

Pattern Recognition and Machine Learning Course:

Introduction.

Bayesian Decision Theory.

lecturer: [Jiří Matas](#), matas@cmp.felk.cvut.cz

authors: Václav Hlaváč, Jiří Matas, Ondřej Drbohlav

Czech Technical University, Faculty of Electrical Engineering
Department of Cybernetics, Center for Machine Perception
121 35 Praha 2, Karlovo nám. 13, Czech Republic

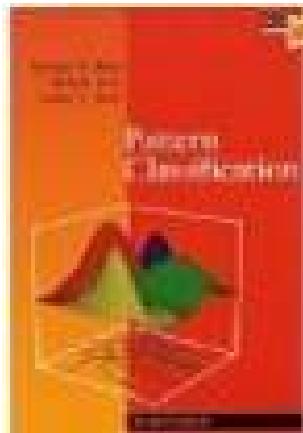
<http://cmp.felk.cvut.cz>

Version: Sep 27, 2019

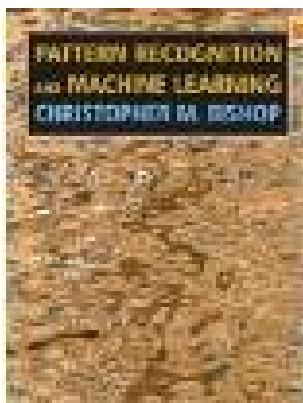
About the Pattern Recognition Course

- ◆ The selection of the topics in the course is mainstream. Besides the course material, a good wiki page is available for almost all topics covered in the course.
- ◆ We strongly recommend attendance of lectures. In PR&ML, many issues are intertwined and it is very difficult to understand the connections (e.g. understanding “why method X should be used instead of Y in case Z”) just by reading about particular methods.
- ◆ Nevertheless, we do not introduce any hard “incentives” e.g. in the form of a written exam during a lecture. But correctly answering questions on lectures merits bonus points which can make up to 10% of the semester total.
- ◆ No single textbooks is ideal for Pattern Recognition and Machine Learning course. The field is still waiting for one.

Textbooks



Duda, Hart, Stork: Pattern Classification. Classical text, 2nd edition, “easy reading”, about 5–10 available at the CMP library (G102, H. Pokorna will lend you a copy); some sections obsolete



Bishop: Pattern Recognition and Machine Learning. New, popular, but certain topics, in my opinion, could be presented in a clearer way



Schlesinger, Hlavac: Ten Lectures on Statistical and Structural Pattern Recognition. Advanced text, for those who want to know more than what is presented in the course; aims at maximum generality

Goodfellow, Bengio and Courville: Deep Learning
<http://www.deeplearningbook.org/>

English/Czech Lectures

- ◆ Those of you who are fulfilling the requirement of OI to choose one course in English should attend the lecture in English, i.e. on Monday. It is acceptable to attend the Friday lectures a few times if you miss the one on Monday (ENG lectures precede the CZ ones.)
- ◆ You may attend both lectures (a couple of students did this last year to gain better understanding.)
- ◆ If English terminology is unclear, ask. As most of the terms will be used repeatedly, language problems will disappear over time.

Pattern Recognition

The course focuses on *statistical pattern recognition*.

We start with an example called “*Dilemma of a lazy short-sighted student*” which introduces most of the basic ingredients of a statistical decision problem.

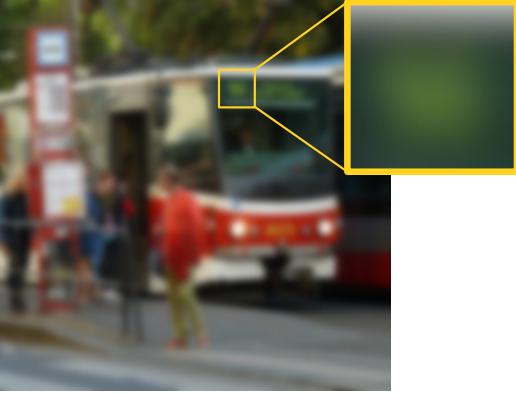
Example: A Lazy Short-Sighted Student Dilemma

A FEE student with **weak eyesight** and strong **dislike for running** is in a hurry. He needs to get to Albertov where he has arranged to play a poker game. He may get there on time, but only if he catches a tram going to Albertov immediately. He will have to pay 100 CZK fine to the poker club if he's late. As he exits the Building A at Karlovo namesti, he sees a tram at the stop:



He needs to decide: (a) *Run and catch the tram, or (b) not run and miss it?*

A Lazy Short-Sighted Student Dilemma

	new (box-style) tram, line 14	old (round-style) tram, line 22
Reality, tram number		
Student's observation		

The student cannot see the tram numbers, but can recognize the tram type (new, old). His decision can solely be based on the tram type.

A Lazy Short-Sighted Student Dilemma

The student:

- ◆ knows that trams 3, 6, 14, 22, 24 stop at Karlovo namesti
- ◆ knows that of those, trams 14 and 24 go to Albertov
- ◆ observes the tram type $x \in \{\text{old, new}\}$
- ◆ knows that the joint probability $p(x, k)$ of a tram of type $x \in \{\text{old, new}\}$ and line number $k \in \{3, 6, 14, 22, 24\}$ is

	3	6	14	22	24	$p(x)$
old	0.05	0.15	0.10	0.25	0.05	0.60
new	0.20	0.00	0.05	0.00	0.15	0.40
$p(k)$	0.25	0.15	0.15	0.25	0.20	

(1)

- ◆ **recall:** $p(a, b) = p(a|b)p(b) = p(b|a)p(a)$, where $p(a|b)$ is conditional probability of a , given b . E.g. $p(3|\text{old}) = p(3, \text{old})/p(\text{old}) = 0.05/0.6$.

A Lazy Short-Sighted Student Dilemma

(copied from the previous slide) $p(x, k)$:

	3	6	14	22	24	$p(x)$
old type	0.05	0.15	0.10	0.25	0.05	0.60
new type	0.20	0.00	0.05	0.00	0.15	0.40
$p(k)$	0.25	0.15	0.15	0.25	0.20	

(1)

From this table, we see that:

$$p(\text{Albertov}|\text{old}) = p(14|\text{old}) + p(24|\text{old}) = 0.1/0.6 + 0.05/0.6 = 0.25 \quad (2)$$

$$p(\text{Albertov}|\text{new}) = p(14|\text{new}) + p(24|\text{new}) = 0.05/0.4 + 0.15/0.4 = 0.5 \quad (3)$$

This gives us an idea of how likely it is that a spotted tram goes in the desired direction, for both old and new types of trams. But the student prefers optimal decisions if possible.

A Lazy Short-Sighted Student Dilemma

We already know that missing a poker game means the loss of 100 CZK. If he misses the game *and* runs to the wrong tram (tram not going to Albertov), he considers this as an additional loss of 50 CZK, thus 150 CZK in total.

The loss (in CZK) can be summarized in the following table:

	he decides to run	he decides not to run
Tram goes to Albertov	0	100
Tram does not go to Albertov	150	100

Assume the probability of spotted tram going to Albertov is p . If he **runs**, the **expected** (average) loss is:

$$p \cdot 0 \text{ CZK} + (1 - p) \cdot 150 \text{ CZK} = (1 - p) \cdot 150 \text{ CZK} \quad (4)$$

↑ ↑
 catches the misses the game and makes
 right tram unnecessary run

Similarly, if he **does not run**, the expected loss is:

$$p \cdot 100 \text{ CZK} + (1 - p) \cdot 100 \text{ CZK} = 100 \text{ CZK}. \quad (5)$$

A Lazy Short-Sighted Student Dilemma

For $p(\text{Albertov}|\text{old}) = 0.25$ and $p(\text{Albertov}|\text{new}) = 0.5$ we have

Average loss (in CZK):

	he decides to run	he decides not to run
old	112.5	100
new	75	100

For each observation (old, new) he selects the decision which produces lower average (expected) loss.

A Lazy Short-Sighted Student Dilemma

(copied from the previous slide:)

Average loss (in CZK):

	he decides to run	he decides not to run
old	112.5	100
new	75	100

Therefore, the best strategy (minimizing average loss) is:

observation		
decision	do not run	run

Notes (1)

Recall the table $p(x, k)$ for $x \in \{\text{old, new}\}$ and $k \in \{3, 6, 14, 22, 24\}$:

	3	6	14	22	24	$p(x)$
old	0.05	0.15	0.10	0.25	0.05	0.60
new	0.20	0.00	0.05	0.00	0.15	0.40
$p(k)$	0.25	0.15	0.15	0.25	0.20	

(6)

The notation $p(x, k)$, $p(x)$, $p(k)$ is a shorthand that can lead to ambiguities. Here, the meanings of $p(\text{old})$, or $p(3)$ are clear.

But if trams were of type 1, 2 and 3, $p(3)$ would have been ambiguous. In that case, the notation $p(x = x')$, $p(x = x', k = k')$ could be used, e.g. $p(x = \text{old}, k = 14)$. We will use a $p_{XK}(x, k)$, $p_K(k)$ notation; $p_K(3)$ is the probability $p(k = 3)$.

The probabilities $p(x)$ and $p(k)$ are called *marginal*.

In pattern recognition literature, $p(k)$ is called *prior probability*; $p(k|x)$ (probability given some observation x) is called *posterior probability*.

Notes (2)

In this example, there were only four strategies possible:

1. if you see an old tram, run, else don't run (and miss it)
2. if you see a new tram, run, else don't run
3. never run
4. always run

Question: Here, the set of observations is $X = \{\text{old type, new type}\}$ and there are two possible decisions, or actions: $\{\text{run, don't run}\}$. What is the number of strategies in the general case with D possible decisions and $|X|$ observations ?

Formulation of the Statistical PR Problem

Let us make a formal abstraction of the student dilemma. Let:

X be the set of observations. An observation (aka measurement, feature vector) $x \in X$ is what is known about an object.

K be the set of classes. A state $k \in K$ is what is not known about an object, it is unobservable (aka hidden parameter, hidden state, state-of-nature, class)

D be the set of possible *decisions* (actions).

p_{XK} : $X \times K \rightarrow \mathbb{R}$ be the joint probability that the object is in the state k and the observation x is made.

W : $K \times D \rightarrow \mathbb{R}$ be a *penalty (loss) function*, $W(k, d)$, $k \in K$, $d \in D$ is the penalty paid if the object is in a state k and the decision made is d . Defined for so-called Bayesian problems (will be dealt with soon).

q : $X \rightarrow D$ be a *decision function* (rule, strategy) assigning for each $x \in X$ the decision $q(x) \in D$. The quality of the strategy q can be measured by a number of ways, the most common being the expected (average) loss $R(q)$:

$$R(q) = \sum_{x \in X} \sum_{k \in K} p_{XK}(x, k) W(k, q(x)) . \quad (7)$$

Statistical PR Problem, Examples (1)

Often, the sets of states K and decisions D coincide.
 Such problem is then called *classification*.

Example 1: Vending machine



Classify coins according to their value. The set of measurements could be, say, weight, diameter and electrical resistance, thus $X \subset \mathbb{R}^3$. The set of classes is $K = \{1, 2, 5, 10, 20, 50\}$, and the set of decisions to make is $D \equiv K$.

Note: in many cases, the designer of the machine will soon discover the need to enlarge the set of decisions D by a “not a coin” class.

Example 2: Optical Character Recognition (OCR)

Prove this identity by considering the eigenvalue expansion of a real, symmetric matrix \mathbf{A} , and making use of the standard results for the determinant and trace of

⇒ Prove this identity by considering the eigenvalue expansion ...

Here, an observation x is an image ($x \in X \subset \mathbb{R}^{1000000}$), $K = \{\text{non-character}, \text{a-z}, \text{A-Z}, \dots\}$

Statistical PR Problem, Examples (2)

The observation x can be a number, symbol, function of one or two variables, a graph, algebraic structure, e.g.:

Application	Measurement	Decisions
license plate recognition	gray-level image	characters, numbers
fingerprint recognition	2D bitmap, gray-level image	personal identity
face recognition	2D bitmap, gray-level image	personal identity
banknote verification	different sensors	{genuine, forgery}
EEG, ECG analysis	$\mathbf{x}(t)$	diagnosis
dictation machine	$x(t)$	words, sentences
speaker identification	$x(t)$	1 of N known identities
speaker verification	$x(t)$	{yes, no}
spam filter	mail content, sender, ...	{spam, ham}
stock value prediction	stock history, economic news, ...	value
recommender systems	purchase history of many users	product recommendation(s)

Examples of Statistical PR Problems: Notes (1)

- ◆ For many examples, most of the possible observations x will never appear, for most of them no x will be observed more than once.
- ◆ For most of the listed examples, there is therefore no hope of knowing $p(x, k)$.
- ◆ For some of the examples, try to estimate the cardinality of the space of observations X .
- ◆ For some of the examples, try to estimate the cardinality of the space of all possible strategies Q .

Examples of Statistical PR Problems: Notes (2)

The given formulation is very general. As seen in the example, the cardinalities of X and D (K) range from 2 to infinite.

For many applications, the formulation captures all important aspects. Nevertheless, other important aspect were ignored, e.g.:

- ◆ The choice of X , which was assumed given. In many applications, the choice of X is left to the designer.
- ◆ The cost and time of making a measurement was ignored. With a cheap camera, observations arrive instantly and at minimum cost (of powering the camera.)
In medical applications, each measurement is costly (disposable material like vials, expensive hardware to take a scan, labor costs.)
- ◆ The time to decision, a strategy was characterized only by its loss.
- ◆ The measurements x were viewed as inputs. In many decision processes, e.g. seeing a doctor, values of initial measurements define what measurements will be made next.

Formulation of the Bayesian Decision Problem

Let the sets X , K and D , the joint probability $p_{XK}: X \times K \rightarrow \mathbb{R}$ and the penalty function $W: K \times D \rightarrow \mathbb{R}$ be given. For a strategy $q: X \rightarrow D$, the expectation of $W(k, q(x))$ is:

$$R(q) = \sum_{x \in X} \sum_{k \in K} p_{XK}(x, k) W(k, q(x)). \quad (8)$$

The quantity $R(q)$ is called the [the Bayesian risk](#). Find the strategy q^* which minimizes Bayesian risk:

$$q^* = \operatorname{argmin}_{q \in X \rightarrow D} R(q) \quad (9)$$

where the minimum is over all possible strategies. The minimizing strategy is called [*Bayesian strategy*](#).

In the following slides, the identity

$$p_{XK}(x, k) = p_{Xk}(x|k)p_K(k) \quad (10)$$

will be used. Here, a handy notation used in the Schlesinger & Hlavac book is adopted: $p_{XK}(x, k)$ is a function of two variables x and k , $p_{Xk}(x|k)$ is a function of a single variable x (k is fixed), and $p_{Xk}(x, k)$ is a single real number.

Finding the Bayesian Strategy (1)

The Bayesian risk $R(q^*)$ for the Bayesian strategy q^* is

$$R(q^*) = \min_{q \in X \rightarrow D} \sum_{x \in X} \sum_{k \in K} p_{XK}(x, k) W(k, q(x)) = \sum_{x \in X} \min_{q(x) \in D} \sum_{k \in K} p_{XK}(x, k) W(k, q(x)) \quad (11)$$

$$= \sum_{x \in X} p(x) \min_{q(x) \in D} \sum_{k \in K} p_{Kx}(k|x) W(k, q(x)) = \sum_{x \in X} p(x) \min_{d \in D} R(x, d), \quad (12)$$

where

$$R(x, d) = \sum_{k \in K} p_{Kx}(k | x) W(k, d) \quad (13)$$

is the expectation of loss conditioned on x , called *partial risk*. From this it follows that minimization of the Bayesian Risk can be done by minimizations of partial risk for each x independently. Thus, the optimal strategy $q^*(x)$ for each x can obtained as

$$q^*(x) = \operatorname{argmin}_{d \in D} \sum_{k \in K} p_{Kx}(k | x) W(k, d). \quad (14)$$

Application: Classification with 0-1 Loss Function

- The set of possible decisions D and of hidden states K coincide, $D = K$.
- The loss function assigns a **unit penalty** if $q(x) \neq k$, and no penalty otherwise, i.e.

$$W(k, q(x)) = \begin{cases} 0 & \text{if } q(x) = k \\ 1 & \text{if } q(x) \neq k \end{cases} \quad (15)$$

The partial risk for x is

$$R(x, d) = \sum_{k \in K} p_{Kx}(k | x) W(k, d) = \sum_{k \neq d} p_{Kx}(k | x) = 1 - p_{Kx}(d | x), \quad (16)$$

and the optimal strategy for this x is then

$$q^*(x) = \operatorname{argmin}_{d \in D} R(x, d) = \operatorname{argmax}_{d \in D} p_{Kx}(d | x). \quad (17)$$

Result: The Bayesian strategy for this problem is: For a given observation x , decide for the state d with the highest **a posteriori** probability $p_{Kx}(d | x)$.

Classification with 0-1 Loss Function: Example 1

Problem: Using a single measurement of sky cloudiness, decide whether it will rain. The cloudiness has four scales, ranging from 1 (no clouds) to 4 (very cloudy). There are two hidden states, 'rain' and 'no rain'. The joint probability $p(x, k)$ for $x \in \{1, 2, 3, 4\}$ and $k \in \{\text{rain}, \text{no rain}\}$ is as follows:

		cloudiness			
		1	2	3	4
rain		0.02	0.12	0.09	0.04
no rain		0.38	0.28	0.06	0.01

There is no need to actually compute $p(k|x)$; it only matters whether or not

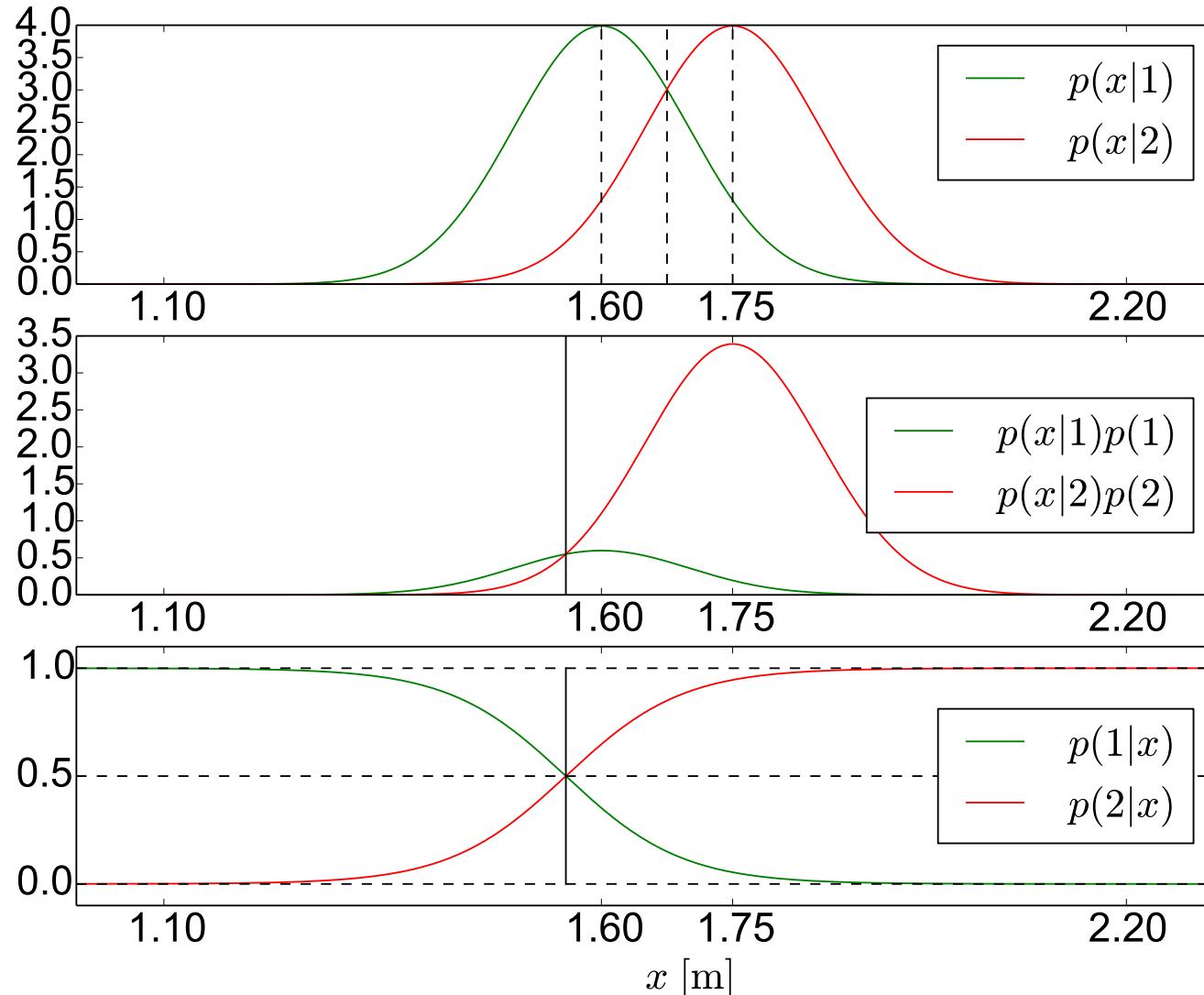
$$p(\text{rain}|x) > p(\text{no rain}|x) \Leftrightarrow p(\text{rain}, x) > p(\text{no rain}, x) \quad (18)$$

for a given observation x . Thus,

$$\begin{aligned} q^*(1) &= \text{no rain ,} \\ q^*(2) &= \text{no rain ,} \\ q^*(3) &= \text{rain ,} \\ q^*(4) &= \text{rain .} \end{aligned} \quad (19)$$

Classification with 0-1 Loss Function: Example 2

Problem: Classify gender based on body height. Observation: x = body height [m], class set $K = D = \{1, 2\}$ (1 = woman, 2 = man.) The conditionals $p(x|1)$ and $p(x|2)$ are given below and the priors (at FEL) are $p(1) = 0.15$, $p(2) = 0.85$. Let the loss function be 0-1.



Note: $p(x|1)$, $p(x|2)$ are probability *densities*. They *integrate* to 1, but their values for a given x may be arbitrary.

Result: The optimal decision strategy is to classify $x < 1.56$ m as women and $x > 1.56$ m as men.

Application: Bayesian Strategy with the Reject Option (1)

Consider an examination where for each question there are three possible answers: yes, no, not known. If your answer is correct, 1 point is added to your score. If your answer is wrong, 3 points are subtracted. If your answer is not known, your score is unchanged. What is the optimal Bayesian strategy if for each question you know the probabilities that $p(\text{yes})$ is the right answer?

Note that adding a fixed amount to all penalties and multiplying all penalties by a fixed amount does not change the optimal strategy. Adding 3 and multiplying by $1/4$ leads to 1 point for correct answer, $3/4$ for not known and 0 points of a wrong answer.

Any problem of this type can be transformed to an equivalent problem with penalty 0 for the correct answer, 1 for the wrong answer, and ϵ for not known. In realistic problems, $\epsilon \in (0, 1)$, since $\epsilon \geq 1$ means it is always better to guess than to say not known; $\epsilon \leq 0$ states that saying not known is preferred to giving the correct answer.

Let us solve the problem formally.

Application: Bayesian Strategy with Reject Option (2)

Let X and K be sets of observations and states, $p_{XK}: X \times K \rightarrow \mathbb{R}$ be a probability distribution and $D = K \cup \{\text{not known}\}$ be a set of decisions.

Let us define $W(k, d)$, $k \in K$, $d \in D$:

$$W(k, d) = \begin{cases} 0, & \text{if } d = k, \\ 1, & \text{if } d \neq k \text{ and } d \neq \text{not known}, \\ \varepsilon, & \text{if } d = \text{not known}. \end{cases}$$

Find the Bayesian strategy $q^*: X \rightarrow D$. The decision $q^*(x)$ corresponding to the observation x has to minimize the partial risk,

$$q^*(x) = \operatorname{argmin}_{d \in D} \sum_{k \in K} p_{Kx}(k | x) W(k, d).$$

Application: Bayesian Strategy with Reject Option (3)

Equivalent formulation of partial risk minimization:

$$q^*(x) = \begin{cases} \underset{d \in K}{\operatorname{argmin}} R(x, d), & \text{if } \min_{d \in K} R(x, d) < R(x, \text{not known}), \\ \text{not known}, & \text{if } \min_{d \in K} R(x, d) \geq R(x, \text{not known}). \end{cases}$$

For $\min_{d \in K} R(x, d)$, there holds (as before for the 0-1 loss function case):

$$\min_{d \in K} R(x, d) = \min_{d \in K} \sum_{k \in K} p_{Kx}(k | x) W(k, d) \quad (20)$$

$$= \min_{d \in K} \sum_{k \in K \setminus \{d\}} p_{Kx}(k | x) \quad (21)$$

$$= \min_{d \in K} \left(\sum_{k \in K} p_{Kx}(k | x) - p_{Kx}(d | x) \right) \quad (22)$$

$$= \min_{d \in K} (1 - p_{Kx}(d | x)) = 1 - \max_{d \in K} p_{Kx}(d | x). \quad (23)$$

Application: Bayesian Strategy with Reject Option (4)

For $R(x, \text{not known})$, there holds

$$\begin{aligned}
 R(x, \text{not known}) &= \sum_{k \in K} p_{K|X}(k | x) W(k, \text{not known}) \\
 &= \sum_{k \in K} p_{K|X}(k | x) \varepsilon = \varepsilon.
 \end{aligned} \tag{24}$$

The decision rule becomes

$$q^*(x) = \begin{cases} \underset{k \in K}{\operatorname{argmax}} p_{K|X}(k | x), & \text{if } 1 - \max_{k \in K} p_{K|X}(k | x) < \varepsilon, \\ \text{not known,} & \text{if } 1 - \max_{k \in K} p_{K|X}(k | x) \geq \varepsilon. \end{cases}$$

Application: Bayesian Strategy with Reject Option (5)

Strategy $q^*(x)$ can thus be described as follows:

First, find the state k which has the largest *a posteriori* probability. Let this probability be denoted by $p_{\max}(x)$.

If $1 - p_{\max}(x) < \varepsilon$ then the optimal decision is k .

If $1 - p_{\max}(x) \geq \varepsilon$ then the optimal decision is not known .

Case of 2 Classes. Likelihood Ratio

- ◆ Let the number of classes be two; $K = \{1, 2\}$.
- ◆ Only conditional probabilities $p_{X|1}(x)$ and $p_{X|2}(x)$ are known.
- ◆ The *a priori* probabilities $p_K(1)$ and $p_K(2)$ and penalties $W(k, d)$,
 $k \in \{1, 2\}$, $d \in D$, are not known.
- ◆ In this situation the Bayesian strategy cannot be created.

Likelihood Ratio (1)

If the *a priori* probabilities $p_K(k)$ and the penalty $W(k, d)$ are known then the decision $q^*(x)$ about the observation x is

$$\begin{aligned}
 q^*(x) &= \operatorname{argmin}_d (p_{XK}(x, 1) W(1, d) + p_{XK}(x, 2) W(2, d)) \\
 &= \operatorname{argmin}_d (p_{X|1}(x) p_K(1) W(1, d) + p_{X|2}(x) p_K(2) W(2, d)) \\
 &= \operatorname{argmin}_d \left(\frac{p_{X|1}(x)}{p_{X|2}(x)} p_K(1) W(1, d) + p_K(2) W(2, d) \right) \\
 &= \operatorname{argmin}_d (\gamma(x) c_1(d) + c_2(d)) .
 \end{aligned}$$

$\gamma(x)$ – likelihood ratio.

Likelihood Ratio (2) – linearity, convex subset of \mathbb{R}

The subset of observations $X(d^*)$ for which the decision d^* should be made is the solution of the system of inequalities

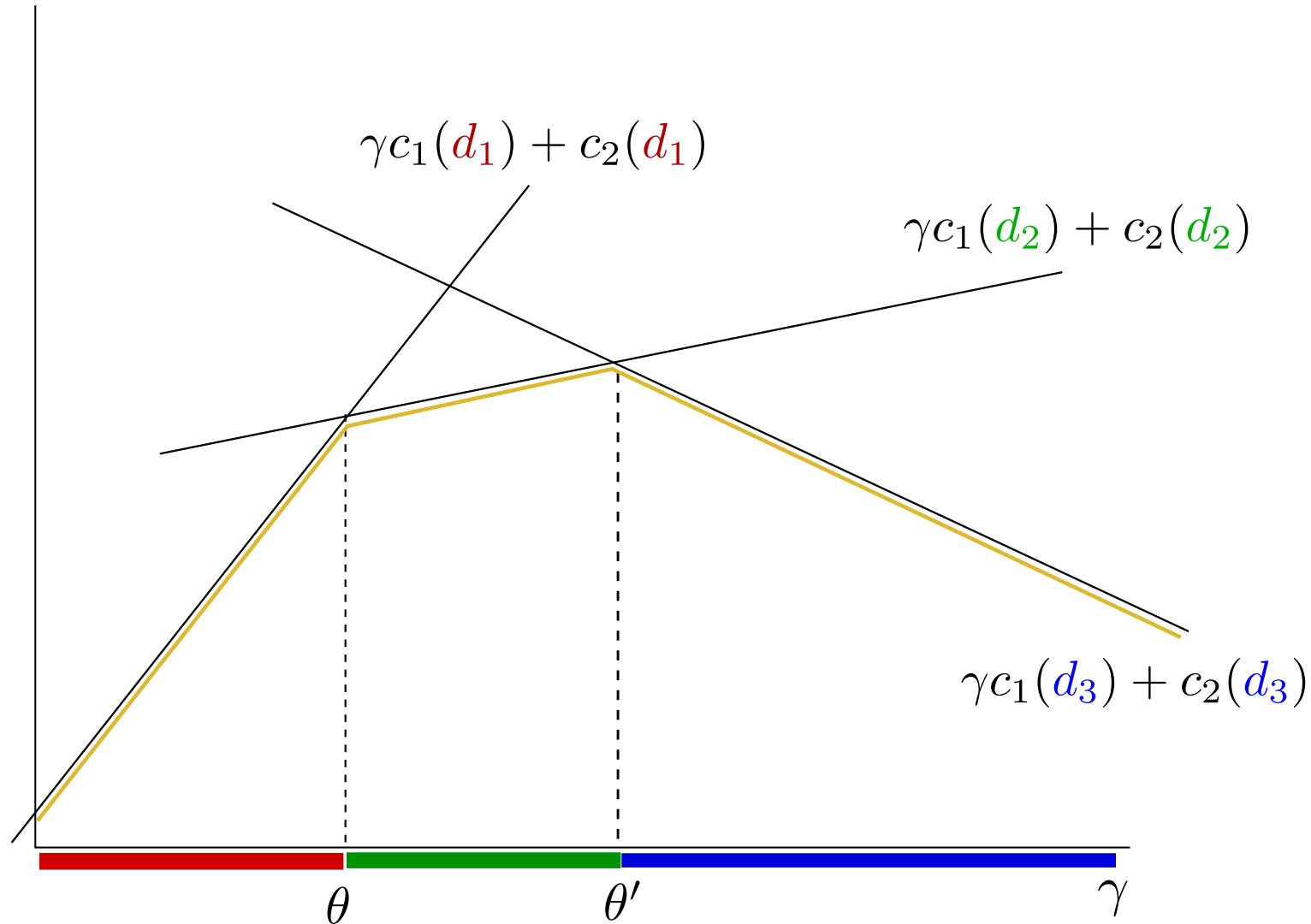
$$\gamma(x) c_1(d^*) + c_2(d^*) \leq \gamma(x) c_1(d) + c_2(d), \quad d \in D \setminus \{d^*\}.$$

- ◆ The system is **linear** with respect to the likelihood ratio $\gamma(x)$.
- ◆ The subset $X(d^*)$ corresponds to a **convex subset** of the values of the likelihood ratio $\gamma(x)$.
- ◆ As $\gamma(x)$ are real numbers, their **convex subsets correspond to the numerical intervals**.

Likelihood Ratio (3) – linearity, convex subset of \mathbb{R}

Example, 2 classes, 3 possible decisions

$$\gamma : X \rightarrow \mathbb{R}$$



Likelihood Ratio (4)

Any Bayesian strategy divides the real axis from 0 to ∞ into $|D|$ intervals $I(d)$, $d \in D$. The decision d is made for observation $x \in X$ when the likelihood ratio $\gamma = p_{X|1}(x)/p_{X|2}(x)$ belongs to the interval $I(d)$.

More particular case which is commonly known:

Two decisions only, $D = \{1, 2\}$. Bayesian strategy is characterised by a single threshold value θ . For an observation x the decision depends only on whether the likelihood ratio is larger or smaller than θ .

Example. 2 Classes, 3 Decisions

Object: a patient examined by the physician.

Observations X : some measurable parameters (temperature, . . .).

2 unobservable states $K = \{\text{healthy, sick}\}$.

3 decisions $D = \{\text{do not cure, weak medicine, strong medicine}\}$.

Penalty function $W: K \times D \rightarrow \mathbb{R}$

$W(k, d)$	do not cure	weak medicine	strong medicine
sick	10	2	0
healthy	0	5	10

Comments on the Bayesian Decision Problem.

Bayesian recognition is decision-making, where

- ◆ Decisions do not influence the state of nature (c.f. Game T., Control T.).
- ◆ A single decisions is made, issues of time are ignored in the model (unlike in Control Theory where decisions are typically taken continuously and in real-time)
- ◆ Cost of obtaining measurements is not modelled (unlike in Sequential Decision Theory).

The hidden parameter k (class information) is considered not observable.

Common situations are:

- ◆ k could be observed, but at a high cost.
- ◆ k is a future state (e.g. of petrol price) and will be observed later.

It is interesting to ponder whether a state can ever be genuinely unobservable.

Generality of the Bayesian task formulation.

Two general properties of Bayesian strategies:

- ◆ Each Bayesian strategy corresponds to separation of the space of probabilities into convex subsets.
- ◆ Deterministic strategies are always better than randomized ones.

Bayesian Strategies are Deterministic

Instead of $q: X \rightarrow D$ consider stochastic strategy (probability distributions) $q_r(d|x)$.

THEOREM

Let X, K, D be finite sets, $p_{XK}: X \times K \rightarrow \mathbb{R}$ be a probability distribution, $W: K \times D \rightarrow \mathbb{R}$ be a penalty function. Let $q_r: D \times X \rightarrow \mathbb{R}$ be a stochastic strategy, i.e a strategy that selects decisions d with probability $q_r(d|x)$. The risk of the stochastic strategy is:

$$R_{\text{rand}} = \sum_{x \in X} \sum_{k \in K} p_{XK}(x, k) \sum_{d \in D} q_r(d|x) W(k, d).$$

In such a case there exists the deterministic strategy $q: X \rightarrow D$ with the risk

$$R_{\text{det}} = \sum_{x \in X} \sum_{k \in K} p_{XK}(x, k) W(k, q(x))$$

which is not greater than R_{rand} .

Note that $q_r(d|x)$ has the following properties for all x : (i) $\sum_{d \in D} q_r(d|x) = 1$ and (ii) $q_r(d|x) \geq 0$, $d \in D$.

PROOF #1 (Bayesian strategy are deterministic)

Comparing the risks associated with deterministic and stochastic strategies

$$R_{\text{rand}} = \sum_{x \in X} \sum_{k \in K} p_{XK}(x, k) \sum_{d \in D} q_r(d | x) W(k, d), \quad R_{\text{det}} = \sum_{x \in X} \sum_{k \in K} p_{XK}(x, k) W(k, q(x))$$

it is clear it is sufficient to prove that for every x

$$\sum_{k \in K} p_{XK}(x, k) \sum_{d \in D} q_r(d | x) W(k, d) \geq \sum_{k \in K} p_{XK}(x, k) W(k, q(x))$$

Let us denote the losses associated with deterministic decision d as

$\alpha_d = \sum_{k \in K} p_{XK}(x, k) W(k, d)$ and let the loss of the best deterministic strategy be denoted $\alpha_{d^*} = \min_{d \in D} \alpha_d$. Expressing the stochastic loss in terms of α_d we obtain:

$$\sum_{k \in K} p_{XK}(x, k) \sum_{d \in D} q_r(d | x) W(k, d) = \sum_{d \in D} q_r(d | x) \sum_{k \in K} p_{XK}(x, k) W(k, d) = \sum_{d \in D} q_r(d | x) \alpha_d$$

To prove the theorem, it is sufficient to show that $\sum_{d \in D} q_r(d | x) \alpha_d \geq \alpha_{d^*}$:

$$\forall d \in D : \alpha_d \geq \alpha_{d^*} \Rightarrow \sum_{d \in D} q_r(d | x) \alpha_d \geq \sum_{d \in D} q_r(d | x) \alpha_{d^*} = \alpha_{d^*} \sum_{d \in D} q_r(d | x) = \alpha_{d^*} \quad \square$$

PROOF #2 (Bayesian strategy are deterministic)

$$R_{\text{rand}} = \sum_{x \in X} \sum_{d \in D} q_r(d | x) \sum_{k \in K} p_{XK}(x, k) W(k, d).$$

$$\sum_{d \in D} q_r(d | x) = 1, \quad x \in X, \quad q_r(d | x) \geq 0, \quad d \in D, \quad x \in X.$$

$$R_{\text{rand}} \geq \sum_{x \in X} \min_{d \in D} \sum_{k \in K} p_{XK}(x, k) W(k, d) \quad \text{holds for all } x \in X, d \in D. \quad (25)$$

Let us denote by $q(x)$ any value d that satisfies the equality

$$\sum_{k \in K} p_{XK}(x, k) W(k, q(x)) = \min_{d \in D} \sum_{k \in K} p_{XK}(x, k) W(k, d). \quad (26)$$

The function $q: X \rightarrow D$ defined in such a way is a deterministic strategy which is not worse than the stochastic strategy q_r . In fact, when we substitute Equation (26) into the inequality (25) then we obtain the inequality

$$R_{\text{rand}} \geq \sum_{x \in X} \sum_{k \in K} p_{XK}(x, k) W(k, q(x)).$$

The risk of the deterministic strategy q can be found on the right-hand side of the preceding inequality. It can be seen that $R_{\text{det}} \leq R_{\text{rand}}$ holds.

What's next? (1)

- ◆ The first part of the course is about solving statistical pattern recognition problems when the model $p_{XK}(x, k)$ is known.
- ◆ It is very rare that $p_{XK}(x, k)$ is known for a given application. Instead, it is almost always possible to obtain a set of representative samples T of (measurement, class) pairs.
Example: Gender recognition. A person labels 1000 face images as male/female.
- ◆ One way to proceed is to find and estimate $p_{XK}(x, k)$ from T and proceed as if the estimate was equal to the true probability. A much more common approach is to obtain a strategy q (= a classifier) with desirable properties directly from T .

What's next? (2)

The next lecture will deal with problems illustrated by a modified version of the **Student Dilemma**:

A student with a weak eyesight and a strong dislike for running is in a hurry. He needs to get to Albertov, where his girlfriend, a medical student is expecting him in 10 minutes. He might get there on time, but he needs to catch a tram immediately.

As he exits Building A at Karlovo namesti, he sees a tram at the stop. He cannot see the tram number as he is short-sighted, but he recognizes the tram is the rectangular shaped “new style” one, not the rounded “old style”.

He knows, as before, the sets X , K , and the joint probability $p_{XK}(x, k)$ for all $x \in X, k \in K$.

He knows that his girlfriend tolerates him being late 20% of the time, and does not even comment. But she'd dump him if gets above that.

When should he run?

Interestingly, in this case, the student need not assign a cost to running or to the loosing his girlfriend (which might be rather difficult).

He needs a strategy that will tell him to run as rarely as possible, given the constraint: he must catch the tram 80% of time else he loses his girlfriend.

Non-Bayesian Methods

lecturer: Jiří Matas, matas@cmp.felk.cvut.cz

authors: Václav Hlaváč, Jiří Matas, Boris Flach, Ondřej Drbohlav

Czech Technical University, Faculty of Electrical Engineering
Department of Cybernetics, Center for Machine Perception
121 35 Praha 2, Karlovo nám. 13, Czech Republic

<http://cmp.felk.cvut.cz>

10/Oct/2016

Last update: 15/Oct/2016

Lecture Outline

1. Limitations of Bayesian Decision Theory
2. Neyman Pearson Task
3. Minimax Task
4. Wald Task

Bayesian Decision Theory

Recall:

X set of observations

K set of hidden states

D set of decisions

p_{XK} : $X \times K \rightarrow \mathbb{R}$: joint probability

W : $K \times D \rightarrow \mathbb{R}$: *loss function*,

q : $X \rightarrow D$: strategy

$R(q)$: risk:

$$R(q) = \sum_{x \in X} \sum_{k \in K} p_{XK}(x, k) W(k, q(x)) \quad (1)$$

Bayesian strategy q^* :

$$q^* = \operatorname{argmin}_{q \in X \rightarrow D} R(q) \quad (2)$$

Limitations of the Bayesian Decision Theory

The limitations follow from the very ingredients of the Bayesian Decision Theory — the necessity to know all the probabilities and the loss function.

