Supervised Learning – Classification

Last revision: 14 March 20

Acknowledgements

Material derived from slides for the book
"Elements of Statistical Learning (2nd Ed.)" by T. Hastie,
R. Tibshirani & J. Friedman. Springer (2009)
http://statweb.stanford.edu/~tibs/ElemStatLearn

Material derived from slides for the book
"Machine Learning: A Probabilistic Perspective" by P. Murphy
MIT Press (2012)
http://www.c

Material derived f
"Machine Learn
Cambridge Univ
http://cs.br

Material derived from slides for the book
"Bayesian Reasoning and Machine Learning" by D. Barber
Cambridge University Press (2012)
http://www.cs.ucl.ac.uk/staff/d.barber/brml

Material derived from slides for the book
"Machine Learning" by T. Mitchell
McGraw-Hill (1997)
http://www-2.cs.cmu.edu/~tom/mlbook.html

Material derived from slides for the course
"Machine Learning" by A. Srinivasan
BITS Pilani, Goa, India (2016)

Aims

This lecture will introduce you to machine learning approaches to the problem of *classification*. Following it you should be able to reproduce theoretical results, outline algorithmic techniques and describe practical applications for the topics:

- des
- outl
- defi
- define the Bayes optimal classification rule in ter
- outline the Naive Bayes classification algorith
- describe typical applications of Naive Bayes for t
- outline the Perceptron classification algorithm
- outline the logistic regression classification algorithm

Note: slides with titles marked * are for background only.

Introduction

Classification (sometimes called *concept learning*) methods dominate machine learning . . .

. . . however, they often don't have convenient mathematical properties like regressio

classifier

one of a set of d

We will mostly focus on their advantages and disadvan

methods first, and point to unifying ideas and approac

applicable. In this lecture we focus on classification me

essentially *linear models* . . .

and in later lectures we will see other, more expressive, classifier learning methods.

Nearest neighbour classification

- Related to the simplest form of learning: rote learning or memorization
  - Training instances are searched for instance that **most closely resembles** new or *query* instance
  -
  - *sed* learning;
- The o beyond simple memorization
- Intuitive idea — instances "close by", i.e., neigh *bars*, should be classified similarly
- Instance-based learning is *lazy* learning
- Methods: *nearest-neighbour*, *k-nearest-neighbour*, *lowess*, ...
- Ideas also important for *unsupervised* methods, e.g., clustering (later lectures)

Minkowski distance

Minkowski distance If $\mathcal{X} = \mathbb{R}^d$, the *Minkowski distance* of order $p > 0$ is defined as

$$j=1$$

where $||\mathbf{z}||_p = \left(\sum_{j=1}^{d} |z_j|^p\right)^{1/p}$ is the p-norm (L_p norm) of the vector $\mathbf{z}$.

Minkowski distance

- The 2-norm refers to the familiar *Euclidean distance*:

$$\text{Dis}_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{j=1}^{d} (x_j - y_j)^2}$$

whi

- The 1-*norm* denotes *Manhattan distan*
distance:

$$\text{Dis}_1(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{d} |x_j - y_j|$$

This is the distance if we can only travel along coordinate axes.

Minkowski distance

- If we now let p grow larger, the distance will be more and more dominated by the largest coordinate-wise distance, from which we can infer that $\text{Dis}_\infty(\mathbf{x}, \mathbf{y}) = \max_j |x_j - y_j|$; this is also called *Chebyshev dist*
- Yo orm) which cou corr vectors $\mathbf{x}$ and $\mathbf{y}$ differ. This is not strictly a Mi however, we can define it as

$$\text{Dis}_0(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{d} (x_j - y_j)^0 = \sum_{j=1}^{d} I[x_j = y_j]$$

under the understanding that $x^0 = 0$ for $x = 0$ and 1 otherwise.

Minkowski distance

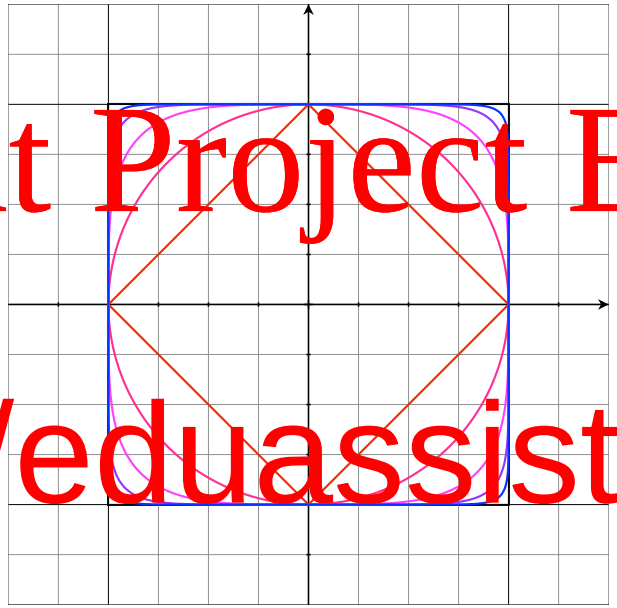
Sometimes the data $\mathcal{X}$ is not naturally in $\mathbb{R}^d$, but if we can turn it into Boolean f

measure

- If $\mathbf{x}$ *ing* *distance*. Alternatively, we can see the Hamming distance as the number of bits that need to be flipped to change

- For non-binary strings of unequal length this ca notion of *edit distance* or *Levenshtein*

Circles and ellipses



Lines connecting points at order-p Minko
for (from inside) $p = 0.8$; $p = 1$ (Manhatt
in red); $p = 1.5$; $p = 2$ (Euclidean distance, the violet circle); $p = 4$; $p = 8$; and $p = \infty$ (Chebyshev distance, the blue rectangle). Notice that for points on the coordinate axes all distances agree. For the other points, our reach increases with p; however, if we require a rotation-invariant distance metric then Euclidean distance is our only choice.

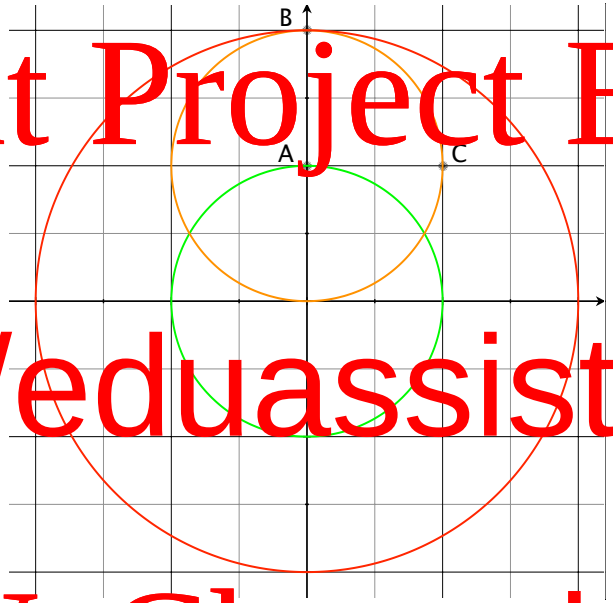
Distance metric

Distance metric – Given an instance space $\mathcal{X}$, a *distance metric* is a function $\mathrm{Dis} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that for any $x, y, z \in \mathcal{X}$:

- dist ;
- all o $x = y$ / $\mathrm{Dis}(x, y) > 0$
- dist
- detours can not shorten the distance: $\mathrm{Dis}(x, z) \leq \mathrm{Dis}(x, y) + \mathrm{Dis}(y, z)$.

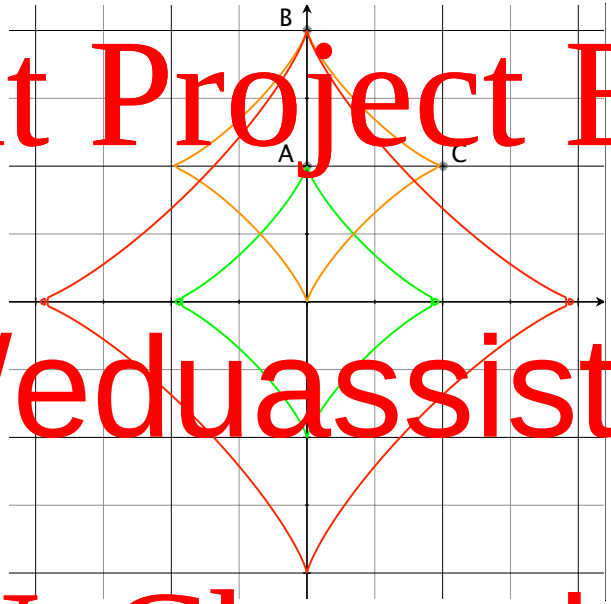
If the second condition is weakened to a non-strict inequ $\mathrm{Dis}(x, y)$ may be zero even if $x \neq y$ – the function Dis is called a *pseudo-metric*.

The triangle inequality – Minkowski distance for $p = 2$



The green circle connects points the same Euclidean di
Minkowski distance of order $p = 2$) away from the origin as A. The orange circle shows that B and C are equidistant from A. The red circle demonstrates that C is closer to the origin than B, which conforms to the triangle inequality.

The triangle inequality – Minkowski distance for $p \leq 1$



With Manhattan distance ($p = 1$), B and C are eq
and also equidistant from A. With $p < 1$ (here, $p = 0.8$) C is further away from the origin than B; since both are again equidistant from A, it follows that travelling from the origin to C via A is quicker than going there directly, which violates the triangle inequality.

Mahalanobis distance *

Often, the shape of the ellipse is estimated from data as the inverse of the covariance matrix: $\mathbf{M} = \mathbf{\Sigma}^{-1}$. This leads to the definition of the *Mahalan*

Using the covariance matrix in this way has the effect of de normalising the features.

Clearly, Euclidean distance is a special case of Mahalanobis distance with the identity matrix $\mathbf{I}$ as covariance matrix: $\text{Dis}_2(\mathbf{x}, \mathbf{y}) = \text{Dis}_M(\mathbf{x}, \mathbf{y}|\mathbf{I})$.

