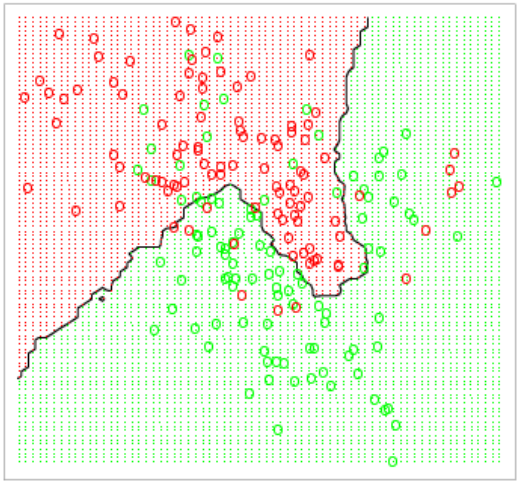
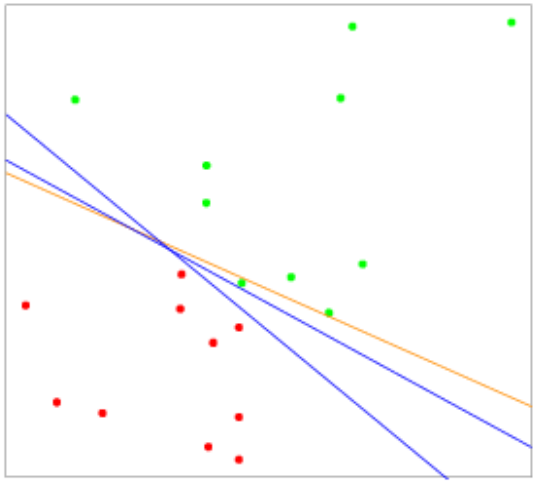
Inductive Concept Learning: Basic Concepts





Univ.-Lektor Dr.techn. Alexander K. Seewald
Österreichisches Forschungsinstitut
für Artificial Intelligence

alex@seewald.at / alex.seewald.at

What is Learning?

"Learning denotes changes in a system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently next time." (Herbert Simon, 1983)

"Learning is constructing or modifying representations of what is being experienced." (Ryszard Michalski, 1986)

"Learning means behaving better as a result of experience." (Stuart Russel & Peter Norvig, 1995)

"Learning is making useful changes in our minds."
(Marvin Minsky, 1985)

alex@seewald.at / alex.seewald.at

Logical Inference

Deduction (logically justified)

known: general rule	$p \Rightarrow q$	("whenever p is true, q is also true")
known/observed	p	("p is true")
Deductive conclusion	q	("q is true")

Abduction (logically not justified)

known: general rule	$p \Rightarrow q$	("whenever p is true, q is also true")
known/observed	q	("q is true")
Abductive conclusion	p	("p is true")

(e.g. p = "It rains", q = "The street is wet")

alex@seewald.at / alex.seewald.at

Logical Inference (contd.)

Induction / Inductive Generalization (logically not justified)

Sets of	p,q,r	("p, q and r occurred together")
observations	p,q,s,t	("p, q, s and t occurred together")
	p,q,s,u,v	("p, q, s, u and v occurred together")
	....	
Inductive conclusion	$p \Rightarrow q$	("whenever p is true, q is also true")
or	$q \Rightarrow p$	("whenever q is true, p is also true")

- Infers general rules/laws from a finite set of specific observations
- Is not logically justified, unless all possible situations were observed
- Likelihood of correct generalization increases w/ number of confirming cases, but a single counter-example(!) can falsify any inductive generalization.

$\Rightarrow$ **Inductive generalization is dangerous, but the only way of producing new knowledge from examples. All non-trivial forms of learning use Induction.**

alex@seewald.at / alex.seewald.at

What makes a learning task?

Given the values of an input vector **x** we want a good prediction of output value y:

$$f(\mathbf{x}) \approx y$$

Regression task: y quantitative (interval/ratio scale) - real numbers

Prediction task: y qualitative (categorical/ordinal scale) - set of distinct values

How to determine function/model f ?