- ◆ The loss function W must make sense, but in many tasks it wouldn't
 - medical diagnosis task (W : price of medicines, staff labor, etc. but what penalty in case of patient's death?) Uncomparable penalties on different axes of X .
 - nuclear plant
 - judicial error
- ◆ The prior probabilities $p_K(k)$: must exist and be known. But in some cases it does not make sense to talk about probabilities because the events are not random.
 - $K = \{1, 2\} \equiv \{\text{own army plane, enemy plane}\}$;
 $p(x|1), p(x|2)$ do exist and can be estimated, but $p(1)$ and $p(2)$ don't.
- ◆ The conditionals may be subject to non-random intervention; $p(x | k, z)$ where $z \in Z = \{1, 2, 3\}$ are different interventions.
 - a system for handwriting recognition: The training set has been prepared by 3 different persons. But the test set has been constructed by one of the 3 persons only. This **cannot** be done:

$$(!) \quad p(x | k) = \sum_z p(z)p(x | k, z) \tag{3}$$

Neyman Pearson Task

- ◆ $K = \{D, N\}$ (dangerous state, normal state)
- ◆ X set of observations
- ◆ Conditionals $p(x | D)$, $p(x | N)$ are given
- ◆ The priors $p(D)$ and $p(N)$ are unknown or do not exist
- ◆ $q: X \rightarrow K$ strategy

The Neyman Person Task looks for the optimal strategy q^* for which

- i) the error of classification of the dangerous state is lower than a predefined threshold $\bar{\epsilon}_D$ ($0 < \bar{\epsilon}_D < 1$), while
- ii) the classification error for the normal state is as low as possible.

This is formulated as an optimization task with an inequality constraint:

$$q^* = \operatorname{argmin}_{q: X \rightarrow K} \sum_{x: q(x) \neq N} p(x | N) \quad (4)$$

subject to: $\sum_{x: q(x) \neq D} p(x | D) \leq \bar{\epsilon}_D .$ (5)

Neyman Pearson Task

(copied from the previous slide:)

$$q^* = \operatorname{argmin}_{q:X \rightarrow K} \sum_{x:q(x) \neq N} p(x | N) \quad (4)$$

subject to: $\sum_{x:q(x) \neq D} p(x | D) \leq \bar{\epsilon}_D .$ (5)

A strategy is characterized by the classification error values ϵ_N and ϵ_D :

$$\epsilon_N = \sum_{x:q(x) \neq N} p(x | N) \quad (\text{false alarm}) \quad (6)$$

$$\epsilon_D = \sum_{x:q(x) \neq D} p(x | D) \quad (\text{overlooked danger}) \quad (7)$$

Example: Male/Female Recognition (Neyman Pearson) (1)

An aging student at CTU wants to marry. He can't afford to miss recognizing a girl when he meets her, therefore he sets the threshold on female classification error to $\bar{\epsilon}_D = 0.2$. At the same time, he wants to minimize mis-classifying boys for girls.

- ◆ $K = \{D, N\} \equiv \{F, M\}$ (female, male)
- ◆ measurements $X = \{\text{short, normal, tall}\} \times \{\text{ultralight, light, avg, heavy}\}$
- ◆ Prior probabilities do not exist.
- ◆ Conditionals are given as follows:

		$p(x F)$			
		short	normal	tall	
short		.197	.145	.094	.017
normal		.077	.299	.145	.017
tall		.001	.008	.000	.000
		u-light	light	avg	heavy

		$p(x M)$			
		short	normal	tall	
short		.011	.005	.011	.011
normal		.005	.071	.408	.038
tall		.002	.014	.255	.169
		u-light	light	avg	heavy

(8)

Neyman Pearson : Solution

The optimal strategy q^* for a given $x \in X$ is constructed using the likelihood ratio $\frac{p(x|N)}{p(x|D)}$.

Let there be a constant $\mu \geq 0$. Given this μ , a strategy q is constructed as follows:

$$\frac{p(x|N)}{p(x|D)} > \mu \quad \Rightarrow \quad q(x) = N, \tag{9}$$

$$\frac{p(x|N)}{p(x|D)} \leq \mu \quad \Rightarrow \quad q(x) = D. \tag{10}$$

The optimal strategy q^* is obtained by selecting the minimal μ for which there still holds that $\epsilon_D \leq \bar{\epsilon}_D$.

Let us show this on an example.

Example: Male/Female Recognition (Neyman Pearson) (2)

$p(x|F)$

short	.197	.145	.094	.017
normal	.077	.299	.145	.017
tall	.001	.008	.000	.000
	u-light	light	avg	heavy

$p(x|M)$

short	.011	.005	.011	.011
normal	.005	.071	.408	.038
tall	.002	.014	.255	.169
	u-light	light	avg	heavy

$$r(x) = p(x|M)/p(x|F)$$

short	0.056	0.034	0.117	0.647
normal	0.065	0.237	2.814	2.235
tall	2.000	1.750	∞	∞
	u-light	light	avg	heavy

$$\text{rank order of } p(x|M)/p(x|F)$$

short	2	1	4	6
normal	3	5	10	9
tall	8	7	11	12
	u-light	light	avg	heavy

Here, different μ 's can produce 11 different strategies.

First, let us take $2.814 < \mu < \infty$, e.g. $\mu = 3$. This produces a strategy $q^*(x) = F$ everywhere except where $p(x|F) = 0$. Obviously, classification error ϵ_F for F is $\epsilon_F = 0$, and $\epsilon_M = 1 - .255 - .169 = .576$.

Example: Male/Female Recognition (Neyman Pearson) (3)

 $p(x|M)$

short	.197	.145	.094	.017
normal	.077	.299	.145	.017
tall	.001	.008	.000	.000
	u-light	light	avg	heavy

 $p(x|F)$

short	.011	.005	.011	.011
normal	.005	.071	.408	.038
tall	.002	.014	.255	.169
	u-light	light	avg	heavy

$$r(x) = p(x|M)/p(x|F)$$

short	0.056	0.034	0.117	0.647
normal	0.065	0.237	2.814	2.235
tall	2.000	1.750	∞	∞
	u-light	light	avg	heavy

rank, and $q^*(x) = \{M, F\}$ for $\mu = 2.5$

short	2	1	4	6
normal	3	5	10	9
tall	8	7	11	12
	u-light	light	avg	heavy

Next, take μ which satisfies

$$r_9 < \mu < r_{10} \quad (\text{e.g. } \mu = 2.5) \tag{11}$$

(where r_i is the likelihood ratios indexed by its rank.)

Here, $\epsilon_F = .145$, and $\epsilon_M = 1 - .255 - .169 - .408 = .168$.

Example: Male/Female Recognition (Neyman Pearson) (4)

 $p(x|M)$

	short	.197	.145	.094	.017
normal	.077	.299	.145	.017	
tall	.001	.008	.000	.000	
	u-light	light	avg	heavy	

 $p(x|F)$

	short	.011	.005	.011	.011
normal	.005	.071	.408	.038	
tall	.002	.014	.255	.169	
	u-light	light	avg	heavy	

$$r(x) = p(x|M)/p(x|F)$$

	short	0.056	0.034	0.117	0.647
normal	0.065	0.237	2.814	2.235	
tall	2.000	1.750	∞	∞	
	u-light	light	avg	heavy	

rank, and $q^*(x) = \{F, M\}$ for $\mu = 2.1$

	short	2	1	4	6
normal	3	5	10	9	
tall	8	7	11	12	
	u-light	light	avg	heavy	

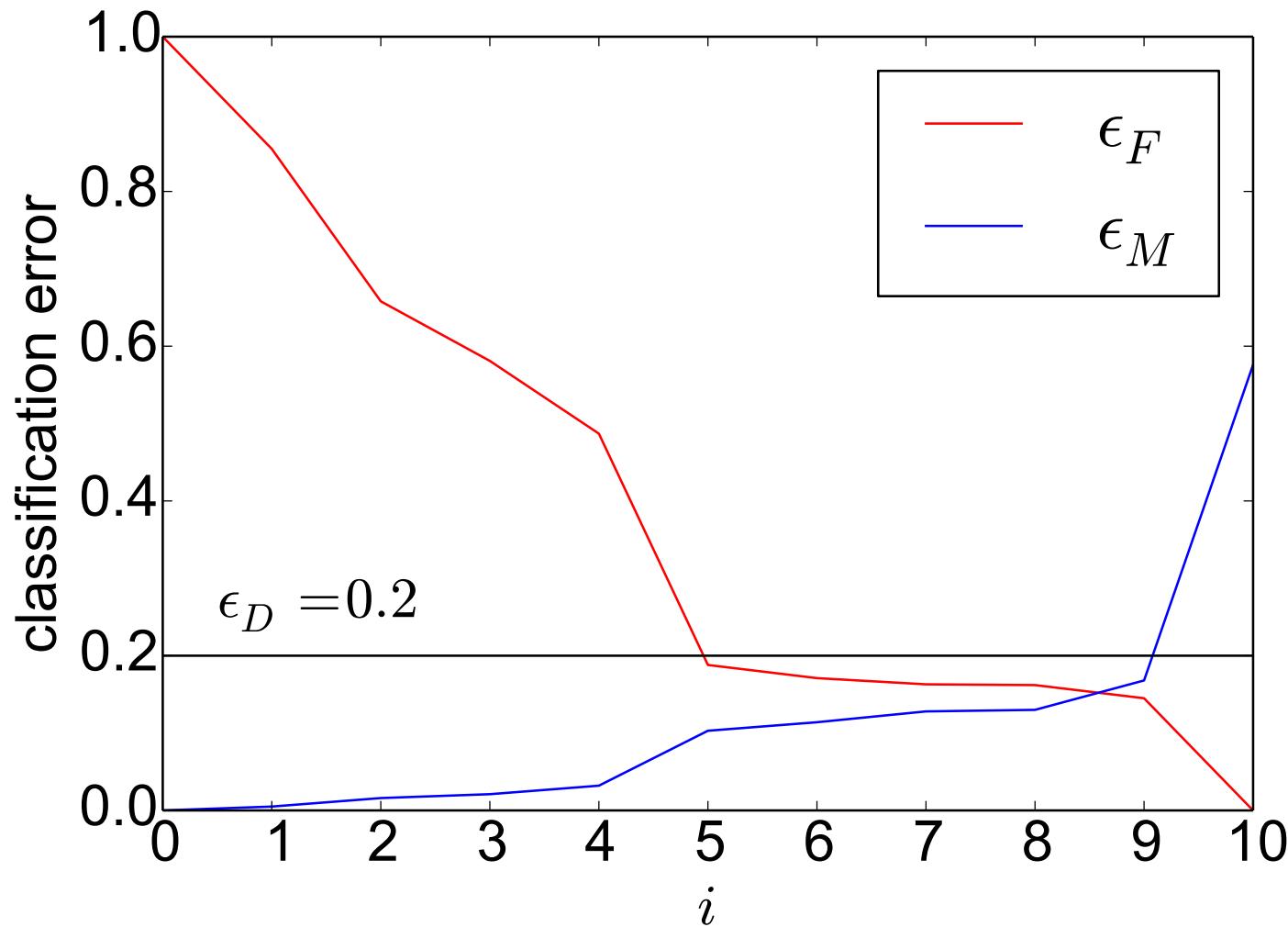
Do the same for μ satisfying

$$r_8 < \mu < r_9 \quad (\text{e.g. } \mu = 2.1) \quad (12)$$

$\Rightarrow \epsilon_F = .162$, and $\epsilon_M = 0.13$.

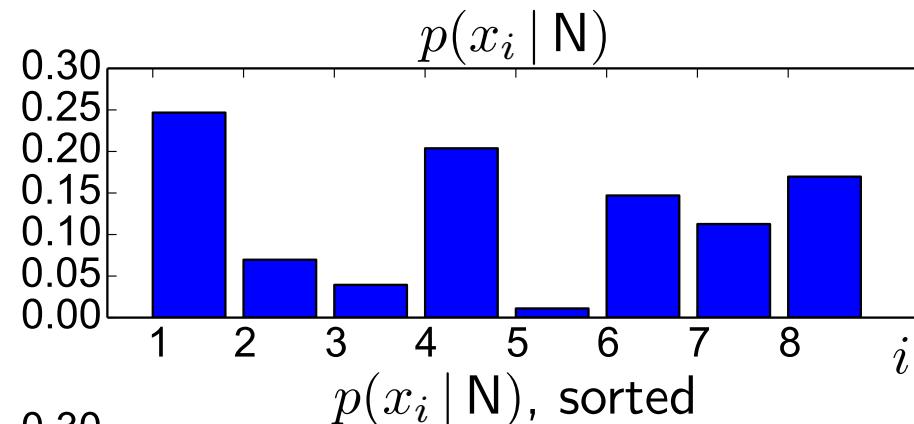
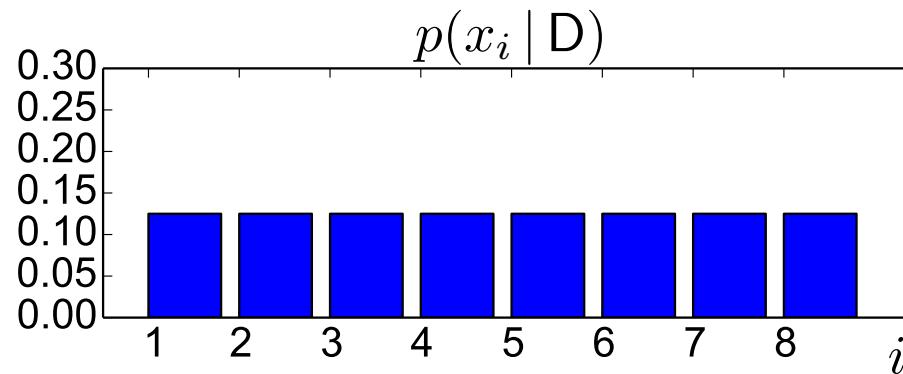
Example: Male/Female Recognition (Neyman Pearson) (5)

Classification errors for F and M, for $\mu_i = \frac{r_i+r_{i+1}}{2}$ and $\mu_0 = 0$.



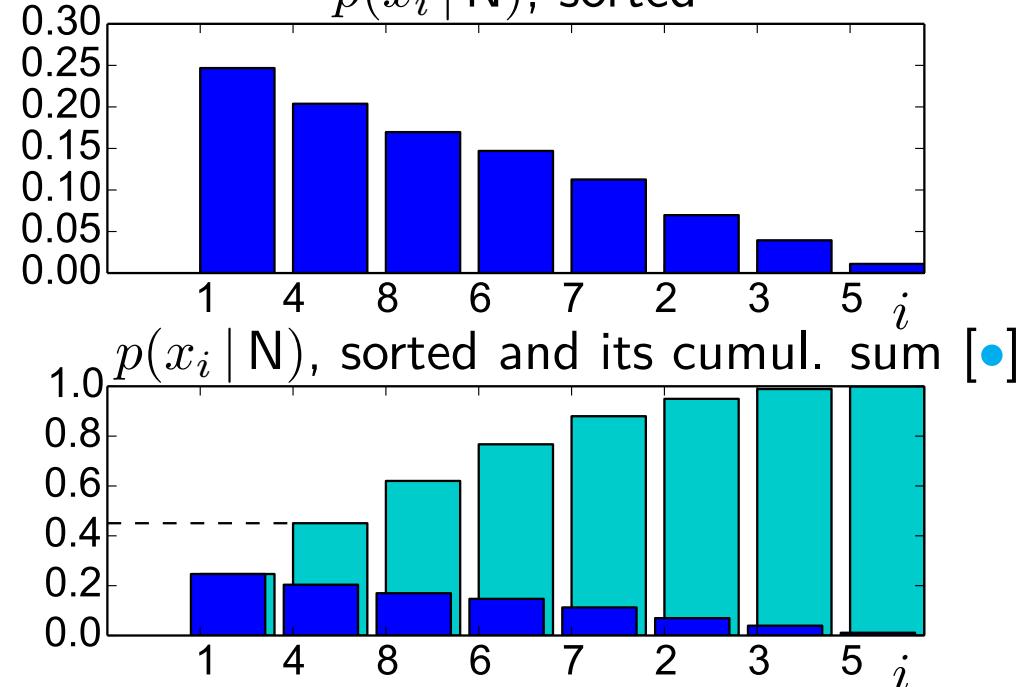
The optimum is reached for $r_5 < \mu < r_6$; $\epsilon_F = .188$, $\epsilon_M = .103$

Neyman Pearson : Solution (1, special case)



Consider first a special case when $p(x_i | D) = \text{const} = \frac{1}{8}$.
 Possible values for ϵ_D are $0, \frac{1}{8}, \frac{2}{8}, \dots, 1$.
 If a strategy q classifies P observations as normal then $\epsilon_D = \frac{P}{8}$.

Let $P = 1$ and thus $\epsilon_D = \frac{1}{8}$. It is clear that ϵ_N will attain minimum if the (one) observation which is classified as normal is the one with the highest $p(x_i | N)$. Similarly, if $P = 2$ then the two observations to be classified as normal are the one with the first two highest $p(x_i | N)$. Etc.



↑ cumulative sum of sorted $p(x_i | N)$ shows the classification success rate for N, that is, $1 - \epsilon_N$, for $\epsilon_D = \frac{1}{8}, \frac{2}{8}, \dots, 1$. For example, for $\epsilon_D = \frac{2}{8}$ ($P = 2$), $\epsilon_N = 1 - 0.45 = 0.55$ (as shown, dashed.)

Neyman Pearson : Solution (2, general case)

In general, $p(x_i | D) \neq \text{const.}$ Consider the following example:

$p(x_i D)$		
x_1	x_2	x_3
0.5	0.25	0.25

$p(x_i N)$		
x_1	x_2	x_3
0.6	0.35	0.05

But this can easily be converted to the previous special case by (only formally) splitting x_1 to two observations x'_1 and x''_1 :

$p(x_i D)$			
x'_1	x''_1	x_2	x_3
0.25	0.25	0.25	0.25

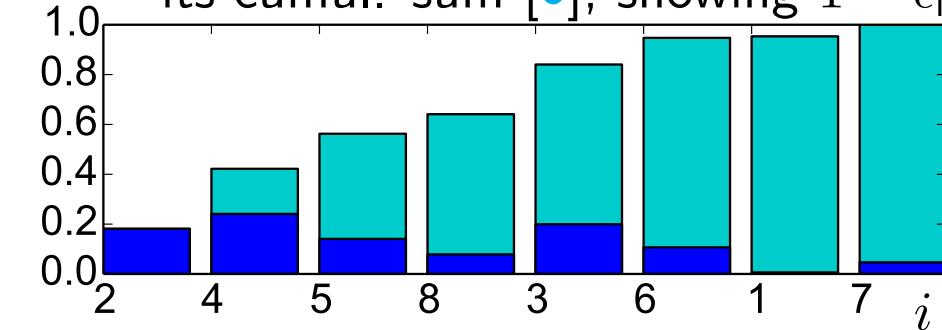
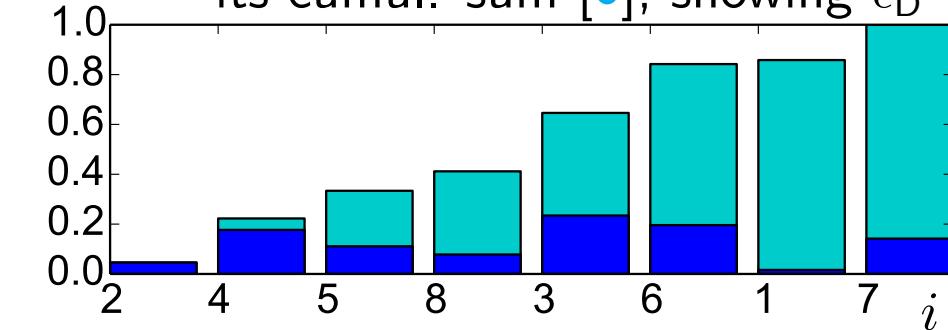
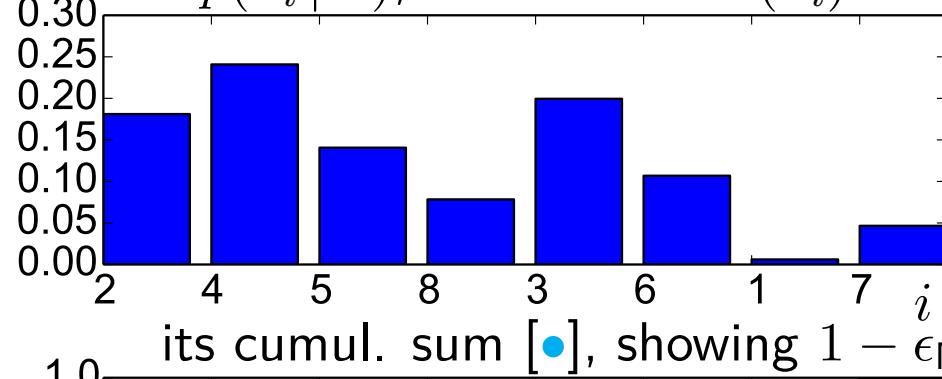
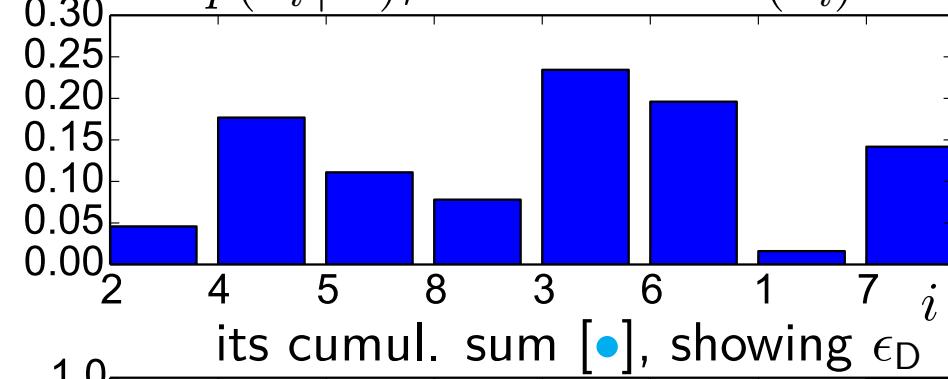
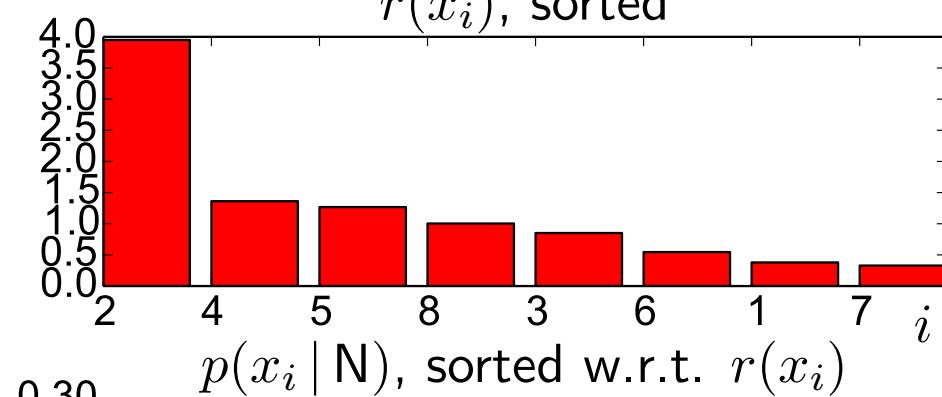
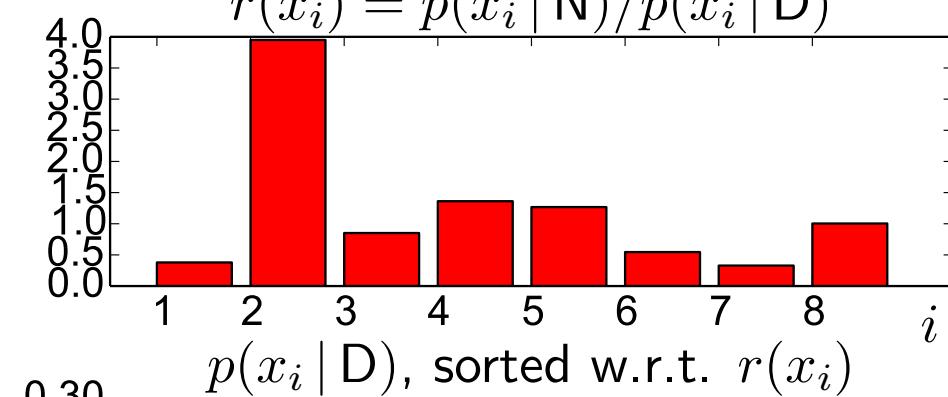
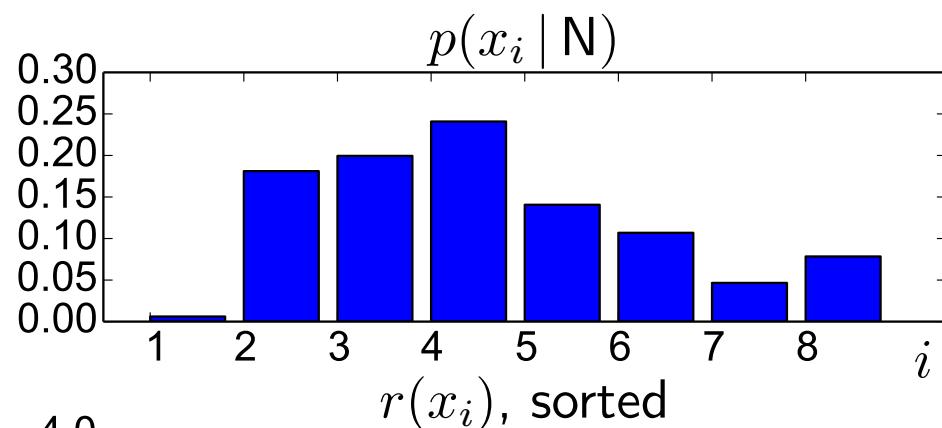
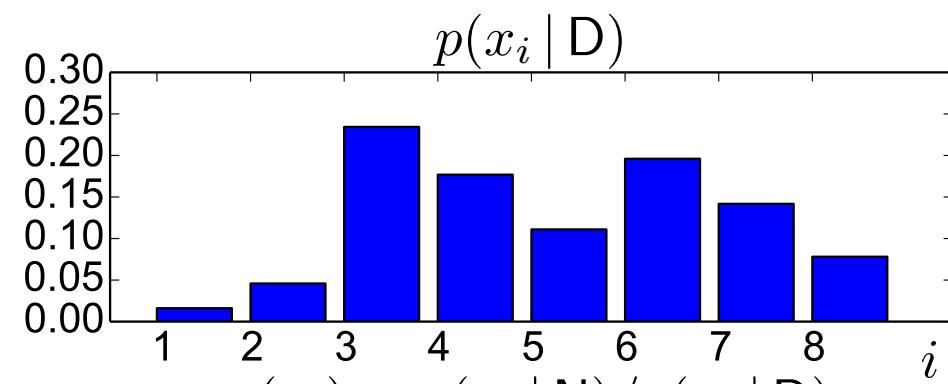
$p(x_i N)$			
x'_1	x''_1	x_2	x_3
0.3	0.3	0.35	0.05

which would result in ordering the observations by decreasing $p(x_i | N)$ as: x_2, x_1, x_3 .

Obviously, the same ordering is obtained when $p(x_i | N)$ is 'normalized' by $p(x_i | D)$, that is, using the likelihood ratio

$$r(x_i) = \frac{p(x_i | N)}{p(x_i | D)}. \quad (13)$$

Neyman Pearson : Solution (3, general case, example)



Neyman Pearson Solution : Illustration of Principle

Lagrangian of the Neyman Pearson Task is

$$L(q) = \underbrace{\sum_{x: q(x)=D} p(x | N)}_{=} + \mu \left(\sum_{x: q(x)=N} p(x | D) - \bar{\epsilon}_D \right) \quad (14)$$

$$= 1 - \overbrace{\sum_{x: q(x)=N} p(x | N)}^= + \mu \left(\sum_{x: q(x)=N} p(x | D) \right) - \mu \bar{\epsilon}_D \quad (15)$$

$$= 1 - \mu \bar{\epsilon}_D + \sum_{x: q(x)=N} \underbrace{\{\mu p(x | D) - p(x | N)\}}_{T(x)} \quad (16)$$

If $T(x)$ is negative for an x then it will decrease the objective function and the optimal strategy q^* will decide $q^*(x) = N$. This illustrates why the solution to the Neyman Pearson Task has the form

$$\frac{p(x | N)}{p(x | D)} > \mu \Rightarrow q(x) = N, \quad (9)$$

$$\frac{p(x | N)}{p(x | D)} \leq \mu \Rightarrow q(x) = D. \quad (10)$$

Neyman Pearson : Derivation (1)

$$q^* = \min_{q: X \rightarrow K} \sum_{x: q(x) \neq N} p(x | N) \quad \text{subject to: } \sum_{x: q(x) \neq D} p(x | D) \leq \bar{\epsilon}_D . \quad (17)$$

Let us rewrite this as

$$q^* = \min_{q: X \rightarrow K} \sum_{x \in X} \alpha(x)p(x | N) \quad \text{subject to: } \sum_{x \in X} [1 - \alpha(x)]p(x | D) \leq \bar{\epsilon}_D . \quad (18)$$

$$\text{and: } \alpha(x) \in \{0, 1\} \quad \forall x \in X \quad (19)$$

This is a combinatorial optimization problem. If the relaxation is done from $\alpha(x) \in \{0, 1\}$ to $0 \leq \alpha(x) \leq 1$, this can be solved by **linear programming** (LP). The Lagrangian of this problem with inequality constraints is:

$$L(\alpha(x_1), \alpha(x_2), \dots, \alpha(x_N)) = \sum_{x \in X} \alpha(x)p(x | N) + \mu \left(\sum_{x \in X} [1 - \alpha(x)]p(x | D) - \bar{\epsilon}_D \right) \quad (20)$$

$$- \sum_{x \in X} \mu_0(x)\alpha(x) + \sum_{x \in X} \mu_1(x)(\alpha(x) - 1) \quad (21)$$

Neyman Pearson : Derivation (2)

$$L(\alpha(x_1), \alpha(x_2), \dots, \alpha(x_N)) = \sum_{x \in X} \alpha(x)p(x | N) + \mu \left(\sum_{x \in X} [1 - \alpha(x)]p(x | D) - \bar{\epsilon}_D \right) \quad (20)$$

$$- \sum_{x \in X} \mu_0(x)\alpha(x) + \sum_{x \in X} \mu_1(x)(\alpha(x) - 1) \quad (21)$$

The conditions for optimality are ($\forall x \in X$):

$$\frac{\partial L}{\partial \alpha(x)} = p(x | N) - \mu p(x | D) - \mu_0(x) + \mu_1(x) = 0, \quad (22)$$

$$\mu \geq 0, \mu_0(x) \geq 0, \mu_1(x) \geq 0, 0 \leq \alpha(x) \leq 1, \quad (23)$$

$$\mu_0(x)\alpha(x) = 0, \mu_1(x)(\alpha(x) - 1) = 0, \mu \left(\sum_{x \in X} [1 - \alpha(x)]p(x | D) - \bar{\epsilon}_D \right) = 0. \quad (24)$$

Case-by-case analysis:

case	implications
$\mu = 0$	L minimized by $\alpha(x) = 0 \quad \forall x$
$\mu \neq 0, \alpha(x) = 0$	$\mu_1(x) = 0 \Rightarrow \mu_0(x) = p(x N) - \mu p(x D) \Rightarrow p(x N)/p(x D) \leq \mu$
$\mu \neq 0, \alpha(x) = 1$	$\mu_0(x) = 0 \Rightarrow \mu_1(x) = -[p(x N) - \mu p(x D)] \Rightarrow p(x N)/p(x D) \geq \mu$
$\mu \neq 0, 0 < \alpha(x) < 1$	$\mu_0(x) = \mu_1(x) = 0 \Rightarrow p(x N)/p(x D) = \mu$

Neyman Pearson : Derivation (3)

Case-by-case analysis:

case	implications
$\mu = 0$	L minimized by $\alpha(x) = 0 \quad \forall x$
$\mu \neq 0, \alpha(x) = 0$	$\mu_1(x) = 0 \Rightarrow \mu_0(x) = p(x N) - \mu p(x D) \Rightarrow p(x N)/p(x D) \leq \mu$
$\mu \neq 0, \alpha(x) = 1$	$\mu_0(x) = 0 \Rightarrow \mu_1(x) = -[p(x N) - \mu p(x D)] \Rightarrow p(x N)/p(x D) \geq \mu$
$\mu \neq 0, 0 < \alpha(x) < 1$	$\mu_0(x) = \mu_1(x) = 0 \Rightarrow p(x N)/p(x D) = \mu$

Optimal Strategy for a given $\mu \geq 0$ and particular $x \in X$:

$$\frac{p(x | N)}{p(x | D)} \begin{cases} < \mu & \Rightarrow q(x) = D \text{ (as } \alpha(x) = 0) \\ > \mu & \Rightarrow q(x) = N \text{ (as } \alpha(x) = 1) \\ = \mu & \Rightarrow \text{LP relaxation does not give the desired solution, as } \alpha \notin \{0, 1\} \end{cases} \quad (25)$$

Neyman Pearson : Note on Randomized Strategies (1)

Consider:

$p(x D)$		
x_1	x_2	x_3
0.9	0.09	0.01

$p(x N)$		
x_1	x_2	x_3
0.09	0.9	0.01

$r(x) = p(x N)/p(x D)$		
x_1	x_2	x_3
0.1	10	1

and $\bar{\epsilon}_D = 0.03$.

- ◆ $q_1 : (x_1, x_2, x_3) \rightarrow (D, D, D) \Rightarrow \epsilon_D = 0.00, \epsilon_N = 1.00$
- ◆ $q_2 : (x_1, x_2, x_3) \rightarrow (D, D, N) \Rightarrow \epsilon_D = 0.01, \epsilon_N = 0.99$
- ◆ no other deterministic strategy q is feasible, that is all other ones have $\epsilon_D > \bar{\epsilon}_D$
- ◆ q_2 is the best deterministic strategy but it does not comply with the previous basic result of constructing the optimal strategy because it decides for N for likelihood ratio 1 but decides for D for likelihood ratios 0.01 and 10. Why is that?
- ◆ we can construct a randomized strategy which attains $\bar{\epsilon}_D$ and reaches lower ϵ_N :

$$q(x_1) = q(x_3) = D, \quad q(x_2) = \begin{cases} N & 1/3 \text{ of the time} \\ D & 2/3 \text{ of the time} \end{cases} \quad (26)$$

For such strategy, $\epsilon_D = 0.03, \epsilon_N = 0.7$.

Neyman Pearson : Note on Randomized Strategies (2)

- ◆ This is not a problem but a feature which is caused by discrete nature of X (does not happen when X is continuous).
- ◆ This is exactly what the case of $\mu = p(x | N)/p(x | D)$ is on slide 18.

Neyman Pearson : Notes (1)

- ◆ The task can be generalized to 3 hidden states, of which 2 are dangerous, $K = \{N, D_1, D_2\}$. It is formulated as an analogous problem with two inequality constraints and minimization of classification error for N.
- ◆ Neyman's and Pearson's work dates to 1928 and 1933.
- ◆ A particular strength of the approach lies in that the likelihood ratio $r(x)$ or even $p(x | N)$ need not be known. For the task to be solved, it is enough to know the $p(x | D)$ and the **rank order** of the likelihood ratio (to be demonstrated on the next page)

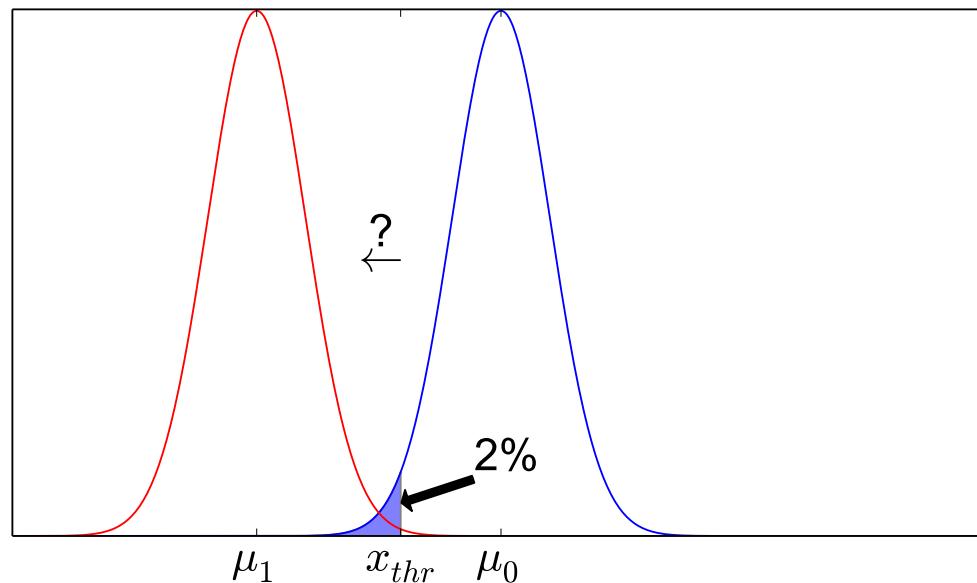
Neyman Pearson : Notes (2)

23/29

- ◆ Consider a medicine for reducing weight. The normal population has a distribution of weight $p(x | D)$ as shown in blue. Let it be normal, $p(x | D) = \mathcal{N}(x | \mu_0, \sigma)$. The distribution of weights after 1 month of taking the medicine is assumed to be normal as well, with the same variance but unknown shift of mean to the left, $p(x | N) = \mathcal{N}(x | \mu_1, \sigma)$, with $\mu_1 < \mu_0$ but otherwise unknown (shown in red). The likelihood ratio is

$r(x) = \exp \frac{1}{2\sigma^2} (-(x - \mu_1)^2 + (x - \mu_0)^2) = \exp \left(\frac{1}{\sigma^2} (\mu_1 - \mu_0)x + \text{const} \right)$. It is thus decreasing (monotone) with x (irrespective of μ_1 , $\mu_1 < \mu_0$).

- ◆ Setting $\bar{\epsilon}_D = 0.02$, we go along the decreasing $r(x)$ and find the point x_{thr} for which $\int_{-\infty}^{x_{thr}} p(x | D) = \bar{\epsilon}_D = 0.02$ (0.02-quantile). Note that the threshold μ on $r(x)$ is still unknown as $p(x | N)$ is unknown.



