (\mathbf{w}^\top \mathbf{x}_i)}{\partial w_k} = \frac{2}{m} \sum_{i=1}^m (\mathbf{w}^\top \mathbf{x}_i - y_i) x_{ik}$$

Hence, to find $\mathbf{w} = (w_1, w_2)^\top$ we need to solve the *linear system* of equations

$$\sum_{i=1}^m (x_{ik} x_{i1} w_1 + x_{ik} x_{i2} w_2) = \sum_{i=1}^m x_{ik} y_i, \quad k = 1, 2$$

Normal equations (cont.)

In vector notations:

$$\sum_{i=1}^m \mathbf{x}_i \mathbf{x}_i^\top \mathbf{w} = \sum_{i=1}^m \mathbf{x}_i y_i$$

In matrix notation:

$$X^\top X \mathbf{w} = X^\top \mathbf{y}$$

where

$$X^\top = \begin{bmatrix} x_{11} & \cdots & x_{m1} \\ \vdots & \ddots & \vdots \\ x_{1n} & \cdots & x_{mn} \end{bmatrix} \equiv [\mathbf{x}_1, \cdots, \mathbf{x}_m], \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

Least square solution

$$X^T X \mathbf{w} = X^T \mathbf{y}$$

For the time being we will assume that the matrix $X^T X$ is invertible, so we conclude that

$$\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y}$$

Otherwise, the solution may not be unique...

Comment:

In matlab use

$$w = X \backslash y$$

Going back to “b” (adding a bias term)

Substituting \mathbf{x}^\top by $(\mathbf{x}^\top, 1)$ and \mathbf{w}^\top by (\mathbf{w}^\top, b) , the above system of equations can be expressed in matrix form as (exercise):

$$(\mathbf{X}^\top \mathbf{X})\mathbf{w} + \mathbf{X}^\top \mathbf{1}b = \mathbf{X}^\top \mathbf{y}$$

$$\mathbf{1}^\top \mathbf{X}\mathbf{w} + mb = \mathbf{1}^\top \mathbf{y}$$

that is
$$\begin{bmatrix} \mathbf{X}^\top \mathbf{X} & \mathbf{X}^\top \mathbf{1} \\ \mathbf{1}^\top \mathbf{X} & m \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{X}^\top \mathbf{y} \\ \mathbf{1}^\top \mathbf{y} \end{bmatrix}$$

where $\mathbf{1} = (1, 1, \dots, 1)^\top$, an $m \times 1$ vector of “ones”

A different approach: k -nearest neighbours

Let $N(\mathbf{x}; k)$ be the set of k nearest training inputs to \mathbf{x} and

$$I_{\mathbf{x}} = \{i : \mathbf{x}_i \in N(\mathbf{x}; k)\}$$

the corresponding index set

$$f(\mathbf{x}) = \begin{cases} \text{red} & \text{if } |\{y_i = \text{red} : i \in I_{\mathbf{x}}\}| > |\{y_i = \text{green} : i \in I_{\mathbf{x}}\}| \\ \text{green} & \text{if } |\{y_i = \text{green} : i \in I_{\mathbf{x}}\}| > |\{y_i = \text{red} : i \in I_{\mathbf{x}}\}| \end{cases}$$

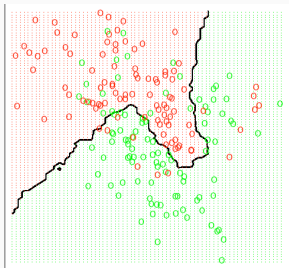
- Closeness is measured using a metric (eg, Euclidean dist.)
- Local rule (compute local majority vote)
- Decision boundary is non-linear

Note: for regression we set $f(\mathbf{x}) = \frac{1}{k} \sum_{i \in I_{\mathbf{x}}} y_i$ (a “local mean”)

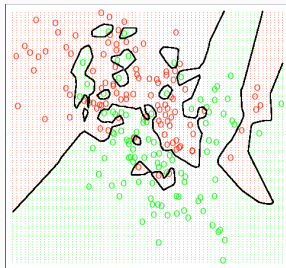
k -NN: the effect of k

- The smaller k the more irregular the decision boundary

$k = 15$



$k = 1$



- How to choose k ? later...