- Estimate from given set of examples $\mathbf{x_i}$ with known output values y_i :
  Training data TD=$\{(\mathbf{x_i}, y_i), i=1..N\}$. TD $\subset$ *Instance space*
  Instance space = (possibly infinite) set of all distinct examples.
- Each learning algorithm works differently, but all of them estimate function f from the training data TD via inductive generalization.
- Structure of f differs between learning algorithms. Need some characterization
  Concept space = (possibly infinite) set of all distinct functions f

$\Rightarrow$ *Language Bias*: restrict the set of all concepts (e.g. linear models)

- Learning algorithms do usually not consider all of *Concept space*

$\Rightarrow$ *Search Bias*: prefer certain functions f over others (e.g. decision tree learning)

Bias is essential for generalization.

alex@seewald.at / alex.seewald.at

A simple learning task

Weather dataset - When to play golf?

Outlook	**Temperature**	**Humidity**	**Windy**	***Play golf?***
overcast	hot	high	false	*yes*
rainy	cool	normal	true	*no*
rainy	mild	high	true	*no*
sunny	cool	normal	false	*yes*

Instance space: 3*3*2*2 = 36 possible instances, of which four are shown with their classification. Since any of these could potentially be either *yes* or *no* classified...
Concept space contains 2^{36} = 68,719,476,736 possible concepts even for this example

Instance space for numerical variables is theoretically infinite. However, computers have only limited numerical precision (2^{32} and 2^{64} for IEEE single and double prec.)

alex@seewald.at / alex.seewald.at

Types of Variables

Categorical Scale (qualitative, discrete, nominal)

- e.g. {Success, Failure}, {ORF1, SAT1, RTL, ...}
- Finite, small number of values; arbitrary ordering of values

Ordinal scale (ordered categorical ~ qualitative)

- e.g. {low, medium, high}, {slow, moderate, fast}
- Ordering is apparent (low<medium<high), but distances are meaningless.

  E.g. the distance between low and medium and between medium and high is not necessarily the same.

Interval/Ratio scale (quantitative, numeric, continuous)

- e.g. temperature in ℃/℉/K, wind speed in km/h
- Interval: Distances between values are meaningful.

  Today it is 2 ℃ colder than yesterday.
- Ratio: Ratios are also meaningful. Zero point has to be known!

  In Florida, the wind blows twice as fast as here.

alex@seewald.at / alex.seewald.at

An (almost) bias-free learner

Rote Learning

- Store all examples/instances from training data
- When predicting, return $f(\mathbf{x}) = \{y_i \mid (\mathbf{x}, y_i) \in \text{TD}\}$
- Logically justified (deduction): exactly those examples which have already been observed are "learned" by RL, and will be returned in the future.
- No Generalization is performed at all! For previously unseen **x**, no prediction is returned ("don't know")

⇒ **A learner which does not restrict concept space either explicitly (language bias) or implicitly (search bias) has not rational basis to classify any unseen examples.**

Definitions

"Bias refers to any criterion for choosing one generalization over another other than strict consistency with the observed training instances"

(Mitchell, 1980)

Variance: Instability of fitting the model to training data. High bias implies low variance (since fewer parameters need to be fitted) and vice versa, so there is a trade-off between bias and variance.

alex@seewald.at / alex.seewald.at

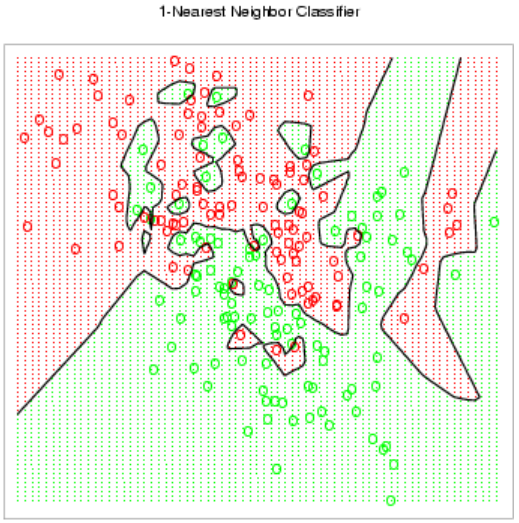
A low-bias learner: Nearest neighbor

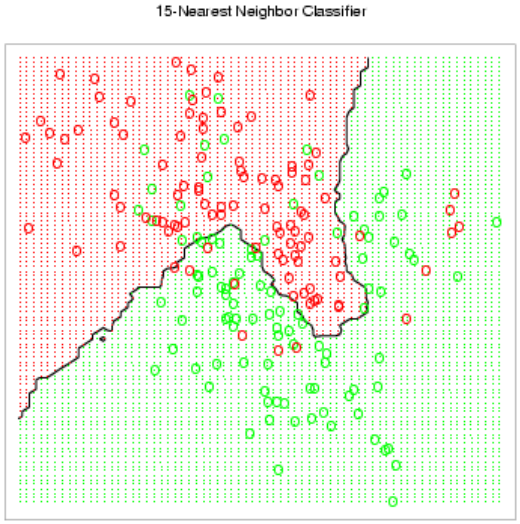
Nearest-neighbor methods (instance-based, lazy l.)