Minimax Task

- ◆ $K = \{1, 2, \dots, N\}$
- ◆ X set of observations
- ◆ Conditionals $p(x | k)$ are known $\forall k \in K$
- ◆ The priors $p(k)$ are unknown or do not exist
- ◆ $q: X \rightarrow K$ strategy

The Minimax Task looks for the optimum strategy q^* which minimizes the classification error of the worst classified class:

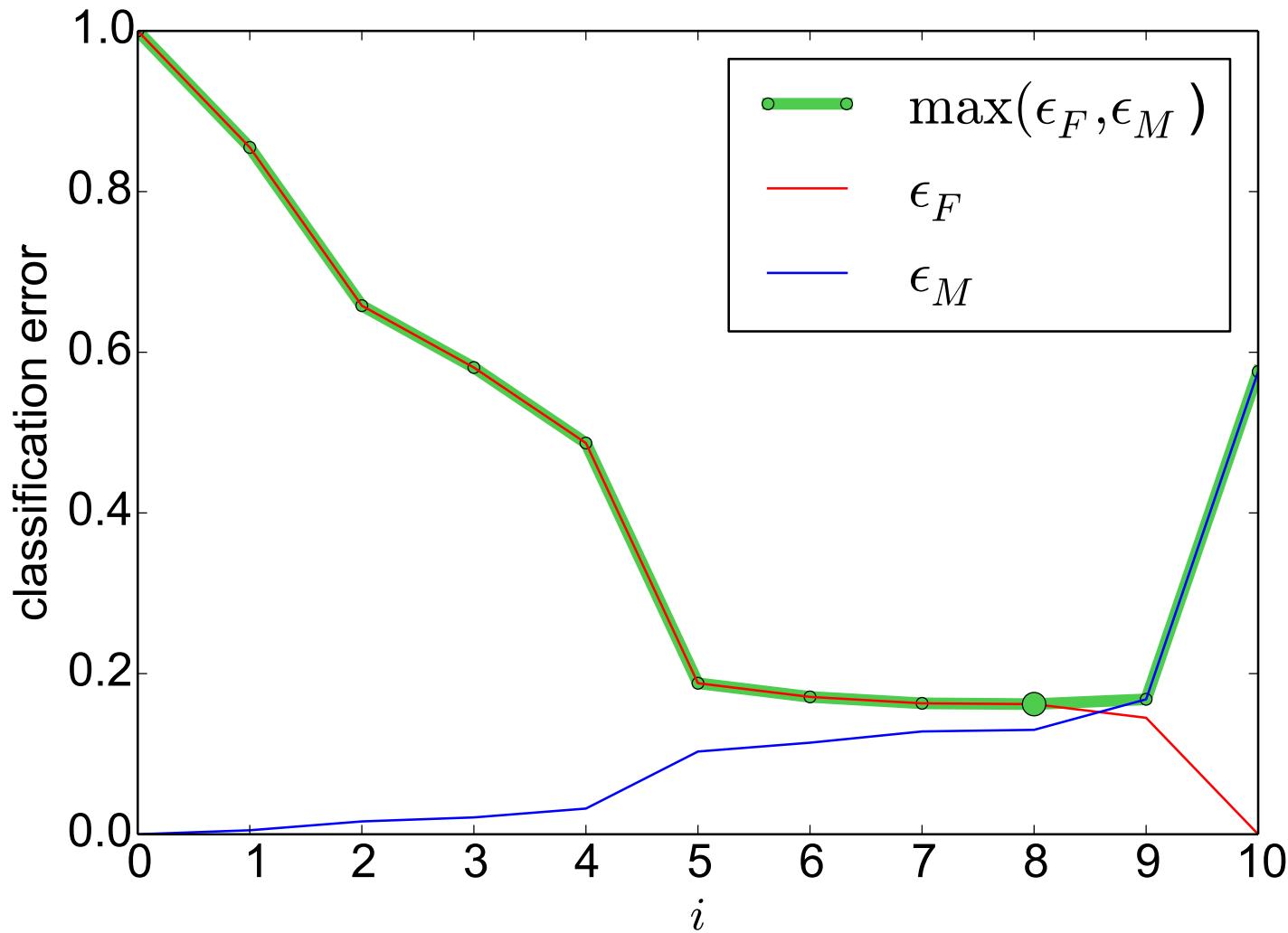
$$q^* = \operatorname{argmin}_{q: X \rightarrow K} \max_{k \in K} \epsilon(k), \quad \text{where} \tag{27}$$

$$\epsilon(k) = \sum_{x: q(x) \neq k} p(x | k) \tag{28}$$

- ◆ Example: A recognition algorithm qualifies for a competition using preliminary tests. During the final competition, only objects from the hardest-to-classify class are used.
- ◆ For a 2-class problem, the strategy is again constructed using the likelihood ratio.
- ◆ In the case of continuous observations space X , equality of classification errors is attained: $\epsilon_1 = \epsilon_2$
- ◆ The derivation can again be done using Linear Programming.

Example: Male/Female Recognition (Minimax)

Classification errors for F and M, for $\mu_i = \frac{r_i + r_{i+1}}{2}$ and $\mu_0 = 0$.



The optimum is attained for $i = 8$, $\epsilon_F = .162$, $\epsilon_M = .13$. The corresponding strategy is as shown on slide 11.

Minimax: Comparison with Bayesian Decision with Unknown Priors

- ◆ Consider the same setting as in the Minimax task, but let the priors $p(k)$ exist but be unknown.
- ◆ The Bayesian error ϵ for strategy q is

$$\epsilon = \sum_k \sum_{x: q(x) \neq k} p(x, k) = \sum_k p(k) \underbrace{\sum_{x: q(x) \neq k} p(x | k)}_{\epsilon(k)} \quad (29)$$

- ◆ We want to minimize ϵ but we do not know $p(k)$'s. What is the maximum it can attain? Obviously, the $p(k)$'s do the convex combination of the class errors $\epsilon(k)$; the maximum Bayesian error will be attained when $p(k) = 1$ for the class k with the highest class error $\epsilon(k)$.
- ◆ Thus, to minimize the Bayesian error ϵ under this setting, the solution is to minimize the error of the hardest-to-classify class.
- ◆ Therefore, Minimax formulation and the Bayesian formulation with Unknown Priors lead to the same solution.

Wald Task (1)

- ◆ Let us consider classification with two states, $K = \{1, 2\}$.
- ◆ We want to set a threshold ϵ on the classification error of both of the classes: $\epsilon_1 \leq \epsilon$, $\epsilon_2 \leq \epsilon$.
- ◆ As the previous analysis shows (Neyman Pearson, Minimax), there may be **no** feasible solution if ϵ is set too low.
- ◆ That is why the possibility of decision “do not know” is introduced. Thus $D = K \cup \{?\}$
- ◆ A strategy $q : X \rightarrow D$ is characterized by:

$$\epsilon_1 = \sum_{x: q(x)=2} p(x | 1) \quad (\text{classification error for 1}) \quad (30)$$

$$\epsilon_2 = \sum_{x: q(x)=1} p(x | 2) \quad (\text{classification error for 2}) \quad (31)$$

$$\kappa_1 = \sum_{x: q(x)=?} p(x | 1) \quad (\text{undecided rate for 1}) \quad (32)$$

$$\kappa_2 = \sum_{x: q(x)=?} p(x | 2) \quad (\text{undecided rate for 2}) \quad (33)$$

Wald Task (2)

- ◆ The optimal strategy q^* :

$$q^* = \operatorname{argmin}_{q:X \rightarrow D} \max_{i=\{1,2\}} \kappa_i \quad (34)$$

$$\text{subject to: } \epsilon_1 \leq \epsilon, \epsilon_2 \leq \epsilon \quad (35)$$

- ◆ The task is again solvable using LP (even for more than 2 classes)
- ◆ The optimal solution is again based on the likelihood ratio

$$r(x) = \frac{p(x|1)}{p(x|2)} \quad (36)$$

- ◆ The optimal strategy is constructed using suitably chosen thresholds μ_l and μ_h such that:

$$q(x) = \begin{cases} 2 & \text{for } r(x) < \mu_l \\ 1 & \text{for } r(x) > \mu_h \\ ? & \text{for } \mu_l \leq r(x) \leq \mu_h \end{cases} \quad (37)$$

Example: Male/Female Recognition (Wald)

Solve the Wald task for $\epsilon = 0.05$.

$$p(x|\mathbf{F})$$

short	.197	.145	.094	.017
normal	.077	.299	.145	.017
tall	.001	.008	.000	.000
	u-light	light	avg	heavy

$$p(x|\mathbf{M})$$

short	.011	.005	.011	.011
normal	.005	.071	.408	.038
tall	.002	.014	.255	.169
	u-light	light	avg	heavy

$$r(x) = p(x|\mathbf{M})/p(x|\mathbf{F})$$

short	0.056	0.034	0.117	0.647
normal	0.065	0.237	2.814	2.235
tall	2.000	1.750	∞	∞
	u-light	light	avg	heavy

$$\text{rank, and } q^*(x) = \{\mathbf{F}, \mathbf{M}, ?\}$$

short	2	1	4	6
normal	3	5	10	9
tall	8	7	11	12
	u-light	light	avg	heavy

Result: $\epsilon_M = 0.032$, $\epsilon_F = 0$, $\kappa_M = 0.544$, $\kappa_F = 0.487$

$$(r_4 < \mu_l < r_5, r_{10} < \mu_h < \infty)$$

Parameter Estimation: Maximum Likelihood (ML), Maximum a Posteriori (MAP), and Bayesian

Lecturer:
Jiří Matas

Authors:
Ondřej Drbohlav, Jiří Matas

Centre for Machine Perception
Czech Technical University, Prague
<http://cmp.felk.cvut.cz>

Last update: 7.10.2020



Probability Estimation

Both in the Bayesian Decision Theory and the Non-Bayesian Methods lectures, it has been assumed that all the necessary probabilities (priors, conditionals) are known.

In practice, the probabilities almost always need to be estimated from the training data.

Probability Distribution Estimation Methods (1/2)

According to the form of the model for the distribution:

- ◆ **Parametric.** The distribution has a known form of a function which has parameters $\theta = (\theta_1, \theta_2, \dots, \theta_D)$. The number of parameters is low.
Example: the normal distribution $\mathcal{N}(x | \mu, \sigma^2)$: the parameters to be estimated are $\theta = \{\mu, \sigma^2\}$. The parameter space is two-dimensional.
- ◆ **Non-parametric.**: The same as with the Parametric models, but the number of parameters to be estimated is very high. Note the apparent contradiction in the terminology (high number of parameters to estimate \rightarrow “non-parametric” method?). This is because the term ‘parameter’ often disappears from the estimating methods procedure.
Example: K-nearest neighbors; Parzen window; histogram.

To be discussed: complexity of estimating e.g. mixtures:

$$p(x) = \sum_{i=1}^D \pi_i \mathcal{N}(\mu_i, \sigma_i^2), \quad (1)$$

or parameters of feed-forward neural nets.

Probability Distribution Estimation Methods (2/2)

Learning principles:

- ◆ Maximum Likelihood
- ◆ Maximum A Posteriori
- ◆ Bayesian Parameter Estimation

Maximum Likelihood (ML) Principle

- ◆ The training set \mathcal{T} is available, $\mathcal{T} = \{(x_1, k_1), (x_2, k_2), \dots, (x_N, k_N)\}$.
- ◆ The parametric form of the likelihood $L(\boldsymbol{\theta}) = p(\mathcal{T}|\boldsymbol{\theta})$ is known.
- ◆ Note that the likelihood function $L(\boldsymbol{\theta})$ is a function of the parameters $\boldsymbol{\theta}$, for fixed observations \mathcal{T} . In particular, $L(\boldsymbol{\theta})$ does not sum up to 1.

ML principle

The maximum likelihood estimate $\hat{\boldsymbol{\theta}}$ for the observed data \mathcal{T} is defined as:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} L(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} p(\mathcal{T}|\boldsymbol{\theta}). \quad (2)$$

The argument for this formulation is, informally, “if the parameters are correct then they will give larger probabilities for the observations, compared to wrong parameters”.

Usually, the parameters for different classes are independent (no shared parameters between classes). In that case, the likelihood function $p(\mathcal{T}|\boldsymbol{\theta})$ can be factorized to

$$p(\mathcal{T}|\boldsymbol{\theta}) = p(\mathcal{T}_1|\boldsymbol{\theta}_1)p(\mathcal{T}_2|\boldsymbol{\theta}_2)\dots p(\mathcal{T}_K|\boldsymbol{\theta}_K) \quad (3)$$

where $\mathcal{T}_k = \{x: (x, l) \in \mathcal{T} \wedge l = k\}$ is the training set for class k . The parameters $\boldsymbol{\theta}_k$ for individual classes can be estimated independently. \Rightarrow **In the subsequent text, we will drop the class index k . All analysis will be done “per class”.**

Maximum Likelihood (ML) Estimation

Consider the observations $\mathcal{T} = \{x_1, x_2, \dots, x_N\}$ and the known parametric form of the likelihood function $L(\boldsymbol{\theta}) = p(\mathcal{T}|\boldsymbol{\theta})$. The ML estimate of $\hat{\boldsymbol{\theta}}$ is

$$\hat{\boldsymbol{\theta}} = \operatorname{argmax}_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \operatorname{argmax}_{\boldsymbol{\theta}} p(\mathcal{T}|\boldsymbol{\theta}). \quad (19a)$$

- ◆ If samples in \mathcal{T} are independent and identically distributed (i.i.d) then $p(\{x_1, x_2, \dots, x_N\}|\boldsymbol{\theta}) = \prod_{i=1}^N p(x_i|\boldsymbol{\theta})$, and the ML estimate for the class is

$$\hat{\boldsymbol{\theta}} = \operatorname{argmax}_{\boldsymbol{\theta}} \prod_{i=1}^N p(x_i|\boldsymbol{\theta}). \quad (4)$$

- ◆ The argument $\hat{\boldsymbol{\theta}}$ maximizing likelihood in Eq. (4) equals the argument maximizing the log-likelihood (as logarithm is an increasing function). This fact will often be taken advantage of in calculations.

Example 1: Binomial Distribution (ML) (1)

There are red and green socks in the drawer. N socks have been drawn randomly from the drawer, with replacement. The result is:

$$R \text{ red socks} \quad (5)$$

$$G \text{ green socks } (G = N - R) \quad (6)$$

Compute the Maximum Likelihood estimate for the actual percentage π of the red socks in the drawer.

Analysis. For an individual draw, $P(\text{red}|\pi) = \pi$ and $P(\text{green}|\pi) = 1 - \pi$. For N independent measurements with an outcome as in Eqs. (5, 6), the likelihood is

$$p(R, N | \pi) = \binom{N}{R} \pi^R (1 - \pi)^{N-R}. \quad (7)$$

Note: Consider a training set in a slightly different form: it is an ordered sequence of observations $\mathcal{T} = (\text{red}, \text{green}, \text{green}, \dots, \text{green})$, with R observations "red" and G observations "green", as before. What is the likelihood function for these observations? How does it differ from Eq. (7)? Will it matter for the ML estimation result?

Example 1: Binomial Distribution (ML) (2)

(copied from the previous slide:)

$$p(R, N | \pi) = \binom{N}{R} \pi^R (1 - \pi)^{N-R}. \quad (7)$$

Taking the derivative of $p(R, N | \pi)$ with respect to π and setting it to zero gives

$$\binom{N}{R} R \pi^{R-1} (1 - \pi)^{N-R} - \binom{N}{R} \pi^R (N - R) (1 - \pi)^{N-R-1} = 0, \quad (8)$$

and thus

$$R(1 - \pi) - (N - R)\pi = 0 \quad (9)$$

which implies

$$\hat{\pi}_{\text{ML}} = \frac{R}{N}. \quad (10)$$

The ML solution is the fraction of the red socks within the socks drawn.

Example 2: Normal Distribution (ML) (1)

Let the conditional probability of a class be normal. Assume that the observations $\mathcal{T} = \{x_1, x_2, \dots, x_N\}$ are i.i.d. and find the ML estimate for the mean and variance. The likelihood to be maximized is:

$$P(\mathcal{T}|\mu, \sigma) = \frac{1}{\sigma^N \sqrt{(2\pi)^N}} \exp \left[-\frac{1}{2\sigma^2} \sum_{i=1}^N (x_i - \mu)^2 \right]. \quad (11)$$

We require that the partial derivatives w.r.t. both μ and σ vanish:

$$\frac{\partial P(\mathcal{T}|\mu, \sigma)}{\partial \mu} = P(\mathcal{T}|\mu, \sigma) \frac{1}{\sigma^2} \left(\sum_{i=1}^N (x_i - \mu) \right) = 0 \quad (12)$$

$$\frac{\partial P(\mathcal{T}|\mu, \sigma)}{\partial \sigma} = -P(\mathcal{T}|\mu, \sigma) \frac{N}{\sigma} + P(\mathcal{T}|\mu, \sigma) \frac{1}{\sigma^3} \left(\sum_{i=1}^N (x_i - \mu)^2 \right) = 0 \quad (13)$$

The first and second equations imply, respectively, the terms on the right. The ML estimator for mean is the sample mean (as before in Example 1) and the ML estimator for variance is the sample variance, with sample mean Eq. (14) plugged into Eq. (15).

$$\hat{\mu}_{\text{ML}} = \frac{1}{N} \sum_{i=1}^N x_i, \quad (14)$$

$$\hat{\sigma}_{\text{ML}}^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \hat{\mu}_{\text{ML}})^2. \quad (15)$$

Example 2: Normal Distribution (ML) (2)

Let us try now with maximizing the log-likelihood:

$$l(\mathcal{T}|\mu, \sigma) = \ln P(\mathcal{T}|\mu, \sigma) = -N \ln \sigma - \frac{N}{2} \ln 2\pi - \frac{1}{2\sigma^2} \sum_{i=1}^N (x_i - \mu)^2. \quad (16)$$

Again, setting the partial derivatives w.r.t. μ and σ to zero yeilds

$$\frac{1}{\sigma^2} \sum_{i=1}^N (x_i - \mu) = 0, \quad (17)$$

$$-\frac{N}{\sigma} + \frac{1}{\sigma^3} \sum_{i=1}^N (x_i - \mu)^2 = 0, \quad (18)$$

which leads to the same solution as before.

Maximum Likelihood—Features, Problems (1)

Why ML estimators?

Under very general conditions, ML is

- ◆ Asymptotically unbiased: as the number of observations N grows to infinity, ML estimate approaches the actual parameters θ_0 ($E(\hat{\theta}) = \theta_0$)
- ◆ Asymptotically consistent: sequence of estimates converges in probability to θ_0 as N grows to infinity ($\lim_{N \rightarrow \infty} \text{prob}\{\|\hat{\theta} - \theta_0\| \leq \epsilon\} = 1$)
- ◆ Asymptotically efficient
- ◆ Asymptotically normal (pdf of ML estimates as $N \rightarrow \infty$ approached Gaussian.)

Maximum Likelihood—Features, Problems (2)

With low number of observations, the ML estimates can be counter-intuitive. Consider the following examples:

- ◆ Binomial distribution, coin tossing, $\mathcal{T} = \{H, H, H\}$. The ML estimate is $\pi_{\text{head}} = 1$ (completely unfair coin). Would you believe that estimate?
- ◆ Normal distribution, estimating x -coordinate of a particle. A range of feasible μ 's can be known a priori, but the sample mean taken from a few observations can be outside this range.

These examples demonstrate that employing a prior knowledge (or belief) about the parameters to be estimated would be beneficial, if available.

Maximum A Posteriori (MAP) Estimation

- ◆ The set of observations \mathcal{T} is $\mathcal{T} = \{x_1, x_2, \dots, x_N\}$.
- ◆ The parametric form of the likelihood function $L(\boldsymbol{\theta}) = p(\mathcal{T}|\boldsymbol{\theta})$ is known.
- ◆ The prior distribution $p(\boldsymbol{\theta})$ of the model parameters $\boldsymbol{\theta}$ is known.

MAP principle

The maximum a posteriori estimate $\hat{\boldsymbol{\theta}}$ of the distribution parameters for the observed data \mathcal{T} is defined as:

$$\hat{\boldsymbol{\theta}} = \operatorname{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{\theta} | \mathcal{T}). \quad (19)$$

The posterior $p(\boldsymbol{\theta}|\mathcal{T})$ can be computed from $p(\mathcal{T}|\boldsymbol{\theta})$ and the prior $p(\boldsymbol{\theta})$ using the Bayes formula:

$$p(\boldsymbol{\theta}|\mathcal{T}) = \frac{p(\mathcal{T}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{T})}. \quad (20)$$

The denominator of Eq. (20) is *independent* of the parameters $\boldsymbol{\theta}$, and the solution $\hat{\boldsymbol{\theta}}$ can be found by maximizing the nominator only:

$$\hat{\boldsymbol{\theta}} = \operatorname{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{\theta} | \mathcal{T}) = \operatorname{argmax}_{\boldsymbol{\theta}} \frac{p(\mathcal{T}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{T})} = \operatorname{argmax}_{\boldsymbol{\theta}} p(\mathcal{T}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \quad (21)$$

which has practical implications and shows the difference w.r.t. the ML approach: The term to be maximized is the **product of the likelihood** (as in the ML) **and the prior** on $\boldsymbol{\theta}$ which “shifts” the optimum $\hat{\boldsymbol{\theta}}$ when the number of observations is low.

Example 1, Binomial Distribution (MAP) (1)

Recall that

$$p(R, N | \pi) = \binom{N}{R} \pi^R (1 - \pi)^{N-R}, \quad (7)$$

where N is the total number of socks drawn, of which R are the red ones, $G = N - R$ are the green ones and π is the percentage of red socks in the sock population to be estimated.

We need a suitable prior on π . A lucky coincidence would be if the prior $p(\pi)$ would take the same functional form in π as the above equation, that is,

$$p(\pi) \sim \pi^A (1 - \pi)^B. \quad (22)$$

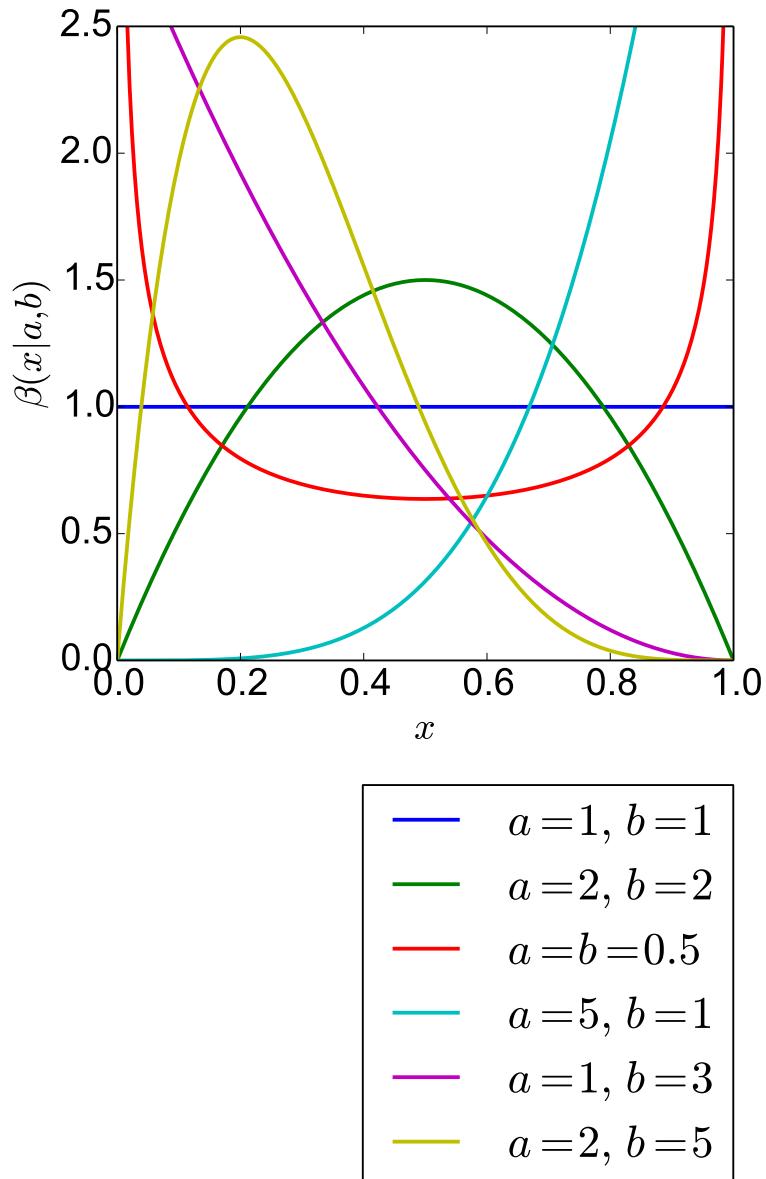
This would imply that the product of likelihood and the prior would be

$$p(R, N | \pi)p(\pi) \sim \pi^R (1 - \pi)^{N-R} \pi^A (1 - \pi)^B = \pi^{R+A} (1 - \pi)^{N-R+B}. \quad (23)$$

Such prior $p(\pi)$ is known under the name Beta distribution. The maximization of this term is already done, as due to this functional form it is the same as the ML solution for $(R + A)$ red socks out of the total number of $(N + A + B)$. Thus, for such a prior,

$$\hat{\pi}_{\text{MAP}} = \frac{R + A}{N + A + B}. \quad (24)$$

Example 1, Binomial Distribution (MAP) (2)



The prior distribution $p(\pi) \sim \pi^A(1 - \pi)^B$ is known as the **Beta distribution**, and is defined as:
 (note the subtle change $A \rightarrow a - 1, B \rightarrow b - 1$)

$$\beta(\pi|a, b) = \frac{\pi^{a-1}(1 - \pi)^{b-1}}{\int_0^1 \pi^{a-1}(1 - \pi)^{b-1} d\pi} = \frac{1}{B(a, b)} \pi^{a-1}(1 - \pi)^{b-1} \quad (25)$$

where $B(a, b)$, the normalizing constant, is the Beta function. Using the β distribution, the term $p(R, N|\pi)p(\pi)$ can be rewritten as

$$p(R, N|\pi)p(\pi) \sim \pi^{R+A}(1 - \pi)^{N-R+B} \quad (26)$$

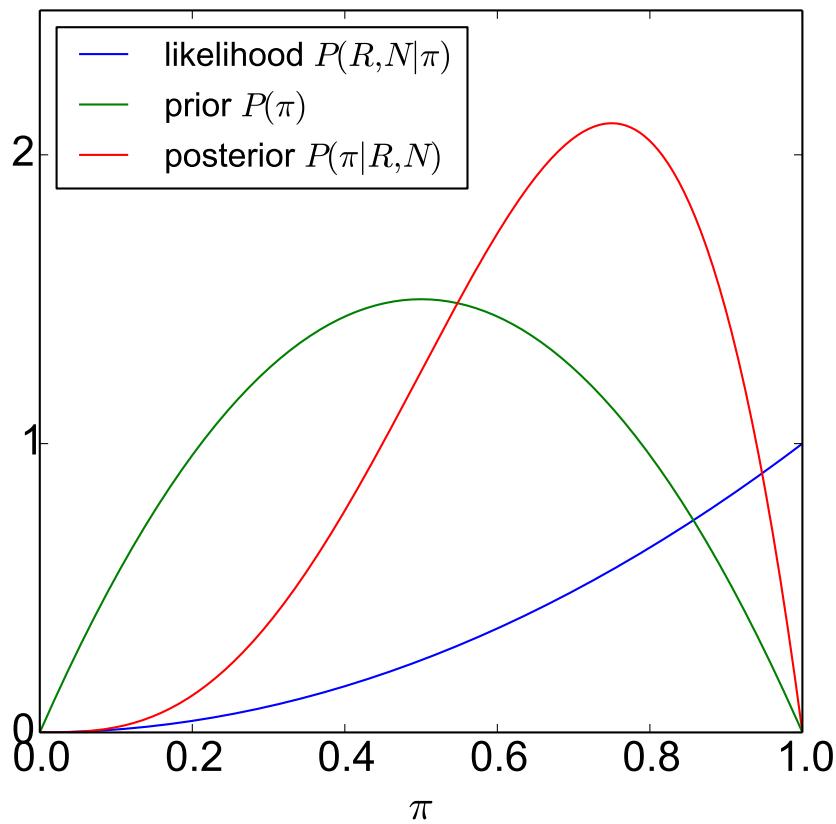
$$\sim \beta(R + A + 1, N - R + B + 1). \quad (27)$$

Note that, indeed, the **posterior** $p(\pi|R, N) = \beta(R + A + 1, N - R + B + 1)$.

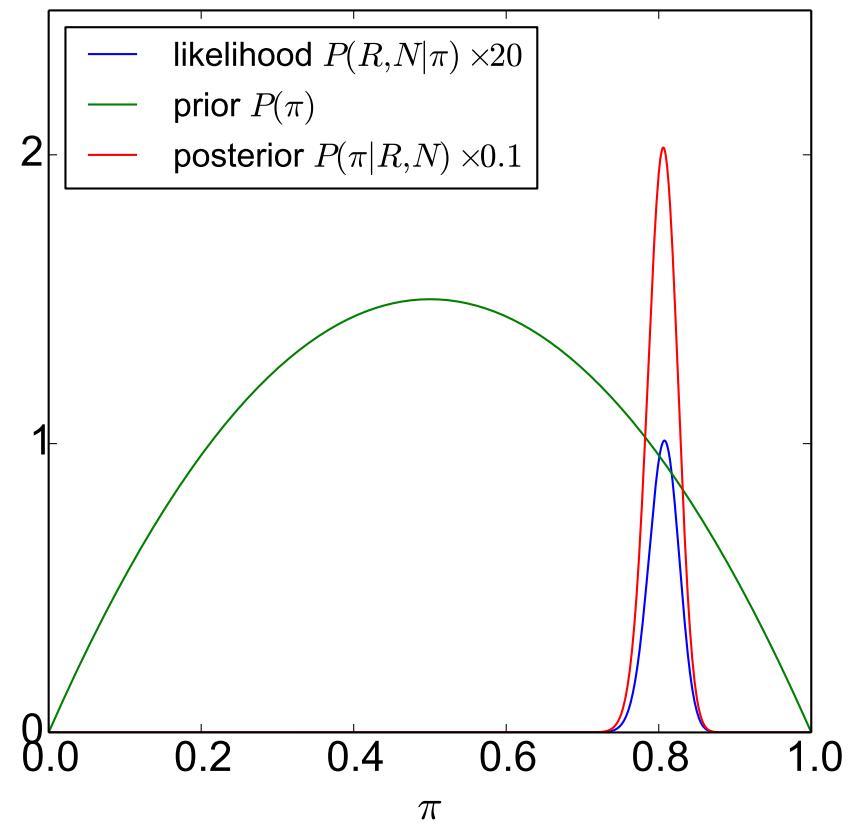
Example 1, Binomial Distribution (MAP) (3)

Examples:

- $N = 2$ socks drawn
- both of them are red ($R = 2$)
- the prior is set to $\beta(r|2, 2) \sim r(1 - r)$



- $N = 400$ socks drawn
- of which $R = 323$ are red ones
- the prior is set to $\beta(r|2, 2) \sim r(1 - r)$



- $\hat{\pi}_{\text{ML}} = 2/2 = 1$
- $\hat{\pi}_{\text{MAP}} = (2 + 1)/(2 + 2) = 3/4$

- $\hat{\pi}_{\text{ML}} = 323/400$
- $\hat{\pi}_{\text{MAP}} = 324/401$

Example 1, Binomial Distribution (MAP) (4)

$$\hat{\pi}_{\text{MAP}} = \frac{R + A}{N + A + B} \quad (24)$$

Note that the parameters A, B of the prior $p(\pi) \sim \pi^A(1 - \pi)^B$ behave as “*virtual*” observations; it is as if A red socks and B green socks have been already observed before any real observation has been done.

The parameters of the prior (here A, B) are generally called **hyperparameters**. This name distinguishes them from the parameters of the probabilistic model which are to be estimated.

Example 2: Normal Distr. with unknown μ (MAP) (1)

Consider the normal distribution $p(\mathcal{T} | \mu, \sigma^2)$, and for simplicity let the variance σ^2 be known and equal to $\sigma^2 = 1$. Estimate the mean μ .

The likelihood is

$$L(\mu) = p(\mathcal{T} | \mu) = \prod_{i=1}^N p(x_i | \mu) = \frac{1}{\sqrt{(2\pi)^N}} \exp \left[-\frac{1}{2} \sum_{i=1}^N (x_i - \mu)^2 \right]. \quad (28)$$

Consider the prior $p(\mu)$ on μ to be distributed normally:

$$p(\mu) = \mathcal{N}(\mu | \mu_0, \sigma_0^2) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp \left[-\frac{1}{2} \frac{(\mu - \mu_0)^2}{\sigma_0^2} \right]. \quad (29)$$

The MAP estimate of μ will be found as

$$\mu_{\text{MAP}} = \underset{\mu}{\operatorname{argmax}} p(\mathcal{T} | \mu) p(\mu). \quad (30)$$

Example 2: Normal Distr. with Unknown μ (MAP) (2)

The exponent of the likelihood $p(\mathcal{T}|\mu)$ (recall $\sigma^2 = 1$ is fixed and known) can be rewritten as

$$-\frac{1}{2} \sum_{i=1}^N (x_i - \mu)^2 = -\frac{1}{2} \left(\sum_{i=1}^N x_i^2 - 2\mu \sum_{i=1}^N x_i + N\mu^2 \right) = \quad (31)$$

$$-\frac{1}{2}N \left(\mu^2 - 2\mu \underbrace{\frac{\sum_{i=1}^N x_i}{N}}_{\mu'} + C_1 \right) = -\frac{N}{2} (\mu - \mu')^2 + C_2, \quad (32)$$

where μ' is the sample mean of \mathcal{T} and C_1 and C_2 are constants independent of μ . The posterior $p(\mu|\mathcal{T})$ can then be written again as a normal distribution (mean μ_c , var. σ_c^2):

$$p(\mu|\mathcal{T}) \sim \frac{\exp \left[-\frac{1}{2}N(\mu - \mu')^2 \right] \mathcal{N}(\mu|\mu_0, \sigma_0^2)}{Z'(\mathcal{T})} = \frac{\mathcal{N}(\mu|\mu', \frac{1}{N}) \mathcal{N}(\mu|\mu_0, \sigma_0^2)}{Z''(\mathcal{T})} = \mathcal{N}(\mu|\mu_c, \sigma_c^2) \quad (33)$$

where $Z'(\mathcal{T})$ and $Z''(\mathcal{T})$ are suitable normalizing constants.

Example 2: Normal Distr. with Unknown μ (MAP) (3)

It is easy to see that making a product of two normal distributions $\mathcal{N}(\mu|\mu_1, \sigma_1^2)$ and $\mathcal{N}(\mu|\mu_2, \sigma_2^2)$ and normalizing it results in another normal distribution $\mathcal{N}(\mu|\mu_c, \sigma_c^2)$, as looking at the exponent of the product, there are terms:

$$\frac{(\mu - \mu_1)^2}{\sigma_1^2} + \frac{(\mu - \mu_2)^2}{\sigma_2^2} = \mu^2 \left[\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} \right] - 2\mu \left[\frac{\mu_1}{\sigma_1^2} + \frac{\mu_2}{\sigma_2^2} \right] + D_1 \quad (34)$$

$$= \left[\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} \right] \left(\mu - \left[\frac{\mu_1}{\sigma_1^2} + \frac{\mu_2}{\sigma_2^2} \right] / \left[\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} \right] \right)^2 + D_2 \quad (35)$$

where D_1 and D_2 are constants which do not even need to be evaluated, as they will be factored into the normalization provided by the term for normal distribution itself. Pairing the parameters and the terms, we obtain

$$\mu_c = \left[\frac{\mu_1}{\sigma_1^2} + \frac{\mu_2}{\sigma_2^2} \right] / \left[\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} \right] \quad \sigma_c^2 = \left[\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} \right]^{-1}. \quad (36)$$

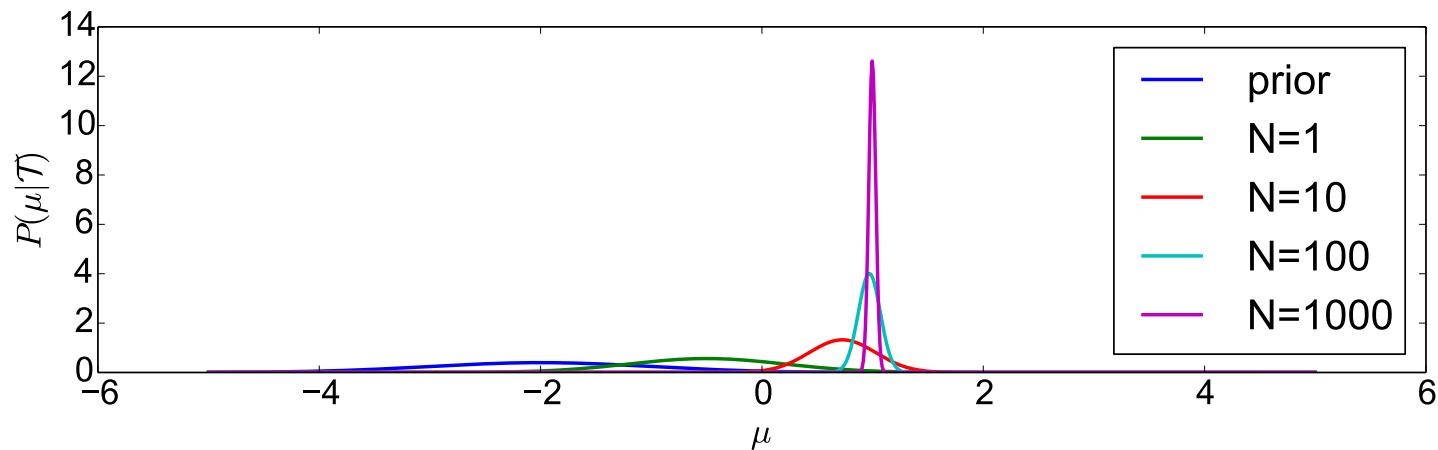
Thus for the case studied here,

$$P(\mu|\mathcal{T}) = \frac{\mathcal{N}(\mu|\mu', \frac{1}{N}) \mathcal{N}(\mu|\mu_0, \sigma_0^2)}{Z''(\mathcal{T})} = \mathcal{N}\left(\mu | \frac{\left[N\mu' + \frac{\mu_0}{\sigma_0^2}\right]}{[N + 1/\sigma_0^2]}, \frac{1}{N + 1/\sigma_0^2}\right). \quad (37)$$

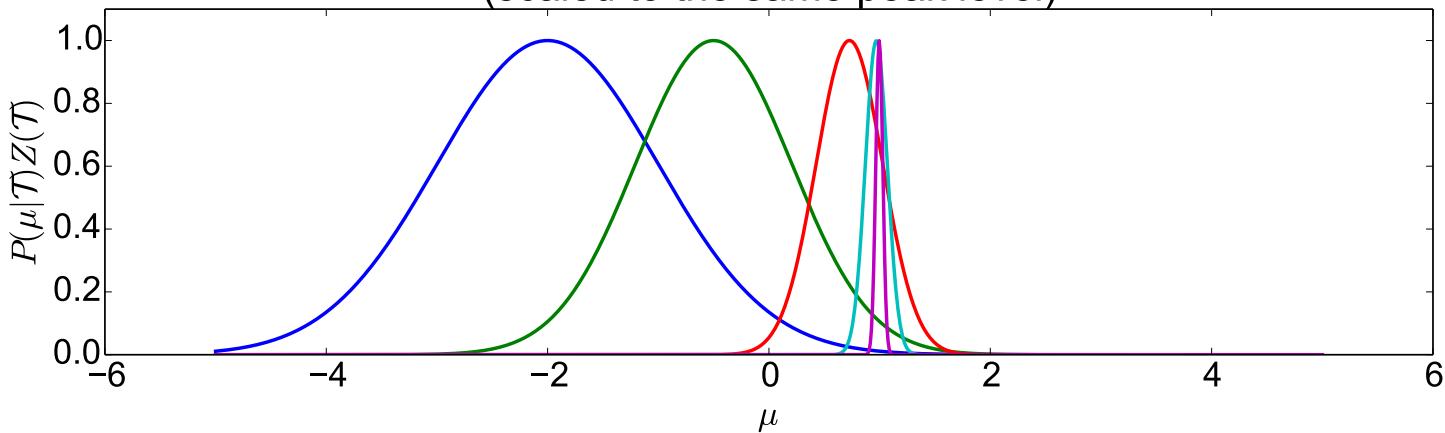
Example 2: Normal Distr. with unknown μ (MAP) (4)

Thus when the likelihood is normal in μ and the prior is also normal in μ , the posterior is normal in μ as well.

Example: The posterior of μ for sample mean $\mu' = 1$ and the prior $p(\mu) = \mathcal{N}(\mu | \mu_0 = -2, \sigma_0^2 = 1)$, for different sizes of the observation set:



(scaled to the same peak level)



Conjugate Prior

In both Example 1 and Example 2, the priors had a functional form which was similar to the form of the likelihood.

This enabled us to make an easy derivation of the MAP formulas.

In general, such “suitable” priors are called **conjugate**.