Perspectives on supervised learning

We consider three influential ideas that underpin SL.

1. What is an optimal predictor? The Bayes estimator.
2. The bias and variance of a learning algorithm.
3. The non-existence of an optimal learning algorithm (without assumptions). NFL theorems.

Optimal Supervised Learning

Model: We assume that the data is obtained by sampling **i.i.d.** from a **fixed but unknown** probability density $P(\mathbf{x}, y)$

Expected error:

$$\mathcal{E}(f) := \mathbf{E} \left[(y - f(\mathbf{x}))^2 \right] = \int (y - f(\mathbf{x}))^2 dP(\mathbf{x}, y)$$

Our goal is to minimize \mathcal{E}

Optimal solution: $f^* := \operatorname{argmin}_f \mathcal{E}(f)$ (called *Bayes estimator*)

Problem A: in order to compute f^* we need to know P !

Note: for binary classification with $\mathcal{Y} = \{0, 1\}$ and $f : \mathcal{X} \rightarrow \mathcal{Y}$, $\mathcal{E}(f)$ counts the average number of mistakes of f (aka expected misclassification error)

Bayes estimator for square loss

Let us compute the optimal solution f^* for regression $\mathcal{Y} = \mathbb{R}$.
Using the decomposition $P(y, \mathbf{x}) = P(y|\mathbf{x})P(\mathbf{x})$ we have

$$\mathcal{E}(f) = \int_{\mathcal{X}} \left\{ \int_{\mathcal{Y}} (y - f(\mathbf{x}))^2 dP(y|\mathbf{x}) \right\} dP(\mathbf{x})$$

Observe that f^* is

$$f^*(\mathbf{x}) = \int_{\mathcal{Y}} y dP(y|\mathbf{x})$$

Deriving the Bayes estimator for square loss

Deriving f^* in the discrete case,

$$\mathcal{E}(f) := \sum_{x \in X} \sum_{y \in Y} (y - f(x))^2 p(x, y)$$

and $\mathcal{E}(f)$ is the expected error of an arbitrary predictor f .

Separating

$$\mathcal{E}(f) = \sum_{x \in X} \left[\sum_{y \in Y} (y - f(x))^2 p(y|x) \right] p(x)$$

now let's find the value of $f^*(x)$ at a specific point. Set $x := x'$ then we have

$$\mathcal{E}(f(x')) = \left[\sum_{y \in Y} (y - f(x'))^2 p(y|x') \right] p(x')$$

Use $e := \mathcal{E}(f(x'))$ and $z := f(x')$, repl. “=” with “ \propto ” rem. $p(x')$

$$e \propto \sum_{y \in Y} (y - z)^2 p(y|x')$$

Derivation – continued

To find the minimum we differentiate e w.r.t. z

$$\frac{\partial e}{\partial z} = -2 \sum_{y \in Y} (y - z)p(y|x').$$

Setting equal to zero and solving,

$$0 = \sum_{y \in Y} yp(y|x') - \sum_{y \in Y} zp(y|x') = \sum_{y \in Y} yp(y|x') - z$$

which implies

$$z = \sum_{y \in Y} yp(y|x')$$

and hence

$$f^*(x) = \sum_{y \in Y} yp(y|x) = E[y|x].$$

Bias and Variance of a learning algorithm

- We now additionally assume there exists some underlying function F such that

$$y = F(x) + \epsilon$$

where ϵ is white noise, i.e., $E[\epsilon] = 0$ and finite variance.

- Thus the optimal prediction is $f^*(x) := E[y|x] = F(x)$ with square loss.
- We would like to understand the expected error by an arbitrary learner $A_S(x)$ (we now drop the S for convenience).
- Our goal will be to understand the expected error at x'

$$\mathcal{E}(A(x')) = E[(y' - A(x'))^2]$$

where y' is a sample from the marginal $P(y|x')$.