- Store all examples/instances ($\mathbf{x_i}$, y_i) from TD
- When predicting, return $f(\mathbf{x}) = 1/k \sum y_i$ over all $\mathbf{x_i} \in N_k(\mathbf{x})$. $N_k(\mathbf{x})$ is the neighborhood of $\mathbf{x}$ defined by the *k* closest points to $\mathbf{x}$ in training data TD. Closeness is measured by an appropriate distance measure, e.g. Euclidean distance. Some kind of normalization is essential for numeric variables.
- Parameter *k* determines smoothness of fit (upper figure: *k*=1, lower: *k*=15) & trades bias and variance explicitly. Effective number of parameters is N/k (one mean is fitted in each neigborhood) May fit data too closely (*overfitting*)

Bias: $f(\mathbf{x})$ is well approximated by a locally constant function. This is a weak assumption.

Variance: Small fluctuations in training data lead to potentially large fluctuations in $f(\mathbf{x})$, because every example contributes to the exact decision boundary. Variance is high for low *k*.





alex@seewald.at / alex.seewald.at

A low-bias learner: Nearest neighbor (2)

Nearest-neighbor methods (cont.)

Nearest-neighbor methods are *universal approximators*

- Can approximate any model to arbitrary precision given enough training data and sufficiently high *k* (more precisely: $N,k \to \infty$ such that $k/N \to 0$)
- But suffer from *curse of dimensionality*:

  With higher input dimensions, the size of a *k*-neighborhood increases exponentially. E.g. for uniformly distributed inputs in ten dimensions, to capture 1% of the data for a neighborhood, we must cover 63% of the range of each input variable. This is no longer a *local* model, and may suffer from more errors due to higher bias. If we reduce *k* accordingly, we may have too few examples in each neighborhood to approximate the concept well, which increases the errors due to variance. In both cases the errors increase.

  For *k*=N, nearest neighbor is equivalent to a simple baseline classifier, *ZeroR*, which predicts the average y over the whole training data. For classification *ZeroR* predicts the most common class and for regression the arithmetic mean over all y_i. This classifier is widely used as a simple benchmark.

alex@seewald.at / alex.seewald.at

Example: Weather dataset

TD =	Outlook	Temperature	Humidity	Windy	*Play?*
$\mathbf{x_1}$	overcast	31 °F	90%	false	*yes*
$\mathbf{x_2}$	rainy	13 °F	65%	true	*no*
$\mathbf{x_3}$	rainy	20 °F	85%	true	*no*
$\mathbf{x_4}$	sunny	14 °F	55%	false	*yes*
	Avg/StD	19.5±8.27	73.8±16.52		

Classify new example $\mathbf{x_n}$ (overcast, 18 °F, 53%, false), k=1:

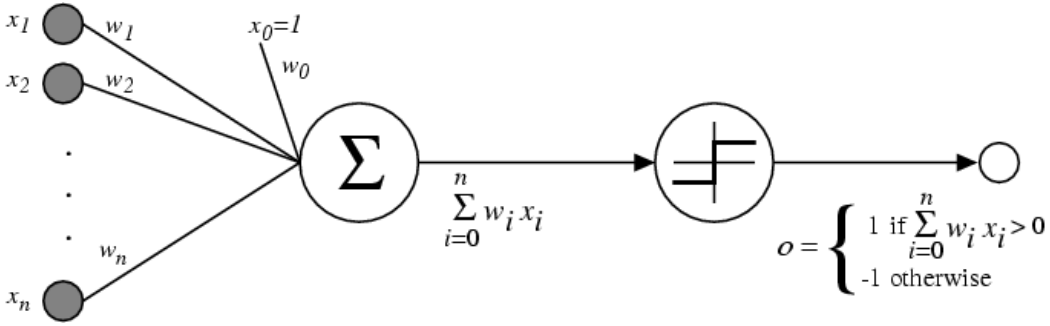
$$D(x,y)=\sqrt{\sum_{\forall i} d(x_i,y_i)} \quad \textit{where} \quad d(x_i,y_i)=\begin{cases}(y_i-x_i)^2 & \text{if i is a numeric attribute}\\ 0 & \text{if } x_i = y_i \text{ and i is qualitative}\\ 1 & \text{if } x_i \neq y_i \text{ and i is qualitative}\end{cases}$$