Bayesian Parameter Estimation

- ◆ The requirements are the same as in MAP (it is necessary to know $p(\mathcal{T}|\theta)$ and $p(\theta)$)
- ◆ ML and MAP search for the maximum of likelihood and likelihood combined with prior, respectively.
- ◆ The Bayesian Parameter Estimation instead minimizes the risk $R(\theta)$ of the estimate θ (quadratic loss function and one-dimensional estimation problem considered here):

$$R(\theta) = \int_{-\infty}^{\infty} p(t|\mathcal{T})(t - \theta)^2 dt \quad (38)$$

$$\theta_{\text{MSE}} = \operatorname{argmin}_{\theta} R(\theta) \quad (39)$$

(MSE = Mean Squared Error)

- ◆ This leads to

$$\theta_{\text{MSE}} = \int_{-\infty}^{\infty} t p(t|\mathcal{T}) dt. \quad (40)$$

- ◆ It is very convenient when the prior has a suitable form (*conjugate prior.*)

Example 1, Binomial Distribution (Bayesian)

We know (from the MAP analysis on slide 15) that for the prior $p(\pi) = \beta(A + 1, B + 1)$, the posterior $p(\pi|R, N)$ is

$$p(\pi|R, N) = \beta(R + A + 1, [N - R] + B + 1). \quad (41)$$

The estimate π_{MSE} obtained is

$$\pi_{\text{MSE}} = \int_0^1 \pi p(\pi|R, N) d\pi = \frac{R+A+1}{N+A+B+2}, \quad (42)$$

where we have used the known fact about the β distribution that the expected value for $\beta(a, b)$ is $a/(a + b)$.

Example: Consider $R = 2$, $N = 2$ and the **uniform** prior, $p(\pi) = r^0(1 - r)^0$ (thus $A = 0, B = 0.$) Then $\pi_{\text{MSE}} = (R + 1)/(N + 2) = 3/4$ ($\pi_{\text{ML}} = \pi_{\text{MAP}} = R/N = 1.$)

Useful known properties of the Beta distribution $\beta(a, b)$:

- ◆ mode (= maximum value): $\frac{a-1}{a+b-2}$ (agrees with our computations)
- ◆ mean (expected value): $\frac{a}{a+b}$
- ◆ variance: $\frac{ab}{(a+b)^2(a+b+1)}$

Estimator Properties: Bias (1/5)

“Is an estimator biased?”

This question has the following meaning. Let us say that we observe $N = 5$ data points x_1, x_2, \dots, x_5 and estimate $\hat{\mu}_{\text{ML}}$ and $\hat{\sigma}_{\text{ML}}^2$ from them. These estimates will most likely be different from the true parameters μ and σ of the distribution $\mathcal{N}(x|\mu, \sigma)$. But how about if we do this repeatedly, that is:

for $i=1$ to K **do**

 get a 5-tuple

 compute $\mu_{\text{ML}}(i), \sigma_{\text{ML}}^2(i)$

end for

Average the obtained values: $\mu'_{\text{ML}} = \frac{1}{K} \sum_{i=1}^K \mu_{\text{ML}}(i)$, $\sigma'^2_{\text{ML}} = \frac{1}{K} \sum_{i=1}^K \sigma_{\text{ML}}^2(i)$

If such a procedure produces true parameter values in the limit as $K \rightarrow \infty$ then the estimator is unbiased. Otherwise, it is biased.

Mathematically, (for an example of μ) this is written as

$$\mu - E \left[\frac{1}{5} \sum_{i=1}^5 x_i \right] = 0 \text{ iff the estimator is unbiased.} \quad (43)$$

where E is the expected value operator which integrates over the entire distribution of 5-tuples.

Estimator Properties: Bias (2/5), Sample Mean

This expected value is:

$$E \left[\frac{1}{5} \sum_{i=1}^5 x_i \right] = \iiint \int_{-\infty}^{\infty} \underbrace{\frac{1}{5} (x_1 + \dots + x_5)}_{\text{estimator}} \underbrace{\mathcal{N}(x_1|\mu, \sigma) \mathcal{N}(x_2|\mu, \sigma) \cdot \dots \cdot \mathcal{N}(x_5|\mu, \sigma)}_{\text{5-tuples distribution}} dx_1 \dots dx_5 \quad (44)$$

$$= \underbrace{\frac{1}{5} \int_{-\infty}^{\infty} x_1 \mathcal{N}(x_1|\mu, \sigma) dx_1}_{\mu} \underbrace{\int_{-\infty}^{\infty} \mathcal{N}(x_2|\mu, \sigma) dx_2 \cdot \dots \cdot \int_{-\infty}^{\infty} \mathcal{N}(x_5|\mu, \sigma) dx_5}_{1} + 4 \text{ analogous terms} \quad (45)$$

$$= \frac{1}{5} 5\mu = \mu. \quad (46)$$

As the expected value of the sample mean estimator is μ , the estimator is unbiased. The same is obviously true for arbitrary N .

Estimator Properties: Bias (3/5), Sample Variance

Is the variance estimator also unbiased? The expected value of the estimator is

$$E[\sigma_{ML}^2] = \int \dots \int_{-\infty}^{\infty} \underbrace{\frac{1}{N} \sum_{i=1}^N (x_i - \mu_{ML})^2}_{\text{estimator}} \mathcal{N}(x_1 | \mu, \sigma) \dots \mathcal{N}(x_N | \mu, \sigma) dx_1 \dots dx_N. \quad (47)$$

The estimator is

$$\frac{1}{N} \sum_{i=1}^N (x_i - \mu_{ML})^2 = \frac{1}{N} \sum_{i=1}^N \left(x_i - \frac{1}{N} \sum_{j=1}^N x_j \right)^2 = \frac{1}{N} \sum_{i=1}^N \left((x_i - \mu) - \frac{1}{N} \sum_{j=1}^N (x_j - \mu) \right)^2 \quad (48)$$

and thus by substituting $x_i \leftarrow (x_i - \mu)$ Eq. (47) is rewritten as

$$E[\sigma_{ML}^2] = \int \dots \int_{-\infty}^{\infty} \frac{1}{N} \sum_{i=1}^N \left(x_i - \frac{1}{N} \sum_{j=1}^N x_j \right)^2 \mathcal{N}(x_1 | 0, \sigma) \dots \mathcal{N}(x_N | 0, \sigma) dx_1 \dots dx_N. \quad (49)$$

Estimator Properties: Bias (4/5), Sample Variance

(copied from the previous slide:)

$$E[\sigma_{ML}^2] = \int \dots \int_{-\infty}^{\infty} \frac{1}{N} \sum_{i=1}^N \left(x_i - \frac{1}{N} \sum_{j=1}^N x_j \right)^2 \mathcal{N}(x_1|0,\sigma) \dots \mathcal{N}(x_N|0,\sigma) dx_1 \dots dx_N. \quad (49)$$

Next, we will use the identity

$$\frac{1}{N} \sum_{i=1}^N \left(x_i - \frac{1}{N} \sum_{j=1}^N x_j \right)^2 = \underbrace{\frac{1}{N} \sum_{i=1}^N x_i^2}_{T_1} - \underbrace{\left(\frac{\sum_{i=1}^N x_i}{N} \right)^2}_{T_2} \quad (50)$$

$E[x_k^2]$ is σ^2 by an analogous construction as used in the derivation of expected value of mean. Thus,

$$E[T_1] = E\left[\frac{1}{N} \sum_{i=1}^N x_i^2 \right] = \frac{1}{N} N \sigma^2 = \sigma^2. \quad (51)$$

Estimator Properties: Bias (5/5), Sample Variance

As for T_2 , it is sufficient to note that

$$E \left[\left(\sum_{i=1}^N x_i \right)^2 \right] = \sum_{i=1}^N \underbrace{E[x_i^2]}_{\sigma^2} - \sum_{i \neq j} \underbrace{E[x_i x_j]}_0 \quad (52)$$

Taking the two results together,

$$E[\sigma_{ML}^2] = \sigma^2 - \frac{1}{N}\sigma^2 = \frac{N-1}{N}\sigma^2. \quad (53)$$

The ML estimator for the variance σ^2 is thus *biased*.

The unbiased version of the variance estimator is

$$\frac{N}{N-1}\sigma_{ML}^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_{ML})^2, \quad (54)$$

where μ_{ML} is the sample mean $\mu_{ML} = \frac{1}{N} \sum_{i=1}^N x_i$.

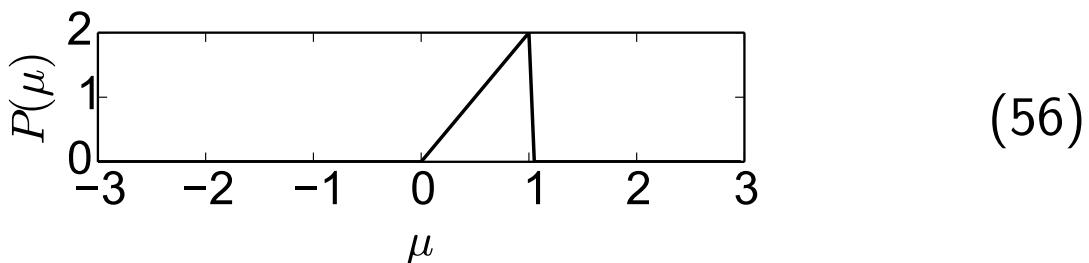
Example 3: Normal Distr. with Unknown μ (MAP), Non-conjugate prior (1)

Consider again the normal distribution $p(\mathcal{T} | \mu, \sigma^2)$, and let the variance σ^2 be known ($\sigma^2 = 1.$) Estimate the mean μ . As before, we have

$$L(\mu) = p(\mathcal{T} | \mu) = \prod_{i=1}^N p(x_i | \mu) = \frac{1}{\sqrt{(2\pi)^N}} \exp \left[-\frac{1}{2} \sum_{i=1}^N (x_i - \mu)^2 \right]. \quad (55)$$

Let us consider the prior on μ to have the form (note: this is clearly not a conjugate prior)

$$p(\mu) = \begin{cases} 2\mu, & 0 < \mu \leq 1 \\ 0, & \text{otherwise.} \end{cases}$$

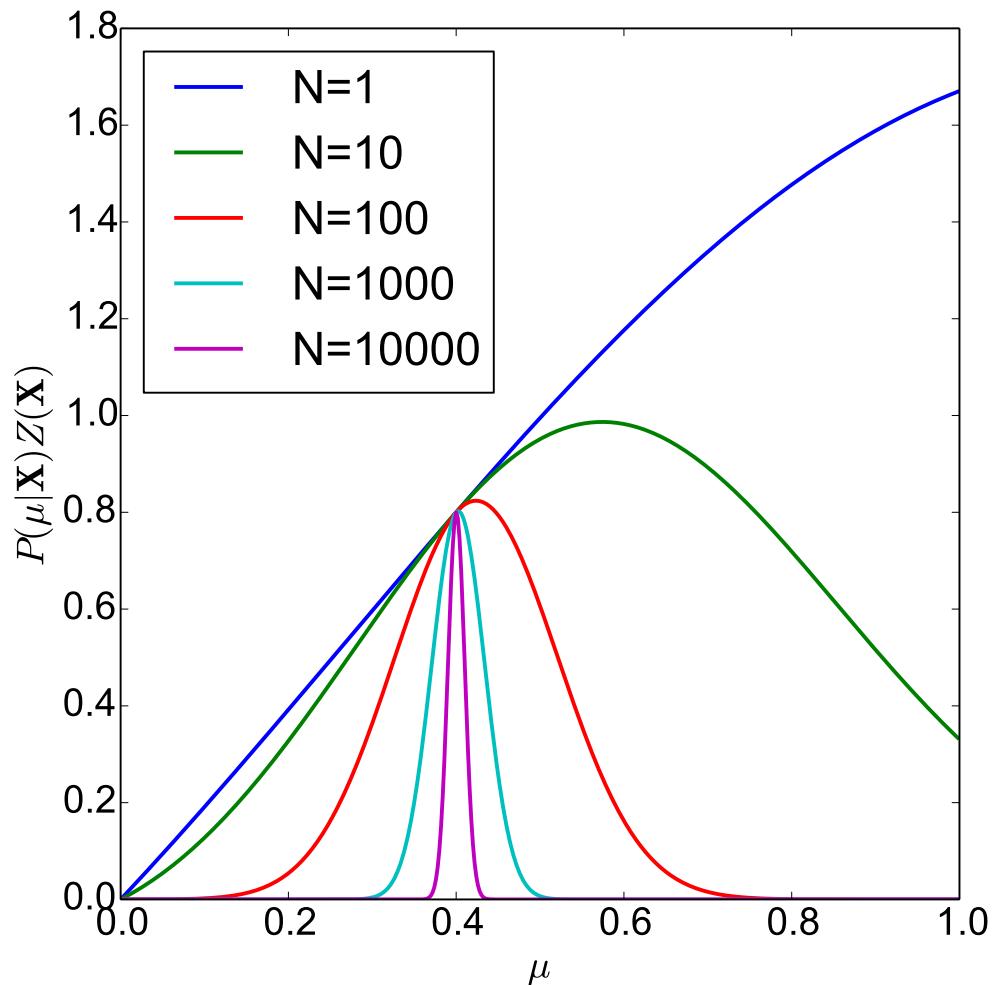


The MAP estimate of μ will be found as

$$\mu_{\text{MAP}} = \underset{\mu}{\operatorname{argmax}} p(\mathcal{T} | \mu) p(\mu). \quad (57)$$

Note that $p(\mathcal{T} | \mu) p(\mu)$ can attain maximum either inside the interval $0 < \mu < 1$, or at its border $\mu = 1$ (not at the other border $\mu = 0$, as $p(0) = 0.$)

Example 3: Normal Distr. with Unknown μ (MAP), Non-conjugate prior (2)



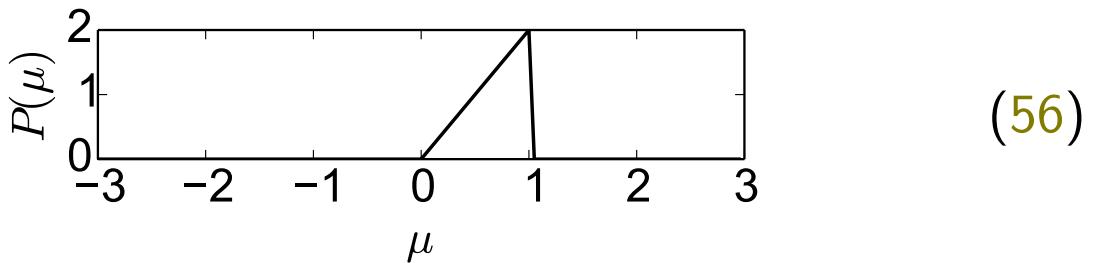
Left: $p(\mathcal{T} | \mu)p(\mu)$ evaluated for sample mean $\mu' = 0.4$ and increasing cardinality of the observation set \mathcal{T} . Note:

- ◆ variance of the distribution decreases as N grows;
- ◆ the distribution is quite close to the prior for $N = 1$;
- ◆ influence of the prior decreases with increasing N .

Example 3: Normal Distr. with Unknown μ (MAP), Non-conjugate prior (3)

$$L(\mu) = p(\mathcal{T}|\mu) = \frac{1}{\sqrt{(2\pi)^N}} \exp \left[-\frac{1}{2} \sum_{i=1}^N (x_i - \mu)^2 \right]. \quad (55)$$

$$p(\mu) = \begin{cases} 2\mu, & 0 < \mu \leq 1 \\ 0, & \text{otherwise.} \end{cases}$$



Taking the log of $p(\mathcal{T}|\mu)p(\mu)$ gives (for the interval $0 < \mu < 1$):

$$\ln p(\mathcal{T}|\mu)p(\mu) = \ln p(\mathcal{T}|\mu) + \ln p(\mu) = -N/2 \ln 2\pi - \frac{1}{2} \sum_{i=1}^N (x_i - \mu)^2 + \ln 2\mu \quad (58)$$

Taking the derivative w.r.t. μ and setting it to zero gives

$$\frac{\partial \ln p(\mathcal{T}|\mu)p(\mu)}{\partial \mu} = \sum_{i=1}^N (x_i - \mu) + \frac{1}{\mu} = 0, \quad (0 < \mu < 1). \quad (59)$$

Note that this is a decreasing function of μ and thus there can be **at most one** solution for μ in the considered interval.

Example 3: Normal Distr. with Unknown μ (MAP), Non-conjugate prior (4)

Denoting $S = \sum_{i=1}^N x_i$, this is rewritten as

$$S - N\mu + \frac{1}{\mu} = 0 \Rightarrow N\mu - \frac{1}{\mu} = S, \quad (0 < \mu < 1). \quad (60)$$

It is easily checked that for any S , $\mu > 0$ can be found such that this equation holds. Taken with the previous observation that there is at most 1 solution, there is exactly 1 solution for $\mu > 0$.

Multiplying by μ , we get

$$N\mu^2 - \mu S - 1 = 0. \quad (61)$$

The roots of this quadratic equation are

$$\mu = \frac{S \pm \sqrt{S^2 + 4N}}{2N} = \frac{1}{2N}S \pm \frac{1}{2}\sqrt{\frac{S^2}{N^2} + \frac{4}{N}} = \begin{cases} \mu^+ > 0 \\ \mu^- < 0 \end{cases}. \quad (62)$$

Only μ^+ (always > 0) can be the solution of Eq. (60) if $\mu^+ < 1$. The root μ^- can never be the solution to it, as it is always < 0 .

If $\mu^+ < 1$ then $\mu_{\text{MAP}} = \mu^+$. Otherwise the maximum is attained at the right border of the interval, and $\mu_{\text{MAP}} = 1$.

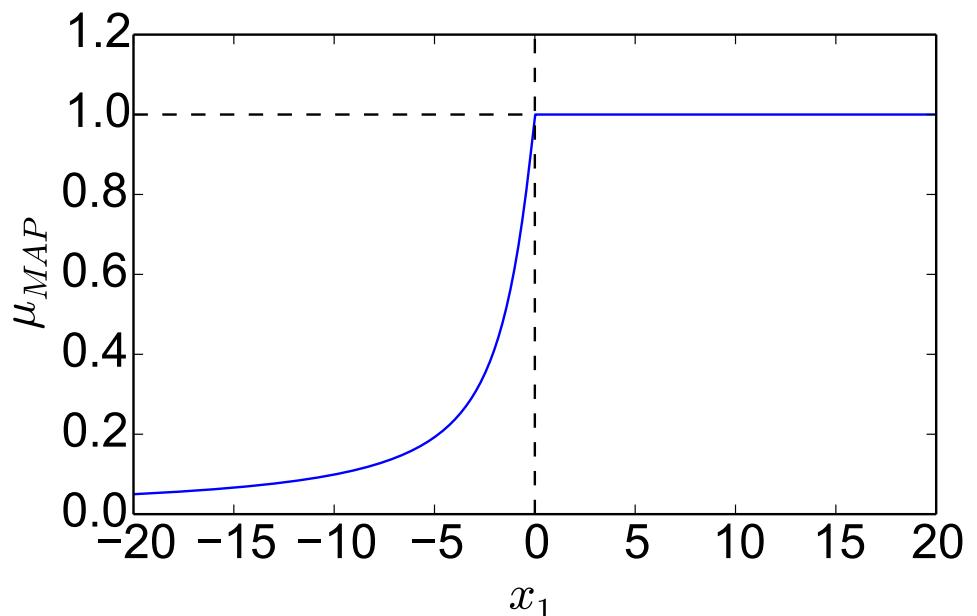
Example 3: Normal Distr. with Unknown μ (MAP), Non-conjugate prior (5)

In conclusion, the MAP solution is the following:

- ◆ Compute $S = \sum_{i=1}^N x_i$
- ◆ Compute $\mu^+ = \frac{1}{2N}S + \frac{1}{2}\sqrt{\frac{S^2}{N^2} + \frac{4}{N}}$.
- ◆ If $\mu^+ < 1$ then $\mu_{\text{MAP}} = \mu^+$ else $\mu_{\text{MAP}} = 1$.

Example: Consider the training set consisting of a single observation $\mathcal{T} = \{x_1\}$.

The estimate μ_{MAP} :

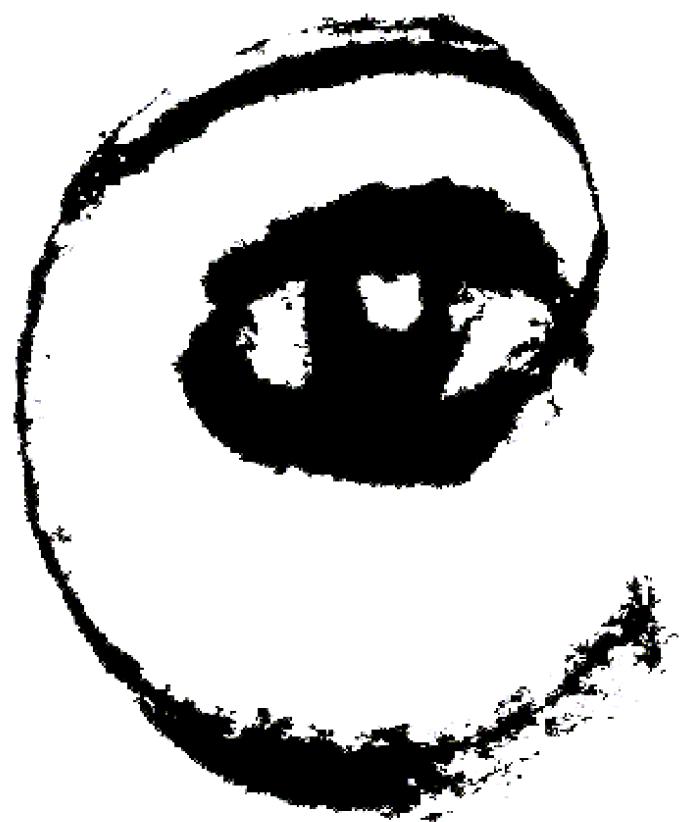


Note that for all $x_1 > 0$, $\mu_{\text{MAP}} = 1$.

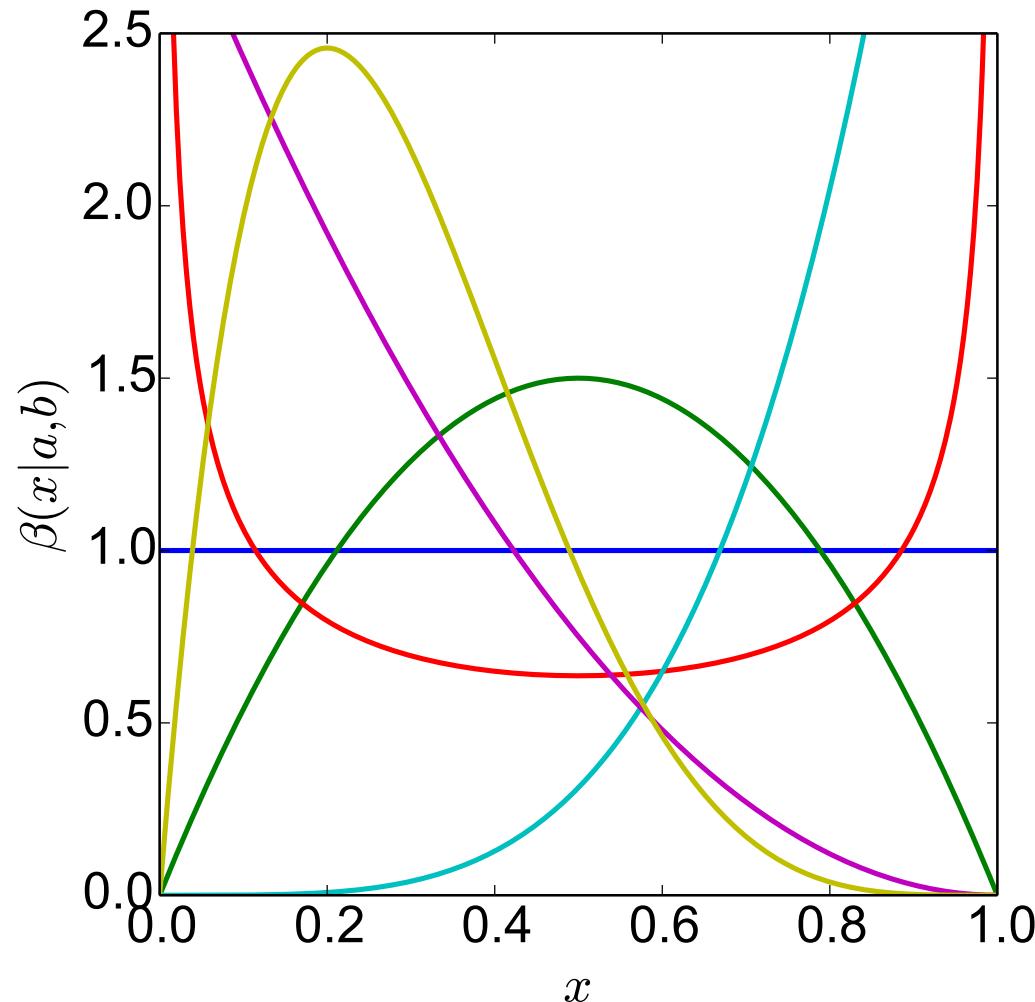
Also note that, as $N \rightarrow \infty$,

$$\mu^+ \rightarrow \frac{1}{2} \left(\frac{S}{N} + \frac{|S|}{N} \right). \quad (63)$$

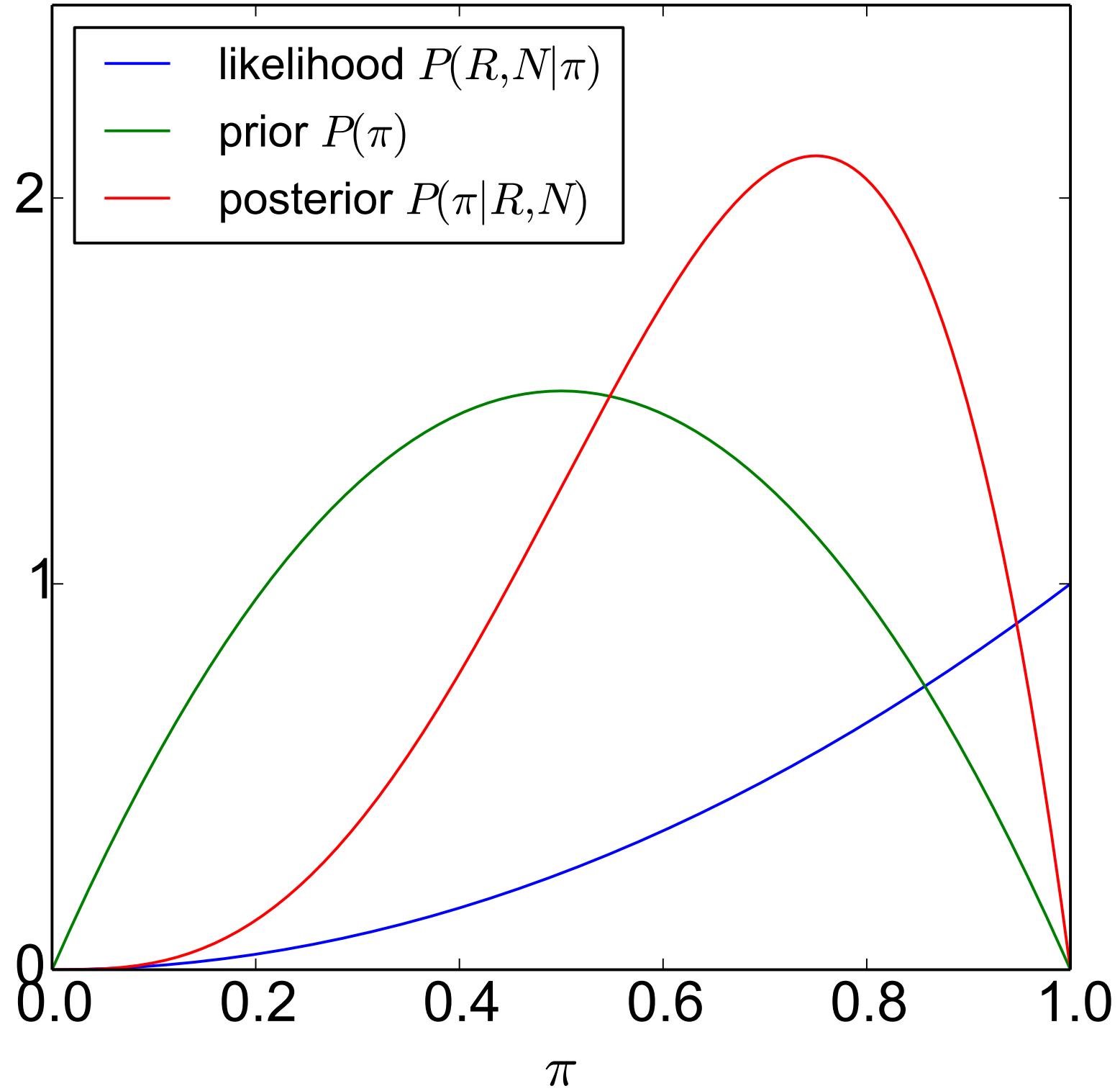
For a high number of observations and $S > 0$, the solution will converge to the ML solution $\hat{\mu} = \frac{\sum_{i=1}^N x_i}{N}$. This is the usual behavior of *MAP* vs. *ML*: The prior distribution of parameters (here μ) shifts the solution towards the values which are a priori more probable, but as evidence grows the influence of the prior shades away.

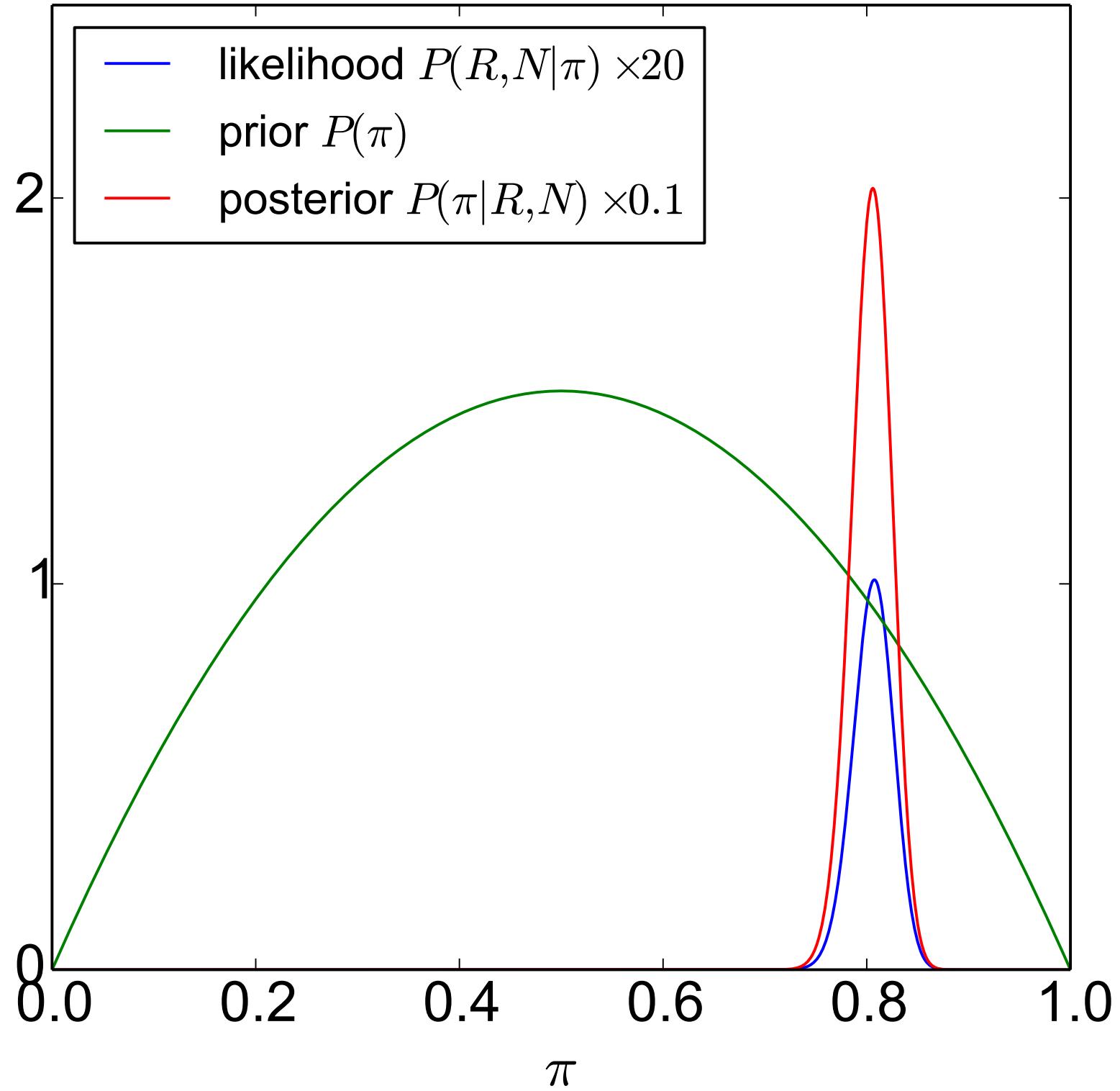


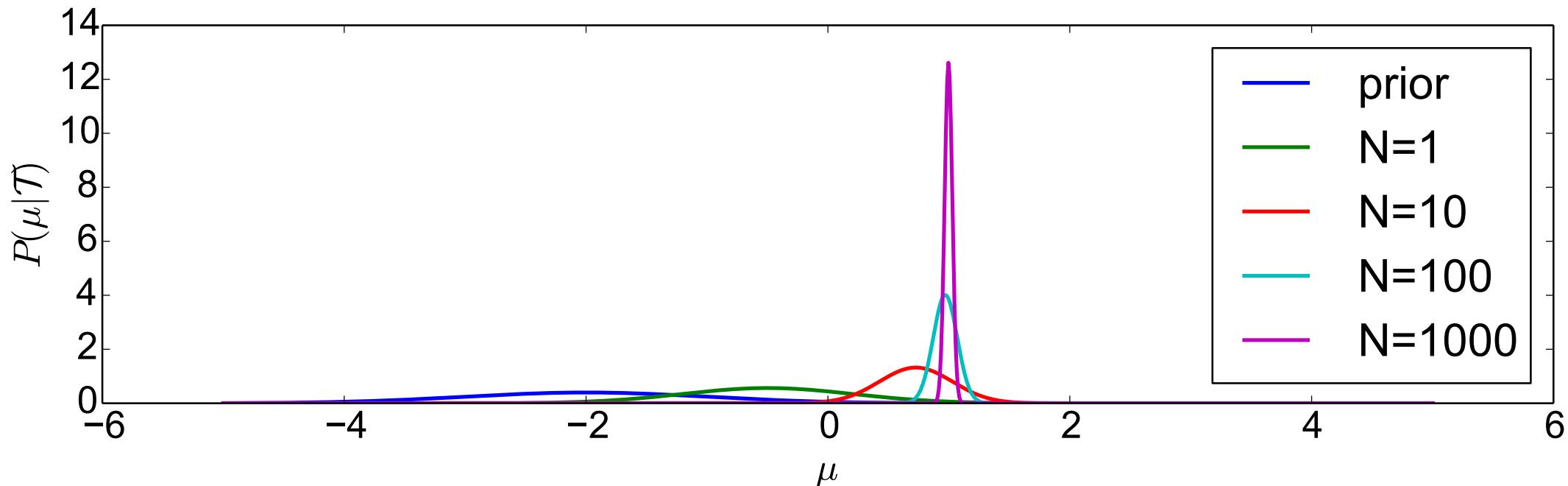
m p



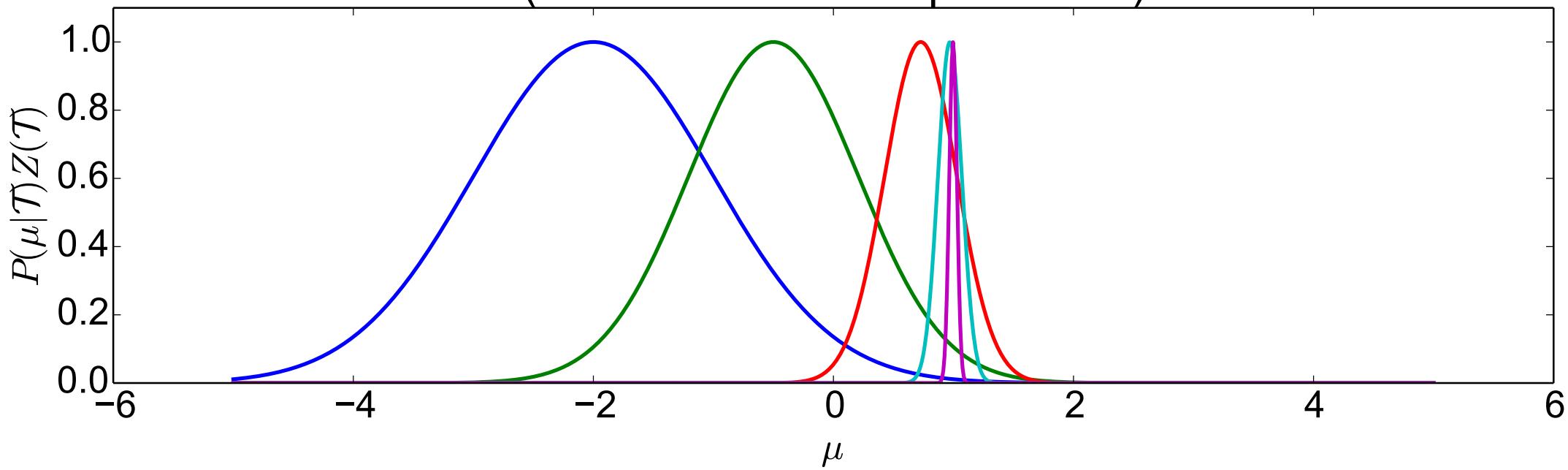
- $a = 1, b = 1$
- $a = 2, b = 2$
- $a = b = 0.5$
- $a = 5, b = 1$
- $a = 1, b = 3$
- $a = 2, b = 5$