Means and distances

The arithmetic mean minimises squared Euclidean distance *The arithmetic mean $\boldsymbol{\mu}$ of a set of data points D in a Euclidean space is the unique point that minimises the sum of squared Euclidean distances to those data*

Proof. We w denotes th vector of partial derivatives with respect to the zero vector:

$$\nabla_{\mathbf{y}} \sum_{\mathbf{x} \in D} ||\mathbf{x} - \mathbf{y}||^2 = -2 \sum_{\mathbf{x} \in D} (\mathbf{x} - \mathbf{y}) \qquad \mathbf{0}$$

from which we derive $\mathbf{y} = \frac{1}{|D|} \sum_{\mathbf{x} \in D} \mathbf{x} = \boldsymbol{\mu}$.

Means and distances

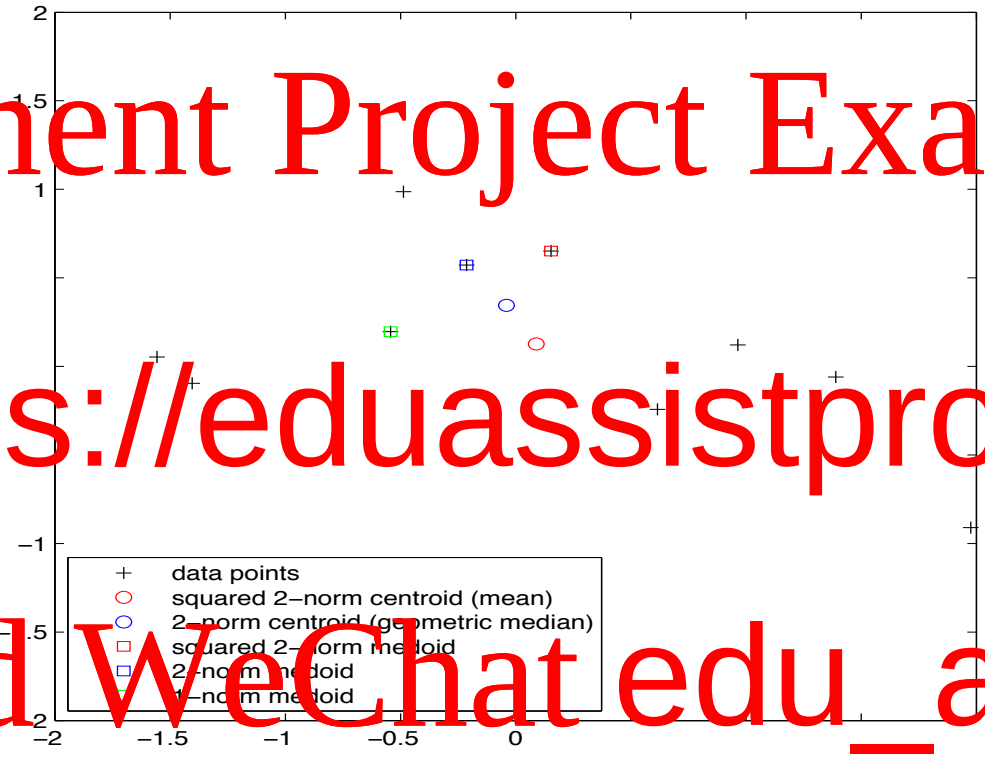
- Notice that minimising the sum of squared Euclidean distances of a given set of points is the same as minimising the *average* squared Euc

- Yo
  it be m
  distance as exemplar?
- This point is known as the *geometric*
  corresponds to the median or 'middle value' of a se
  However, for multivariate data there is no close
  the geometric median, which needs to be calculated by successive approximation.

Means and distances

- In certain situations it makes sense to restrict an exemplar to be one of the given data points. In that case, we speak of a *medoid*, to
  dist e
  to oc
- Fin https://eduassistpro.github.i
  tota
  that minimises it. Regardless of the distance met
  $O(n^2)$ operation for n points.
- So for medoids there is no computational reason t
  distance metric over another.
- There may be more than one medoid.

Centroids and medoids

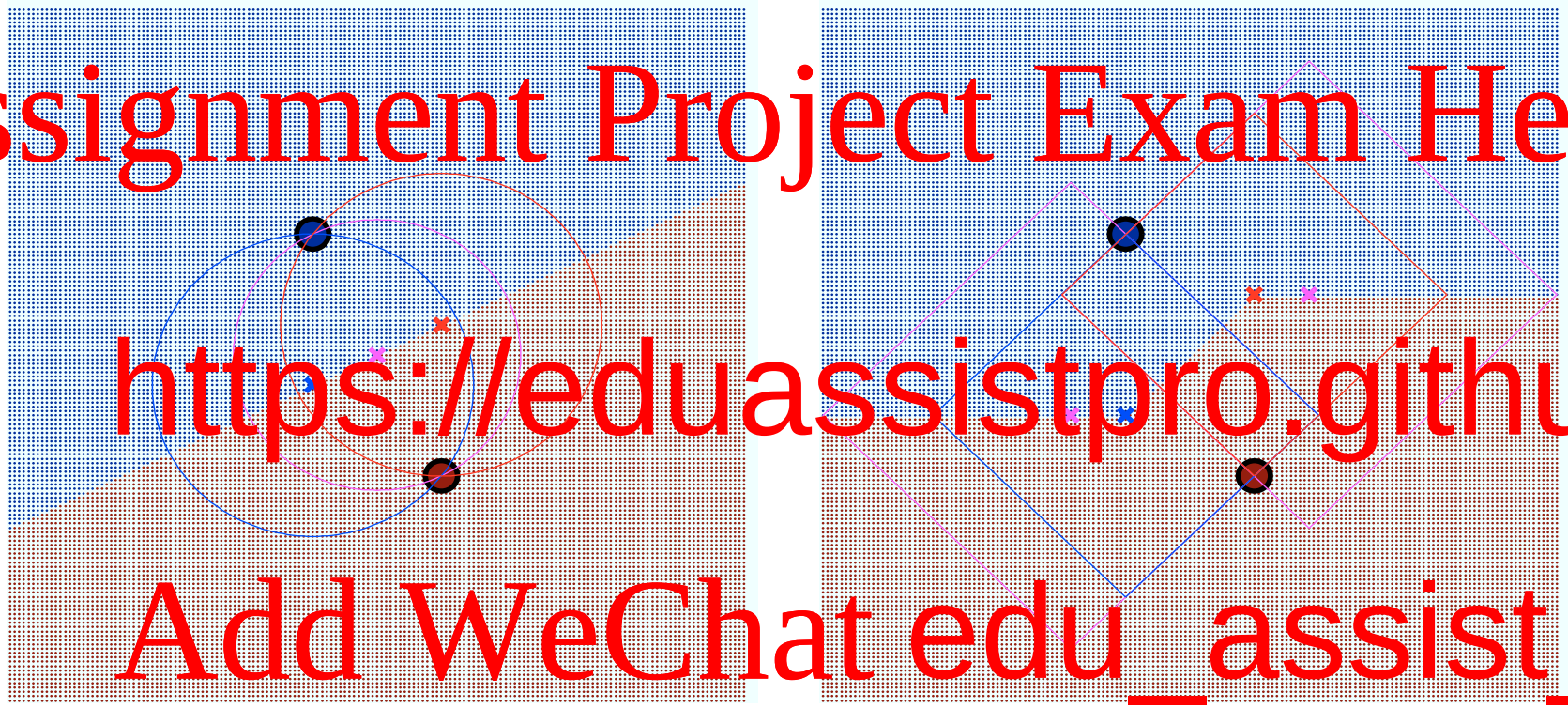


A small data set of 10 points, with circles indicating centroids and squares indicating medoids (the latter must be data points), for different distance metrics. Notice how the outlier on the bottom-right 'pulls' the mean away from the geometric median; as a result the corresponding medoid changes as well.

The basic linear classifier is distance-based

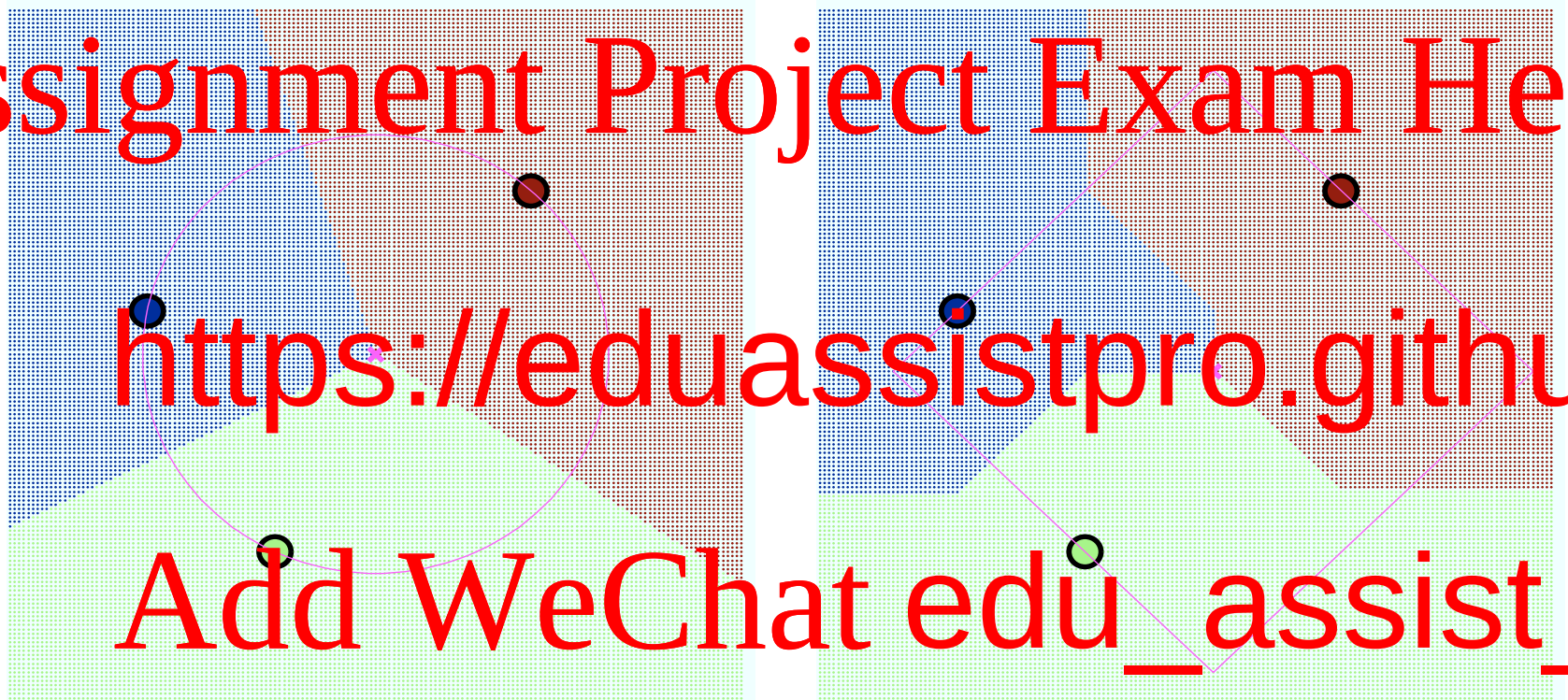
- The basic linear classifier constructs the decision boundary as the perpendicular bisector of the line segment connecting the two exemplars (one for each class).
- An al
  refe x
  is ne
  equivalently, classify an instance to the class of the *nearest* exemplar.
- If we use Euclidean distance as our closeness mea
  geometry tells us we get exactly the same decision
- So the basic linear classifier can be interpreted fro
  perspective as constructing exemplars that minimise squared Euclidean distance within each class, and then applying a nearest-exemplar decision rule.