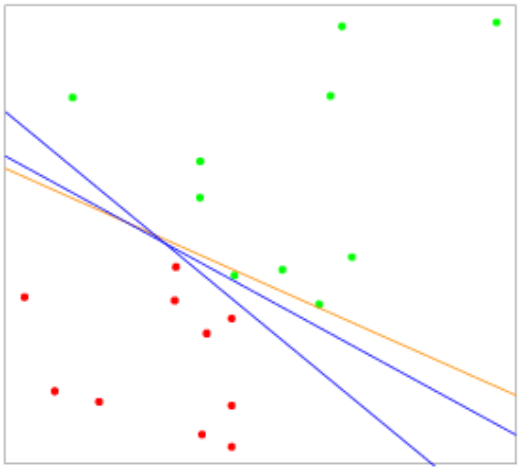
Nearest neighbor $\mathbf{x_4} \Rightarrow$ predict *yes*: $D(x_4,x_n)=\sqrt{1^2+\left(\frac{14-18}{8.27}\right)^2+\left(\frac{55-53}{16.52}\right)^2+0^2}=1.12$

Normalization to zero mean and unit variance (i.e. subtract mean and divide by standard deviation)

alex@seewald.at / alex.seewald.at

A high-bias learner: Perceptron





Perceptron (linear binary threshold unit, linear model)

- Computes a linear function of **x** (assume adding an $x_0=1$ to **x**, so that constant term w_0 can be handled). $f(\mathbf{x}) = \text{sign}(\mathbf{x}^T.\mathbf{w})$. **w** is initialized randomly.
- *Perceptron training rule*: $\mathbf{w} \leftarrow \mathbf{w}+\eta(y-f(\mathbf{x})).\mathbf{x}^T$, where y is the true output value from training data (±1), and η is the learning rate.
- Intuitively, concept boundary is a hyperplane which separates classes +1 & -1.

  Update rule is applied to each training example in turn, repeating until all training examples are classified correctly. Provided η is small enough, and the training set is linearly separable, this algorithm converges in a finite number of steps. If data is not linearly separable, convergence is not assured.

alex@seewald.at / alex.seewald.at

A high-bias learner: Perceptron (2)

Perceptron (cont.)

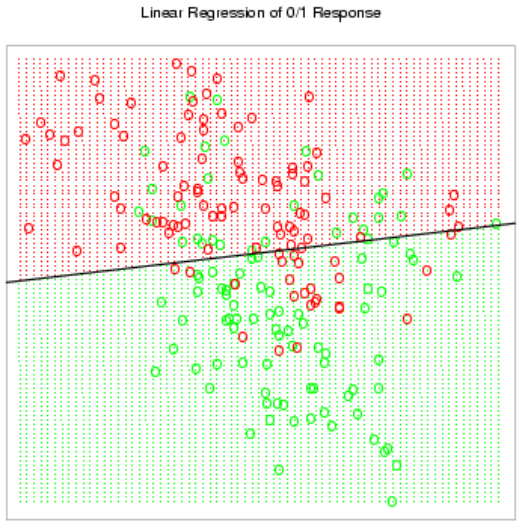
- Number of parameters: p+1 (one for each input dimension plus one constant)

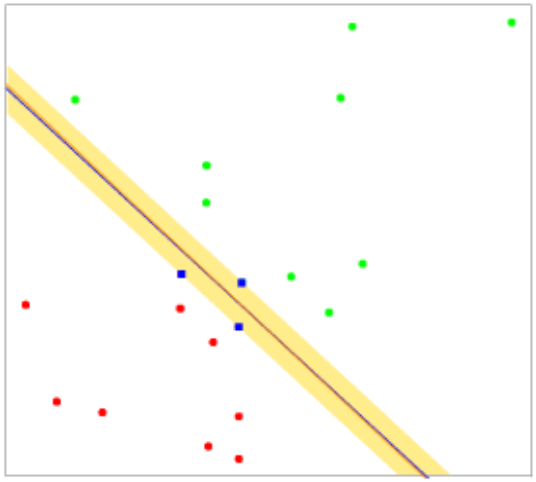
Bias: f(**x**) is well approximated by a linear function.

Variance: small fluctuations in training data have little to no effect. Variance is usually quite low.