Useful Lemma and Corollary

Lemma

$$E[(Z - E[Z])^2] = E[Z^2] - E[Z]^2$$

Proof

$$E[(Z - E[Z])^2] = E[Z^2 - 2ZE[Z] + E[Z]^2]$$

$$\begin{aligned} E[(Z - E[Z])^2] &= E[Z^2] - 2E[Z]^2 + E[Z]^2 \\ &= E[Z^2] - E[Z]^2 \end{aligned}$$

Corollary

$$E[Z^2] = E[(Z - E[Z])^2] + E[Z]^2$$

Decomposing $\mathcal{E}(A(x'))$

$$\begin{aligned} E[(y' - A(x'))^2] &= E[(y')^2 - 2y'A(x') + A(x')^2] \\ &= E[(y' - f^*(x'))^2] + f^*(x')^2 + \\ &\quad - 2f^*(x')E[A(x')] + \\ &\quad E[(A(x') - E[A(x')])^2] + E[A(x')]^2 \\ &= E[(y' - f^*(x'))^2] + \quad \text{[Bayes error]} \\ &\quad (f^*(x') - E[A(x')])^2 + \quad \text{[bias}^2\text{]} \\ &\quad E[(A(x') - E[A(x')])^2] \quad \text{[variance]} \end{aligned}$$

- Bayes error : $E[(y' - f^*(x'))^2]$ is the irreducible noise.
- Bias : $f^*(x') - E[A(x')]$
describes the discrepancy between the algorithm and “truth” .
- Variance : $E[(A(x') - E[A(x')])^2]$
captures the variance of the algorithm between training sets

Subtlety: There are two *independent* random quantities. $A(\cdot)$ depends on the random draw of the training sequence

and y' depends on the draw on from the marginal $p(y|x')$.

Bias and Variance Dilemma

- The bias and variance tend to trade off against one another
- Many parameters better flexibility to fit the data thus low bias but high variance
- Few parameters give high bias but the fit between different data sets will not change much thus low variance
- **Caveat:** this exact decomposition only holds for the square loss.

Optimal Algorithms and No Free Lunch Theorems

- Is there an optimal machine learning algorithm?
- I.e., an algorithm that does almost as well as the bayes estimator.
- There are a variety of such (NFL) results showing that no such algorithm exists.
- We give an example NFL Theorem in classification setting

Bayes estimator for classification

More generally the *Bayes classifier* (estimator) is the minimiser of the expected loss

- for C-class classification, f^* (called the Bayes classifier) is

$$f^*(\mathbf{x}) = \operatorname{argmax}_{c \in \{1, \dots, C\}} P(Y = c | \mathbf{x})$$

where the loss is 0 if we predict correctly and 1 otherwise.

- The *Bayes error rate* (optimal) is then

$$\int (1 - P(Y = f^*(\mathbf{x}) | \mathbf{x})) dP(\mathbf{x})$$

Challenge: What is the Bayes estimator of

$$\mathcal{E}(f) := \sum_{x \in X} \sum_{y \in Y} |y - f(x)| p(x, y)$$

An NFL theorem for classification

Theorem (Shalev-Shwartz and Ben-David 2014, Section 5.1)

Let A_S be any learning algorithm for binary classification $\mathcal{Y} = \{0, 1\}$ over the domain \mathcal{X} . Let $m < |\mathcal{X}|/2$ then there exists some distribution P over $\mathcal{X} \times \{0, 1\}$ such that,

1. There exists a function $f : \mathcal{X} \rightarrow \{0, 1\}$ the Bayes error is 0.
2. With probability of at least $1/7$ with respect to a random draw S of m examples from P and a draw of a $m + 1$ st example (x', y') we have that

$$\text{Prob}[A_S(x') \neq y'] \geq 1/8$$

Proof Intuitions (Not examined)