Two-exemplar decision boundaries



(left) For two exemplars the nearest-exemplar decision rule with Euclidean distance results in a linear decision boundary coinciding with the perpendicular bisector of the line connecting the two exemplars. (right) Using Manhattan distance the circles are replaced by diamonds.

Three-exemplar decision boundaries



(left) Decision regions defined by the 2-norm nearest-exemplar decision rule for three exemplars. (right) With Manhattan distance the decision regions become non-convex.

Distance-based models

To summarise, the main ingredients of distance-based models are

- dist
  Ma
- exe
  distance metric, or medoids that find the most centrally located data point; and
- distance-based decision rules, which take a vot nearest exemplars.

Nearest Neighbour

Stores all training examples $\langle x_i, f(x_i) \rangle$.

Nearest neighbour:

- Giv x_n, then esti

k-Neare

- Given x_q, take vote among its k nearest neighbours (if discrete-valued target function)
- take mean of f values of k nearest ne

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

k-Nearest Neighbour Algorithm

Training algorithm:

- For each training example $\langle x_i, f(x_i) \rangle$, add the example to the list *trai* _

Classific

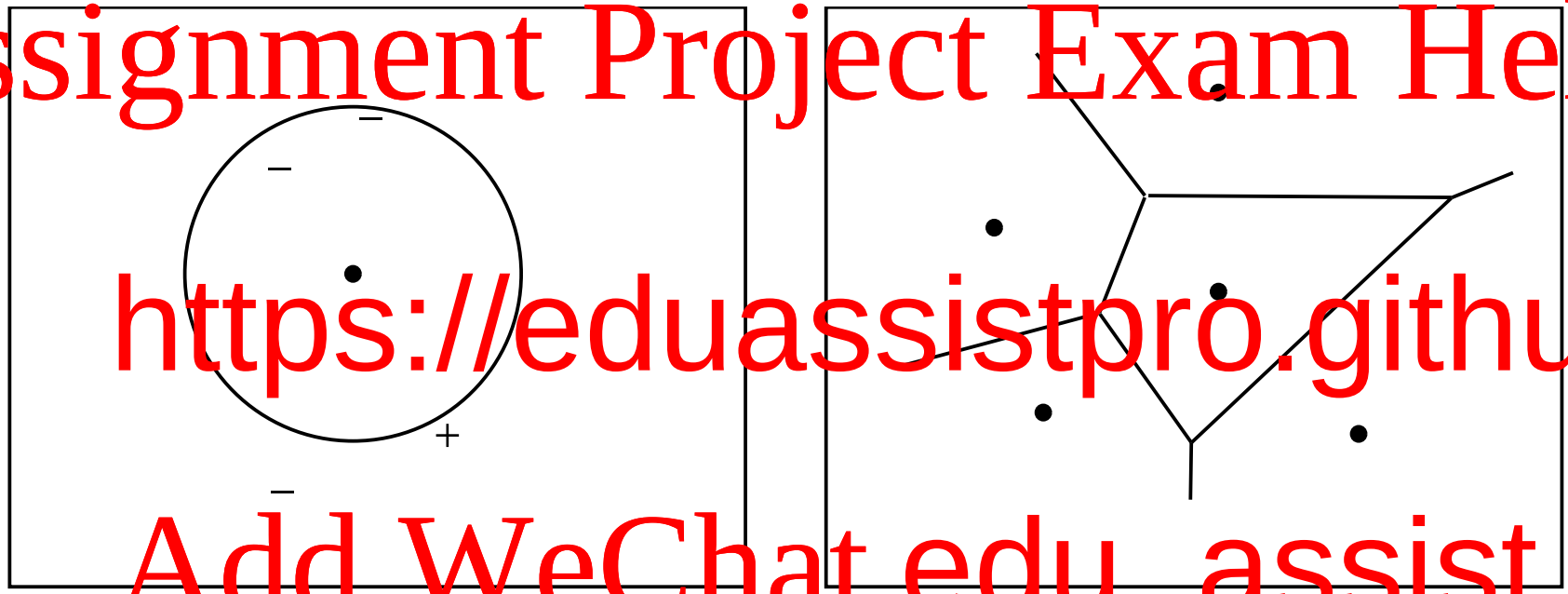
- Giv

  - Let $x_1 \ldots x_k$ be the k instances from *training_examples* that are *nearest* to x_q by the distance function
  - Return

$$\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}$$

where $\delta(a, b) = 1$ if $a = b$ and 0 otherwise.

"Hypothesis Space" for Nearest Neighbour



2 classes, $+$ and $-$ and query point x_q. On left, note effect of varying k. On right, $1-$NN induces a Voronoi tessellation of the instance space. Formed by the perpendicular bisectors of lines between points.

Distance function again

- The distance function defines what is learned.
- Instance x is described by a feature vector (list of attribute-value pairs)

where a_r

Most com

- distance between two instances x_i an

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{m} (a_r(x_i) - a_r(x_j))^2}$$

Distance function again

Many other distance functions could be used ...

- e.g. diff

$$d(x_i, x_j) = \sum_{r=1} |a_r(x\ ) \quad a\ (x\ )$$

Vector-based formalization – use *norm* L_1,

The idea of distance functions will appear again in *kernel methods*.

Normalization and other issues

- Different attributes measured on different scales
- Need to be *normalized* (why ?)

where v_r is the actual value of attribute

- Nominal attributes: distance either 0 or 1
- Common policy for missing values: assumed to b (given normalized attributes)

When To Consider Nearest Neighbour

- Inst
- Les
- Lots of training data
- No requirement for "explanatory" model to be le

When To Consider Nearest Neighbour

Advantages:

- Sta
- Can
- Training is very fast
- Can learn complex target functions
- Don't lose information by generalization - keep

When To Consider Nearest Neighbour

Disadvantages:

- Slow at query time: basic algorithm scans entire training data to deri
- "Cu
- Ass irrelevant attributes
  - Remedy: attribute selection or weights
- Problem of noisy instances:
  - Remedy: remove from data set
  - not easy – how to know which are noisy ?

When To Consider Nearest Neighbour

What is the inductive bias of k-NN?

- an assumption that the classification of query instance x_q will be
  mos
  acc

k-NN uses

- *Regression* approximating a real-valued ta
- *Residual* the error $\hat{f}(x) - f(x)$ in appr
- *Kernel function* function of distance used to
  training example, i.e., kernel function is the function K s.t.
  $w_i = K(d(x_i, x_q))$

Nearest-neighbour classifier

- kNN uses the training data as exemplars, so training is $O(n)$ (but prediction is also $O(n)$!)
- 1N
- By in d dec

- Easily adapted to real-valued targets, and even to structured objects (nearest-neighbour retrieval). Can also outp $k > 1$

- Warning: in high-dimensional spaces everyt everything and so pairwise distances are uninformative (curse of dimensionality)

Distance-Weighted kNN

- Might want to weight nearer neighbours more heavily ...
- Use distance function to construct a weight w_i
- Rep

$$\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \quad w \; \delta(v, f(x\,)) \qquad i=$$

where

$$w_i \equiv \frac{}{d(x_q, x_i)^2}$$

and $d(x_q, x_i)$ is distance between x_q and x_i

Distance-Weighted kNN

For real-valued target functions replace the final line of the algorithm by:

(denominator normalizes contribution of individu

Now we can consider using *all* the training ex k

→ using all examples (i.e., when $k = n$) w
Shepard's method

Evaluation

Lazy learners do not construct an explicit model, so how do we evaluate the output of the learning process ?

- 1-N
- k-N

Leave-one-out cross-validation (LOOCV) – leave o
predict it given the rest:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_{i-1}, y_{i-1}), \quad i+1 \quad i+1 \qquad n \quad n$$

Error is mean over all predicted examples. Fast – no models to be built !

Curse of Dimensionality

Bellman (1960) coined this term in the context of dynamic programming

Imagine instances described by 20 attributes, but only 2 are relevant to target function — "similar" examples will appear "distant".