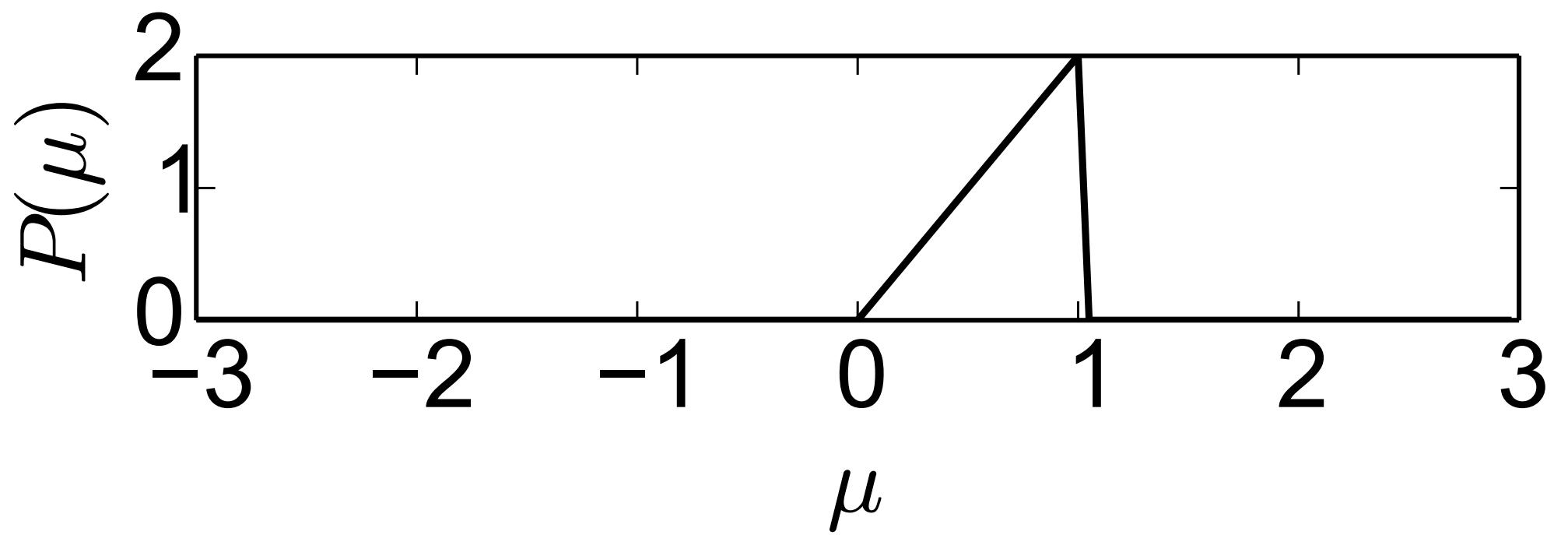


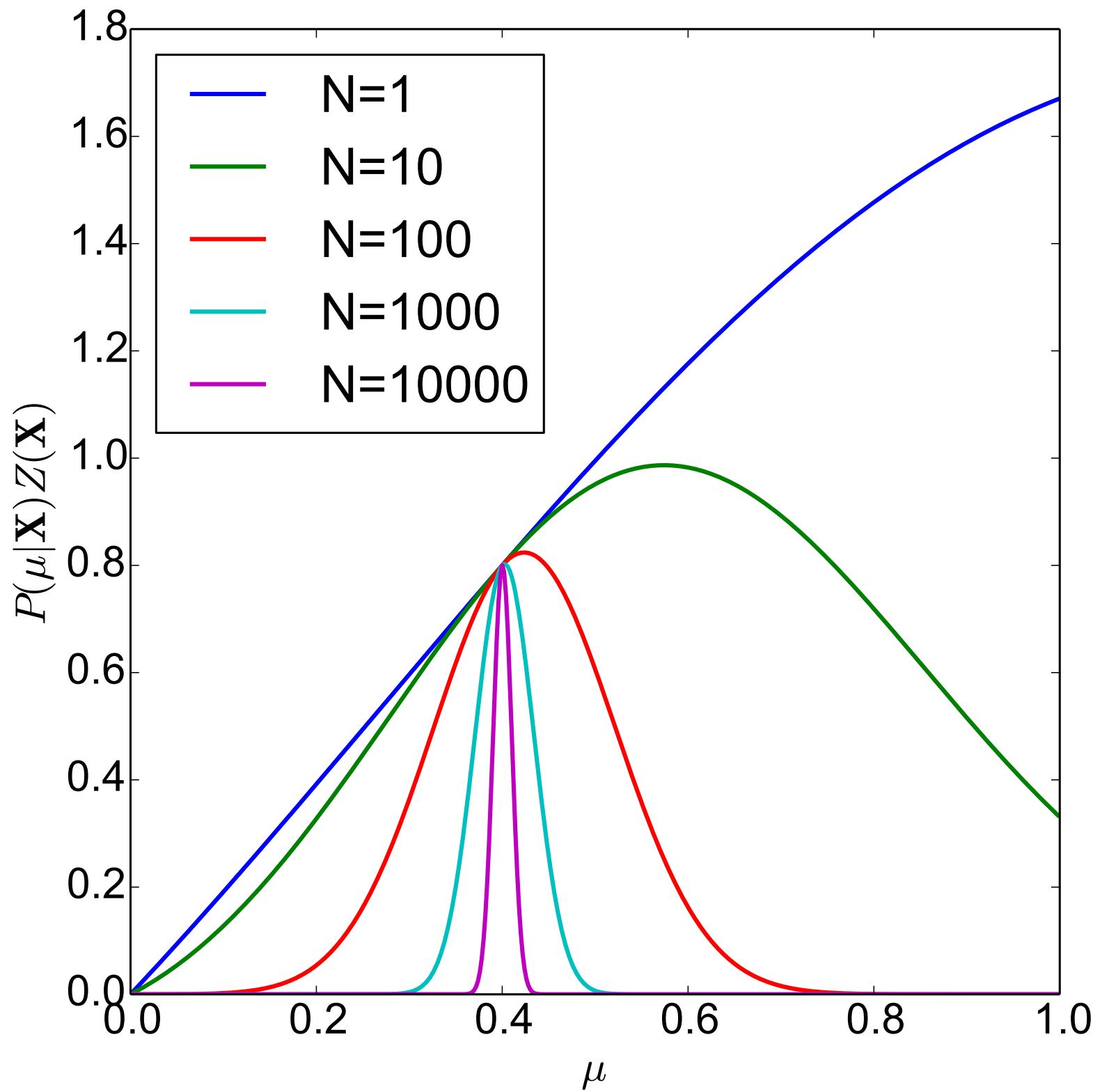


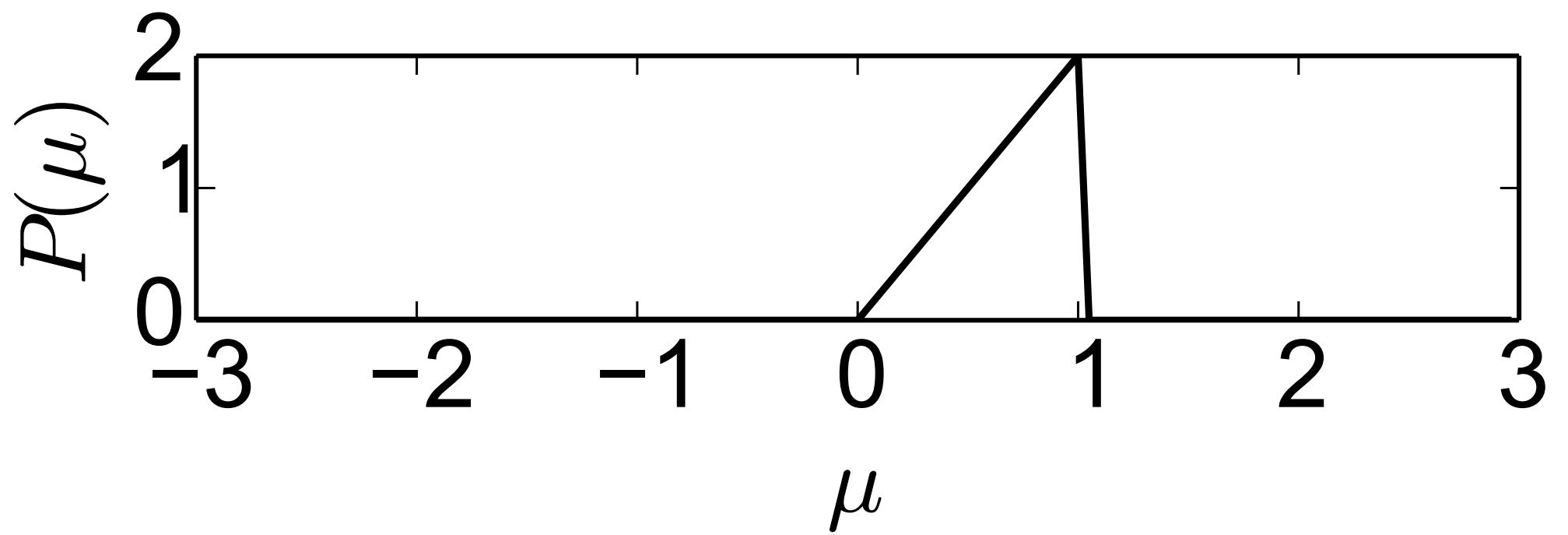


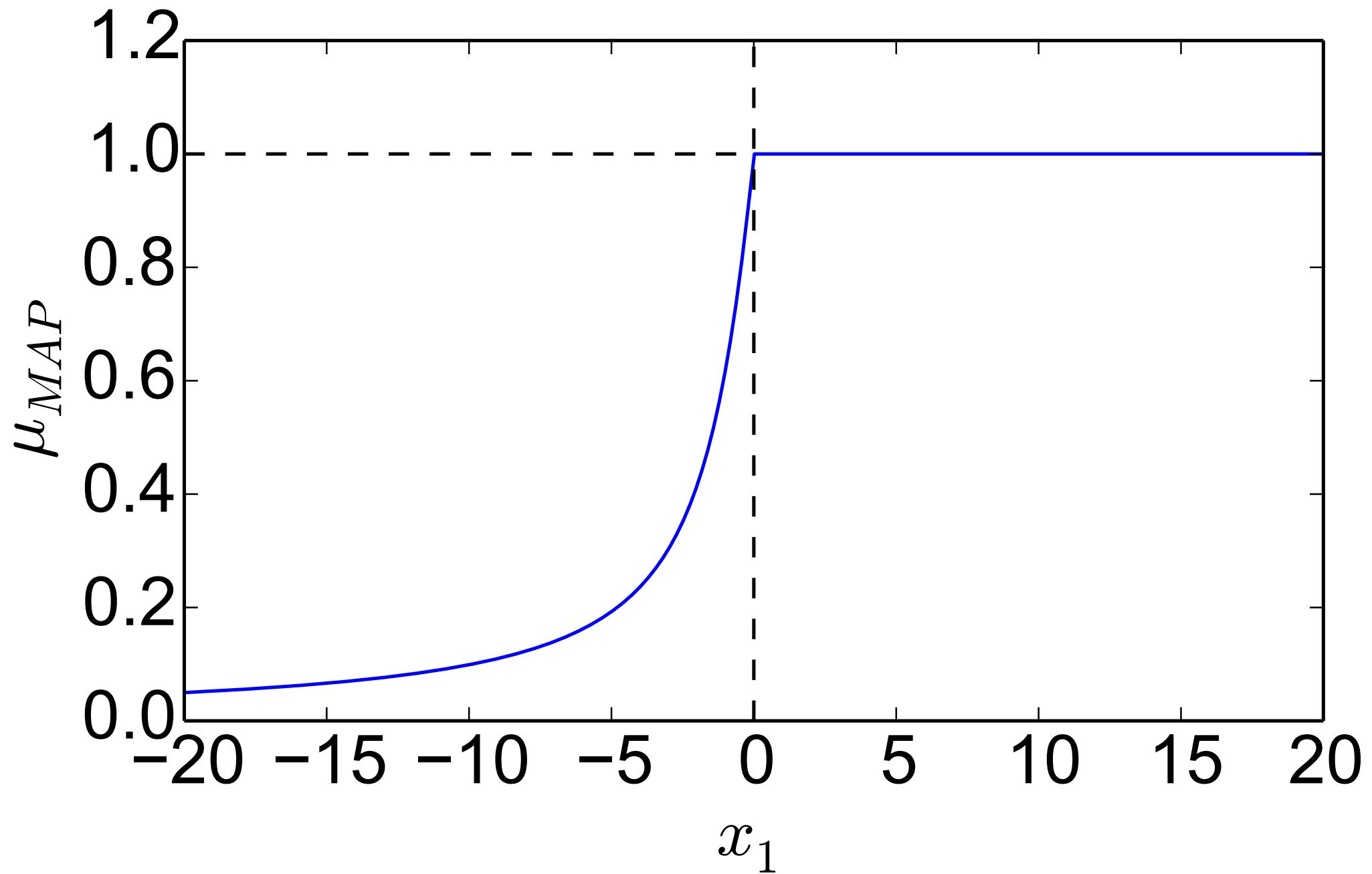
(scaled to the same peak level)











Nonparametric Methods for Density Estimation

Nearest Neighbour Classification

Lecturer:
Jiří Matas

Authors:
Ondřej Drbohlav, Jiří Matas

Centre for Machine Perception
Czech Technical University, Prague
<http://cmp.felk.cvut.cz>

Last update: 15.10.2020



Probability Density Estimation

Parametric Methods for Density Estimation

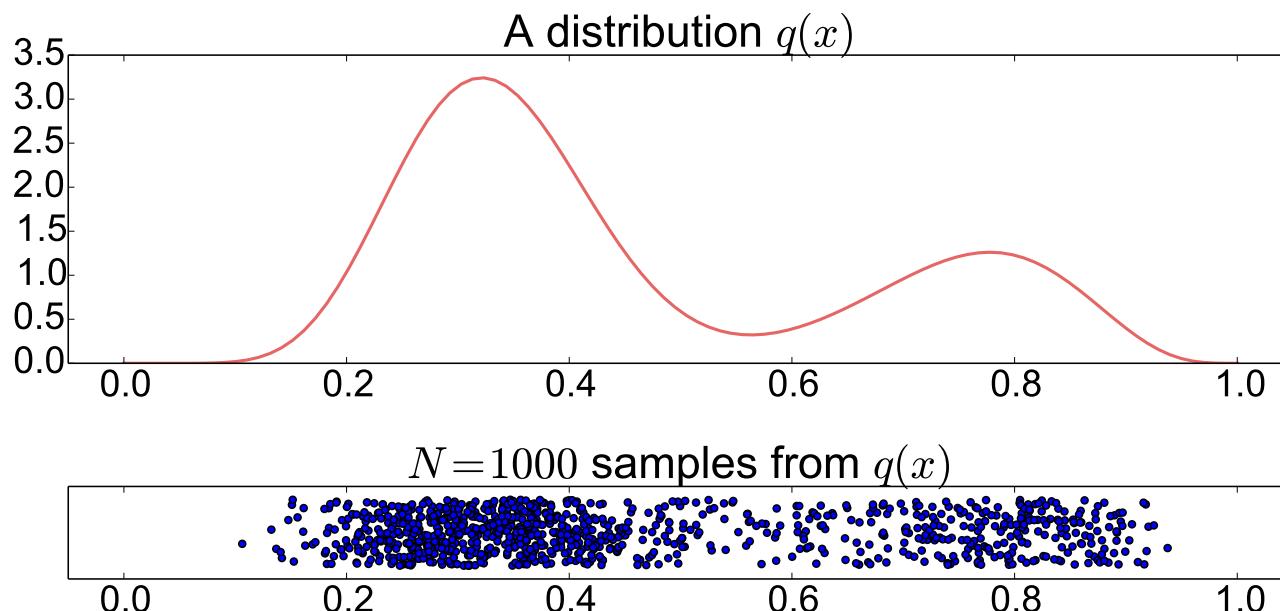
- ◆ Have been dealt with in the previous lecture
- ◆ Advantage: Low number of parameters to estimate
- ◆ Disadvantage: The resulting estimated density can be arbitrarily wrong if the underlying distribution does not agree with the assumed parametric model.

Non-Parametric Methods for Density Estimation

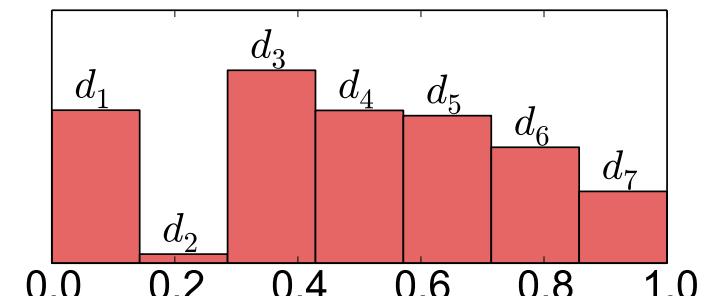
- ◆ Histogram
- ◆ Nearest Neighbor approach

Histogram as piecewise constant density estimate:

Consider the following distribution $q(x)$ on the interval $[0, 1]$, and i.i.d. sampling from it. We will fit the distribution by a 'histogram' with B bins. More precisely, we will estimate a piecewise-constant function on the interval $[0, 1]$ with B segments of the same width. For a given B , the parameters of this piecewise-constant function are the heights d_1, d_2, \dots, d_B of the individual bins. This function is denoted $p(x|\{d_1, d_2, \dots, d_B\})$.



$p(x|\{d_1, d_2, \dots, d_B\})$ to be estimated



For the given number of bins B , d_1, d_2, \dots, d_B must conform to the constraint that the area under the function must sum up to one,

$$1 = \int_{-\infty}^{\infty} p(x|\{d_1, d_2, \dots, d_B\}) dx = \sum_{i=1}^B \int_{\frac{i-1}{B}}^{\frac{i}{B}} d_i dx = \sum_{i=1}^B d_i w = \sum_{i=1}^B \frac{d_i}{B}. \quad (1)$$

bin width

Histogram as piecewise constant density estimate:

Finding d_i 's using Maximum Likelihood

Let us estimate $\{d_i, i = 1, 2, \dots, B\}$ by Maximum Likelihood (ML) approach. Let N_i denote the number of samples which belong the i -th bin (thus clearly, $\sum_{i=1}^B N_i = N$). The likelihood $L(\mathcal{T})$ of observing the samples $\mathcal{T} = \{x_1, x_2, \dots, x_N\}$ given the parameters $\theta = \{d_1, d_2, \dots, d_B\}$ is

$$L(\mathcal{T}) = p(\mathcal{T}|\theta) = \prod_{i=1}^N p(x_i|\theta) = \prod_{j=1}^B \overbrace{\left(\prod_{k=1}^{N_j} d_j \right)}^{\text{points in } j\text{-th bin}} = \prod_{j=1}^B d_j^{N_j}. \quad (2)$$

The maximization task is then

$$\ell(\mathcal{T}) = \sum_{j=1}^B N_j \log d_j \rightarrow \max, \quad \text{subject to } \frac{1}{B} \sum_{j=1}^B d_j = 1, \quad (3)$$

where maximization has been formulated using the log-likelihood $\ell(\mathcal{T})$. The Lagrangian of the optimization task and the conditions of optimality (using the derivative $\partial/\partial d_k$) are then:

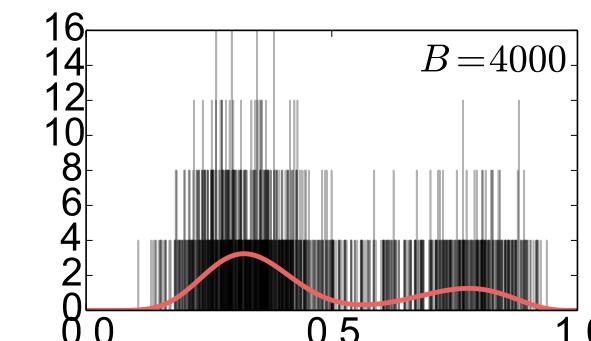
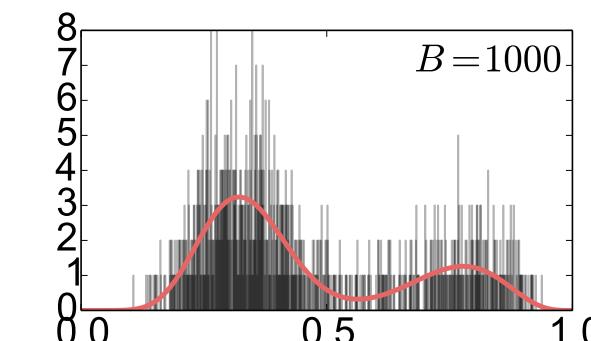
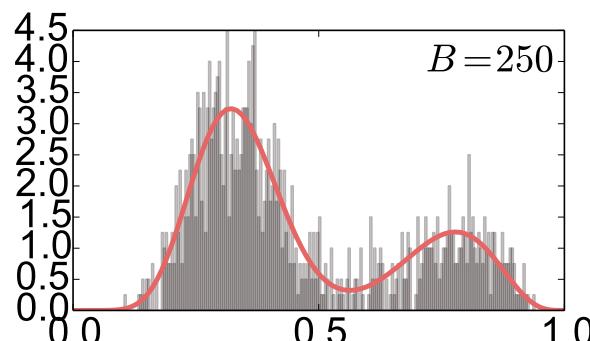
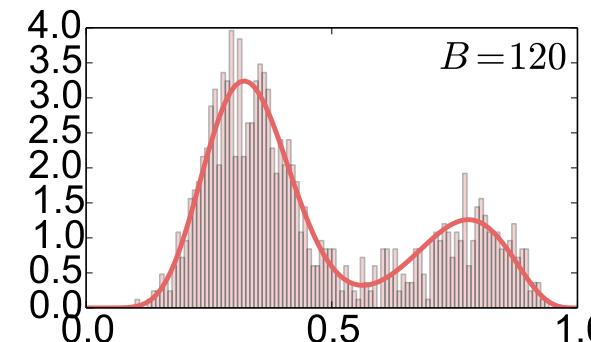
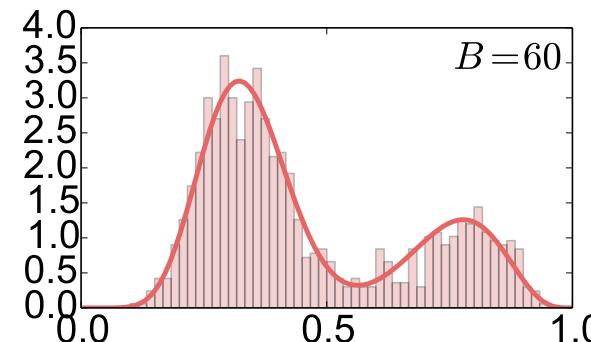
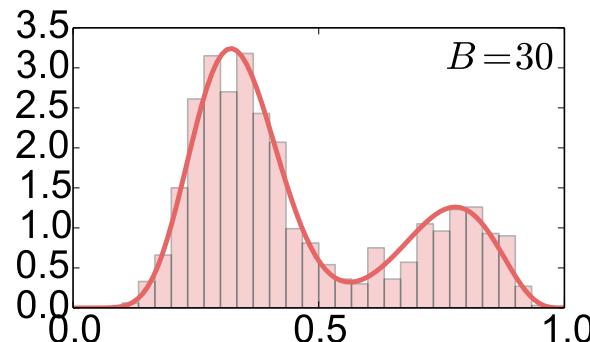
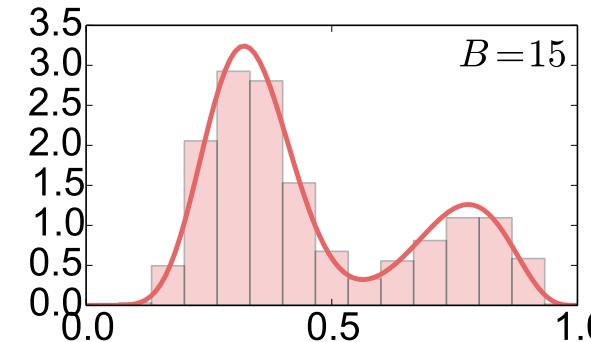
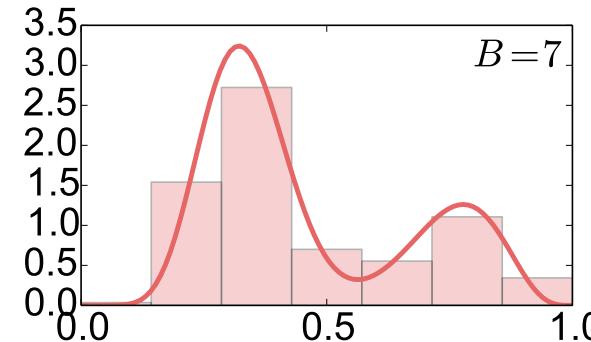
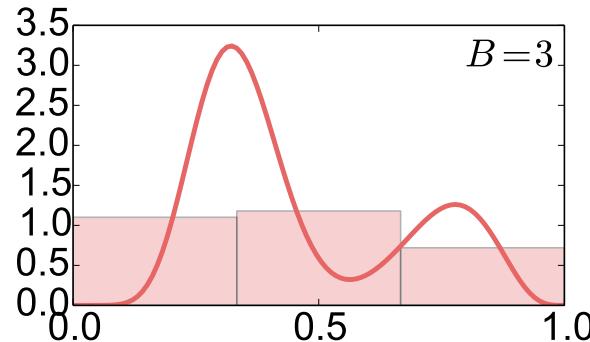
$$\text{Lagrangian: } \sum_{j=1}^B N_j \log d_j + \lambda \left(\frac{1}{B} \sum_{j=1}^B d_j - 1 \right) \quad (4)$$

$$\frac{N_k}{d_k} + \frac{\lambda}{B} = 0 \Rightarrow \frac{d_k}{N_k} = \text{const.} \Rightarrow d_k = B \frac{N_k}{N}. \quad (5)$$

Histogram as piecewise constant density estimate: Example, different number of bins

$$d_k = B \frac{N_k}{N} \quad (6)$$

This result is in line with the common use of histograms for approximating pdf's. Results for different B 's:



Histogram as piecewise constant density estimate: What number of bins produces closest pdf approximation?

Let us measure the differences between the (actual) source distribution $q(x)$ and the piecewise-constant density estimate $p(x) = p(x|\{d_1, d_2, \dots, d_B\})$ from the $N = 1000$ samples, using B bins.

Measures used:

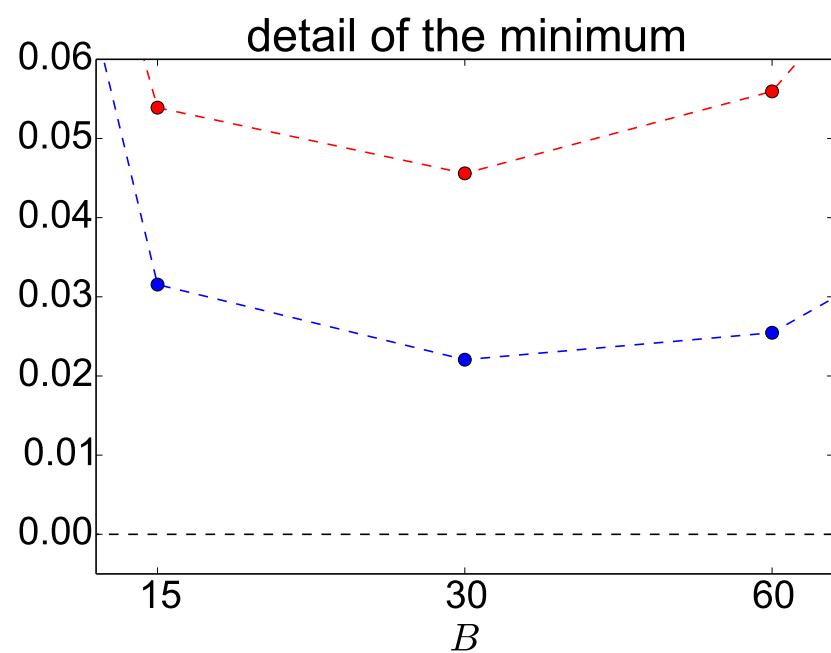
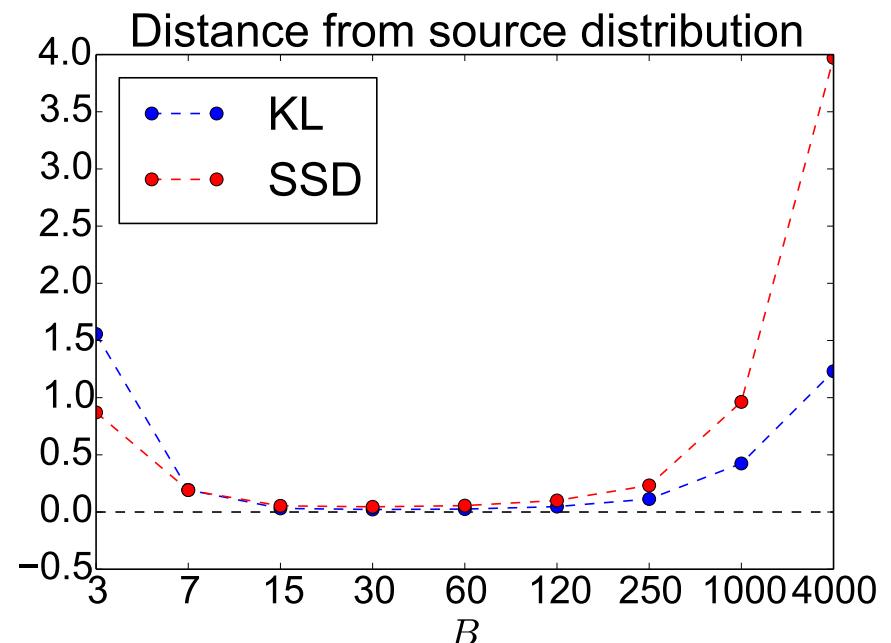
Kullback-Leibler divergence D_{KL} :

$$D_{KL}(p||q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx. \quad (7)$$

(Note that KL div. is not a metric.)

Sum of squared differences D_{SSD} :

$$D_{SSD}(p, q) = \int_{-\infty}^{\infty} (p(x) - q(x))^2 dx. \quad (8)$$



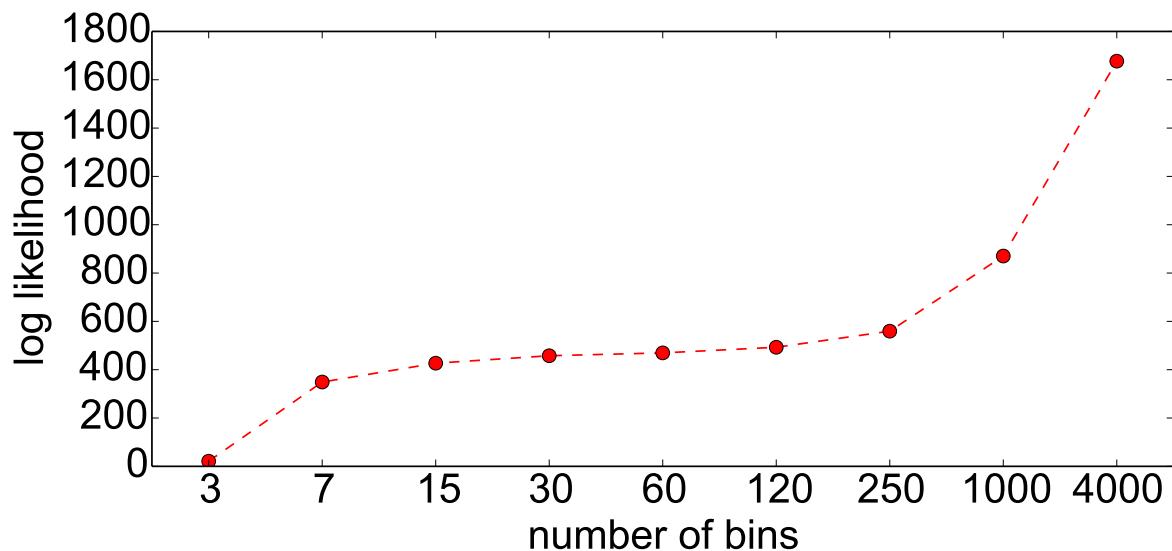
Histogram as piecewise constant density estimate: Choosing the number of bins B by ML

How can we find the optimal number of bins B ? Let us try to employ the ML approach again: find the B which maximizes the likelihood. Recall that:

$$\text{parameters } d_j : \quad d_j = B \frac{N_j}{N} \quad (\text{ML estimate}) \quad (9)$$

$$\text{likelihood } L(\mathcal{T}) : \quad L(\mathcal{T}) = p(\mathcal{T} | \{d_1, d_2, \dots, d_B\}) = \prod_{j=1}^B d_j^{N_j} = \prod_{j=1}^B \left(\frac{BN_j}{N} \right)^{N_j} \quad (10)$$

$$\text{log-likelihood } \ell(\mathcal{T}) : \quad \ell(\mathcal{T}) = \sum_{j=1}^B N_j \log d_j = \sum_{j=1}^B N_j \log \frac{BN_j}{N} \quad (11)$$



For $B = 4000$, the log-likelihood ℓ is the highest. But the pdf estimate with this B is poor, and very different from the source distribution as measured by D_{KL} or D_{SSD} . For $B = 10^5$, $\ell(\mathcal{T}) \sim 4600$. What went wrong?

Histogram, choosing the number of bins B :

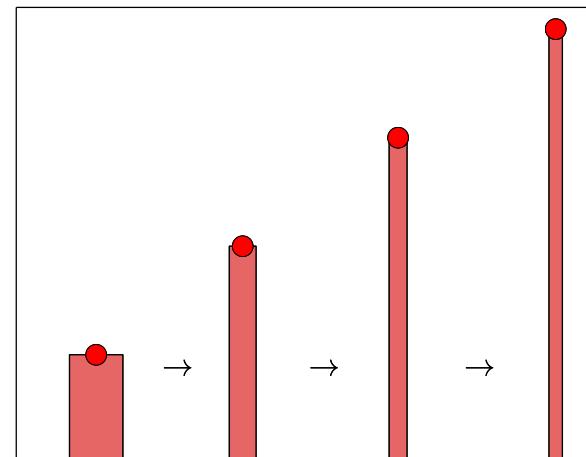
ML overfits and produces $B = \infty$

When B grows, eventually it will reach a number \hat{B} such that there is either *no* or *one* point in every bin (assuming no two points in the data are identical), and this will stay true for any $B > \hat{B}$.

In such cases,

$$d_j = \begin{cases} \frac{B}{N} & \text{if the bin is populated by a point,} \\ 0 & \text{if the bin is not populated.} \end{cases} \quad (12)$$

As the number of bins B grows, the widths of occupied bins get narrower and the heights d_j 's higher. If $B \rightarrow \infty$ then also $d_j \rightarrow \infty$ for the occupied bins, and therefore also $\ell(\mathcal{T}) \rightarrow \infty$. Thus, such an approach cannot produce a “reasonable” answer to choosing B , as the solution it provides is $B = \infty$.



The problem is that the log-likelihood ℓ is computed using the same data used for fitting the model (computing d_i 's). This is a similar concept to training a classifier on certain data and testing on the same data, which is prone to over-fitting and poor generalization.

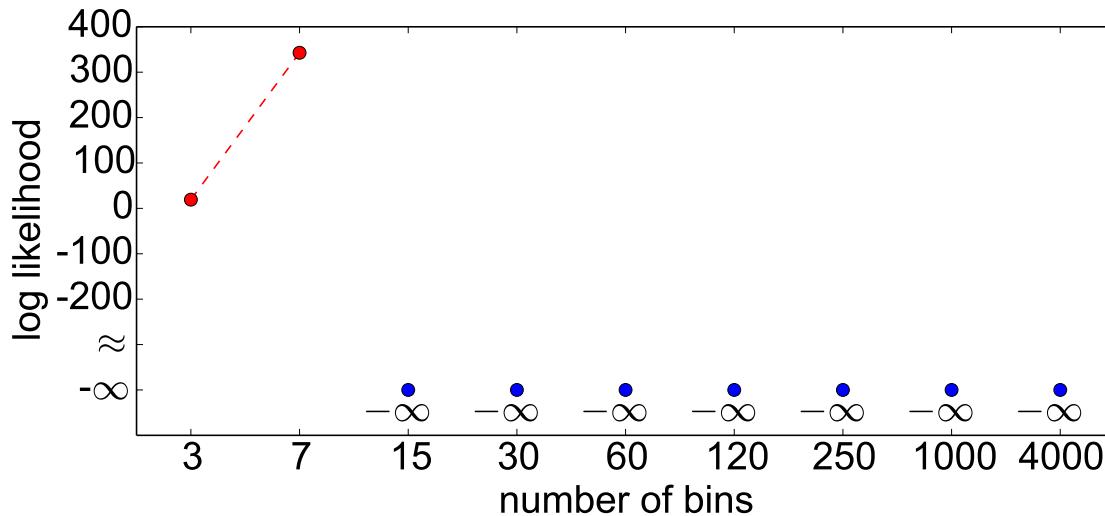
Histogram, choosing the number of bins B : Employing cross-validation

Let us compute the log likelihood using the following procedure: remove a given point from the dataset for computing d_i 's and evaluate its contribution to the log-likelihood. Do this for all the points. This approach is related to cross-validation technique (leave-one-out) for choosing parameters of a classifier.

Let the point in question belong to the j -th bin. The ML estimate for d_j , after removing this point from the dataset, is

$$d_j = B \frac{N_j - 1}{N - 1}, \quad (N_j \geq 1), \quad (13)$$

where the subtractions of 1 reflect the fact that the considered point is not used for estimating d_j . Computing the log likelihood ℓ this way produces the following result:



$$\ell = \sum_{\substack{j=1 \\ N_j \geq 1}}^B N_j \log d_j,$$

$$\text{with } d_j = B \frac{N_j - 1}{N - 1}$$

The 'failure' for $B > 7$ is caused by singly-occupied bins ($N_j = 1$) for which the modified ML estimate for d_j becomes zero. This will be fixed by using different estimates for d_j 's.

Histogram, choosing the number of bins B :

More suitable estimates for d_j 's

The problem of d_i being estimated as 0 is similar to the one encountered previously: Recall the example of tossing a coin three times, always getting *heads* ($\mathcal{T} = \{H, H, H\}$). The ML estimate is a fully unfair coin (probability of getting *heads* is 1, $\pi_{\text{head}} = 1$), thus making the likelihood of any sequence containing *tails* zero. We have seen before that employing the prior for the parameters to be estimated can mitigate this problem.

A (conjugate) prior for the histogram bin counts is the Dirichlet Distribution, with the pdf $p(d_1, d_2, \dots, d_B | \alpha_1, \alpha_2, \dots, \alpha_B) \sim \prod d_i^{\alpha_i - 1}$.

MAP Estimate:

Bayes Estimate:

$$d_i = B \frac{N_i + \alpha_i - 1}{N + \sum_{i=1}^B \alpha_i - B} \quad (14)$$

$$d_i = B \frac{N_i + \alpha_i}{N + \sum_{i=1}^B \alpha_i} \quad (15)$$

Interpretation: The parameters α_i 's can be interpreted as 'virtual' observations, as if α_k points have already been assigned to the k -th bin.

Example: The Bayes estimate using $\alpha_i = 1$ for all $i = 1, 2, \dots, B$ is

$$d_i = B \frac{N_i + 1}{N + B}. \quad (16)$$

Using this estimate will enable us to make reasonable computation of likelihood for all B 's.

Histogram, choosing the number of bins B :

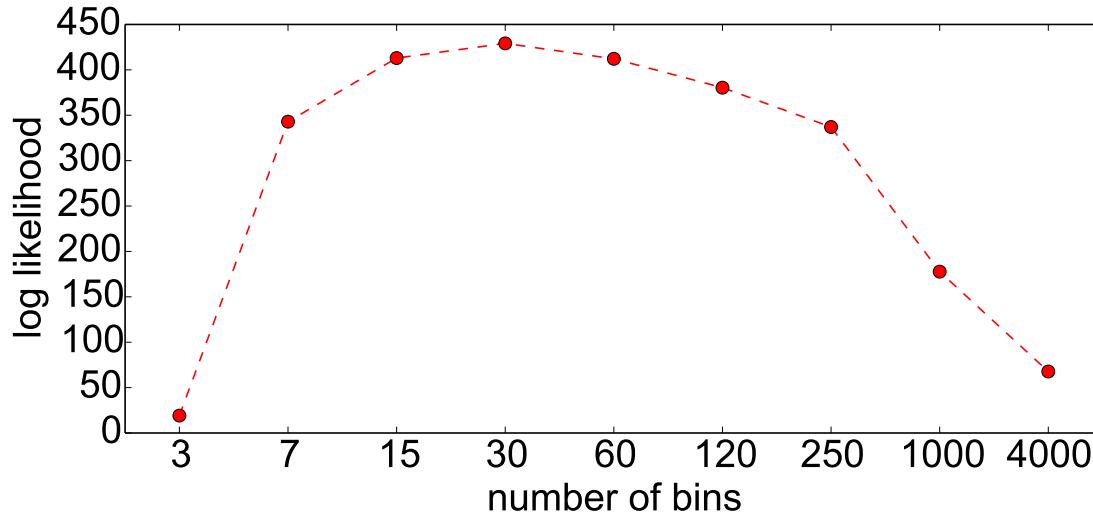
ML to find B , cross-validation, Bayes estimate for d_j 's

Let us now return to the previous task. Compute the log likelihood using the following procedure: remove a given point from the dataset for computing d_i 's and evaluate its contribution to the log-likelihood. Do this for all the points.