Proof Sketch. Theorem (Shalev-Shwartz and Ben-David 2014, Section 5.1)

1. Intuition: if an $f : \mathcal{X} \rightarrow \{0, 1\}$ is chosen uniformly at random. There is no “traction” for any learning algorithm except “memorization”.
2. Hence suppose $m = |\mathcal{X}|/2$.
3. The intuition suggests we will be able to predict accurately on up to 50% of the points and on the remaining points it will only be 50% accurate.
4. Thus

$$E[\text{Prob}[A_S(x') \neq y']] \geq 1/4$$

5. Let $\theta := \text{Prob}[A_S(x') \neq y']$
6. θ is random variable wrt to the draw of the training sequence S
7. Using Markov's inequality for a random variable $\theta \in [0, 1]$ we have that $\text{Prob}[\theta \geq a] \geq \frac{E[\theta] - a}{1 - a}$.
8. Thus since by the “intuition” $E[\theta] \geq 1/4$ and then if we set $a := 1/7$ we have ...
9. With probability of at least $1/7$ with respect to a random draw S of m examples from P and a draw of a $m + 1$ st example (x', y') we have that

$$\text{Prob}[A_S(x') \neq y'] \geq 1/8$$

Asymptotic Optimality of k -NN

Revisiting k -NN

k -NN attempts approximate $P(Y = c|\mathbf{x})$ as $\frac{|\{i: y_i = c, i \in I_{\mathbf{x}}\}|}{k}$

- Expectation is replaced by averaging over sample data
- Conditioning at \mathbf{x} is relaxed to conditioning on some region close to \mathbf{x}

As the number of samples goes to infinity ($m \rightarrow \infty$) 1-NN and k -NN become “good” estimators.

1-NN is near asymptotically optimal

Theorem

As the number samples goes to infinity the error rate is no more than twice the Bayes error rate.

Proof Sketch

Abbreviate notation $P(c|\mathbf{x}) := P(Y = c|\mathbf{x})$.

The expected (Bayes) error of the Bayes classifier (at x) is

$$1 - \max_{c \in \{1, \dots, C\}} P(c|\mathbf{x})$$

and the expected rate of 1-NN (at x) is

$$\sum_{c=1}^C P(c|\mathbf{x}_{nn})[1 - P(c|\mathbf{x})].$$

Proof Sketch – continued

Proof Sketch

Observe that as the number samples goes to infinity, $m \rightarrow \infty$,

$$P(c|\mathbf{x}) \approx P(c|\mathbf{x}_{nn})$$

thus the expected rate of 1-NN (at x) is

$$\sum_{c=1}^C P(c|\mathbf{x})[1 - P(c|\mathbf{x})].$$

We need to show

$$\sum_{c=1}^C P(c|\mathbf{x})[1 - P(c|\mathbf{x})] \leq 2[1 - \max_{c \in \{1, \dots, C\}} P(c|\mathbf{x})]$$

Proof Sketch – continued

Proof Sketch – continued

Let $c^* = \operatorname{argmax}_{c \in \{1, \dots, C\}} P(c|\mathbf{x})$ and $p^* = P(c^*|\mathbf{x})$. Observe that

$$\begin{aligned}\sum_{c=1}^C P(c|\mathbf{x})[1 - P(c|\mathbf{x})] &= \sum_{c \neq c^*}^C P(c|\mathbf{x})[1 - P(c|\mathbf{x})] + p^*(1 - p^*) \\ &\leq (C-1) \frac{1-p^*}{C-1} \left[1 - \frac{1-p^*}{C-1}\right] + p^*(1 - p^*) \\ &= (1 - p^*) \left[1 - \frac{1-p^*}{C-1} + p^*\right]\end{aligned}$$

Where the second line follows since the sum is maximised when all “ $P(c|\mathbf{x})$ ” have the same value. And since $p^* < 1$ we are done. □

k -NN is asymptotically optimal

One can show that $\mathcal{E}(k(m) - NN) \rightarrow \mathcal{E}(f^*)$ as $m \rightarrow \infty$ provided that:

1. $k(m) \rightarrow \infty$
2. $\frac{k(m)}{m} \rightarrow 0$

Weakness: the rate of convergence depends exponentially on the input dimension. An example of the **curse of dimensionality**.