Curse of di
high-dim

One approach:

- Stretch jth axis by weight z_j, where e
  prediction error
- Use cross-validation to automatically choose weights $z_1, \ldots, z_n$
- Note setting z_j to zero eliminates this dimension altogether

Curse of Dimensionality

- number of "cells" in the instance space grows exponentially in the number of features
- with exponentially many cells we would need exponentially many data points to ensure that each cell is sufficiently populated to make nearest-neighbour predictions reliably

Curse of Dimensionality

Some ideas to address this for instance-based (nearest-neighbour) learning

- Euclidean distance with weights on attributes

- updating of weights based on nearest neighbou
  - class correct/incorrect: weight increased
  - can be useful if not all features used in classificati

See Moore and Lee (1994) "*Efficient Algorithms for Minimizing Cross Validation Error*"

Instance-based (nearest-neighbour) learning

Recap – Practical problems of 1-NN scheme:

- Slow (but fast k d tree-based approaches exist)
- Noi
- All attributes deemed equally important
  - Remedy: attribute weighting (or simply sele
- Doesn't perform explicit generalization
  - Remedy: rule-based or tree-based NN approaches

Refinements of instance-based (nearest-neighbour) classifiers *

- Edi
  ma
- Sav
- IB2: *incremental* NN learner: only incorporates misclassified instances into the classifier
  - Problem: noisy data gets incorporated
- IB3: store *classification performance* i e
  & only use in prediction if above a threshold

Dealing with noise *

Use larger values of k (why ? How to find the "right" k ?

- One way: cross-validation-based k-NN classifier (but slow)
- Diff
  kee
  (IB

  - Computes confidence interval for an instance's success rate and for default accuracy of its class
  - If lower limit of first interval is above upper limit of s instance is accepted (IB3: 5%-level)
  - If upper limit of first interval is below lower limit of s instance is rejected (IB3: 12.5%-level)

Uncertainty

–Albert Einstein

Two Roles for Bayesian Methods

Provides practical learning algorithms:

- Naive Bayes classifier learning
- Bay
- Co
- Ho

Provides useful conceptual framework:

- Provides a "gold standard" for evaluating othe
- Some additional insight into Occam's razor

Bayes Theorem

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

where

- $P(h)$ = prior probability of hypothesis h
- $P(D)$ = prior probability of training data
- $P(h|D)$ = probability of h given D
- $P(D|h)$ = probability of D given h

Choosing Hypotheses

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Generall

Maximu

$$h_{MAP} = \arg\max_{h \in H}$$

$$= \arg\max_{h \in H}$$

$$= \arg\max_{h \in H} P(D|h)P(h)$$

Choosing Hypotheses

If assume

Maximu

$$h_{ML} = \arg\max_{h_i \in H} P(D\ h\ )$$

Applying Bayes Theorem

Does patient have cancer or not?

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in wh
resu
pre
can

$$P(cancer) = \qquad P(\qquad$$

$$P(\oplus \mid cancer) = \qquad P(\ominus \mid cancer) =$$

$$P(\oplus \mid \neg cancer) = \qquad P(\ominus \mid \neg cancer) =$$

Applying Bayes Theorem

Does patient have cancer or not?

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% of the cases in wh
resu
pre
can

$$P(cancer) = .008 \qquad P($$

$$P(\oplus \mid cancer) = .98 \qquad P(\ominus \mid cancer) = .02$$

$$P(\oplus \mid \neg cancer) = .03 \qquad P(\ominus \mid \neg cancer) = .97$$

Applying Bayes Theorem

Does patient have cancer or not?

We can fin

$$P(\oplus \mid \neg cancer)P(\neg cancer) \quad = \quad 0.03 \quad 0.992 = 0.02976$$

Thus $h_{MAP} = \ldots$.

Applying Bayes Theorem

Does patient have cancer or not?

We can find the maximum a posteriori (MAP) hypothesis

$$\frac{784}{976}$$

Thus $h_{MAP} = \neg cancer$.

Also note: posterior probability of hypothesis $cancer$ higher than prior.

Applying Bayes Theorem

How to get the posterior probability of a hypothesis h?
Divide by $P(\oplus)$, probability of data, to normalize result for h:

$$P(h|\oplus) = \frac{P(\oplus|h)P(h)}{P(\oplus)}$$

Denominator ensures we obtain posterior probabil
Sum for all possible numerator values, since hypothes
exclusive (e.g., patient either has cancer or does not).
Marginal likelihood (marginalizing out over hypoth
probability of the data.

Basic Formulas for Probabilities

Product Rule: probability $P(A \wedge B)$ of conjunction of two events A and B:

$$P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$$

Sum Rule

Theorem of total probability: if events A_1 ive with $\sum_{i=1}^{n} P(A_i) = 1$, then:

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

Basic Formulas for Probabilities

Also worth remembering

- *Conditional Probability*: probability of A given B:

- Rearrange sum rule to get:

$$P(A \wedge B) = P(A) +$$

Exercise: Derive Bayes Theorem.

Brute Force MAP Hypothesis Learner

- Idea: view learning as finding the *most probable* hypothesis

- For e

- Output the hypothesis h_{MAP} with the h

$$h_{MAP} = \arg\max_{h \in H} P(h|D)$$

Relation to Concept Learning (i.e., classification)

Canonical concept learning task:

- inst D
- con
  fro
  "zero-error" classification rules)

What would Bayes rule produce as the MAP hypothesi

Does this algorithm output a MAP hypothesis??

Relation to Concept Learning

Brute-Force MAP Framework for Concept Learning:

Assume fixed set of instances $x_1, \ldots, x_m$

Assume

Choose

- $P(h) = \frac{1}{|H|}$ for all h in H

Choose $P(D|h)$:

- $P(D|h) = 1$ if h consistent with D
- $P(D|h) = 0$ otherwise

Relation to Concept Learning

Then:

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{other} \end{cases}$$

Relation to Concept Learning

Note that since likelihood is zero if h is inconsistent then the posterior is also zero. But how did we obtain the posterior for consistent h ?

$$= \frac{1}{\frac{|V S_{H,D}|}{|V S_{H,D}|}}$$

$$= \frac{1}{|V S_{H,D}|}$$

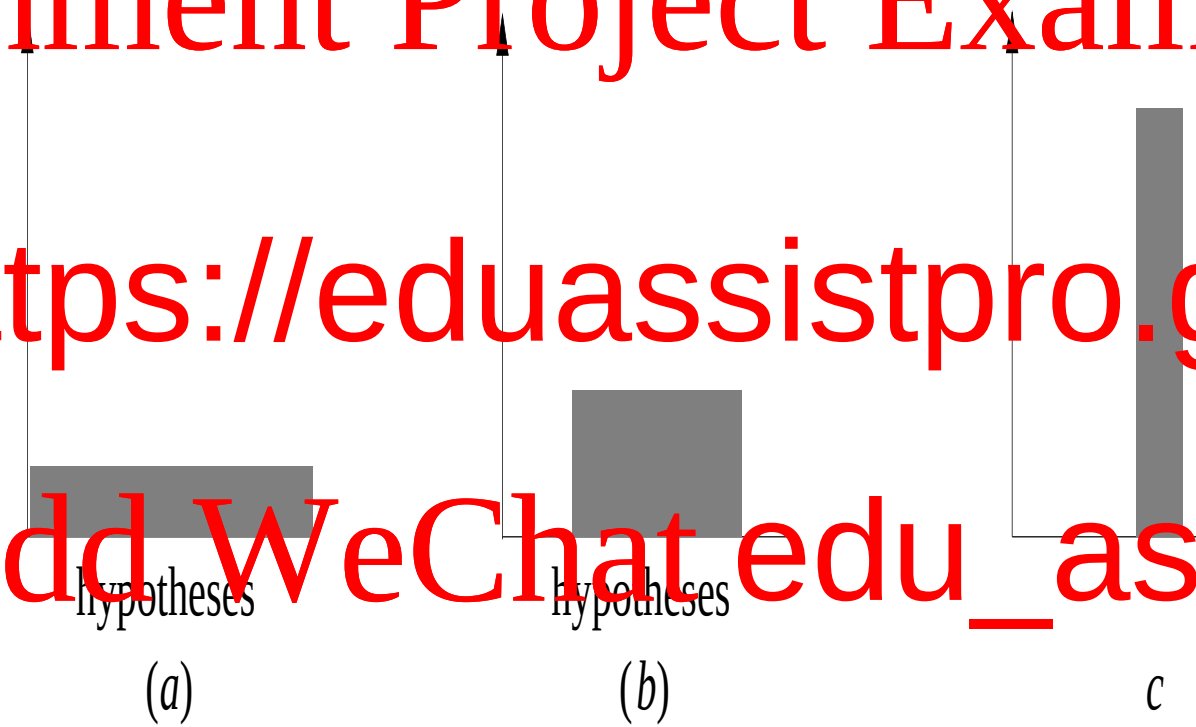
Relation to Concept Learning

How did we obtain $P(D)$? From theorem of total probability:

$$= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} \Big/ \sum_{h_i \in VS_{H,D}}$$