Use the Bayes estimate for d_j from the previous example, $d_j = B \frac{N_j+1}{N+B}$. The modified estimation of d_j (omitting the point in question) will become

$$d_j = B \frac{N_j}{N - 1 + B}. \quad (17)$$

This leads to the following result:



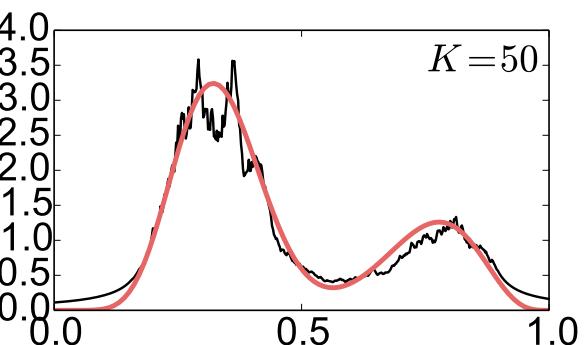
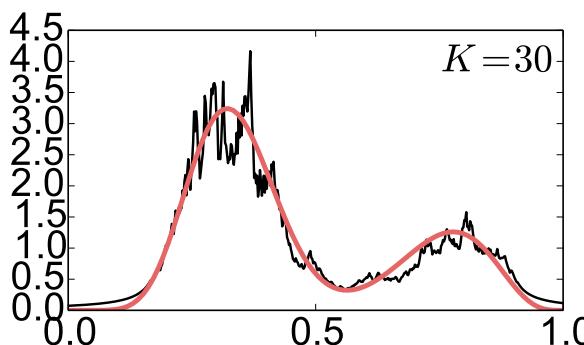
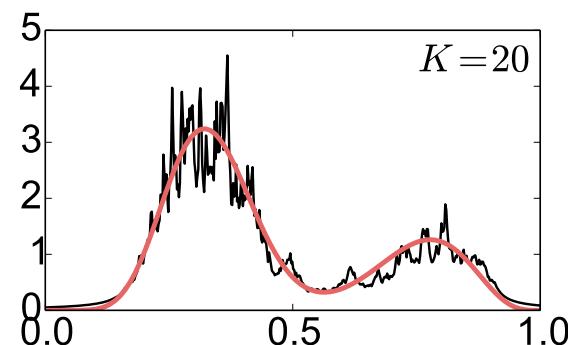
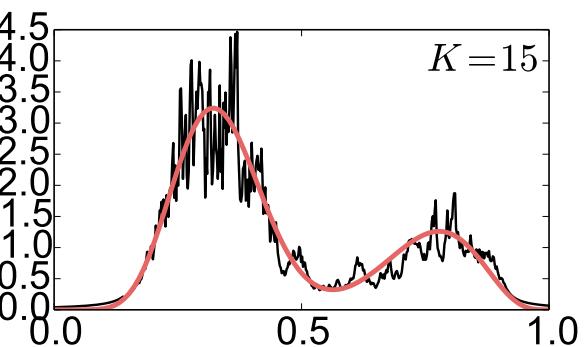
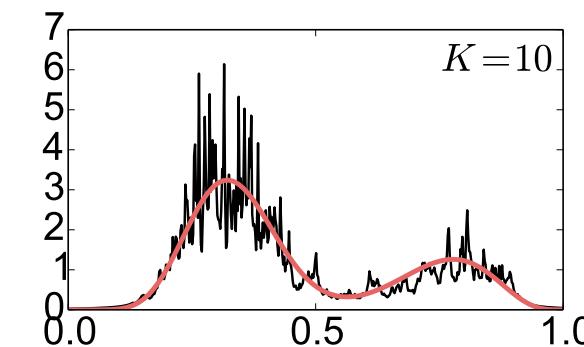
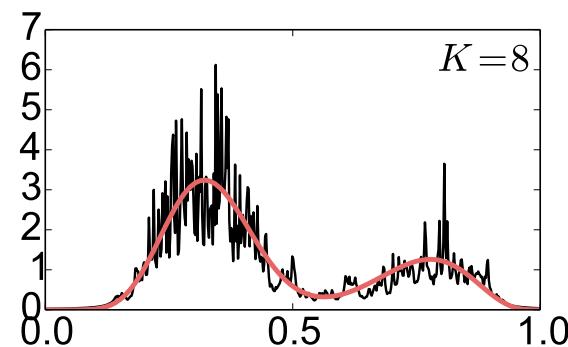
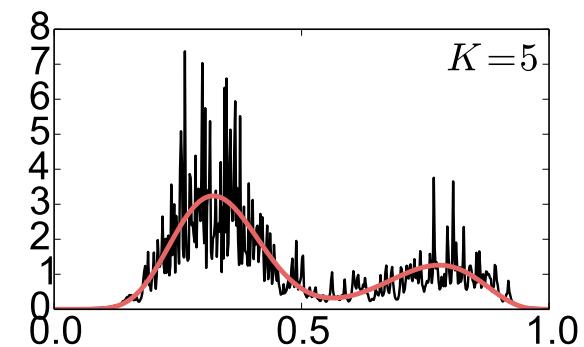
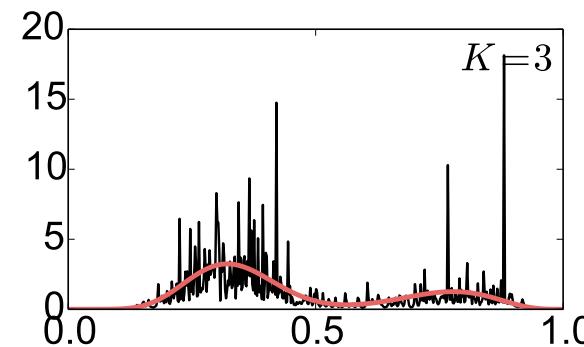
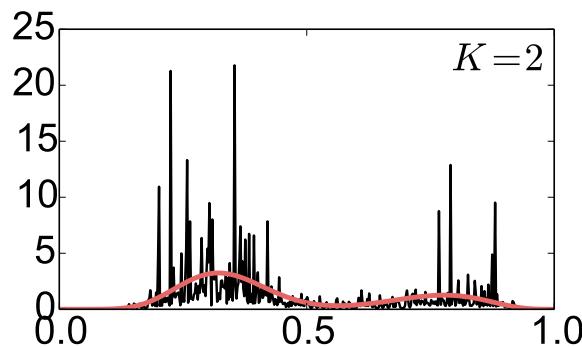
$$\ell = \sum_{j=1}^B N_j \log d_j,$$

$$\text{with } d_j = B \frac{N_j}{N - 1 + B}$$

This result is in agreement with distribution differences as measured by D_{KL} or D_{SSD} . In particular, $B = 30$ is identified as the best-approximating number of bins.

K -Nearest Neighbor Approach to Density Estimation

Find K neighbors, the density estimate is then $p \sim 1/V$ where V is the volume of a minimum cell containing K NNs. Example ($p \sim$ inverse distance to K -th NN, same 1000 samples as before):



***K*-Nearest Neighbor Approach to Classification**

Outline:

- ◆ Definition
- ◆ Properties
- ◆ Asymptotic error of NN classifier
- ◆ Error reduction by edit operation on the training class
- ◆ Fast NN search

K -NN Classification Definition

Assumption:

- ◆ Training set $\mathcal{T} = \{(x_1, k_1), (x_2, k_2), \dots, (x_N, k_N)\}$. There are R classes (letter K is reserved for K -NN in this lecture)
- ◆ A distance function $d : X \times X \mapsto \mathbb{R}_0^+$

Algorithm:

1. Given x , find K points $S = \{(x'_1, k'_1), (x'_2, k'_2), \dots, (x'_K, k'_K)\}$ from the training set \mathcal{T} which are closest to x in the metric d :

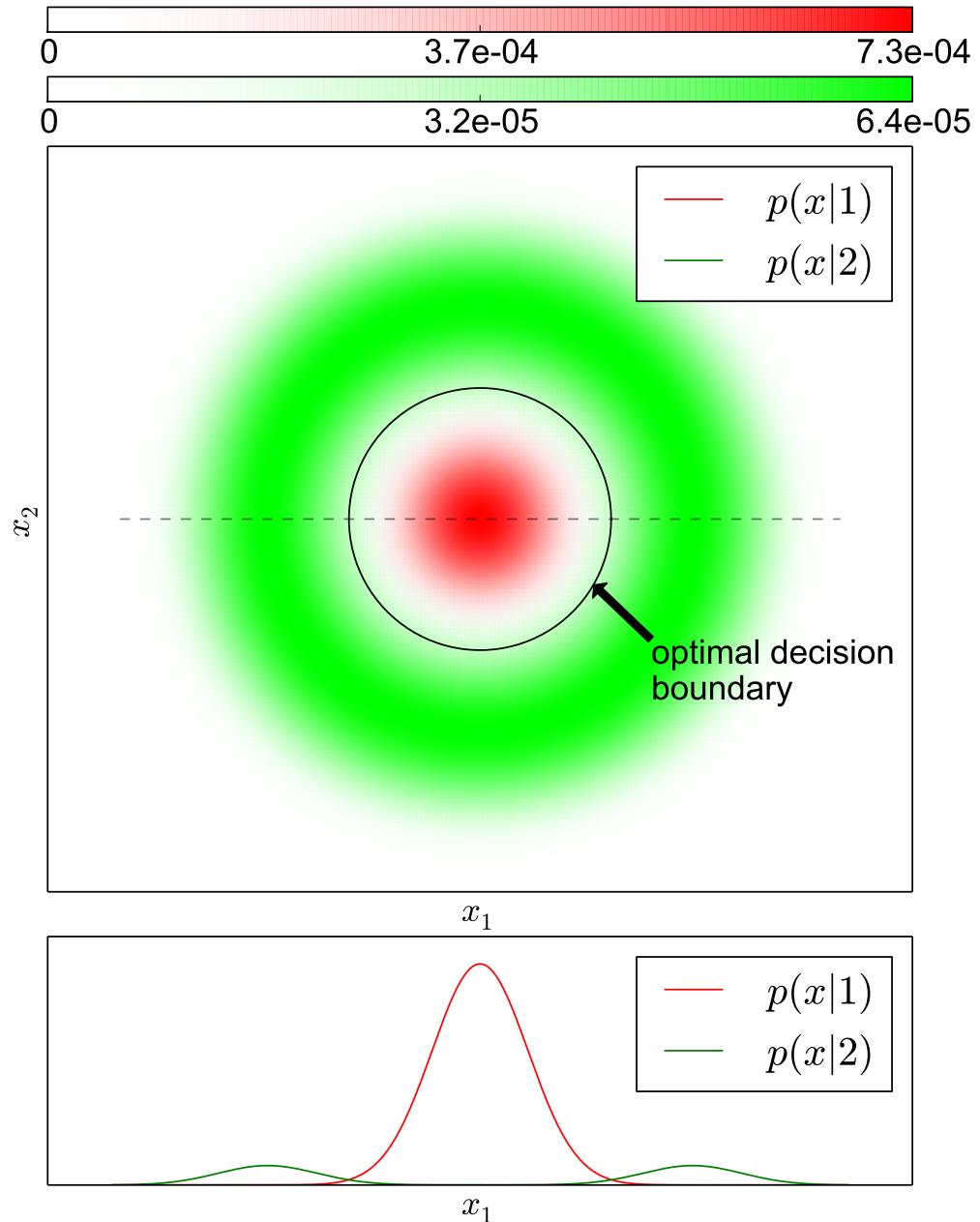
$$S = \{(x'_1, k'_1), (x'_2, k'_2), \dots, (x'_K, k'_K)\} \equiv \{(x_{r_1}, k_{r_1}), (x_{r_2}, k_{r_2}), \dots, (x_{r_K}, k_{r_K})\} \quad (18)$$

r_i : the rank of $(x_i, k_i) \in \mathcal{T}$ as given by the ordering $d(x, x_i)$ (19)

2. Classify x to the class k which has majority in S :

$$k = \operatorname{argmax}_{l \in R} \sum_{i=1}^K \llbracket k'_i = l \rrbracket \quad (x'_i, k'_i) \in S \quad (20)$$

K -NN Example (1)



Consider the two distributions shown. The priors are assumed to be the same,

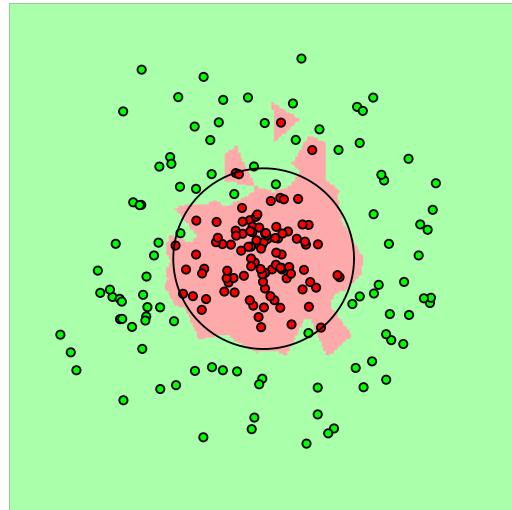
$$p(1) = p(2) = 0.5.$$

Bayesian optimal decision boundary is shown by the black circle.

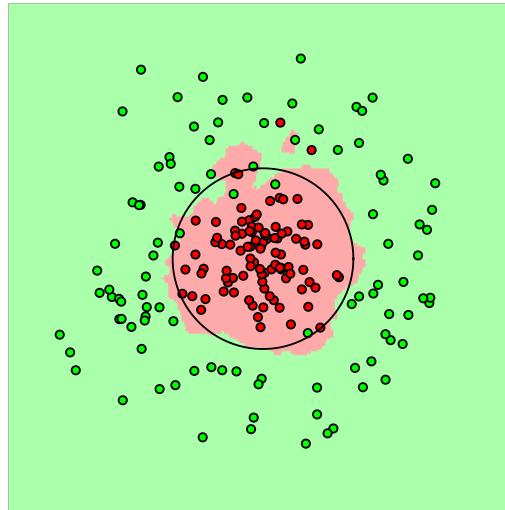
Bayesian error is $\epsilon_B = 0.026$.

K -NN Example (2)

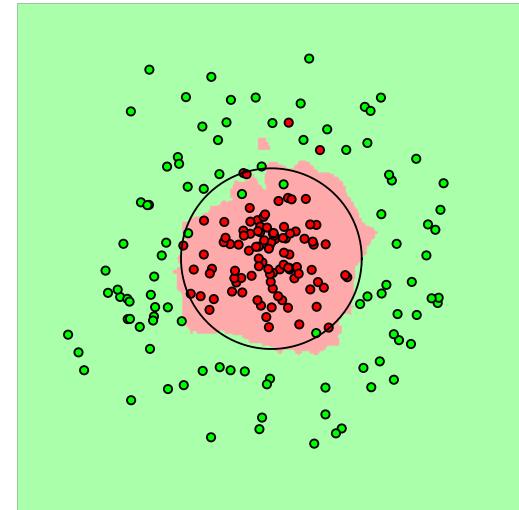
$K = 1$, error $\epsilon = 0.044$



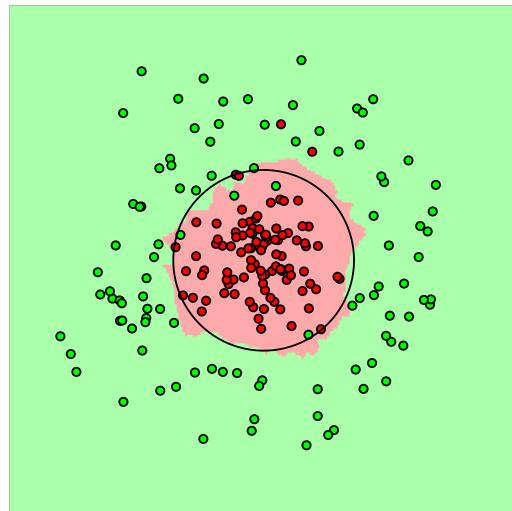
$K = 3$, error $\epsilon = 0.034$



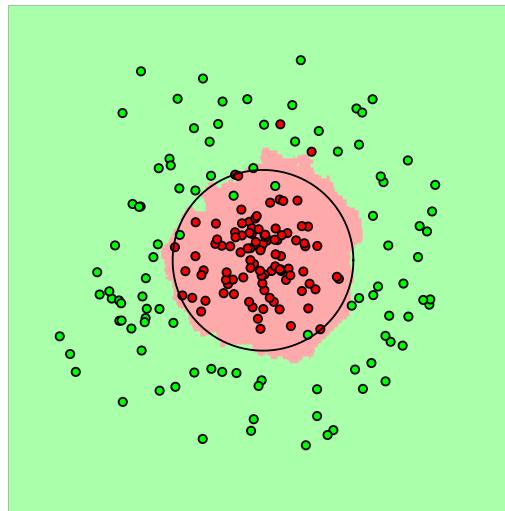
$K = 5$, error $\epsilon = 0.032$



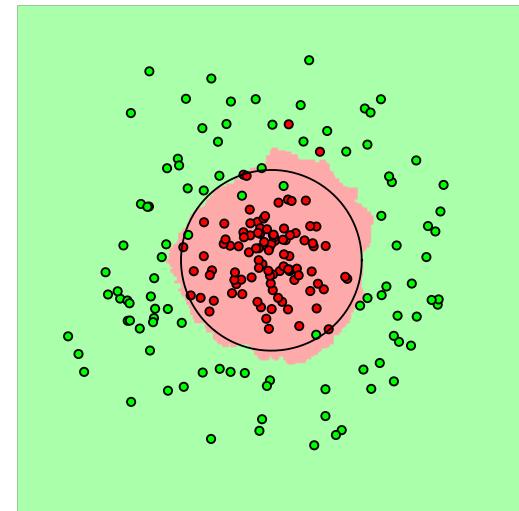
$K = 7$, error $\epsilon = 0.030$



$K = 9$, error $\epsilon = 0.031$

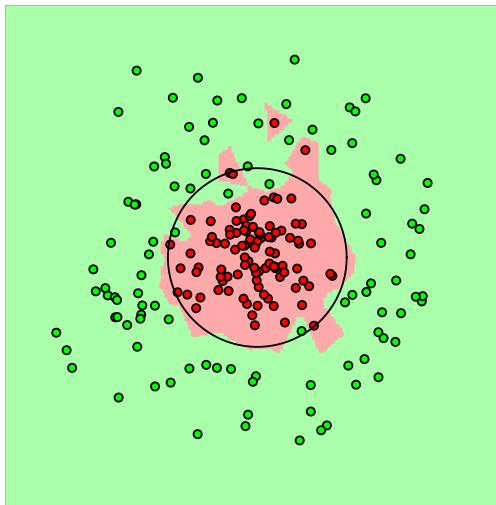
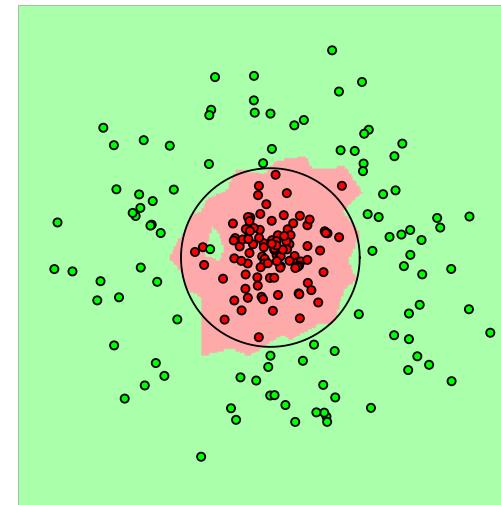
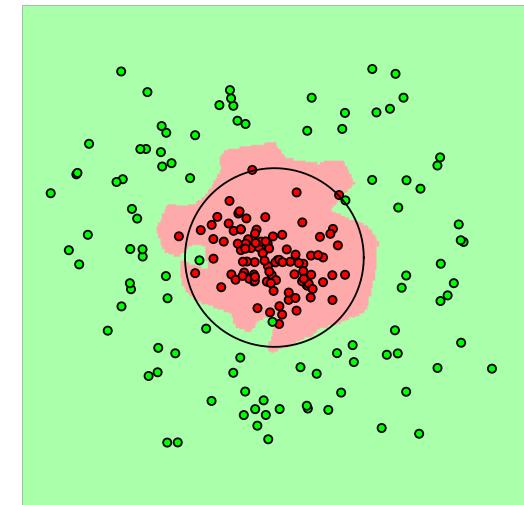
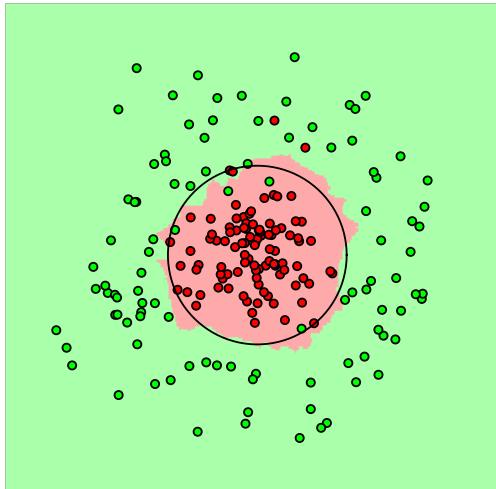
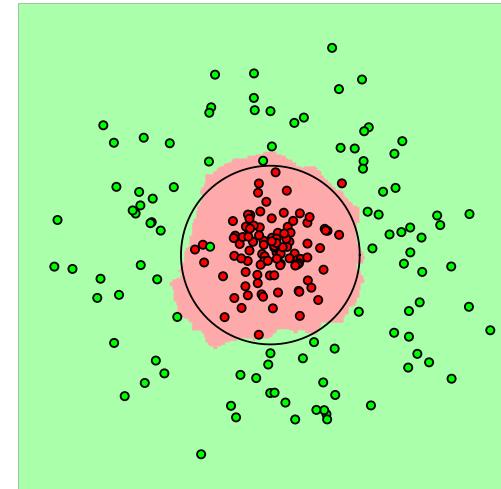
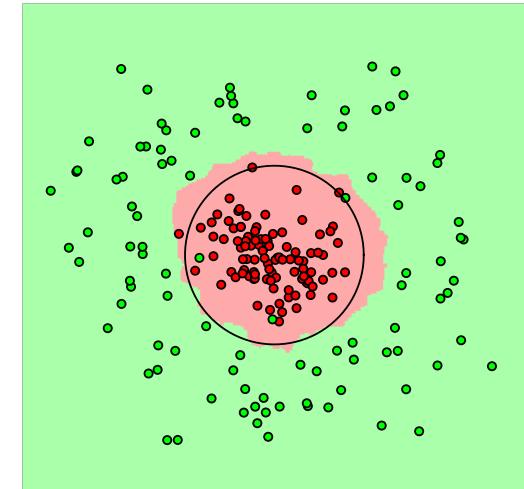


$K = 11$, error $\epsilon = 0.032$



$N = 100$ samples for each class. Bayes error $\epsilon_B = 0.026$.

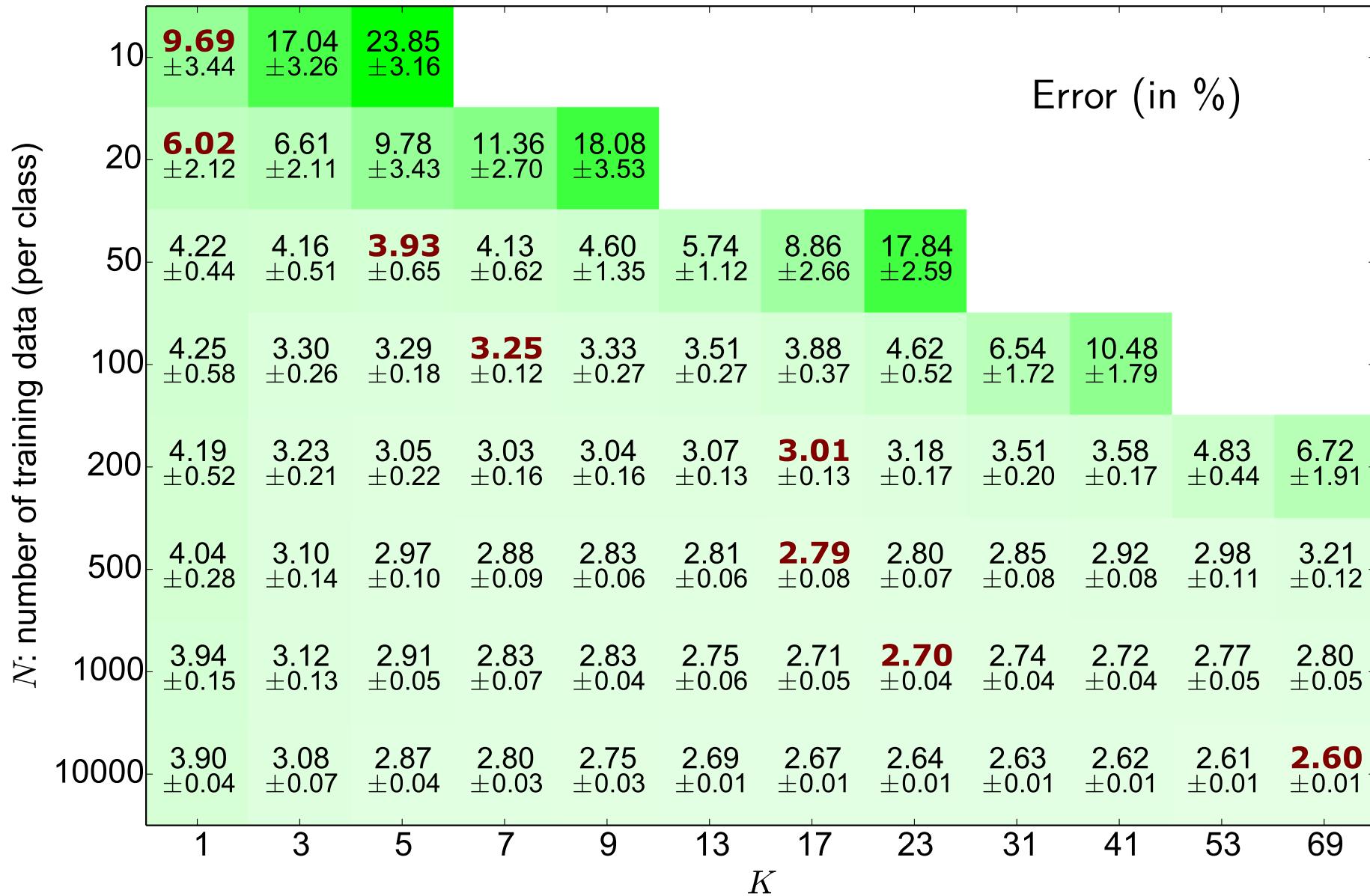
K -NN Example (3)

 \mathcal{T}_1
 $K = 1, \text{error } \epsilon = 0.044$

 \mathcal{T}_2
 $K = 1, \text{error } \epsilon = 0.038$

 \mathcal{T}_3
 $K = 1, \text{error } \epsilon = 0.043$

 $K = 7, \text{error } \epsilon = 0.030$

 $K = 7, \text{error } \epsilon = 0.031$

 $K = 7, \text{error } \epsilon = 0.036$


The results depend on the training set (result of a random process.)
 Each of the training sets $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ contain 100 points for each class.

K-NN Example (4)

K -NN error for different K and different sizes of the training set (N samples per class). 10 training sets have been generated randomly for each setting of K and N . Average error and its std is shown. Minimum average error is highlighted for each N . Bayes err. $\epsilon_B = 2.58\%$.



K-NN Properties

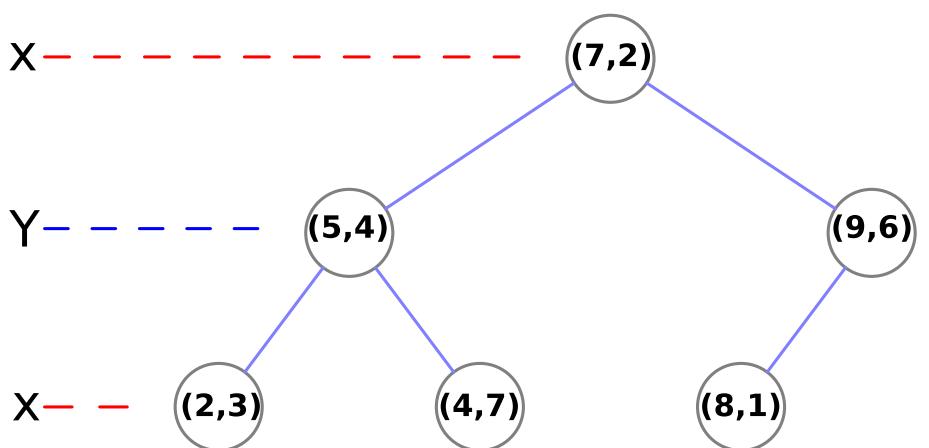
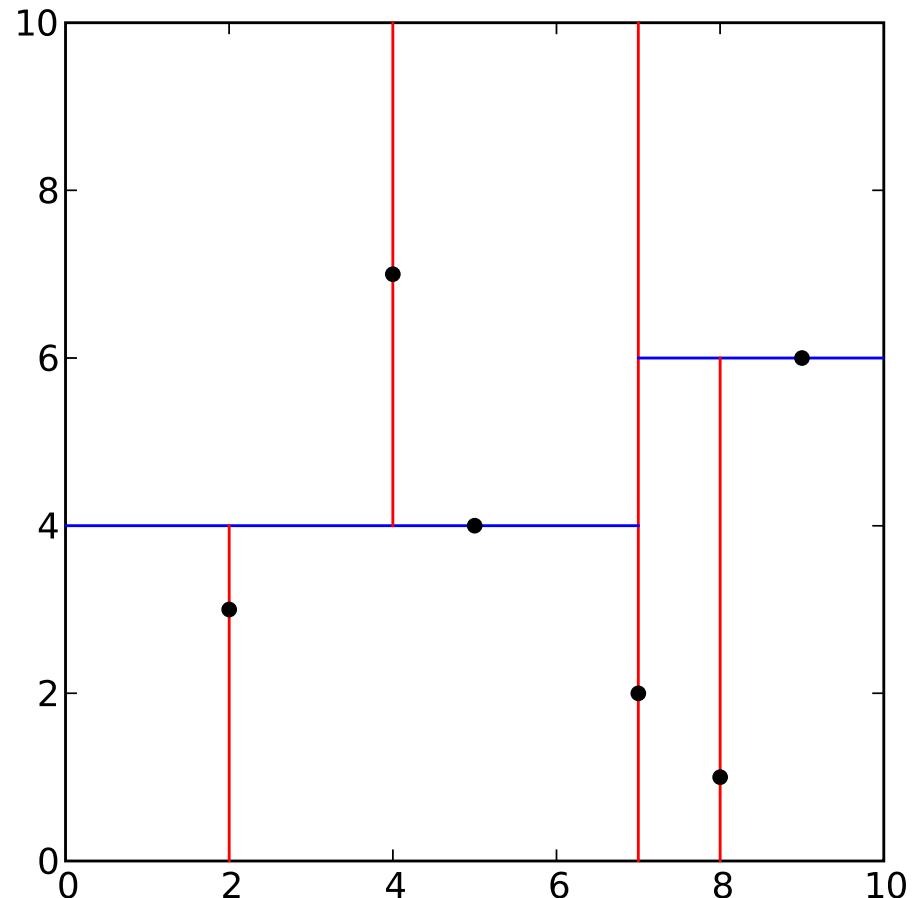
- ◆ Trivial implementation (\rightarrow good baseline method)
- ◆ 1-NN: Bayes error ϵ_B is the lower bound on error of classification ϵ_{NN} (in the asymptotic case $N \rightarrow \infty$.) Upper bounds can also be constructed, e.g. $\epsilon_{NN} \leq 2\epsilon_B$
- ◆ Slow when implemented naively, but can be sped up (Voronoi, k-D trees)
- ◆ High computer memory requirements (but training set can be edited and its cardinality decreased)
- ◆ How to construct the metric d ? (problem of scales in different axes)

K -NN : Speeding Up the Classification

- ◆ Sophisticated algorithms for NN search:
 - Classical problem in Comp. Geometry
 - k-D trees
- ◆ Removing the samples from the training class \mathcal{T} which do not change the result of classification
 - Exactly: using Voronoi diagram
 - Approximately: E.g. use Gabriel graph instead of Voronoi
 - Condensation algorithm: iterative, also approximate.

K-d Tree

k-d tree decomposition for the point set $(2,3)$, $(5,4)$, $(9,6)$, $(4,7)$, $(8,1)$, $(7,2)$



Condensation Algorithm

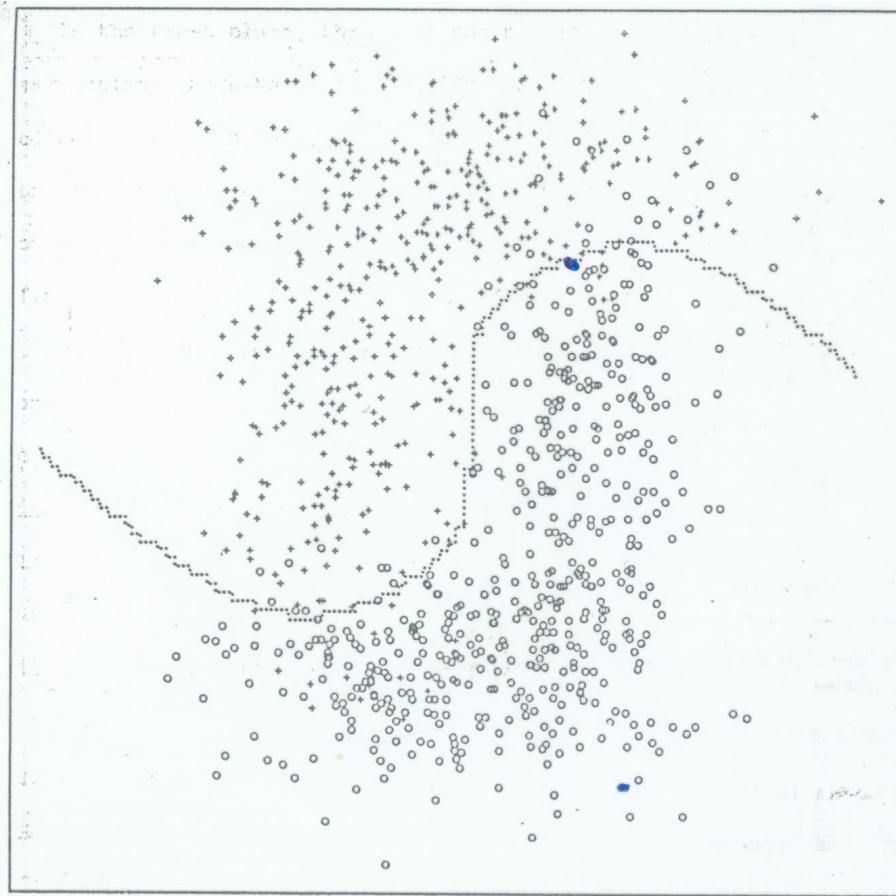
Input: The training set \mathcal{T} .

Algorithm

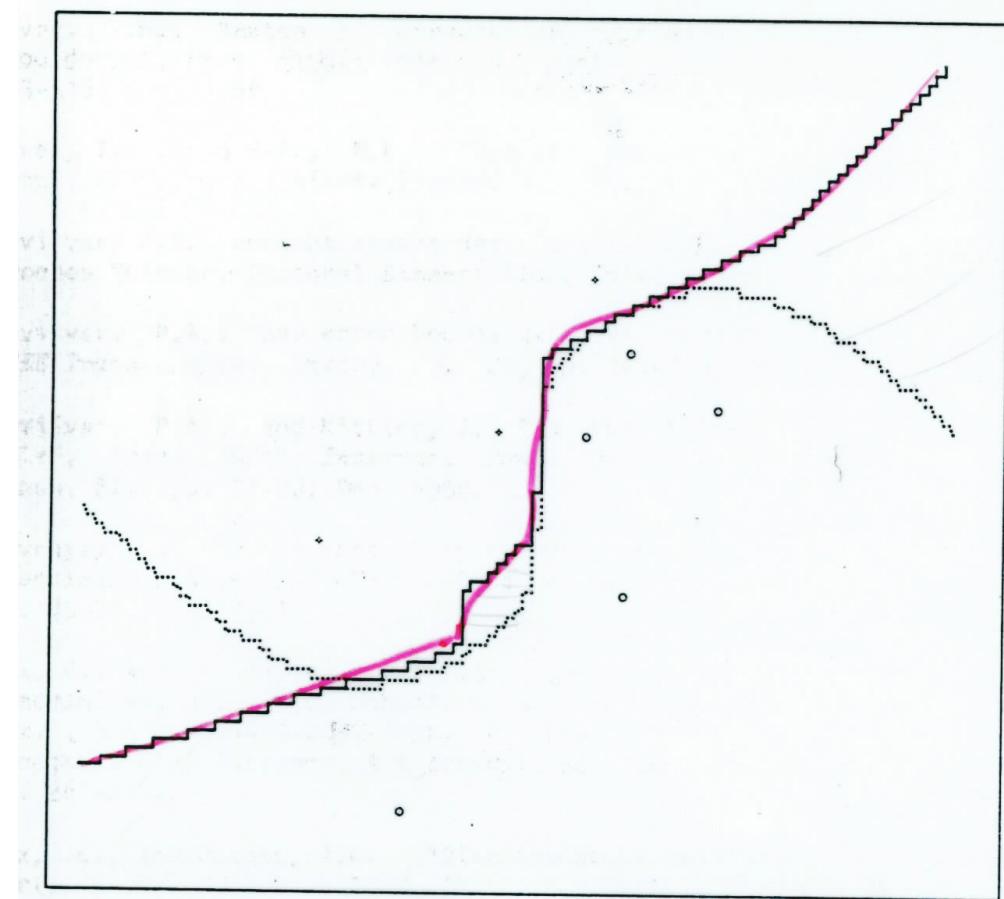
1. Create two lists, A and B . Insert a randomly selected sample from \mathcal{T} to A . Insert the rest of the training samples to B .
2. Classify samples from B using 1NN with training set A . If an $x \in B$ is mis-classified, move it from B to A .
3. If a move has been triggered in Step 2., goto Step 2.

Output: A (the condensed training set for 1NN classification)

Condensation Algorithm, Example



The training dataset



The dataset after the condensation.
Shown with the new decision boundary.

1-NN Classification Error

Recall that a classification error $\bar{\epsilon}$ for strategy $q: X \rightarrow R$ is computed as

$$\bar{\epsilon} = \int \sum_{k:q(x) \neq k} p(x, k) dx = \int \underbrace{\sum_{k:q(x) \neq k} p(k|x) p(x)}_{\epsilon(x)} dx = \int \epsilon(x) p(x) dx. \quad (21)$$

We know that the Bayesian strategy q_B decides for the highest posterior probability $q(x) = \operatorname{argmax}_k p(k|x)$, thus the partial error $\epsilon_B(x)$ for a given x is

$$\epsilon_B(x) = 1 - \max_k p(k|x). \quad (22)$$

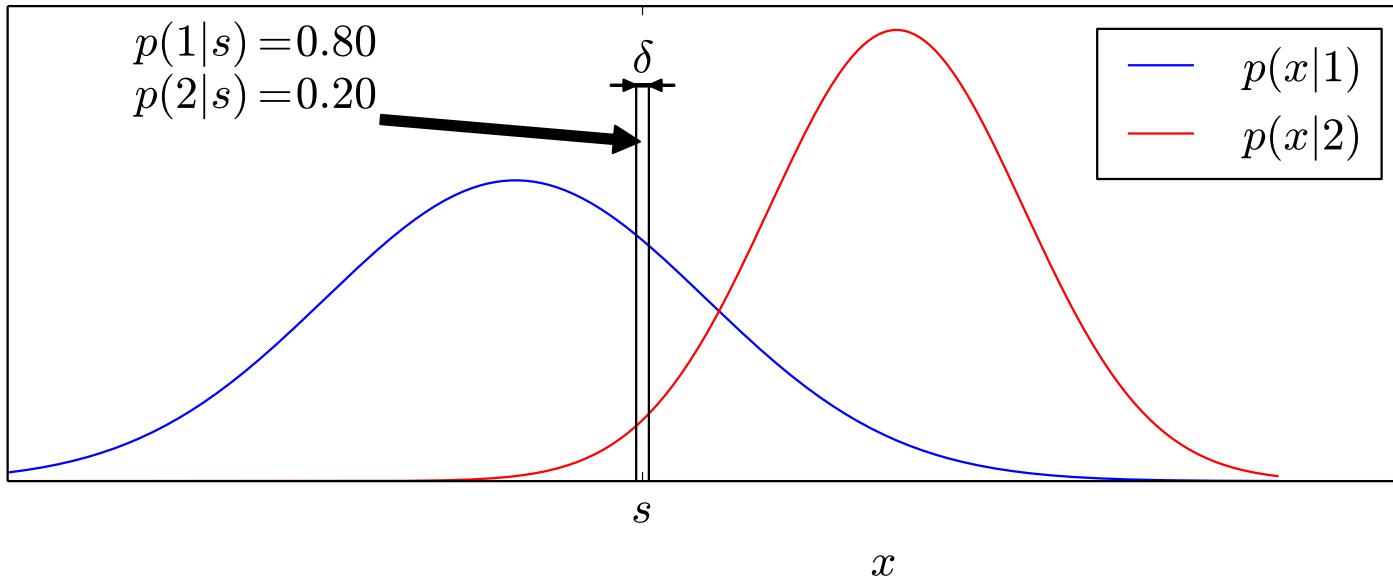
Assume the asymptotic case. We will show that the following bounds hold for the partial error $\epsilon_{NN}(x)$ and classification error $\bar{\epsilon}_{NN}$ in the 1-NN classification,

$$\epsilon_B(x) \leq \epsilon_{NN}(x) \leq 2\epsilon_B(x) - \frac{R}{R-1}\epsilon_B^2(x), \quad (23)$$

$$\bar{\epsilon}_B \leq \bar{\epsilon}_{NN} \leq 2\bar{\epsilon}_B - \frac{R}{R-1}\bar{\epsilon}_B^2, \quad (24)$$

where $\bar{\epsilon}_B$ is the Bayes classification error and R is the number of classes.