Reference for 1-NN near optimality

Cover & Hart : *Nearest Neighbor Pattern Classification*, 1967

Curse of dimensionality

1. “Curse of dimensionality” is an observation that many methods suffer as the dimensionality of data increases.
 2. The intuition is that since volume increases *exponentially* with the dimension then the data required to “cover” the space to perform estimates also increase exponentially.
 3. E.g., consider the unit d -dimensional ball centered at the origin versus the $1/2$ -unit ball at the origin the ratios of their volumes is $\frac{1}{2}^d$.
- Potential “solutions” include limiting the number of hypotheses (functions) – i.e., introducing a *hypothesis space*. Alternately associating a “complexity” with each hypothesis and then “prefer” simpler hypothesis.

Linear regression vs. k -NN (informal)

- Parametric vs. non-parametric
- Global vs. local
- Linear vs. non-linear
- Bias / variance considerations:
 - LR relies heavily on linear assumption (may have large bias) k -NN does not
 - LR is stable (solution does not change much if data are perturbed) 1-NN isn't!
- k -NN sensitive to input dimension d : if d is high, the inputs tends to be far away from each other!

Hypothesis Space

Solving the “Problem A”

$P(\mathbf{x}, y)$ is unknown \Rightarrow cannot compute $f^* = \operatorname{argmin}_f \mathcal{E}(f)$

We are only given a sample (training set) from P

A natural approach: we approximate the expected error $\mathcal{E}(f)$ by the empirical error

$$\mathcal{E}_{\text{emp}}(\mathcal{S}, f) = \frac{1}{m} \sum_{i=1}^m (y_i - f(\mathbf{x}_i))^2$$

Problem B: If we minimize \mathcal{E}_{emp} over all possible functions, we can always find a function with zero empirical error! (if the Bayes error is zero).

Why is this a problem?

Solving the “Problem B”

A Proposed solution: we introduce a **restricted** space of functions \mathcal{H} called the **hypothesis space**

We minimize $\mathcal{E}_{\text{emp}}(\mathcal{S}, f)$ within \mathcal{H} . That is, our learning algorithm is:

$$f_{\mathcal{S}} = \operatorname{argmin}_{f \in \mathcal{H}} \mathcal{E}_{\text{emp}}(\mathcal{S}, f)$$

This approach is usually called **empirical error (risk) minimization**

For example (Least Squares) :

$$\mathcal{H} = \{f(\mathbf{x}) = \mathbf{w}^{\top} \mathbf{x} : \mathbf{w} \in \mathbb{R}^n\}$$

Problem C: How do we choose a space \mathcal{H} (discuss later)?

Choosing a Hypothesis Space (returning to prob. “C”)

Given the training data $y_i = f^*(\mathbf{x}_i) + \epsilon_i$, the goal is to compute an “approximation” of f^* .

We look for an approximant of f^* within a prescribed hypothesis space \mathcal{H}

- Unless prior knowledge is available on f^* (eg, f^* is linear) we cannot expect $f^* \in \mathcal{H}$
- Choosing \mathcal{H} “very large” leads to **overfitting!** (we’ll see an example of this in a moment)

Summary

- Data S sampled i.i.d from P (fixed but unknown)
- f^* is what we want, f_S is what we get
- Different approaches to attempt to estimate/approximate f^* :
 - Minimize \mathcal{E}_{emp} in some restricted space of functions (eg, linear)
 - Compute local approximation of f^* (k -NN)
 - Estimate P and then use Bayes rule...