$$= \frac{|VS_{H,D}|}{|H|}$$

Evolution of Posterior Probabilities



Relation to Concept Learning

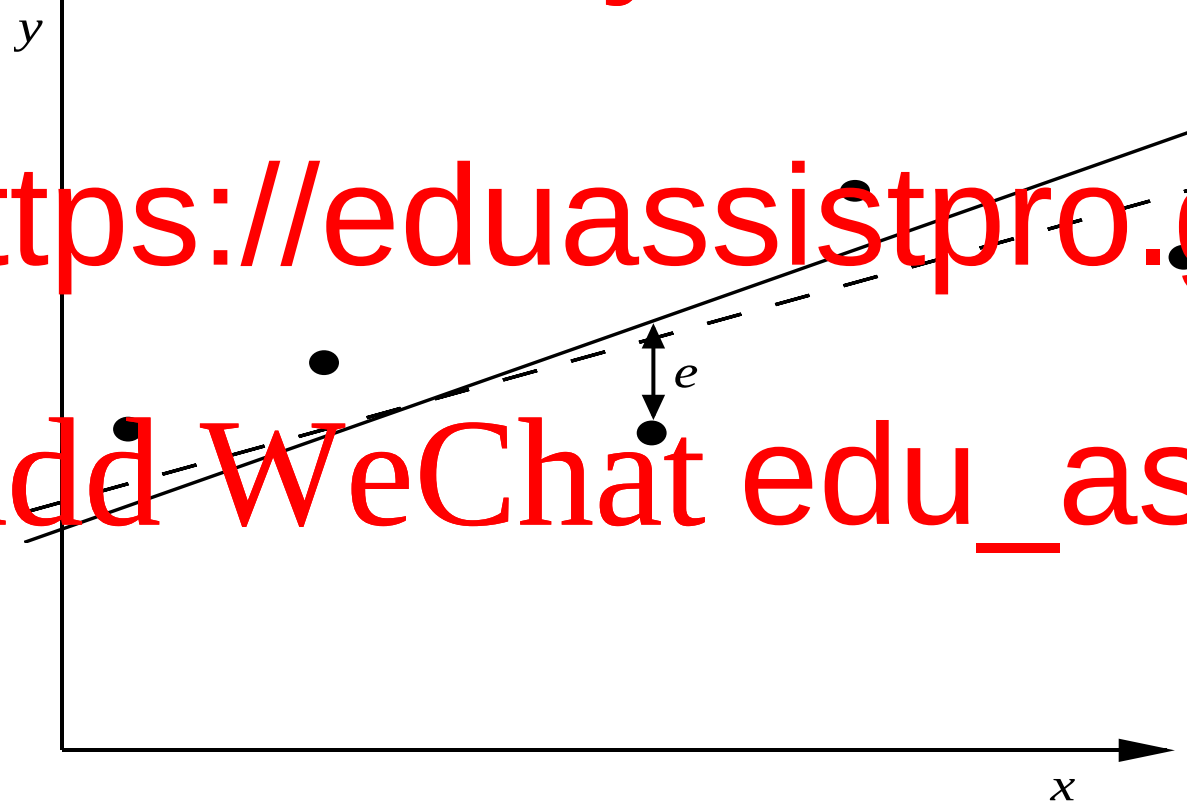
Every hypothesis consistent with D is a MAP hypothesis, if we assume

- unif
- targ
- det
- etc. (see above)

So, this learning algorithm *will* output a MA
does not explicitly use *probabilities* in learni

Learning A Real Valued Function

E.g., learning a linear target function f from noisy examples:



Learning A Real Valued Function

Consider any real-valued target function f

Training examples $\langle x_i, d_i \rangle$, where d_i is noisy training value

- d_i
- e_i i
  acc

x_i

Then the **maximum likelihood** hypothesi
minimizes the sum of squared errors:

$$h_{ML} = \arg\min_{h \in H} \sum_{i=1}^{m} (d_i - h(x_i))$$

Learning A Real Valued Function

How did we obtain this ?

$$= \arg\max_{h \in H} \prod_{i=1}^{m} \frac{1}{\sqrt{2}}$$

Recall that we treat each probability $p(D|h)$ **as if** $h = f$, i.e., **we assume** $\mu = f(x_i) = h(x_i)$, which is the key idea behind maximum likelihood !

Learning A Real Valued Function

Maximize natural log to give simpler expression:

$$h_{ML} = \arg\max_{h \in H} \sum_{i=1}^{m} \ln \frac{1}{\sqrt{\sigma^2}} - \frac{1}{2} \left(\frac{d_i - h(x_i)}{\sigma} \right)^2$$

$$= \arg\max_{h \in H} \sum_{i=1}^{m} -(d_i -$$

Equivalently, we can minimize the positive version of the expression:

$$h_{ML} = \arg\min_{h \in H} \sum_{i=1}^{m} (d_i - h(x_i))^2$$

Discriminative and generative probabilistic models

- *Discriminative models* model the posterior probability distribution $P(Y|X)$, where Y is the target variable and X are the features. That is, given X they return a probability distribution over Y.
- *Generative models* model the joint distribution $P(Y, X)$ of the target Y and the fe
deri
In par
distribution can be obtained as $P(Y|X) = \frac{P(Y,X)}{\quad}$.

- Alternatively, generative models can be described $P(X|Y)$, since $P(Y, X) = P(X|Y)P(Y)$ (usually abbreviated to 'prior') can be easily estimat

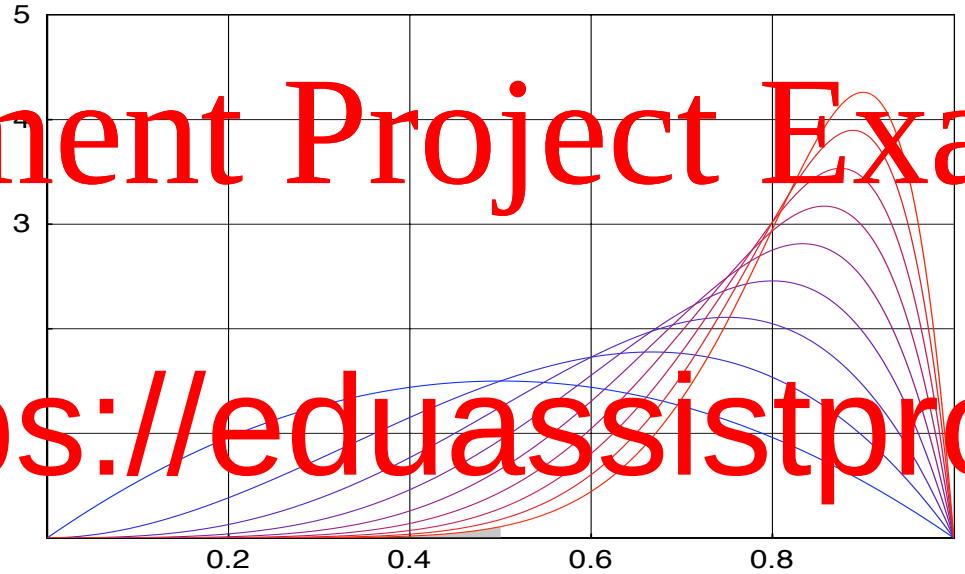
- Such models are called 'generative' because we can sample from the joint distribution to obtain new data points together with their labels. Alternatively, we can use $P(Y)$ to sample a class and $P(X|Y)$ to sample an instance for that class.

Assessing uncertainty in estimates

Suppose we want to estimate the probability θ that an arbitrary e-mail is spam, so that we can use the appropriate prior distribution.

- The natural thing to do is to inspect n e-mails, determine the number of spam e-mails d, and set $\hat{\theta} = d/n$; we don't really need any com
- Ho ximum a pos θ are completely ruled out.
- We model this by a probability distribution over tion in this case) which is updated each time new inform This is further illustrated in the figure for a distribution that is more and more skewed towards spam.
- For each curve, its bias towards spam is given by the area under the curve and to the right of $\theta = 1/2$.

Assessing uncertainty in estimates



Each time we inspect an e-mail, we are reducing our uncertainty about the prior spam probability θ. After we inspect two e-mails, one spam, the possible θ values are characterised by a symmetric distribution around $1/2$. If we inspect a third, fourth, ..., tenth e-mail and each time (except the first one) it is spam, then this distribution narrows and shifts a little bit to the right each time. The distribution for n e-mails reaches its maximum at $\hat{\theta}_{\text{MAP}} = \frac{n-1}{n}$ (e.g., $\hat{\theta}_{\text{MAP}} = 0.8$ for $n = 5$).

The Bayesian perspective

Explicitly modelling the posterior distribution over the parameter θ has a number of advantages that are usually associated with the "Bayesian perspecti

- We c
esti
- We can obtain a generative model for the parameter by sampling from the posterior distribution, which contain
information than a summary statistic such as th
convey – so, rather than using a single e-mail with ur
generative model can contain a number of e-mails with θ sampled from the posterior distribution.

The Bayesian perspective

- We can quantify the probability of statements such as 'e-mails are biased towards ham' (the tiny shaded area in the figure demonstrates that after observing one ham and nine spam e-mails this probability is very
- We c
  if we b
  50–
  mmetric
  one in the figure on the previous slide) as our prior.

The key point is that probabilities do not have to be interpr
estimates of relative frequencies, but can carry the mo
of (possibly subjective) degrees of belief.

Consequently, we can attach a probability distribution to almost anything: not just features and targets, but also model parameters and even models.

Minimum Description Length Principle

Once again, the MAP hypothesis

$$h = \arg\max P(D\ h)P(h)$$

Which is eq

$$h_{MAP} = \arg\max_{h \in H} \log_2 P($$

Or

$$h_{MAP} = \arg\min_{h \in H} -\log_2 P(D|h) - \log_2 P(h)$$

Minimum Description Length Principle

Interestingly, this is an expression about a quantity of *bits*.

$$(1)$$

From information theory:

> *The optimal (shortest expected coding length with probability p is $-\log_2 p$ bits.*

Minimum Description Length Principle

So interpr

- – lo
- – lo

Note well: assumes *optimal* encodings, w

are known. In practice, this is difficult and makes a differ

Minimum Description Length Principle

Occam's razor: prefer the shortest hypothesis

MDL: pre

$$MDL \qquad C_1 \qquad C_2$$
$$h \in H$$

where $L_C(x)$ is the description length of x C

Minimum Description Length Principle

Without loss of generality, classifier is here assumed to be a decision tree

Example: H = decision trees, D = training data labels

- L_{C_1}
- L_{C_2}
  - C_2 ed only describe exceptions
- Hence h_{MDL} trades off tree size for training er
  - i.e., prefer the hypothesis that minimizes

$$length(h) + length(misclassifications)$$

Most Probable Classification of New Instances

So far we'v D (i.e., h_{MAP})

Given new instance x, what is its most probable *classification*?