Linear methods assume a simple structure of f(**x**). If this assumption is correct, very little training data is sufficient to model f(**x**) well. If not, this will contribute to a high error in estimating the output y (see upper fig.) = *underfitting*. When squared error is used, **w** can be determined analytically (=linear regression).

Linear discriminant analysis, logistic regression and support vector machines (see lower fig.) are refinements of linear models.





alex@seewald.at / alex.seewald.at

Linear Regression

Estimate w from training data analytically (not equivalent to Perceptron)

- Choose to minize residual squared error (RSS) yields:

  RSS(**w**) = $\sum(y_i - \mathbf{x_i}^T.\mathbf{w})^2$ = $(\mathbf{y} - \mathbf{Xw})^T(\mathbf{y}-\mathbf{Xw})$ where **X** is an N x p matrix with each row an input vector $\mathbf{x_i}$ and **y** is an N-vector of the outputs from TD.

  Since we seek the minimum, differentiating the above yields:

  $\mathbf{X^T(y-Xw)}=0 \Leftrightarrow \mathbf{w} = (\mathbf{X^TX})^{-1}\mathbf{X^Ty}$, which can be solved analytically to directly estimate **w** without an iterative method. This is called *linear regression*.

Example: logical AND (i.e. $x_1 \wedge x_2$)

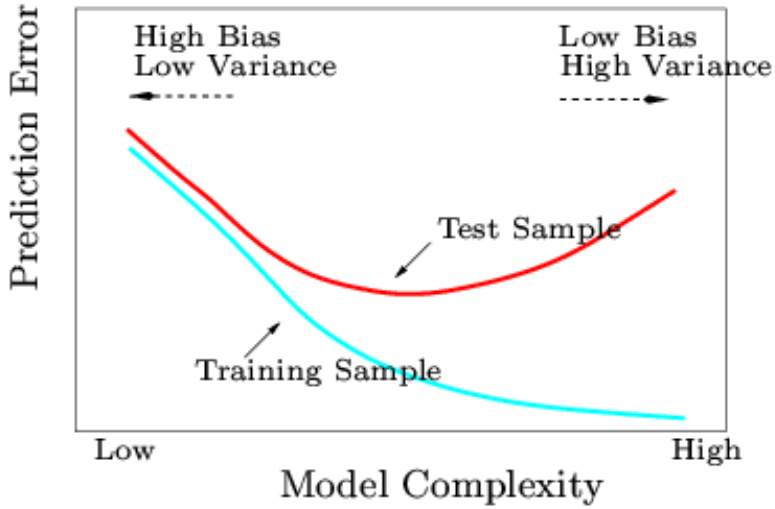
$$X = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}, y = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \rightarrow X^T = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}, X^TX = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} = Q \rightarrow QQ^{-1} = I \Leftrightarrow \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\Leftrightarrow \begin{array}{l} 2a+c=1 \\ 2b+d=0 \\ a+2c=0 \\ b+2d=1 \end{array} \Leftrightarrow \begin{array}{l} a=\frac{2}{3} \\ b=c=-\frac{1}{3} \\ d=\frac{2}{3} \end{array} \rightarrow (X^TX)^{-1} = \begin{pmatrix} \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} \end{pmatrix} \rightarrow \hat{w} = (X^TX)^{-1}X^Ty = \begin{pmatrix} \frac{1}{3} \\ \frac{1}{3} \end{pmatrix} \rightarrow f\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{cases} \frac{1}{3}x_1 + \frac{1}{3}x_2 > 0.5 : 1 \\ \frac{1}{3}x_1 + \frac{1}{3}x_2 \leq 0.5 : 0 \end{cases}$$

Note: For simplicity, we have not used the constant term x_0 and weight w_0 here.

alex@seewald.at / alex.seewald.at

Bias and Variance



Trade-off between Bias and Variance

Low bias: high model complexity, can potentially approximate any f(**x**) well.

However, overfitting can occur - model adapts too closely to training sample and does not generalize well beyond training data, i.e. on test sample.

High bias: low model complexity, can estimate parameters reliably from little data

However, underfitting can occur - model cannot appropriately fit training sample and does not generalize well beyond training data, i.e. on test sample.

Prediction error can be decomposed into *irreducible model error* (due to imperfect data), *variance* (due to model fluctuations), and *squared bias* (due to the fit between model class and data) = *Bias-Variance Decomposition*

alex@seewald.at / alex.seewald.at