Model Selection

Polynomial fitting

As an example of hypothesis spaces of increasing “complexity” consider regression in one dimension

$$H_0 = \{f(x) = b : b \in \mathbb{R}\}$$

$$H_1 = \{f(x) = ax + b : a, b \in \mathbb{R}\}$$

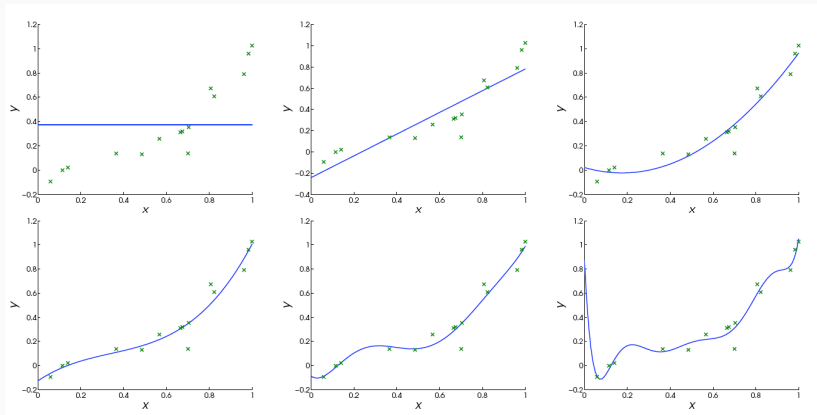
$$H_2 = \{f(x) = a_1x + a_2x^2 + b : a_1, a_2, b \in \mathbb{R}\}$$

$$\vdots$$

$$H_n = \left\{ f(x) = \sum_{\ell=1}^n a_\ell x^\ell + b : a_1, \dots, a_n, b \in \mathbb{R} \right\}$$

Consider minimizing the empirical error in \mathcal{H}_r (r = “polynomial degree”)

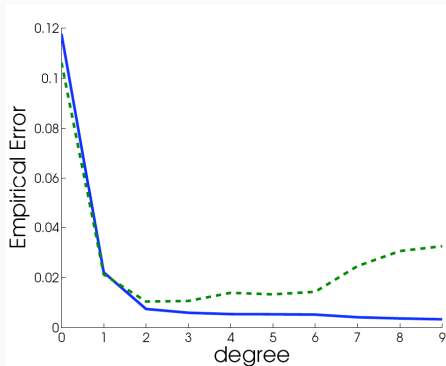
Polynomial fitting (simulation)



$r = 0, 1, 2, 3, 4, 5$. As r increases the fit to the data improves
(empirical error decreases)

Overfitting vs. Underfitting

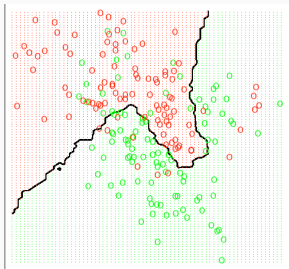
- Compare the empirical error (solid line) with expected error (dashed line)
 - r small: underfitting
 - r large: overfitting
- The larger r the lower the empirical error of f_S ! \Rightarrow We cannot rely on the training error!



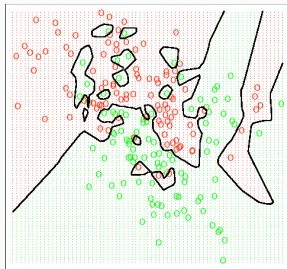
k -NN: the effect of k

- The smaller k the more irregular the decision boundary

$k = 15$



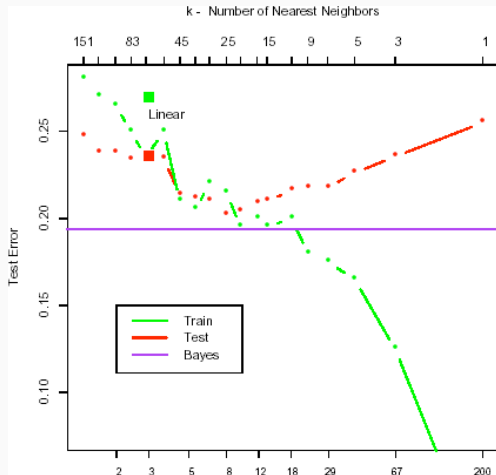
$k = 1$



- How to choose k ? later...