- $h_{MAP}(x)$ is not the most probable classificat

Most Probable Classification of New Instances

Consider

- Thr
- Giv

$$h_1(x) = +,\ h_2(x) = -,\ h_3(x) = -$$

- What's most probable classification of

Bayes Optimal Classifier

Bayes optimal classification:

Example

$$P(h_1|D) = .4, \quad P(-|h_1) = 0$$
$$P(h_2|D) = .3, \quad P(-|h_2) = 1$$
$$P(h_3|D) = .3, \quad P(-|h_3) = 1, \quad P(+|h_3) = 0$$

Bayes Optimal Classifier

therefore

$$P(+|h_i)P(h_i|D) = .4$$

and

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P$$

No other classification method using the same hypothesis space and same prior knowledge can outperform this method on average

Bayes Error

What is the best performance attainable by a (two-class) classifier ?

Define the probability of error for classifying some instance x by

$$P(\text{error } x) = P(\text{class}_1\ x) \quad \text{if we predict class}_2$$

This gives

$$\Sigma_x \quad P(\text{error}) = P(\text{er}$$

So we can justify the use of the decision rule

$$\text{if} \quad P(\text{class}_1|x) > P(\text{class}_2|x) \quad \text{then} \quad \text{predict class}_1$$
$$\text{else} \quad \text{predict class}_2$$

On average, this decision rule minimises probability of classification error.

Naive Bayes Classifier

Along with decision trees, neural networks, nearest neighbour, one of the most practical learning methods.

When to us

- Mo
- Attributes that describe instances are conditionally independent given classification

Successful applications:

- Classifying text documents
- Gaussian Naive Bayes for real-valued data

Naive Bayes Classifier

Assume target function $f : X \rightarrow V$, where each instance x described by attributes $\langle a_1, a_2 \ldots a_n \rangle$.

Most prob

$$v_{MAP} = \arg\max_{v_j \in V} \frac{P(a_1, a}{P}$$

$$= \arg\max_{v_j \in V} P(a_1, a$$

Naive Bayes Classifier

Naive Bayes assumption:

$$P(a_1, a_2 \ldots a_n \; v_j) = \qquad P(a_i \; v_j)$$

- Att
  - which means knowledge about the value of a particular attribute tells us nothing about the value of another attribute

which gives

Naive Bayes classifier: $v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i|v_j)$

Naive Bayes Algorithm

Naive_Bayes_Learn($examples$)

For each target value v_j

$\hat{P}(v_j) \leftarrow$ estimate $P(v_j)$

Classify_New_Instance(x)

$$v_{NB} = \arg\max_{v_j \in V} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)$$

Naive Bayes Example

Consider *PlayTennis* again...

									dy
									es
Sun	2	3							
Ove	4	0							
Rai	3	2							
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5
Overcast	4/9	0/5	Mild	4/9	2/5	Nor	6	1/			
Rainy	3/9	2/5	Cool	3/9	1/5						
Play											
Yes		No									
9		5									
9/14		5/14									

Naive Bayes Example

Say we have the new instance:

$$\langle \quad true \rangle$$

We want to

$$v_{NB} = \underset{v_j \in \{\text{"yes"}, \text{"no"}\}}{\arg\max} \quad P$$

Naive Bayes Example

So we first calculate the likelihood of the two classes, "yes" and "no".

for "yes" $P(true|y)$

$$0.00 = \frac{}{} \times \frac{}{} \times \frac{}{} \times \frac{}{} \times \frac{}{}$$

for "no" $= P(n) \times P(sun|n) \times P(co \quad true|n)$

$$0.0206 = \frac{5}{14} \times \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5}$$

Naive Bayes Example

Then convert to a probability by normalisation

$$P(\text{“no”}) = \frac{\quad}{(0.005}$$

$$= \frac{}{0.795}$$

The Naive Bayes classification is “no”.

Naive Bayes: Subtleties

Conditional independence assumption is often violated

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

- ...b
pos

$$\arg\max_{v_j \in V} \hat{P}(v_j) \qquad \hat{P}(a_i | v_j) = \arg\max_{v_j \in} \qquad)$$

i.e. maximum probability is assigned to correct class

- see [Domingos & Pazzani, 1996] for analysis
- Naive Bayes posteriors often unrealistically close to 1 or 0
- adding too many redundant attributes will cause problems (e.g. identical attributes)

Naive Bayes: "zero-frequency" problem

What if none of the training instances with target value v have attribute value a_i

$$P(v_j) \qquad P(a_i|v_j) = 0$$

Pseudo-counts add 1 to each count (a version of the i *ator*)

Naive Bayes: "zero-frequency" problem

- In some cases adding a constant different from 1 might be more appropriate
- Exa

$$\frac{\overline{2}}{9+\mu} \qquad \frac{\overline{3}}{9+\mu} \qquad \frac{\overline{3}}{}$$

- Weights don't need to be equal (if they sum to 1) – a for *prior*

Sunny *Overcast* *Rain*

$$\frac{2+\mu p_1}{9+\mu} \qquad \frac{4+\mu p_2}{9+\mu} \qquad \frac{3+\mu p_3}{9+\mu}$$

Naive Bayes: "zero-frequency" problem

This generalisation is a Bayesian estimate for $\hat{P}(a_i|v_j)$

$$\frac{n_c + mp}{}$$

where

- n is number of training examples for which $v = v_j$,
- n_c number of examples for which $v =$
- p is prior estimate for $\hat{P}(a_i|v_j)$
- m is weight given to prior (i.e. number of "virtual" examples)

This is called the m-estimate of probability.

Naive Bayes: missing values

- Tra
valu
- Classification: attribute will be omitted from ca

Naive Bayes: numeric attributes

- Usual assumption: attributes have a *normal* or *Gaussian* probability distribution (given the class)
- The ned by t

$$\mu = \frac{}{n} \quad x_i$$

The standard deviation σ:

$$\sigma = \frac{1}{n-1} \sum_{i=1} (x_i - \mu)^2$$

Naive Bayes: numeric attributes

Then we have the density function $f(x)$:

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

Example: mean = 73, and standard deviation = 6.2. Density value

$$f(\text{temperature} = 66 | \text{“yes”}) = \frac{1}{\sqrt{2\pi}6.2} e^{\frac{(66-73)^2}{2\cdot 6.2^2}}$$

Missing values during training are not included in calculation of mean and standard deviation.

Naive Bayes: numeric attributes

Note: the normal distribution is based on the simple exponential function

$$f(x) = e^{-|x|^m}$$

As the pow
function.
Where $m = 2$

$$f(x) = e^{-|x}$$

and this is the basis of the normal distribution – the variou
the result of scaling so that the integral (the area under the
$-\infty$ to $+\infty$) is equal to 1.

from "Statistical Computing" by Michael J. Crawley (2002) Wiley.

Categorical random variables

Categorical variables or features (also called discrete or nominal) are ubiquitous in machine learning.

- Perhaps the most common form of the Bernoulli distribution models whether or not a word occurs in a document. That is, for the i-th word in the vocabulary, $X_i = 1$ if the word is generated by a Bernoulli distribution.

- Variables with more than two outcomes are also common. For example, every word position in an e-mail corresponds to a categorical variable with k outcomes, where k is the size of the vocabulary. The multinomial distribution manifests itself as a histogram of the number of occurrences of all vocabulary words in a document. This establishes an alternative way of modelling text documents that allows the number of occurrences of a word to influence the classification of a document.

Categorical random variables

Both these document models are in common use. Despite their differences, they both assume independence between word occurrences, generally referred to as the *naive Bayes assumption*.

- In the multinomial document model, this follows from the very use of the
  wor
  dist
- In th
  vector are statistically independent, which all
  joint probability of a particular bit vector uct of the probabilities of each component

- In practice, such word independence assumptions are often not true: if we know that an e-mail contains the word 'Viagra', we can be quite sure that it will also contain the word 'pill'. Violated independence assumptions reduce the quality of probability estimates but may still allow good classification performance.

Example application: Learning to Classify Text

In machin
learning t

Here is a simplified version in the multinomial docume

Learning to Classify Text

Why?

- Lea
- Lea

Naive Bayes is among most effective algorithms

What attributes shall we use to represent text docume

Learning to Classify Text

Target concept $Interesting? : Document \rightarrow \{+, -\}$

1. Rep
2. Lea
   - $P(+)$
   - $P(-)$
   - $P(doc|+)$
   - $P(doc|-)$

Learning to Classify Text

Naive Bayes conditional independence assumption

where $P(a_i = w_k|v_j)$ is probability that word in p n v_j

one more assumption: $P(a_i = w_k, v_j) = P$

"bag of words"

Learning to Classify Text

LEARN_NAIVE_BAYES_TEXT($Examples, V$)

// collect all words and other tokens that occur in Examples

$Vocabulary \leftarrow$ all distinct words and other tokens in $Examples$

// calculat

for each tar

$docs_j$

$P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$

$Text_j \leftarrow$ a single document created by concatenati $docs_j$

$n \leftarrow$ total number of words in $Text_j$ (count s)

for each word w_k in $Vocabulary$

$n_k \leftarrow$ number of times word w_k occurs in $Text_j$

$P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

Learning to Classify Text

CLASSIFY_NAIVE_BAYES_TEXT(Doc)