1-NN Classification Error, Example (1)



Consider two distributions as shown, a small interval δ on an x -axis, and a point $s \in \delta$. Let the class priors be $p(1) = p(2) = 0.5$. Assume $\delta \rightarrow 0$ and number of samples $N \rightarrow \infty$.

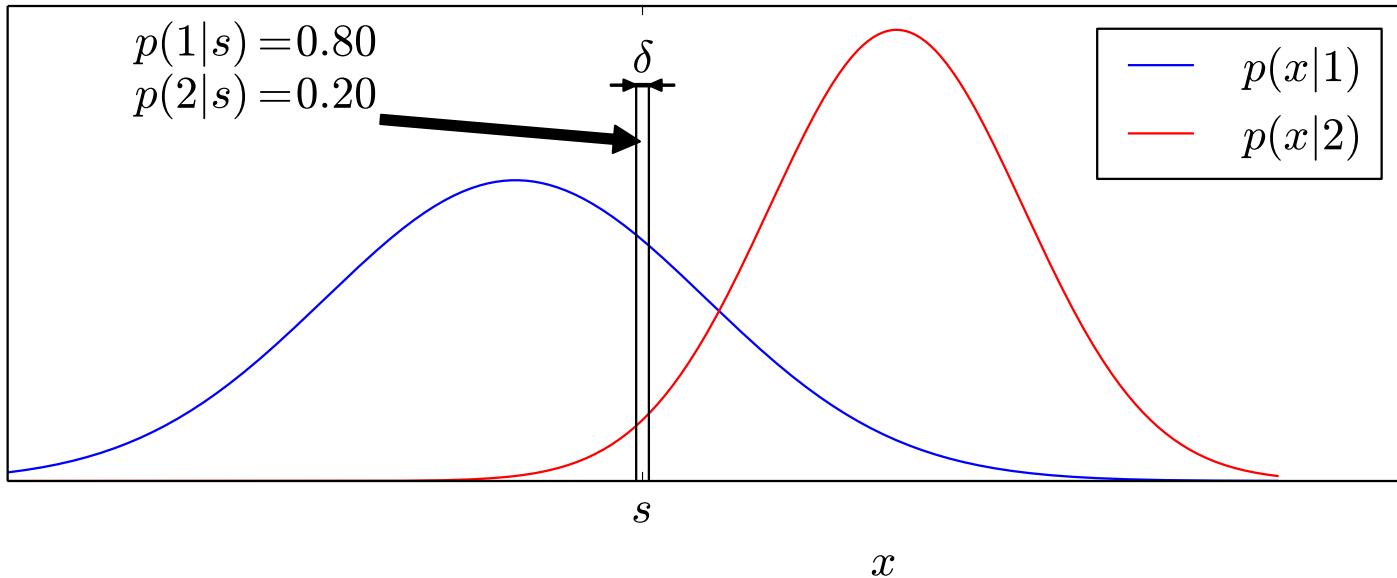
Observe the following:

$$p(1|s) = 0.8, \quad p(2|s) = 0.2, \quad (25)$$

$$p(NN=1|s) = p(1|s) = 0.8, \quad p(NN=2|s) = p(2|s) = 0.2, \quad (26)$$

where $p(NN=k|s)$ is the probability that the 1-NN of s is from class k ($k = 1, 2$) and thus s is classified as k .

1-NN Classification Error, Example (2)



The error $\epsilon_{NN}(s)$ at s is

$$\epsilon_{NN}(s) = p(1|s) p(NN=2|s) + p(2|s) p(NN=1|s) \quad (27)$$

$$= 1 - p(1|s) p(NN=1|s) - p(2|s) p(NN=2|s) \quad (28)$$

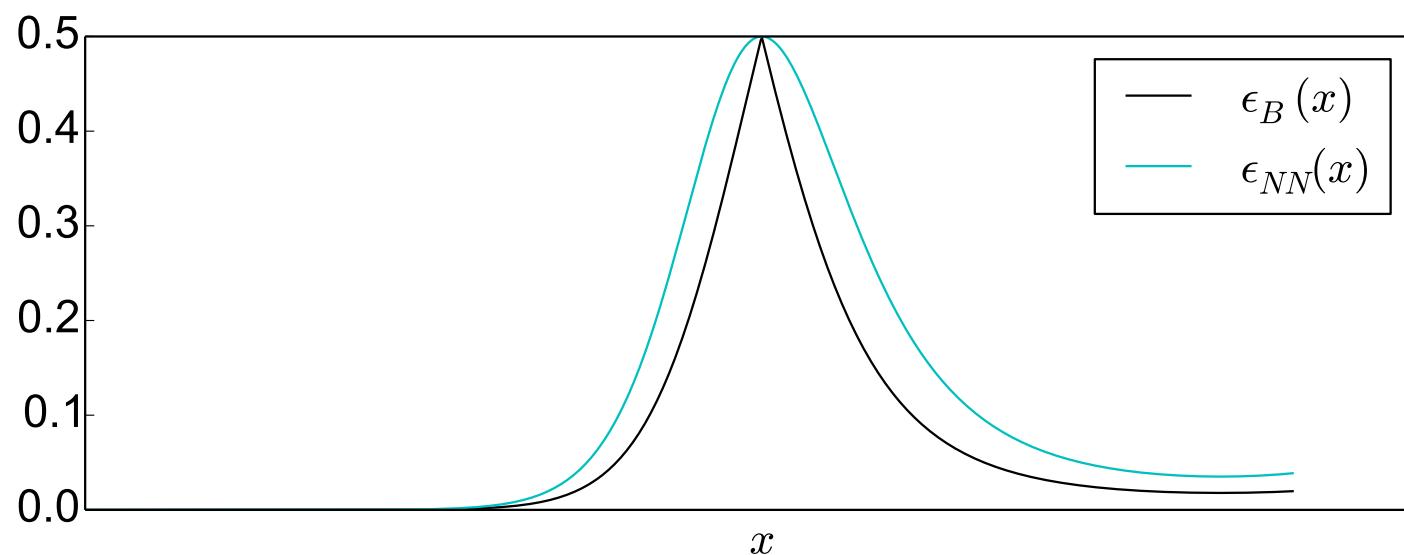
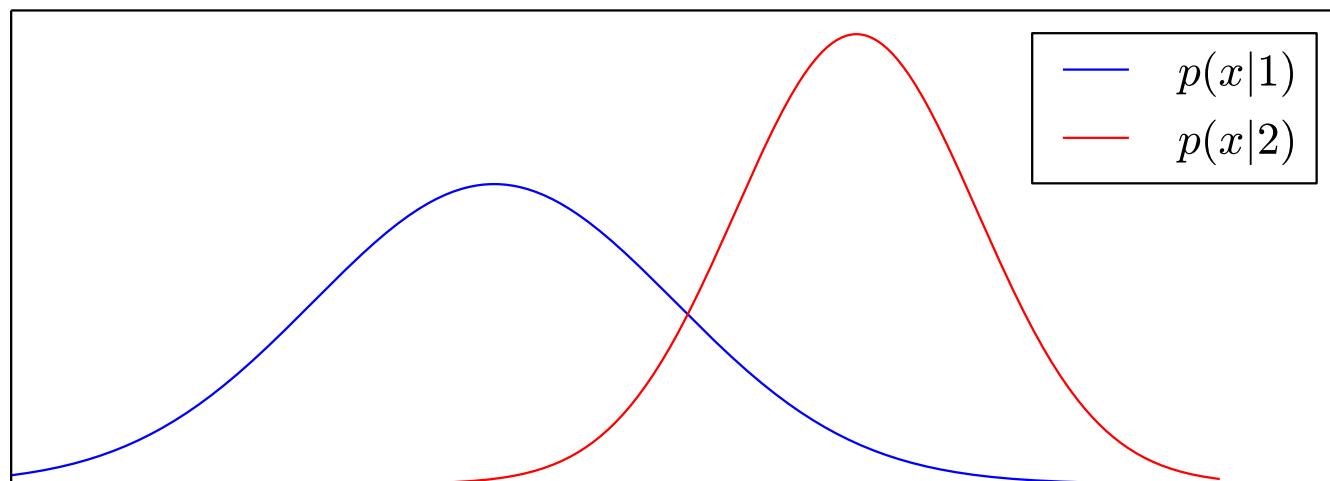
$$= 1 - p^2(1|s) - p^2(2|s). \quad (29)$$

Generally, for R classes, the error will be

$$\epsilon_{NN}(s) = 1 - \sum_{k \in R} p^2(k|s). \quad (30)$$

1-NN Classification Error, Example (3)

The two distributions and the partial errors
(the Bayesian error $\epsilon_B(x)$ and the 1-NN error $\epsilon_{NN}(x)$)



1-NN Classification Error Bounds (1)

Let us now return to the inequalities and prove them:

$$\epsilon_B(x) \leq \epsilon_{NN}(x) \leq 2\epsilon_B(x) - \frac{R}{R-1}\epsilon_B^2(x), \quad (31)$$

The **first** inequality follows from the fact that Bayes strategies are optimal.

To prove the **second** inequality, let $P(x)$ denote the maximum posterior for x :

$$P(x) = \max_k p(k|x) \quad (32)$$

$$\Rightarrow \epsilon_B(x) = 1 - P(x). \quad (33)$$

Let us rewrite the partial error $\epsilon_{NN}(x)$ using the Bayesian entities $P(x)$ and $q(x)$:

$$\epsilon_{NN}(x) = 1 - \sum_{k \in R} p^2(k|x) = 1 - P^2(x) - \sum_{k \neq q(x)} p^2(k|x). \quad (34)$$

We know that $p(q(x)|x) = P(x)$, but the remaining posteriors can be arbitrary. Let us consider the worst case. i.e. set $p(k|x)$ for $k \neq q(x)$ such that Eq. (34) is maximized. This will provide the upper bound.

1-NN Classification Error Bounds (2)

There are the following constraints on $p(k|x)$ ($k \neq q(x)$):

$$\sum_{k \neq q(x)} p(k|x) + P(x) = 1 \quad (\text{posteriors sum to 1}) \quad (35)$$

$$\sum_{k \neq q(x)} p^2(k|x) \rightarrow \min \quad (36)$$

It is easy to show that this optimization problem is solved by setting all the posteriors to the same number. Thus,

$$p(k|x) = \frac{1 - P(x)}{R - 1} = \frac{\epsilon_B(x)}{R - 1} \quad (k \neq q(x)) \quad (37)$$

The upper bound can then be rewritten in terms of the Bayes partial error $\epsilon_B(x) = 1 - P(x)$:

$$\epsilon_{NN}(x) \leq 1 - P^2(x) - \sum_{k \neq q(x)} p^2(k|x) = 1 - (1 - \epsilon_B(x))^2 - (R - 1) \frac{\epsilon_B^2(x)}{(R - 1)^2}. \quad (38)$$

1-NN Classification Error Bounds (3)

$$\epsilon_{NN}(x) \leq 1 - P^2(x) - \sum_{k \neq q(x)} p^2(k|x) = 1 - (1 - \epsilon_B(x))^2 - \frac{\epsilon_B^2(x)}{R-1}. \quad (39)$$

After expanding this, we get

$$\epsilon_{NN}(x) \leq 1 - (1 - \epsilon_B(x))^2 - \frac{\epsilon_B^2(x)}{(R-1)} \quad (40)$$

$$= 1 - 1 + 2\epsilon_B(x) - \epsilon_B^2(x) - \epsilon_B^2(x)\frac{R}{R-1} \quad (41)$$

$$= 2\epsilon_B(x) - \epsilon_B^2(x)\frac{R}{R-1} \quad (42)$$

Note that for $R = 2$, the bound is tight because using $\epsilon_B(x) = 1 - P(x)$ in Eq. (39) gives

$$\epsilon_{NN}(x) \leq 1 - P^2(x) - \frac{(1 - P(x))^2}{1} = \epsilon_{NN}(x). \quad (43)$$

1-NN Classification Error Bounds (4)

The inequality for the local errors has been proven:

$$\epsilon_{NN}(x) \leq 2\epsilon_B(x) - \epsilon_B^2(x) \frac{R}{R-1} \quad (44)$$

Is there a similar upper bound for the classification error $\bar{\epsilon}_{NN} = \int \epsilon_{NN}(x)p(x)dx$, based on the Bayes error $\bar{\epsilon}_B = \int \epsilon_B(x)p(x)dx$?

Multiplying Eq. (44) by $p(x)$, and integrating, gives

$$\bar{\epsilon}_{NN} \leq 2\bar{\epsilon}_B(x) - \frac{R}{R-1} \int \epsilon_B^2(x)p(x)dx \quad (45)$$

Let us use the known identity and inequality (where $E(\cdot)$ is the expectation operator)

$$\text{var}(x) = E(x^2) - E^2(x), \text{ var}(x) \geq 0 \quad \Rightarrow \quad E(x^2) \geq E^2(x) \quad (46)$$

Thus, $\int \epsilon_B^2(x)p(x)dx \geq (\int \epsilon_B(x)p(x)dx)^2$, and

$$\bar{\epsilon}_{NN} \leq 2\bar{\epsilon}_B(x) - \frac{R}{R-1} \int \epsilon_B^2(x)p(x)dx \leq 2\bar{\epsilon}_B(x) - \frac{R}{R-1}\bar{\epsilon}_B^2 \quad . \quad (47)$$

K -NN Classification Error Bound

It can be shown that for K -NN, the following inequality holds:

$$\bar{\epsilon}_{KNN} \leq \bar{\epsilon}_B + \bar{\epsilon}_{1NN} / \sqrt{K \text{ const}} \quad (48)$$

Edit algorithm

The primary goal of this method is to reduce the classification error (not the speed-up of classification.)

Input: The training set \mathcal{T} .

Algorithm

1. Partition \mathcal{T} to two sets, A and B ($\mathcal{T} = A \cup B$, $A \cap B = \emptyset$.)
2. Classify samples in B using K-NN with training set A . Remove all samples from B which have been mis-classified.

Output: B the training set for 1-NN classification.

Asymptotic property:

$$\bar{\epsilon}_{edit} = \bar{\epsilon}_B \frac{1 - \bar{\epsilon}_B}{1 - \bar{\epsilon}_{KNN}} \quad (49)$$

If $\bar{\epsilon}_{KNN}$ is small (e.g. 0.05) then the edited 1NN is quasi-Bayes (almost the same performance as Bayesian Classification.)

Logistic Regression

Lecturer:
Jiří Matas

Authors:
Ondřej Drbohlav, Jiří Matas

Centre for Machine Perception
Czech Technical University, Prague

<http://cmp.felk.cvut.cz>

Last update: 24/Oct/2019



Logistic Regression

Outline of the talk

- ◆ Motivation
- ◆ Model, relationship between log odds and posteriors
- ◆ Cross entropy objective function
- ◆ Gradient descent for fitting the model
- ◆ Examples
- ◆ Conclusion

Logistic Regression, Motivation

Consider a classification problem with 0/1 loss matrix. Recall that given an observation \mathbf{x} , the optimal Bayesian strategy $q(\mathbf{x})$ decides for a class k which maximizes the posterior:

$$q(\mathbf{x}) = \operatorname{argmax}_k p(k|\mathbf{x}) = \operatorname{argmax}_k p(\mathbf{x}, k) = \operatorname{argmax}_k p(\mathbf{x}|k)p(k). \quad (1)$$

For a binary (2-class) classification,

$$q(\mathbf{x}) = 1 \quad \text{if} \quad p(1|\mathbf{x}) > p(2|\mathbf{x}) \quad \Leftrightarrow \quad \frac{p(1|\mathbf{x})}{p(2|\mathbf{x})} > 1 \quad \Leftrightarrow \quad \ln \frac{p(1|\mathbf{x})}{p(2|\mathbf{x})} > 0, \quad (2)$$

$$q(\mathbf{x}) = 2 \quad \text{if} \quad p(1|\mathbf{x}) < p(2|\mathbf{x}) \quad \Leftrightarrow \quad \frac{p(1|\mathbf{x})}{p(2|\mathbf{x})} < 1 \quad \Leftrightarrow \quad \ln \frac{p(1|\mathbf{x})}{p(2|\mathbf{x})} < 0. \quad (3)$$

The ratio of posteriors $\frac{p(1|\mathbf{x})}{p(2|\mathbf{x})}$ is called **odds ratio**, the logarithm of this ratio, $\ln \frac{p(1|\mathbf{x})}{p(2|\mathbf{x})}$, is called **log odds**.

Logistic Regression, Motivation

Why are we interested in log odds? Because for the following problems, the log odds are linear (therefore simple) function of the observation variable \mathbf{x} :

- ◆ Normal distributions with equal variances
- ◆ Independent features with binary outcomes
- ◆ Multinomial naive Bayes

Normal distributions with equal covariance matrices

$$p(\mathbf{x}|1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu}_1)}, \quad (4)$$

$$p(\mathbf{x}|2) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu}_2)}, \quad (5)$$

$$(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2 \in \mathbb{R}^D, \boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}, \boldsymbol{\Sigma} \succ 0, \boldsymbol{\Sigma} = \boldsymbol{\Sigma}^\top) \quad (6)$$

The log odds:

$$\ln \frac{p(1|\mathbf{x})}{p(2|\mathbf{x})} = \ln \frac{p(\mathbf{x}|1)p(1)}{p(\mathbf{x}|2)p(2)} = \ln \frac{p(\mathbf{x}|1)}{p(\mathbf{x}|2)} + \ln \frac{p(1)}{p(2)} = \quad (7)$$

$$= \ln \frac{(2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}}}{(2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}}} e^{-\frac{1}{2}\{(\mathbf{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu}_1) - (\mathbf{x}-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu}_2)\}} + \ln \frac{p(1)}{p(2)} \quad (8)$$

Logistic Regression, Motivation

$$\ln \frac{p(1|x)}{p(2|x)} = \ln \frac{(2\pi)^{-\frac{D}{2}} |\Sigma|^{-\frac{1}{2}}}{(2\pi)^{-\frac{D}{2}} |\Sigma|^{-\frac{1}{2}}} e^{-\frac{1}{2} \{(x - \mu_1)^\top \Sigma^{-1} (x - \mu_1) - (x - \mu_2)^\top \Sigma^{-1} (x - \mu_2)\}} + \ln \frac{p(1)}{p(2)} \quad (9)$$

$$= -\frac{1}{2}(x^\top \cancel{\Sigma^{-1}} x - x^\top \Sigma^{-1} \mu_1 - \mu_1^\top \Sigma^{-1} x + \mu_1^\top \Sigma^{-1} \mu_1) \quad (10)$$

$$+ \frac{1}{2}(x^\top \cancel{\Sigma^{-1}} x - x^\top \Sigma^{-1} \mu_2 - \mu_2^\top \Sigma^{-1} x + \mu_2^\top \Sigma^{-1} \mu_2) + \ln \frac{p(1)}{p(2)} \quad (11)$$

(Note: $x^\top \Sigma^{-1} y = y^\top \Sigma^{-1} x$ because $\Sigma = \Sigma^\top$)

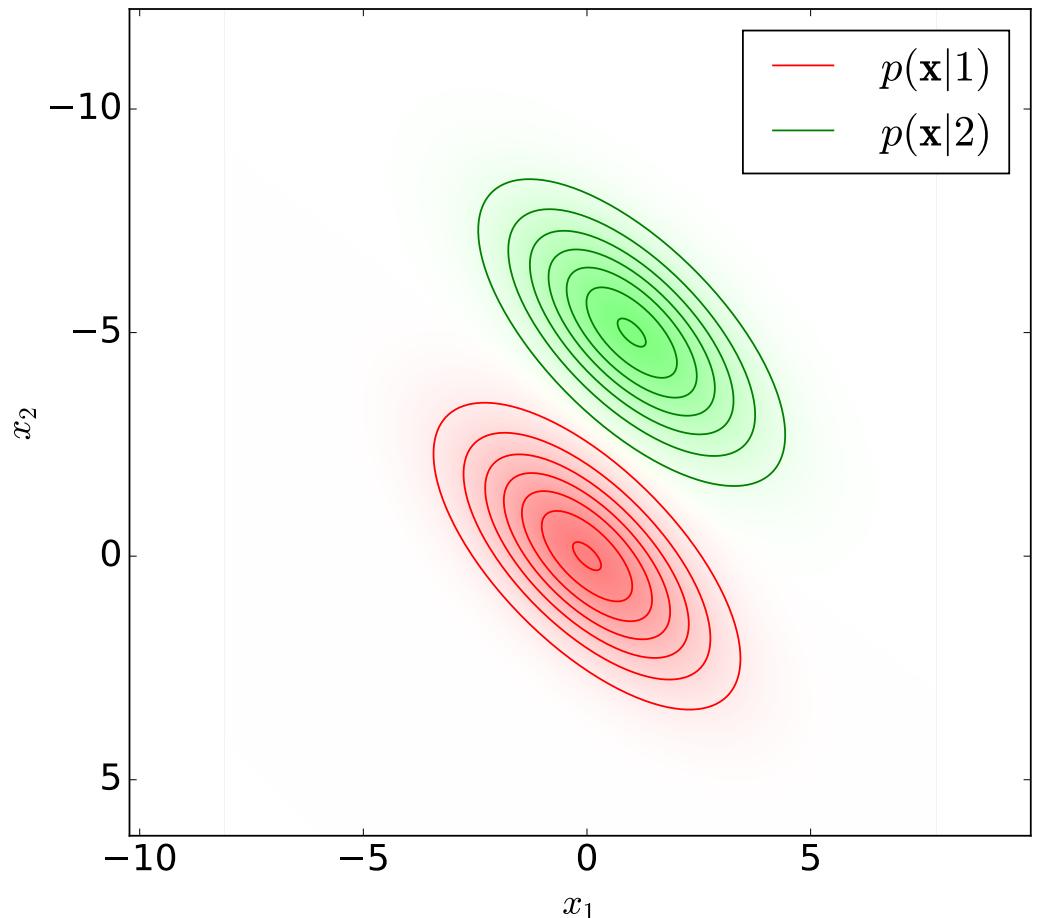
$$= -\frac{1}{2}(-2\mu_1^\top \Sigma^{-1} x + \mu_1^\top \Sigma^{-1} \mu_1 - (-2\mu_2^\top \Sigma^{-1} x + \mu_2^\top \Sigma^{-1} \mu_2)) + \ln \frac{p(1)}{p(2)} \quad (12)$$

$$= \underbrace{[(\mu_1 - \mu_2)^\top \Sigma^{-1}] x}_{\mathbf{w}} + \underbrace{\frac{1}{2}(\mu_1^\top \Sigma^{-1} \mu_1 - \mu_2^\top \Sigma^{-1} \mu_2) + \ln \frac{p(1)}{p(2)}}_{w_0} \quad (13)$$

$$= \mathbf{w} \cdot \mathbf{x} + w_0, \quad (\mathbf{w} \in \mathbb{R}^D, w_0 \in R) \quad (14)$$

Conclusion: When two classes are normally distributed with equal covariance matrices, the log odds are a **linear** function of observation vector \mathbf{x} .

Example



Multinomial normal distribution,
two classes

Centers:

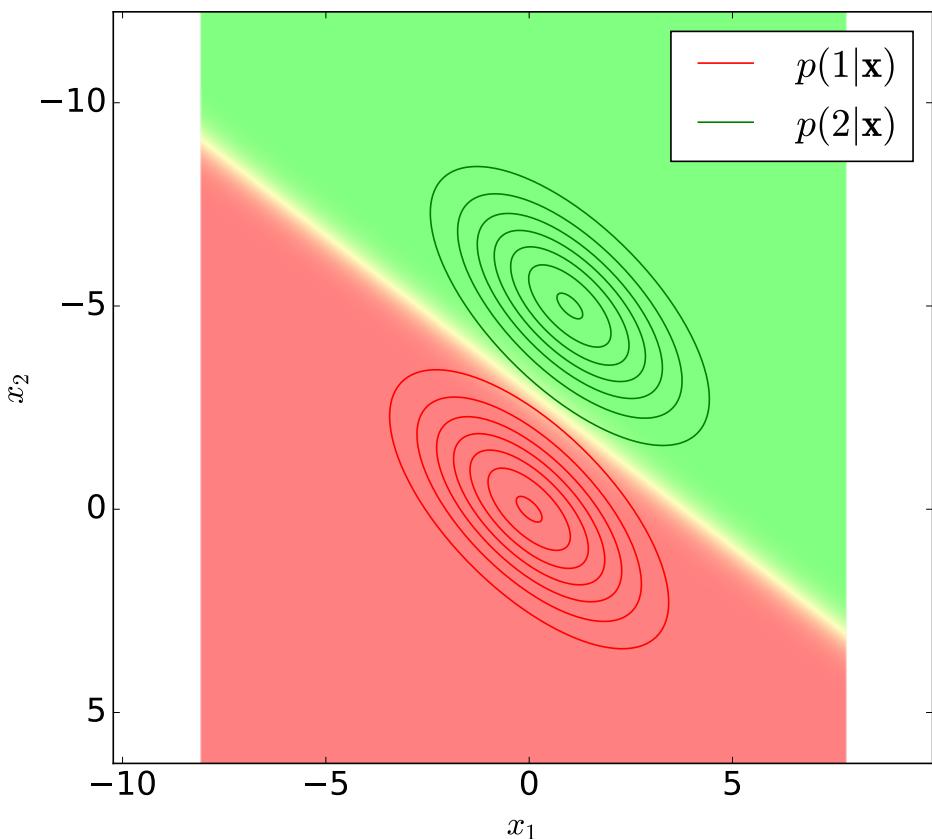
$$(0, 0), (1, -5).$$

Cov. matrices (are equal):

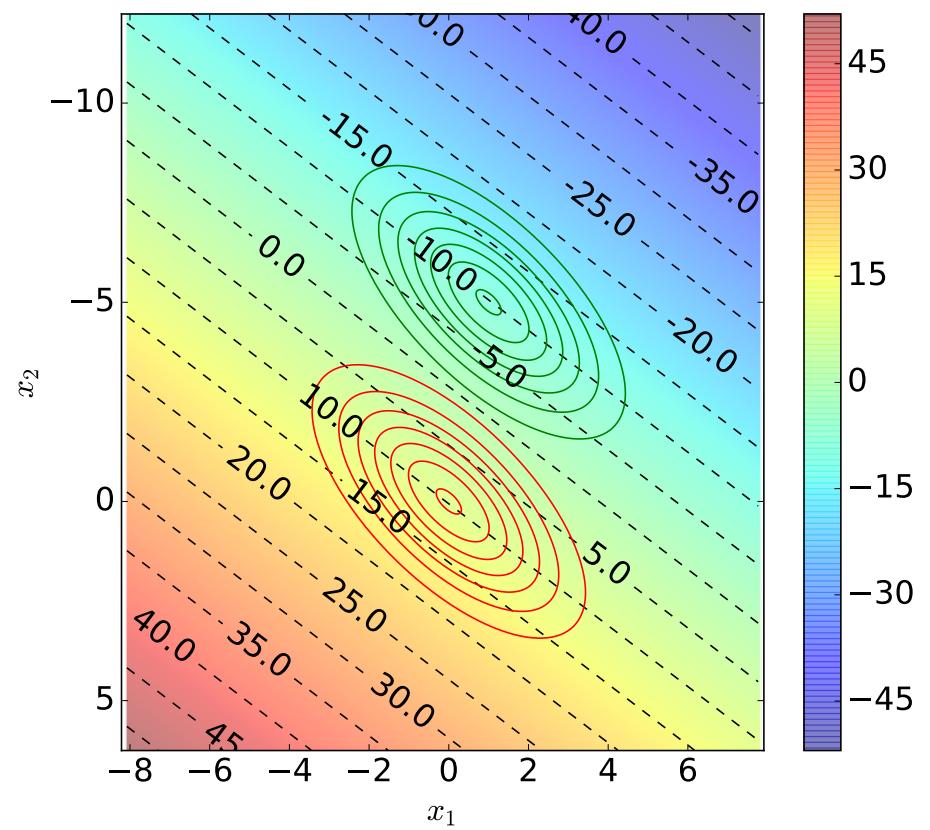
$$\Sigma = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}$$

Example (contd.)

Priors: $p(\text{red}) = 0.5, p(\text{green}) = 0.5$



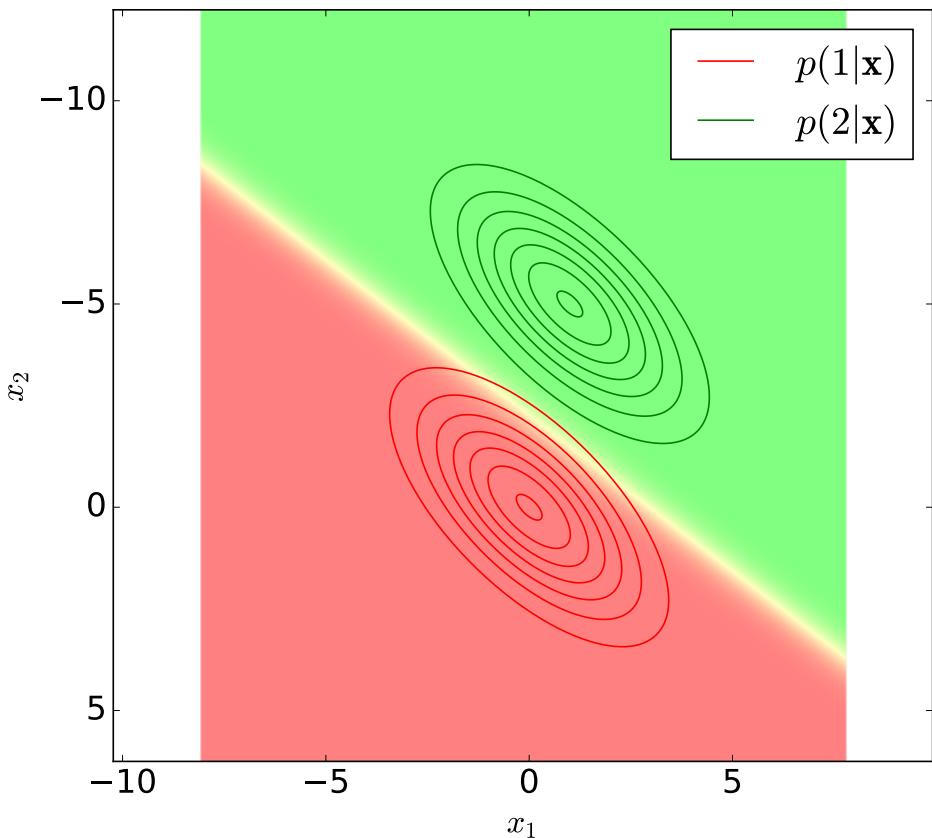
Probabilities $p(1|\mathbf{x})$ and $p(2|\mathbf{x})$.



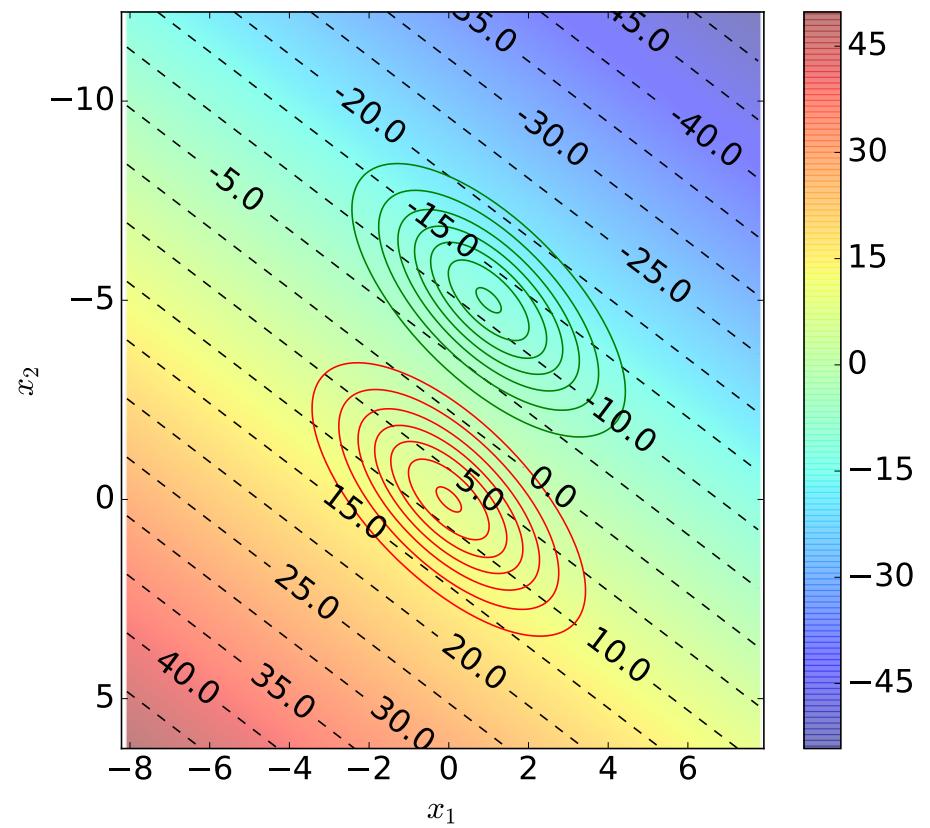
Log odds, shown as a map with contour plot.
Note that the log odds are a linear function.

Example (contd.)

Priors: $p(\text{red}) = 0.1$, $p(\text{green}) = 0.9$



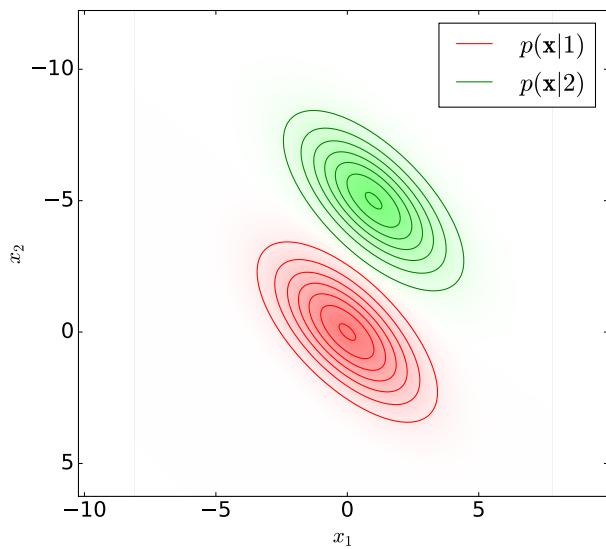
Probabilities $p(1|\mathbf{x})$ and $p(2|\mathbf{x})$.



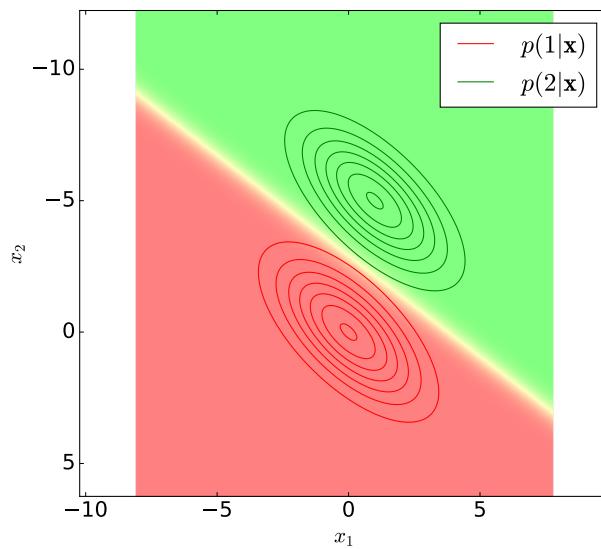
Log odds, shown as a map with contour plot.
 Note that the log odds are a linear function.
 Also note the shift of the zero level w.r.t. previous slide.

Example (contd.)

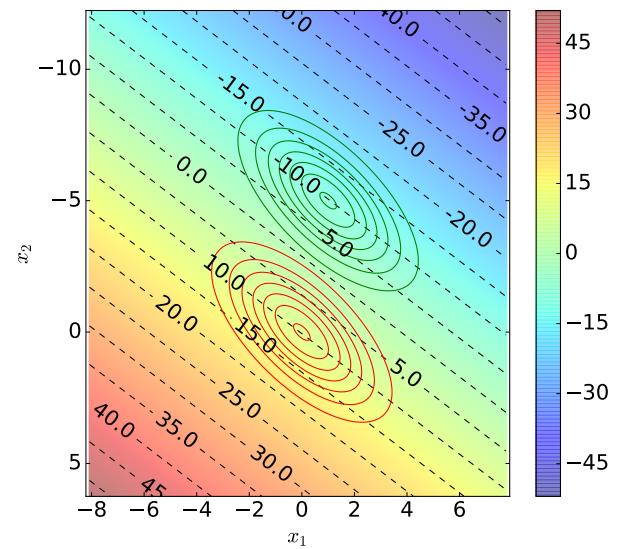
conditionals



posteriors



log odds



It will soon be shown for linear log odds, $a(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + w_0$, the posteriors can be written as

$$p(1|\mathbf{x}) = \frac{1}{1 + e^{-a(\mathbf{x})}} \quad (15)$$

$$p(2|\mathbf{x}) = \frac{1}{1 + e^{a(\mathbf{x})}} \quad (16)$$

The idea of logistic regression is that posteriors are modeled this way directly, without first estimating the priors and conditionals.

Other models with linear log odds (1)

Independent features with binary outcomes.

Feature vector $\mathbf{x} = (x_1, x_2, \dots, x_D) \in \mathbb{R}^D$ $x_i \in \{0, 1\}$ (binary outcomes)

Each feature:

$$p(x_i = 1) = \pi_i, \quad (17)$$

$$p(x_i = 0) = 1 - \pi_i, \quad (18)$$

$$\Rightarrow \text{ can be written as: } \quad p(x_i) = \pi_i^{x_i} (1 - \pi_i)^{1-x_i} \quad (19)$$

Conditional probabilities for two classes 1, 2:

$$p(\mathbf{x}|1) = \prod_{i=1}^D \pi_i^{x_i} (1 - \pi_i)^{1-x_i} \quad (20)$$

$$p(\mathbf{x}|2) = \prod_{i=1}^D \kappa_i^{x_i} (1 - \kappa_i)^{1-x_i} \quad (\pi_i, \kappa_i \in (0, 1), x_i \in \{0, 1\}) \quad (21)$$

Other models with linear log odds (1)

$$p(\mathbf{x}|1) = \prod_{i=1}^D \pi_i^{x_i} (1 - \pi_i)^{1-x_i} \quad (22)$$

$$p(\mathbf{x}|2) = \prod_{i=1}^D \kappa_i^{x_i} (1 - \kappa_i)^{1-x_i} \quad (\pi_i, \kappa_i \in (0, 1), x_i \in \{0, 1\}) \quad (23)$$

The log odds are:

$$a(\mathbf{x}) = \ln \frac{p(\mathbf{x}|1)p(1)}{p(\mathbf{x}|2)p(2)} = \sum_{i=1}^D \{x_i \ln \pi_i + (1 - x_i) \ln(1 - \pi_i) - x_i \ln \kappa_i - (1 - x_i) \ln(1 - \kappa_i)\} \quad (24)$$

$$+ \ln \frac{p(1)}{p(2)} = \mathbf{w} \cdot \mathbf{x} + w_0 \quad (\mathbf{w} \in \mathbb{R}^D, w_0 \in \mathbb{R}) \quad (25)$$

Note that the assumption that the features are independent may be quite strong. If this assumption is true (or anyway adopted), we talk about **naive Bayes** approach.

Other models with linear log odds (1)

Example

Problem: male/female classification

x_1 : hair length > 5cm

x_2 : shoe size > 41

Other models with linear log odds (2)

Multinomial naive Bayes

The analysis is similar to the case of binary outcomes. Here, the feature components x_i are not binary but they represent counts, summing to certain constant n ($\sum_{i=1}^D x_i = n$). The probabilities of observing the counts $\{x_i, i = 1, 2, \dots, D\}$ are

$$p(\mathbf{x}|1) = \frac{n!}{\prod_{i=1}^D x_i!} \prod_{i=1}^D \pi_i^{x_i} \quad (26)$$

$$p(\mathbf{x}|2) = \frac{n!}{\prod_{i=1}^D x_i!} \prod_{i=1}^D \kappa_i^{x_i} \quad (x_i \in \mathbb{N}_0, \pi_i > 0, \kappa_i > 0,$$

$$\sum_i \pi_i = 1, \sum_i \kappa_i = 1, \sum_{i=1}^D x_i = n) \quad (27)$$

It is easy to see that the log odds $a(\mathbf{x})$ are again linear in \mathbf{x} .