k -NN: the effect of k

$\frac{m}{k}$ large: overfitting versus $\frac{m}{k}$ small: underfitting



Model Selection

1. How to choose k in k -NN?
2. How to choose the degree r for polynomial regression?
3. The simplest approach is to use part of the training data (say 2/3) for training and the rest as a validation set for each \mathcal{H}_r
4. Another approach is K -fold cross-validation – see next slide
5. Then choose the “best” r relative to the error(s) on the validation set
6. We will return to model selection later in the course

Cross-validation

1. we split the data in K parts (of roughly equal sizes)
2. repeatedly train on $K - 1$ parts and test on the part “left out”
3. average the errors of K “validation” sets to give so-called cross-validation error
4. smaller K is less expensive but poorer estimate as size of training set is smaller and random fluctuations larger

For a dataset of size m , m -fold cross-validation is referred to as leave-one-out (LOO) testing

- Cross validation is good in “practice.”
- There are a variety of theoretical-based approaches (not covered today)
- Examples
 1. “Bayesian” model selection via the “evidence”
 2. Structural Risk Minimization

Other learning paradigms

- **Online learning:** we observe the data sequentially and we make a prediction and update the learner after every datum.
- **Active learning:** we are given many inputs and we can choose which ones to request a label.
- **Unsupervised learning:** we have only input examples. Here we may want to find data clusters, estimate the probability density of the data, find important features/variable (dimensionality reduction problem), detect anomalies, etc.
- **Semi-supervised learning:** the ‘learning environment’ may give us access to many input examples but only few of them are labeled.
- **Reinforcement learning:** Similar to online learning where data is received sequentially but feedback is often delayed.

Suggested Readings

- Elements of Statistical Learning, 2ed, Chapter 2 : spans most of material of this lecture, with the following exceptions
- Cover & Hart : *Nearest Neighbor Pattern Classification*, 1967 : for Asymptotic Optimality of k -NN
- *Understanding Machine Learning from Theory to Algorithms*, Chapter 5.1 : For the NFL Theorem. See section 19.2.2 for a discussion of the curse of dimensionality wrt $K - nn$.

- *Reconciling modern machine learning practice and the bias-variance trade-off.* Suggests that the classic U-shaped curve (see page 64) can be replaced by a double U, in a number of scenarios.
- *An adaptive nearest neighbor rule for classification.* Gives bounds for a k -NN variant that adapts k locally for each neighbourhood.

Problems – 1

1. True or False?

“For the K -nearest neighbour algorithm, larger K values will tend to lead to overfitting.”

Explain your answer.

2.
 - First, give Cover's bound on the Bayes error of the 1-nearest neighbour algorithm as the number of examples goes to infinity.
 - Second, give an example where the 1-nearest neighbour algorithm achieves the Bayes error as the number of examples goes to infinity, explain your reasoning.
3. How can we choose k to ensure that the k -NN makes an unambiguous decision in the two-class case provided the test point is not equidistant to any pair of points in the training set? What values of k give an unambiguous decision under similar assumptions for the 3-class case?

Problems – 2

1. One of the drawbacks of the nearest-neighbour algorithm is that we must retain all of the training data. Describe a situation where a training point can be removed without affecting the resulting 1-NN classification for any test point in the input space.
2. In linear regression it is common to transform the training data

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$$

to

$$((\mathbf{x}_1, 1), y_1), \dots, ((\mathbf{x}_m, 1), y_m)$$

i.e., we add an additional component to each input vector and set it to 1. What is the motivation for this procedure?

3. Given we have a dataset $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ where $\mathbf{x} \in \mathbb{R}^n$ and $y \in \{1, 2, \dots, K\}$. Thus the inputs are real vectors and the labels denote one of K classes. How can linear regression (least squares), be used or adapted to tackle such a multi-class classification problem?