- pos nd in Voc
- Ret

$$v_{NB} = \arg\max_{v_i \in V} P(v_i)$$

Application: 20 Newsgroups

Given: 1000 training documents from each group

Learning task: classify each new document by newsgroup it came from

comp.graphics

misc.forsale

alt.atheism

soc.religion.christian

talk.religion.misc

talk.politics.mideast

talk.politics.misc

talk.politics.guns

sci.space

sci.crypt

sci.electro

sci.med

Naive Bayes: 89% classification accuracy

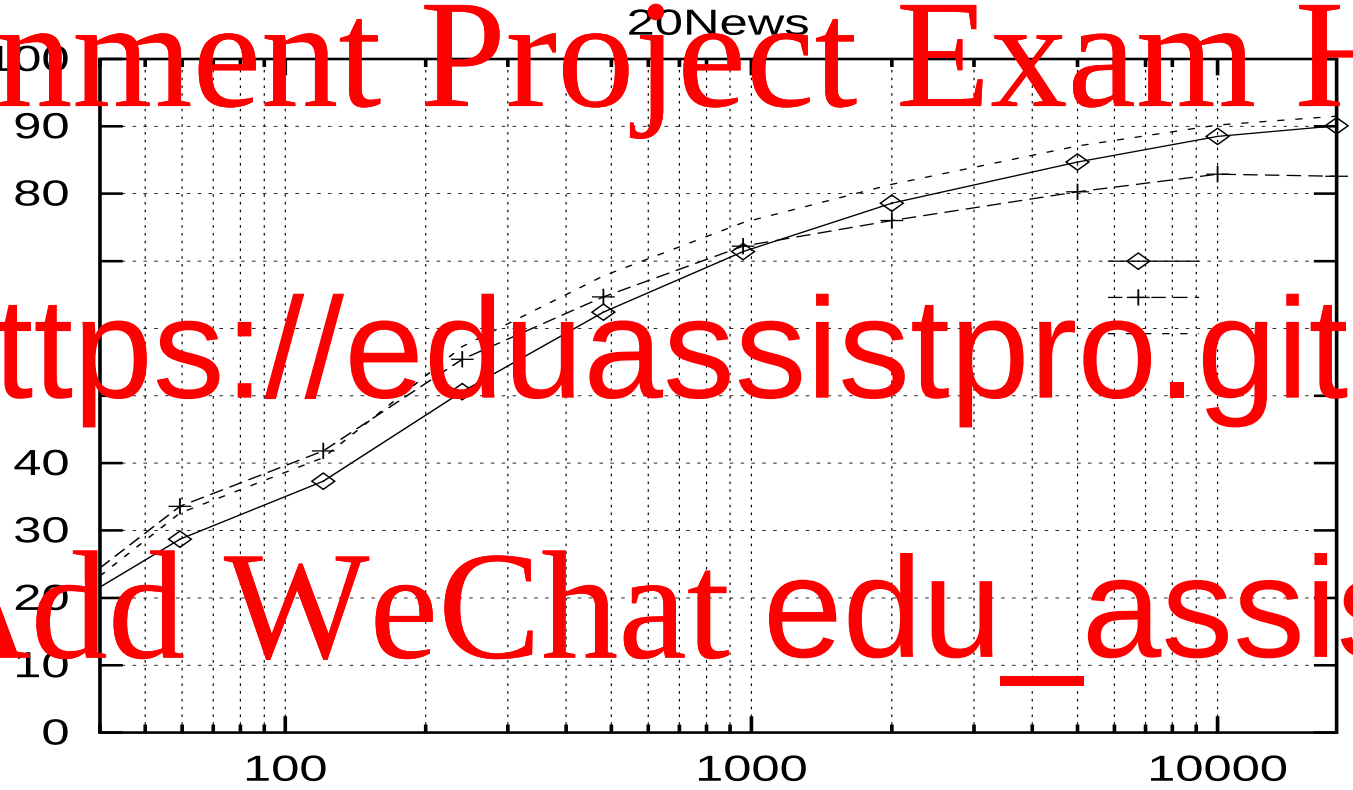
Article from rec.sport.hockey

Path: cantaloupe.srv.cs.cmu.edu!das-news.harvard.edu!ogicse!uwm.edu
From: xxxyyy@zzz.edu (John Doe)
Subject: Re: This year's biggest and worst (opinion)...
Date: 5 Apr 9

I can only co
for pleas
a defensive defenseman, but he's clearly much more than that.
Great skater and hard shot (though wish he were more accu

In fact, he pretty much allowed the Kings to trade away tha
huge defensive liability Paul Coffey. Kelly Hrude
biggest disappointment if you thought he was any good to begin
with. But, at best, he's only a mediocre goaltender. A better
choice would be Tomas Sandstrom, though not through any fault
of his own, but because some thugs in Toronto decided ...

Learning Curve for 20 Newsgroups



Accuracy vs. Training set size (1/3 withheld for test)

Probabilistic decision rules

We have chosen one of the possible distributions to model our data X as coming from either class

- The more different $P(X \mid Y = \text{spam})$ and $P(X \mid Y = \text{ham})$ are, the mor
- Thu $Y = \text{spam})$ and rules:

maximum likelihood (ML) – predict $\arg\max_y P(X = x|Y = y)$

maximum a posteriori (MAP) – predict $\arg\max_y P(X = x|Y = y)P(Y = y)$;

The relation between the first two decision rules is that ML classification is equivalent to MAP classification with a uniform class distribution.

Probabilistic decision rules

We again
illustrate

Prediction using a naive Bayes model

Suppose our vocabulary contains three words a, b and c, and we use a multivariate Bernoulli model for our e-mails, with parameters

$$)$$

This mean am
(+), com
The e-mail to be classified contains words e is
described by the bit vector $\mathbf{x} = (1, 1, 0)$. We

$$P(\mathbf{x}|\oplus) = 0.5 \cdot 0.67 \cdot$$

$$P(\mathbf{x}|\ominus) = 0.67 \cdot 0.33 \cdot (1 - 0.33) = 0.148$$

The ML classification of $\mathbf{x}$ is thus spam.

Prediction using a naive Bayes model

In the case of two classes it is often convenient to work with likelihood ratios and odds.

- The likelihood ratio can be calculated as

$$\frac{P(\mathbf{x}\quad)}{\quad} \quad \frac{0.5}{\quad} \frac{0.67}{\quad} \frac{1}{\quad} \quad \frac{0.33}{\quad}$$

- Thi nor odds are more than $2/3$, but ham if they are less than that.
- For example, with 33% spam and 67 $\frac{P(\oplus)}{P(\ominus)} = \frac{0.3}{0.67} = 1/2$, resulting in a posterior odd

$$\frac{P(\oplus|\mathbf{x})}{P(\ominus|\mathbf{x})} = \frac{P(\mathbf{x}|\oplus)}{P(\mathbf{x}|\ominus)} \frac{P(\oplus)}{P(\ominus)} = 3/2 \cdot 1/2 = 3/4 < 1$$

In this case the likelihood ratio for $\mathbf{x}$ is not strong enough to push the decision away from the prior.

Prediction using a naive Bayes model

Alternatively, we can employ a multinomial model. The parameters of a multinomial establish a distribution over the words in the vocabulary, say

$$\theta^{\oplus} = (0.3, 0.5, 0.2) \qquad \theta^{\ominus} = (0.6, 0.2, 0.2)$$

The e-mai a, one single occurren cribed by the co occurren

$$P(\mathbf{x}|\oplus) = 4! \frac{0.3^3}{3!} \frac{}{} \frac{}{}$$

$$P(\mathbf{x}|\ominus) = 4! \frac{0.6^3}{3!} \frac{0.}{1!} \frac{}{0!}$$

The likelihood ratio is $\left(\frac{0.3}{0.6}\right)^3 \left(\frac{0.5}{0.2}\right)^1 \left(\frac{0.2}{0.2}\right)^0 = 5/16$. The ML classification of $\mathbf{x}$ is thus ham, the opposite of the multivariate Bernoulli model. This is mainly because of the three occurrences of word a, which provide strong evidence for ham.

Training data for naive Bayes

A small e-mail data set described by count vectors:

e_3				
e_4	2	3	0	
e_5	4	3	0	
e_6	4	0	3	
e_7	3	0	0	–
e_8	0	0	0	–

Training data for naive Bayes

The same data set described by bit vectors

	3			
e_4	1	1	0	
e_5	1	1	0	
e_6	1	0	1	
e_7	1	0	0	–
e_8	0	0	0	–

Training a naive Bayes model

Consider the following e-mails consisting of five words a, b, c, d, e:

e_1: $b\ d\ e\ b\ b\ d\ e$

e_2

e_3

e_4

e_5: $a\ b\ a\ b\ a\ b\ a\ e\ d$

We are told that the e-mails on the left are spam and those on the right are ham, and so we use them as a small training set to train our classifier.

- First, we decide that d and e are so-cal o common to convey class information.
- The remaining words, a, b and c, constitute our vocabulary.

Training a naive Bayes model

For the multinomial model, we represent each e-mail as a count vector, as before.

- In or
  the c am and
  $(11,$

- To s
  each vocabulary word, which brings the total nu
  of vocabulary words to 20 for each class.

- The estimated parameter vectors are thus
  $\hat{\boldsymbol{\theta}}^{\oplus} = (6/20, 10/20, 4/20) = (0.3, 0.5, 0.2)$ for spam and
  $\hat{\boldsymbol{\theta}}^{\ominus} = (12/20, 4/20, 4/20) = (0.6, 0.2, 0.2)$ for ham.

Training a naive Bayes model

In the multivariate Bernoulli model e-mails are represented by bit vectors, as before.

- Adding the bit vectors for each class results in $(2, 3, 1)$ for spam and $(3,$
- Eac
ord
particular vocabulary word.
- Probability smoothing now means adding tw
containing each word and one containing none o
- This results in the estimated parameter vectors
$\hat{\boldsymbol{\theta}}^{\oplus} = (3/6, 4/6, 2/6) = (0.5, 0.67, 0.33)$ for spam and
$\hat{\boldsymbol{\theta}}^{\ominus} = (4/6, 2/6, 2/6) = (0.67, 0.33, 0.33)$ for ham.

Linear discriminants

Many forms of linear discriminant from statistics and machine learning, e.g.,

- Fish
- Log
- Perceptron
  - a linear threshold classifier
  - an early version of an artificial "neuron"
  - still a useful method, and source of ideas

Logistic regression

In the case of a two-class problem, model the probability of one class $P(Y=1)$ vs. the alternative $(1-P(Y=1))$:

$$\frac{1}{}$$

or

$$\ln \frac{P(Y=1|\mathbf{x})}{1-P(Y=1|\mathbf{x})}$$

The quantity on the l.h.s. is called the *log*

model for the logit.

Note: to fit this is actually more complex than linear regression, so we omit the details.

Generalises to multiple class versions (Y can have more than two values).

Perceptron

A linear classifier that can achieve perfect separation on linearly separable data is the *perceptron*, originally proposed as a simple *neural network* by F. Rosenblatt in the late 1950s.

Perceptron

Originally implemented in software (based on the McCulloch-Pitts neuron from the 1940s), then in hardware as a 20x20 visual sensor array with potentiometers for adaptive weights.

Source `http://en.wikipedia.org/w/index.php?curid=47541432`

Perceptron

Output o is thresholded sum of products of inputs and their weights:

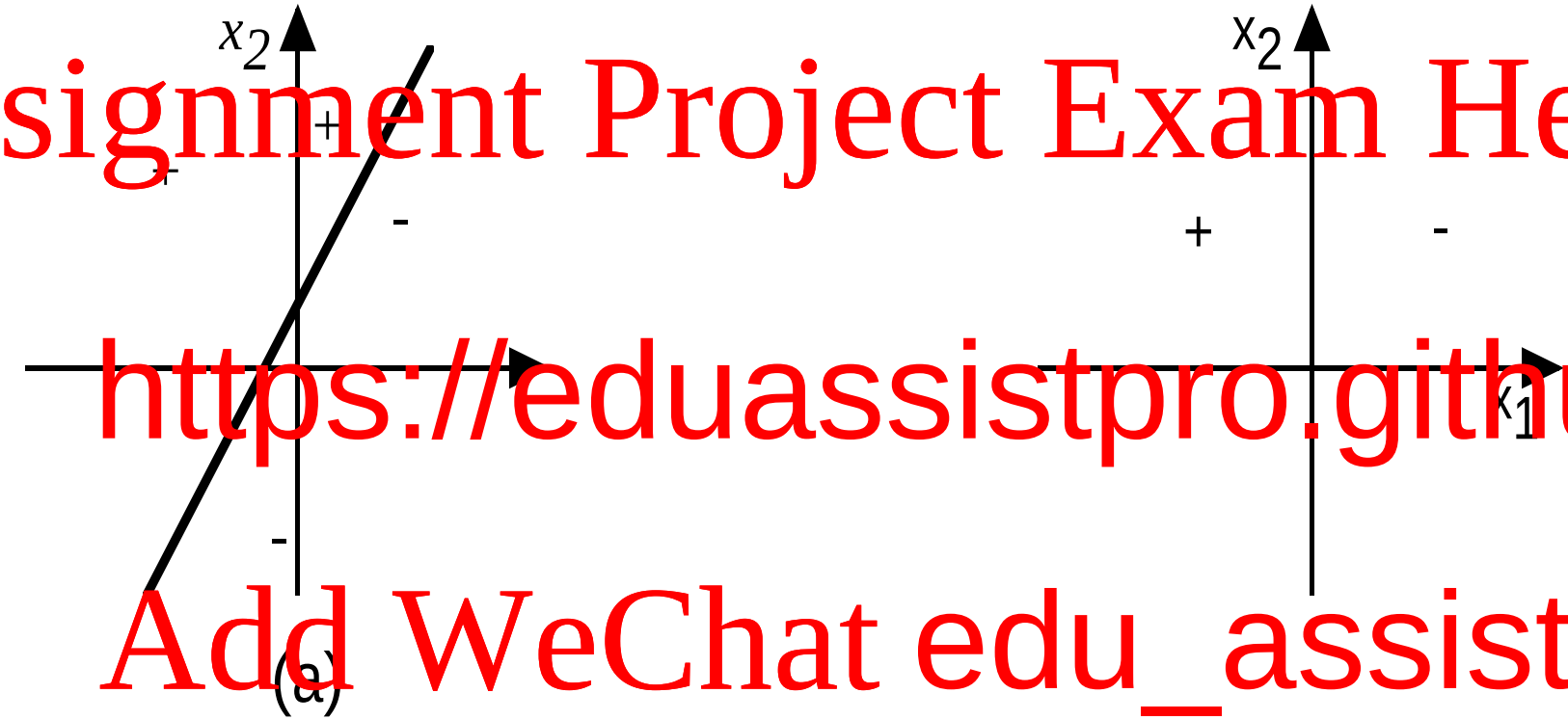
$$o(x_1, \ldots, x_n) = \begin{cases} +1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Perceptron

Or in vector notation:

$$o(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Decision Surface of a Perceptron



Represents some useful functions

- What weights represent $o(x_1, x_2) = AND(x_1, x_2)$?
- What weights represent $o(x_1, x_2) = XOR(x_1, x_2)$?

Decision Surface of a Perceptron

So some functions not representable

- e.g.
  -
- for non-linearly separable data we'll need some
  - e.g., networks of these ...
  - the start of "deep" networks ...

Perceptron learning

Key idea:

Learning is "finding a good set of weights"

Perceptr

where the weight update Δw_i depends only o les and is modulated by a "smoothing" parameter η (also known as the "learning rate").

Can prove that perceptron learning will converge:

- if training data is linearly separable
- and η sufficiently small

Perceptron learning

The perceptron iterates over the training set, updating the weight vector every time it encounters an incorrectly classified example.

- For example, let $\mathbf{x}_i$ be a misclassified positive example, then we have y_i ch that $\mathbf{w}' \cdot$ hop

- This can be achieved by calculating the new weig $\mathbf{w}' = \mathbf{w} + \eta \mathbf{x}_i$, where $0 < \eta \leq 1$ is the me set to 1). We then have $\mathbf{w}' \cdot \mathbf{x}_i = \mathbf{w} \cdot \mathbf{x}_i$ ired

- Similarly, if $\mathbf{x}_j$ is a misclassified negative example, then we have $y_j = -1$ and $\mathbf{w} \cdot \mathbf{x}_j > t$. In this case we calculate the new weight vector as $\mathbf{w}' = \mathbf{w} - \eta \mathbf{x}_j$, and thus $\mathbf{w}' \cdot \mathbf{x}_j = \mathbf{w} \cdot \mathbf{x}_j - \eta \mathbf{x}_j \cdot \mathbf{x}_j < \mathbf{w} \cdot \mathbf{x}_j$.

Perceptron learning

- The two cases can be combined in a single update rule:

$$\mathbf{w}' = \mathbf{w} + \eta y_i \mathbf{x}_i$$

- Her
whe
- Thi
ear
classification
- The algorithm just iterates over the training exa
weight update rule until all the examples are corre
- If there is a linear model that separates the positive from the negative examples, i.e., the data is linearly separable, it can be shown that the perceptron training algorithm will converge in a finite number of steps.

Perceptron training algorithm

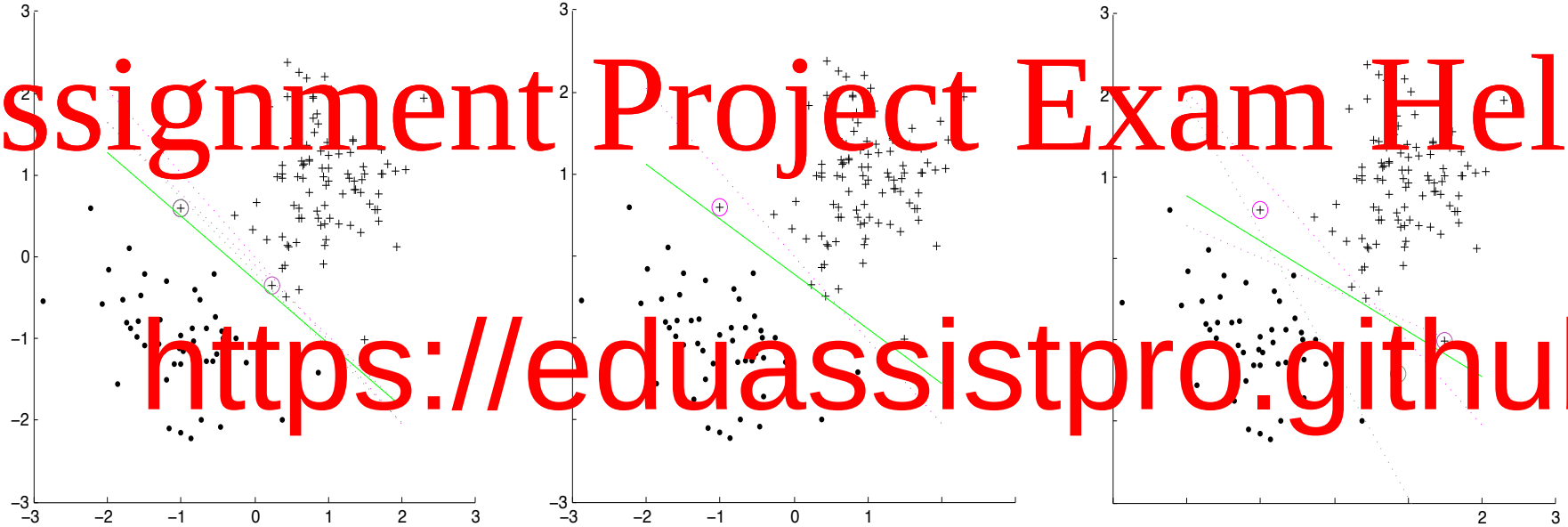
Algorithm Perceptron(D, η) // perceptron training for linear classification

Input: labelled training data D in homogeneous coordinates, learning rate η.

Output: weight vector $\mathbf{w}$ defining classifier $\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x})$.

$\mathbf{w} \leftarrow \mathbf{0}$
conver
while *c*
  con
  for $i = 1$ to $|D|$ **do**
    if $y_i \mathbf{w} \cdot \mathbf{x}_i \leq 0$ **then** // i.e., ˆ /
      $\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i$
      converged←false // We changed
    end
  end
end

Perceptron training – varying learning rate



(left) A perceptron trained with a small learning rate (circled examples are the ones that trigger the weight update. (middle) Increasing the learning rate to $\eta = 0.5$ leads in this case to a rapid convergence. (right) Increasing the learning rate further to $\eta = 1$ may lead to too aggressive weight updating, which harms convergence.
The starting point in all three cases was the basic linear classifier.

Summary

- Two major frameworks for classification by linear models

  Dist

  Pro
- We a
  form of threshold model.
- These classifiers are also, in some sense, linear mo
- So we have established the basis for learning class
- Later we will see how to extend by building on these ideas