Summary

In many real-world problems, the log odds $a(\mathbf{x})$ are a linear function of the observations \mathbf{x} .

Logistic Regression, Model

Idea: Let us look for the log of the ratio of posteriors (log odds) $a(\mathbf{x})$ directly as a linear function of the input vector $\mathbf{x} = (x_1, x_2, \dots, x_D) \in \mathbb{R}^D$ (D is the dimensionality of the feature space):

$$a(\mathbf{x}) = \ln \frac{p(1|\mathbf{x})}{p(2|\mathbf{x})} = \mathbf{w} \cdot \mathbf{x} + w_0, \quad \mathbf{w} = (w_1, w_2, \dots, w_D) \in \mathbb{R}^D, \\ w_0 \in \mathbb{R}, \tag{28}$$

where w_0 is the bias term.

Let us rewrite this as

$$a(\mathbf{x}) = w_0 \cdot 1 + \mathbf{w} \cdot \mathbf{x} = [w_0, w_1, w_2, \dots, w_D] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_D \end{bmatrix} = \mathbf{w}' \cdot \mathbf{x}'. \quad (x_0 = 1) \tag{29}$$

Note: From now on, we will drop the dash sign and write again only ' \mathbf{x} ' or ' \mathbf{w} ', with the understanding that these **include the zero-index components** $x_0 = 1$ and $w_0 \in \mathbb{R}$ implementing the **bias**.

Logistic Regression, Log Odds and Posteriors (1)

Here is the relationship between the log odds $a(\mathbf{x})$ and the posterior probabilities $p(1|\mathbf{x})$ and $p(2|\mathbf{x})$.

The log odds $a(\mathbf{x})$ is (remember the bias term is consumed in the \mathbf{x} and \mathbf{w})

$$a(\mathbf{x}) = \ln \frac{p(1|\mathbf{x})}{p(2|\mathbf{x})} = \mathbf{w} \cdot \mathbf{x}. \quad (30)$$

From this, it follows that

$$\frac{p(1|\mathbf{x})}{p(2|\mathbf{x})} = \exp(\mathbf{w} \cdot \mathbf{x}) \quad \frac{p(2|\mathbf{x})}{p(1|\mathbf{x})} = \exp(-\mathbf{w} \cdot \mathbf{x}) \quad (31)$$

$$p(1|\mathbf{x}) = p(2|\mathbf{x}) \exp(\mathbf{w} \cdot \mathbf{x}) \quad p(2|\mathbf{x}) = p(1|\mathbf{x}) \exp(-\mathbf{w} \cdot \mathbf{x}) \quad (32)$$

$$1 = p(1|\mathbf{x}) + p(2|\mathbf{x}) = p(1|\mathbf{x}) (1 + e^{-\mathbf{w} \cdot \mathbf{x}}) = p(2|\mathbf{x}) (1 + e^{\mathbf{w} \cdot \mathbf{x}}) \quad (33)$$

$$p(1|\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}, \quad p(2|\mathbf{x}) = \frac{1}{1 + e^{\mathbf{w} \cdot \mathbf{x}}}. \quad (34)$$

Logistic Regression, Log Odds and Posteriors (2)

Again,

$$a(\mathbf{x}) = \ln \frac{p(1|\mathbf{x})}{p(2|\mathbf{x})} = \mathbf{w} \cdot \mathbf{x} \quad (35)$$

$$p(1|\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}} = \sigma(\mathbf{w} \cdot \mathbf{x}) \quad (36)$$

$$p(2|\mathbf{x}) = \frac{1}{1 + e^{\mathbf{w} \cdot \mathbf{x}}} = \sigma(-\mathbf{w} \cdot \mathbf{x}) \quad (37)$$

where $\sigma(u) = 1/(1 + \exp(-u))$ is the **logistic sigmoid** function.

It will be advantageous to rename the classes from $(1, 2)$ to $(1, -1)$. Then we can rewrite the equations (36, 37) as

$$p(k|\mathbf{x}) = \frac{1}{1 + e^{-k\mathbf{w} \cdot \mathbf{x}}}, \quad k \in \{-1, 1\} \quad (38)$$

Finding \mathbf{w} : Objective $E(\mathbf{w})$

$$p(k|\mathbf{x}) = \frac{1}{1 + e^{-k\mathbf{w} \cdot \mathbf{x}}}, \quad k \in \{-1, 1\} \quad (39)$$

How do we find \mathbf{w} ?

We apply the Maximum Likelihood approach for finding \mathbf{w} . Let us have the training set $\mathcal{T} = \{(\mathbf{x}_1, k_1), (\mathbf{x}_2, k_2), \dots, (\mathbf{x}_N, k_N)\}$. The optimal \mathbf{w}^* would be the one maximizing the log-likelihood $l(\mathbf{w})$,

$$l(\mathbf{w}) = \sum_{(\mathbf{x}, k) \in \mathcal{T}} \ln p(\mathbf{x}, k)_{[\mathbf{w}]} = \sum_{(\mathbf{x}, k) \in \mathcal{T}} \ln p(k|\mathbf{x})_{[\mathbf{w}]} + \sum_{(\mathbf{x}, k) \in \mathcal{T}} \ln p(\mathbf{x})_{[\mathbf{w}]}, \quad (40)$$

where for the sake of clarity, dependence on \mathbf{w} is denoted by a subscript $[\mathbf{w}]$.

As there are no assumptions about the form of $p(\mathbf{x})$ as a function of \mathbf{w} , logistic regression employs instead the maximization of *conditional log-likelihood* $l'(\mathbf{w})$

$$l'(\mathbf{w}) = \sum_{(\mathbf{x}, k) \in \mathcal{T}} \ln p(k|\mathbf{x})_{[\mathbf{w}]} = - \sum_{(\mathbf{x}, k) \in \mathcal{T}} \ln(1 + e^{-k\mathbf{w} \cdot \mathbf{x}}) \quad (\text{conditional log likelihood}) \quad (41)$$

$$\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} l'(\mathbf{w}) \quad (\text{optimal } \mathbf{w}^*) \quad (42)$$

Finding \mathbf{w} : Objective $E(\mathbf{w})$

(copied from previous slide)

$$l'(\mathbf{w}) = \sum_{(\mathbf{x}, k) \in \mathcal{T}} \ln p(k|\mathbf{x})_{[\mathbf{w}]} = - \sum_{(\mathbf{x}, k) \in \mathcal{T}} \ln(1 + e^{-k\mathbf{w} \cdot \mathbf{x}}) \quad (\text{conditional log likelihood}) \quad (43)$$

$$\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} l'(\mathbf{w}) \quad (\text{optimal } \mathbf{w}^*) \quad (44)$$

In order for the optimization to fit into the **minimization** framework, we define the objective function $E(\mathbf{w})$ as the negative conditional log likelihood, $E(\mathbf{w}) = -l'(\mathbf{w})$. This objective function corresponds to **cross entropy**. Let us now analyze the properties of $E(\mathbf{w})$.

Finding \mathbf{w} : Gradient of $E(\mathbf{w})$

$$E(\mathbf{w}) = - \sum_{(\mathbf{x}, k) \in \mathcal{T}} \ln p(k|\mathbf{x}) = \sum_{(\mathbf{x}, k) \in \mathcal{T}} \ln(1 + e^{-k\mathbf{w} \cdot \mathbf{x}}) \quad (45)$$

(46)

The gradient vector $g(\mathbf{w})$ of E is:

$$g(\mathbf{w}) = \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \sum_{(\mathbf{x}, k) \in \mathcal{T}} \frac{e^{-k\mathbf{w} \cdot \mathbf{x}}}{1 + e^{-k\mathbf{w} \cdot \mathbf{x}}} (-k\mathbf{x}) = - \sum_{(\mathbf{x}, k) \in \mathcal{T}} \underbrace{\frac{1}{1 + e^{k\mathbf{w} \cdot \mathbf{x}}} k\mathbf{x}}_{p(-k|\mathbf{x})} \quad (47)$$

$$= - \sum_{(\mathbf{x}, k) \in \mathcal{T}} (1 - p(k|\mathbf{x})) k\mathbf{x}. \quad (48)$$

We require $g(\mathbf{w}) = 0$ (the necessary condition for optimality). However, it seems that these equations **cannot be solved analytically**. We will need to resort to the numerical optimization methods. Let us continue and check the second order derivatives.

Finding \mathbf{w} : $E(\mathbf{w})$ is convex

The Hessian matrix $H(\mathbf{w})$ of the objective function E is

$$H(\mathbf{w}) = \frac{\partial^2 E(\mathbf{x})}{\partial \mathbf{w}^2} = \frac{\partial g(\mathbf{w})}{\partial \mathbf{w}} = -\frac{\partial}{\partial \mathbf{w}} \sum_{(\mathbf{x}, k) \in \mathcal{T}} \frac{1}{1 + e^{k\mathbf{w} \cdot \mathbf{x}}} k \mathbf{x} \quad (49)$$

$$= \sum_{(\mathbf{x}, k) \in \mathcal{T}} \underbrace{\frac{e^{k\mathbf{w} \cdot \mathbf{x}}}{(1 + e^{k\mathbf{w} \cdot \mathbf{x}})^2} k^2}_{> 0} \mathbf{x} \mathbf{x}^\top = \sum_{(\mathbf{x}, k) \in \mathcal{T}} p(-1|\mathbf{x}) p(1|\mathbf{x}) \mathbf{x} \mathbf{x}^\top \quad (50)$$

This is a very important result. It shows that the Hessian matrix $H(\mathbf{w})$ is **positive definite** in every point \mathbf{w} and, therefore, the function $E(\mathbf{w})$ is **convex**. As a consequence, $E(\mathbf{w})$ has a **unique minimum**.

Note. Can you show that $H(\mathbf{w})$ is positive definite?

Finding w : Gradient Descent

Any method of convex optimization can be used to find the optimal w^* . For the examples in this lecture, the following gradient descent method with adaptive step size has been used:

```
# input: x (observations), k (class labels), w_init (initial w)

# init:
w = w_init
step_size = 1.0
E, g = compute_E_and_gradient(x, k, w)

# iterate:
while not TERMINATION_CONDITION:
    E_new, g_new = compute_E_and_gradient(x, k, w - step_size * g)

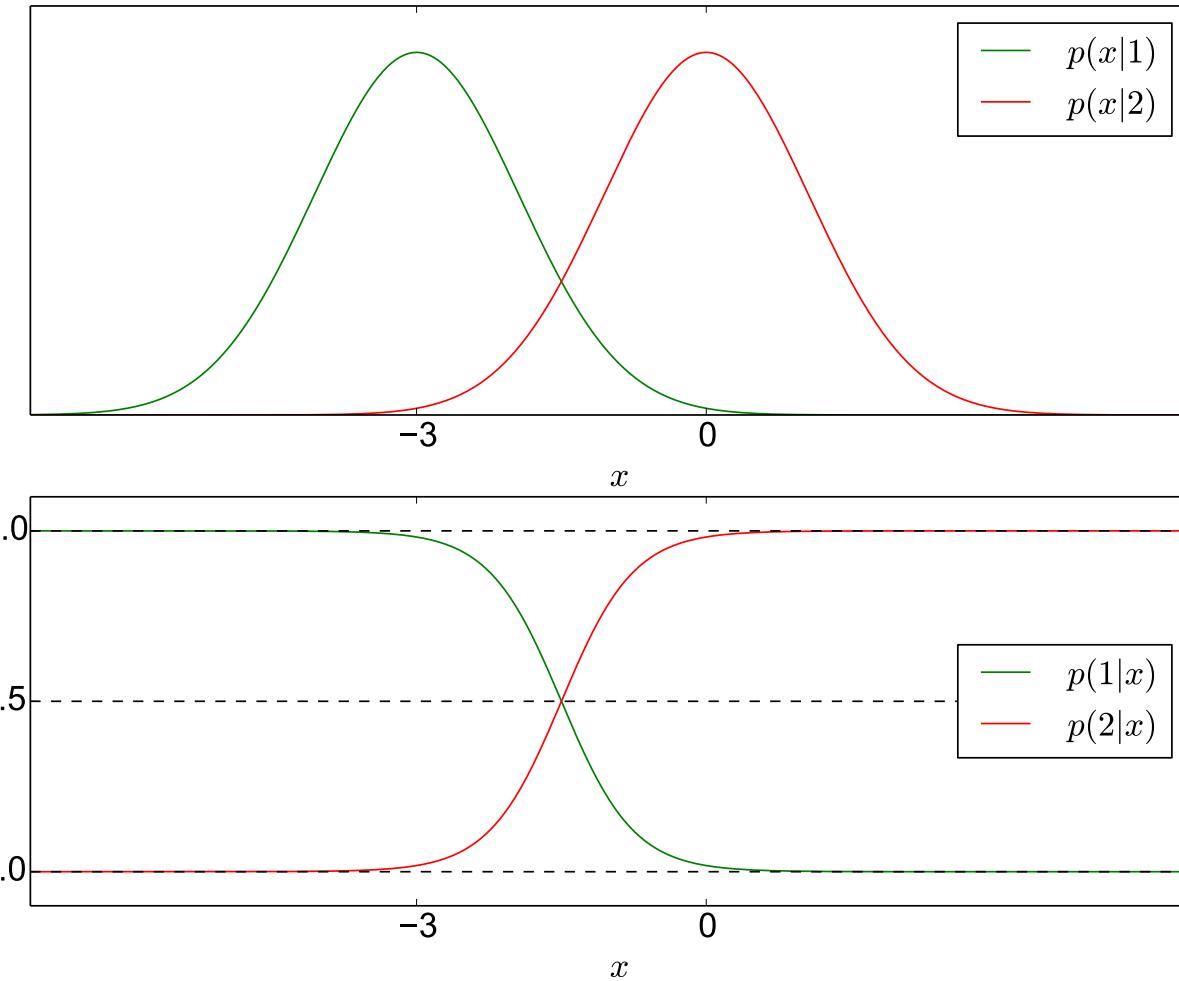
    if E_new < E:
        # success.
        w -= step_size * g
        g = g_new
        E = E_new
        step_size *= 2
    else:
        step_size /= 2

return w
```

Notes:

- i) Iteration is accepted if $E(w)$ decreases. If it hasn't decreased, either the step size is too high (thus it is halved), or optimum has been already found.
- ii) We normalize the gradient by the number of training data N because otherwise its magnitude scales linearly with N , causing the necessity for smaller step sizes with higher N .

Example 1, Two Normal Dists with Equal Variance (1)



$$p(x|1) = \mathcal{N}(x|\mu_1 = -3, \sigma_1^2 = 1.5)$$

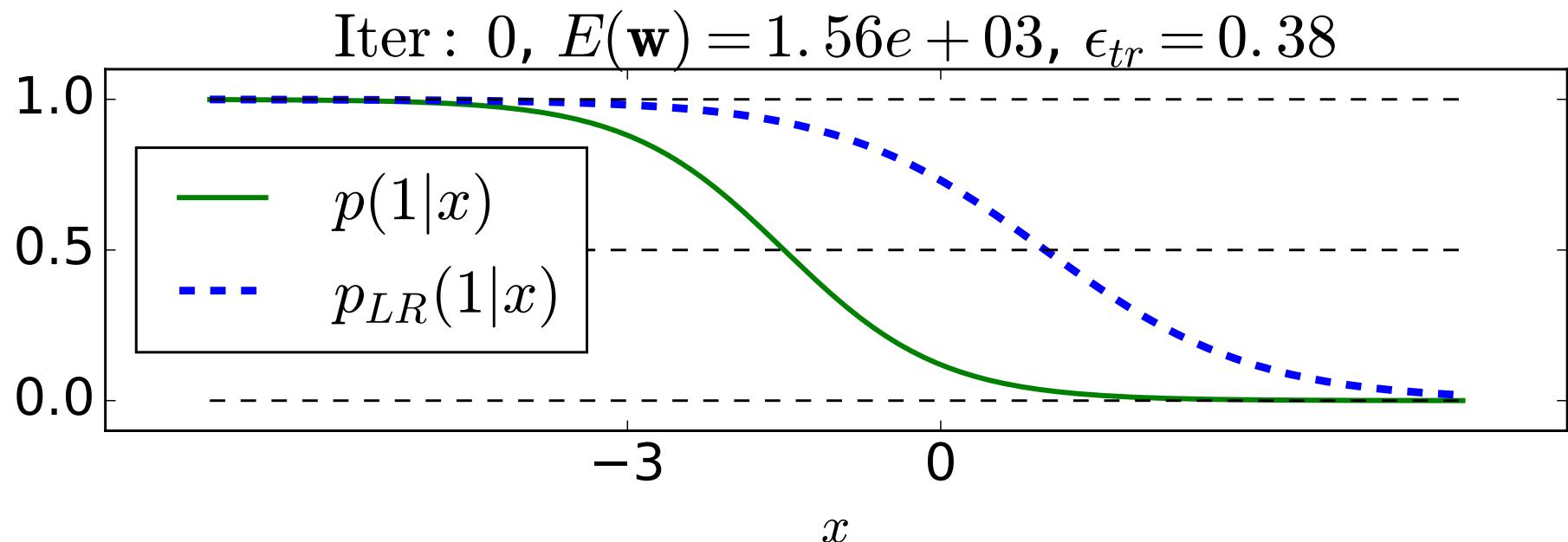
$$p(x|2) = \mathcal{N}(x|\mu_2 = 0, \sigma_2^2 = 1.5)$$

$p(1) = p(2) = 0.5$. **Bayesian error is** $\epsilon_B = 0.16$.

Example 1, Two Normal Dists with Equal Variance (2)

Initial state.

Training set: 1000 samples from each of the distributions.



$p(1|x)$: The actual conditional for the 1st class.

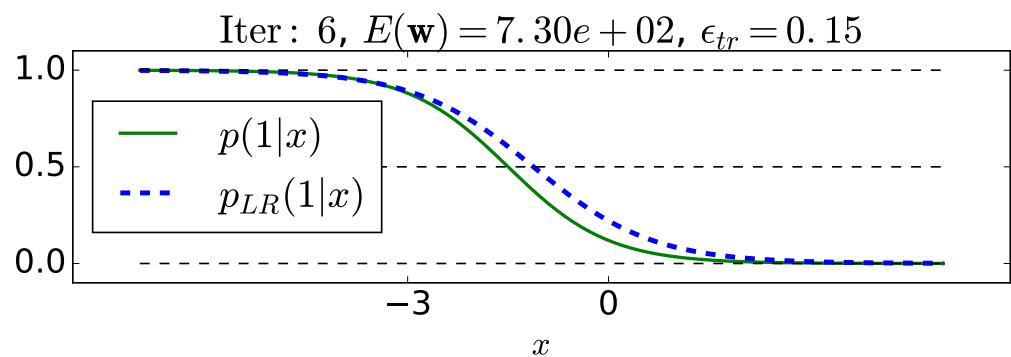
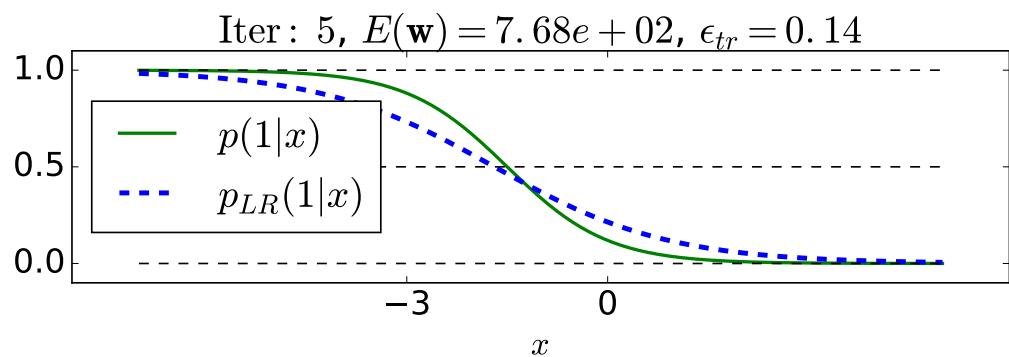
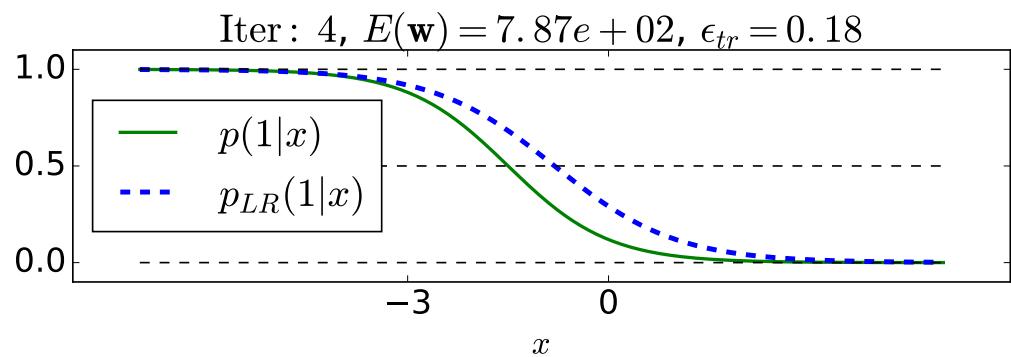
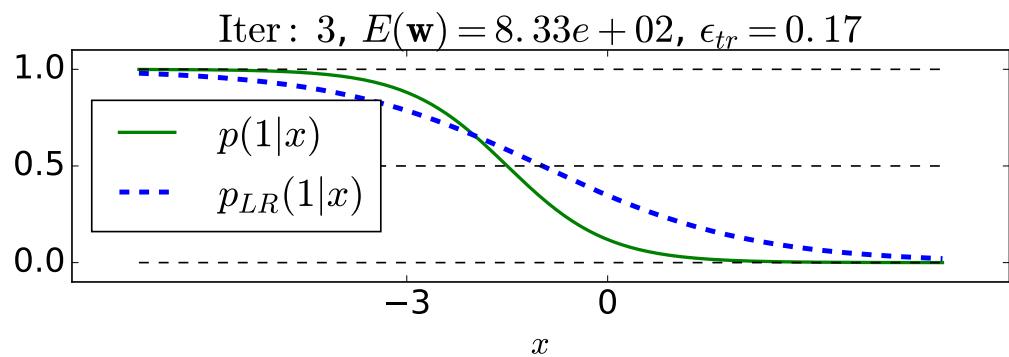
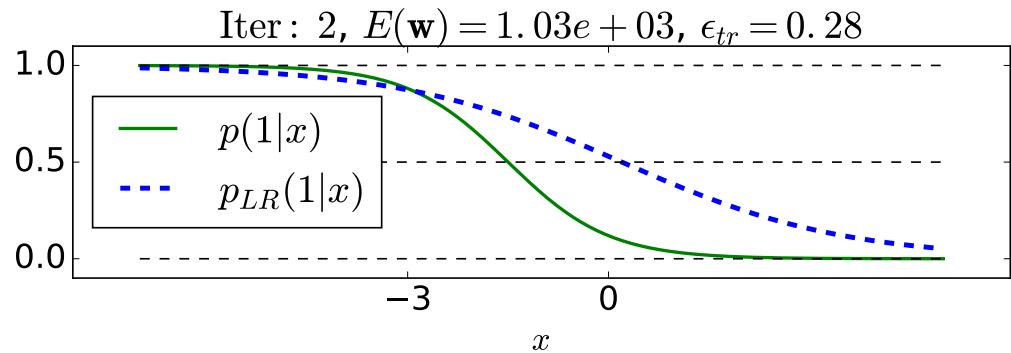
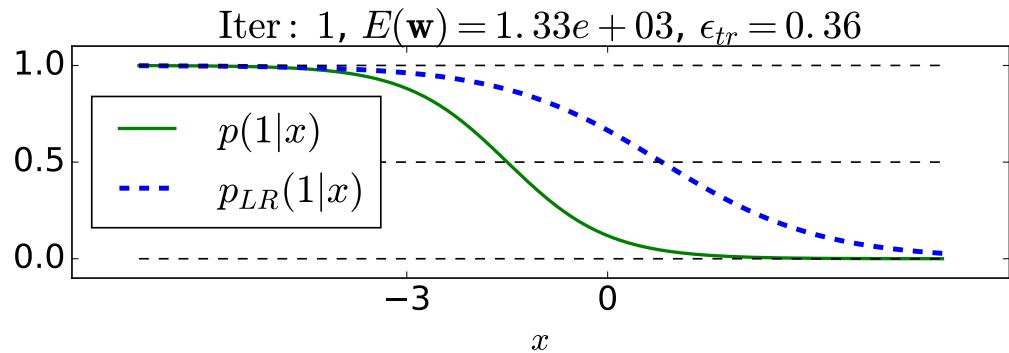
$p_{LR}(1|x)$: The conditional for the 1st class predicted by logistic regression.

$E(\mathbf{w})$: the value of cross entropy.

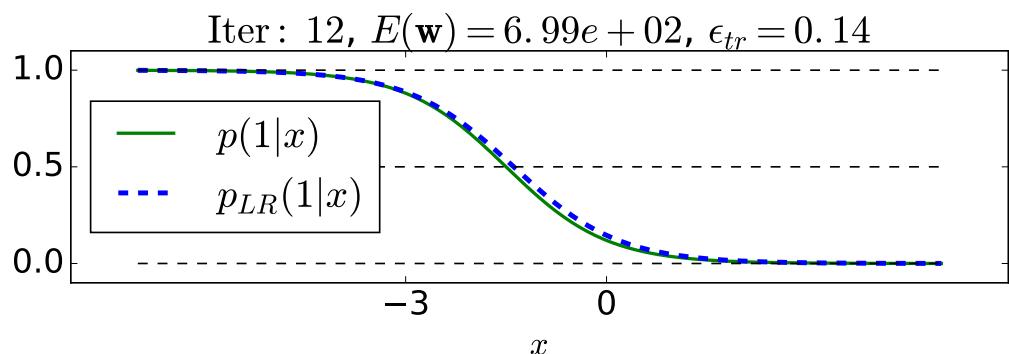
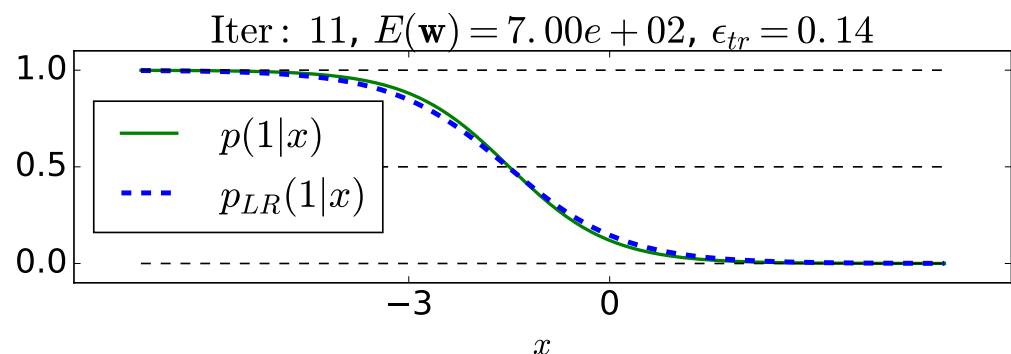
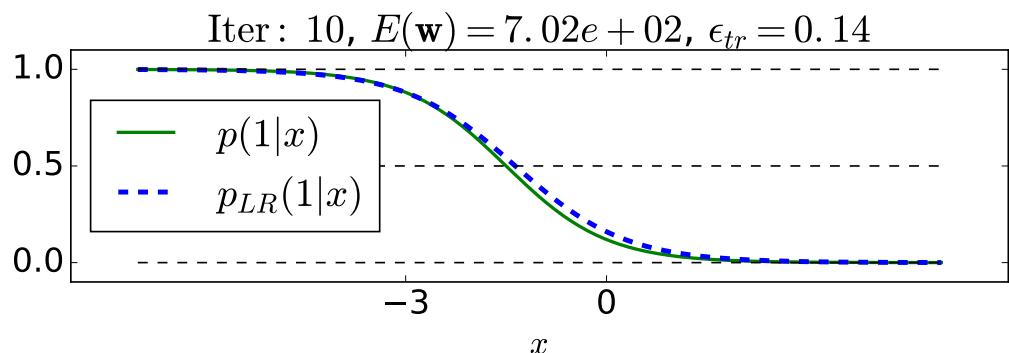
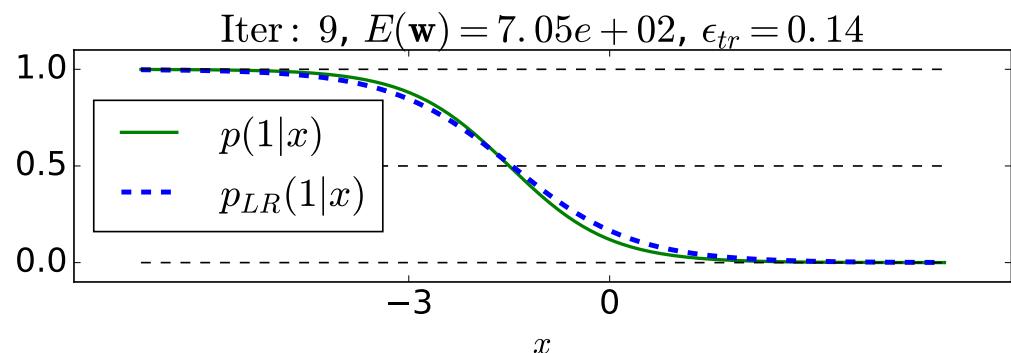
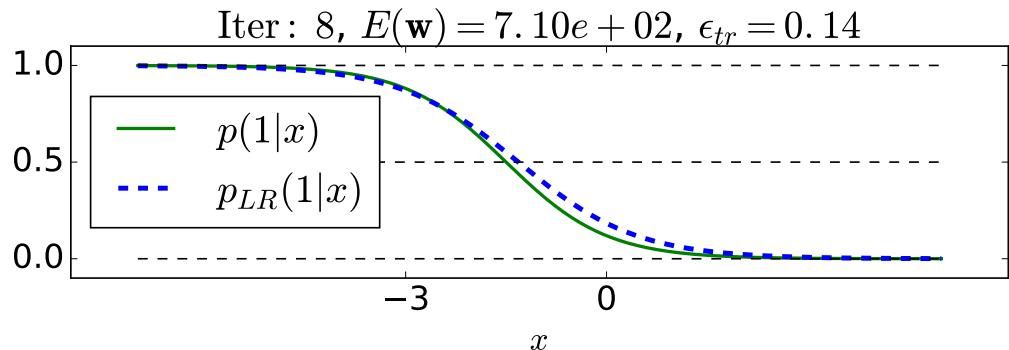
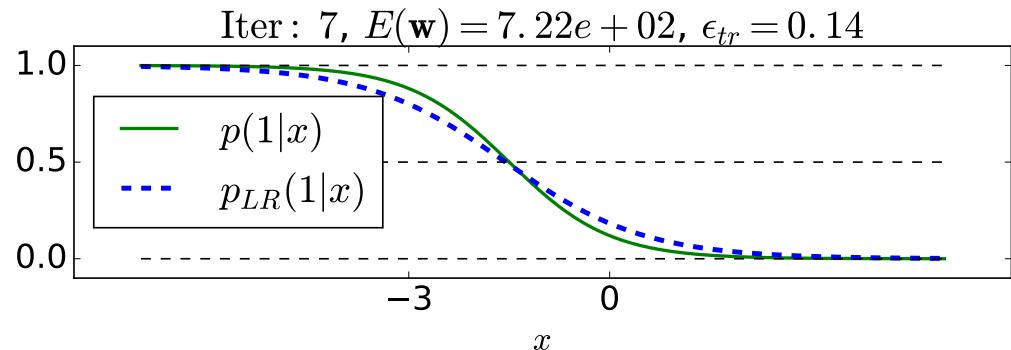
ϵ_{tr} : the training error (error on the training set.)

(initial $w = [1, -1]^\top$)

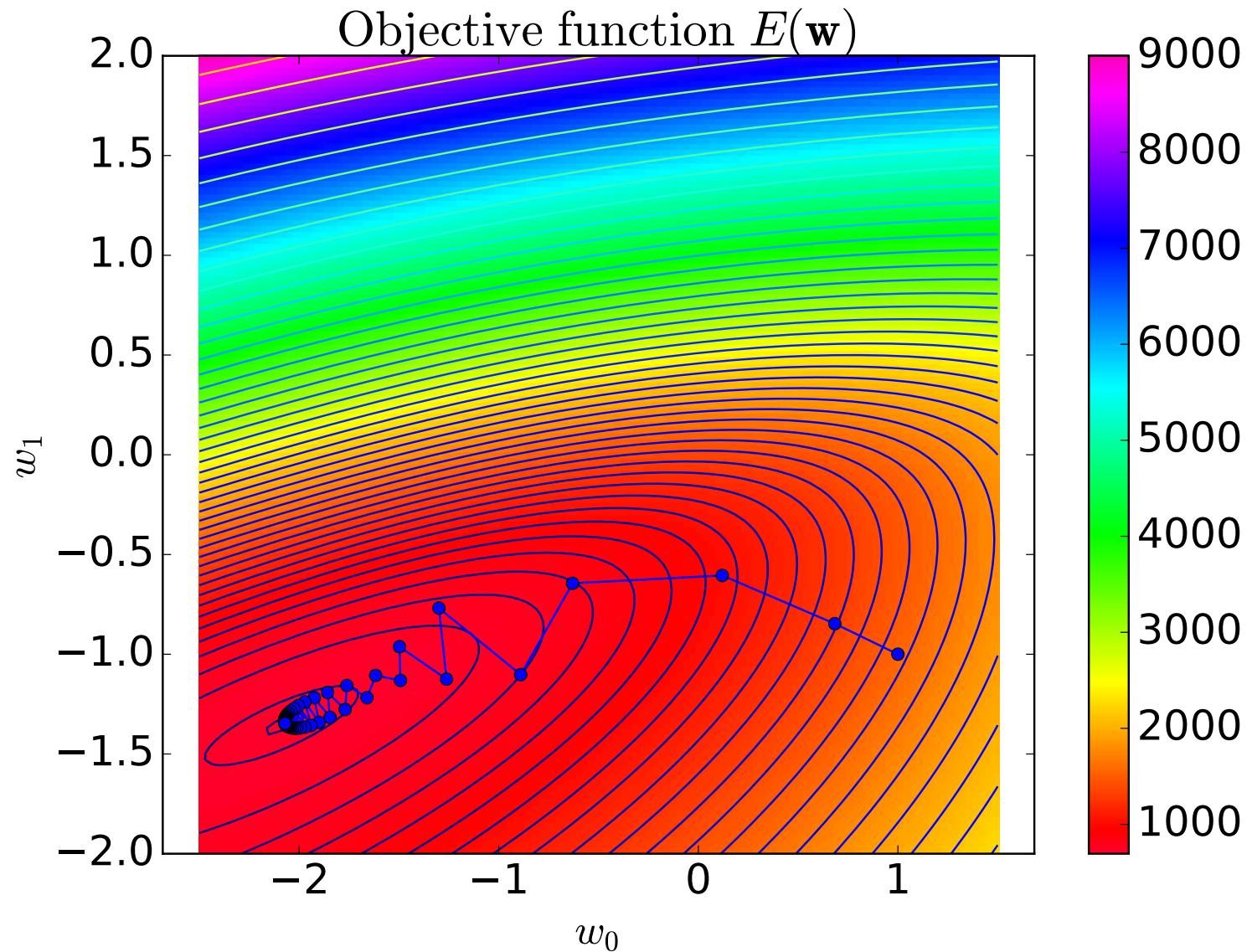
Example 1, Two Normal Distributions with Equal Variance (3)



Example 1, Two Normal Distributions with Equal Variance (4)

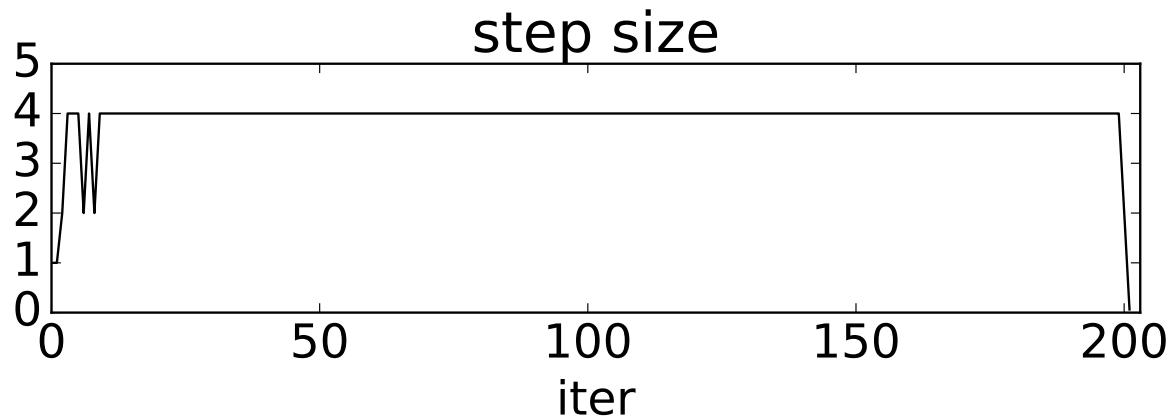
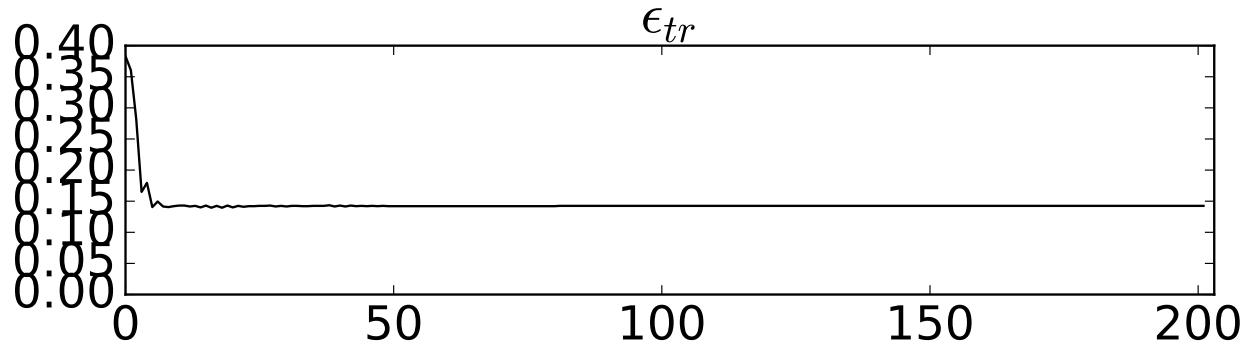
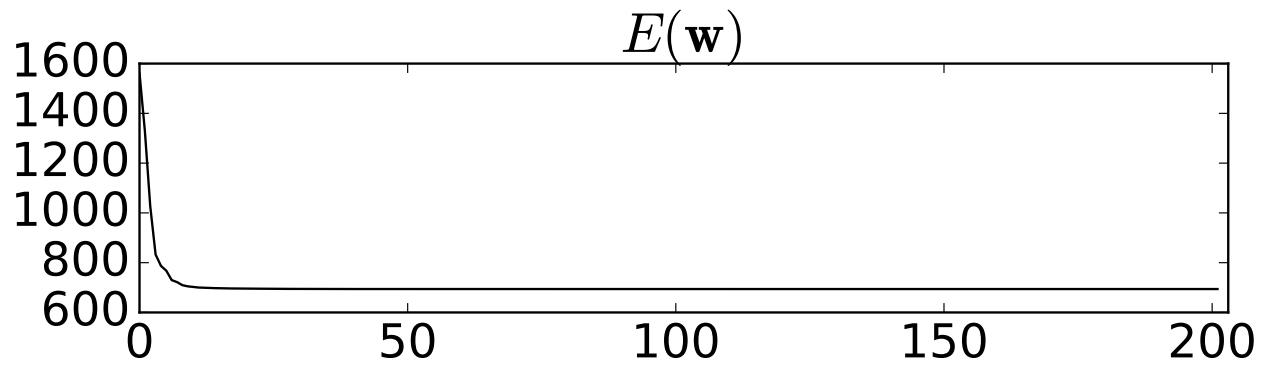


Example 1, Two Normal Distrib with Equal Variance (5)



The cross-entropy $E(\mathbf{w})$ and the progress of \mathbf{w} with iterations.

Example 1, Two Normal Distrib with Equal Variance (6)



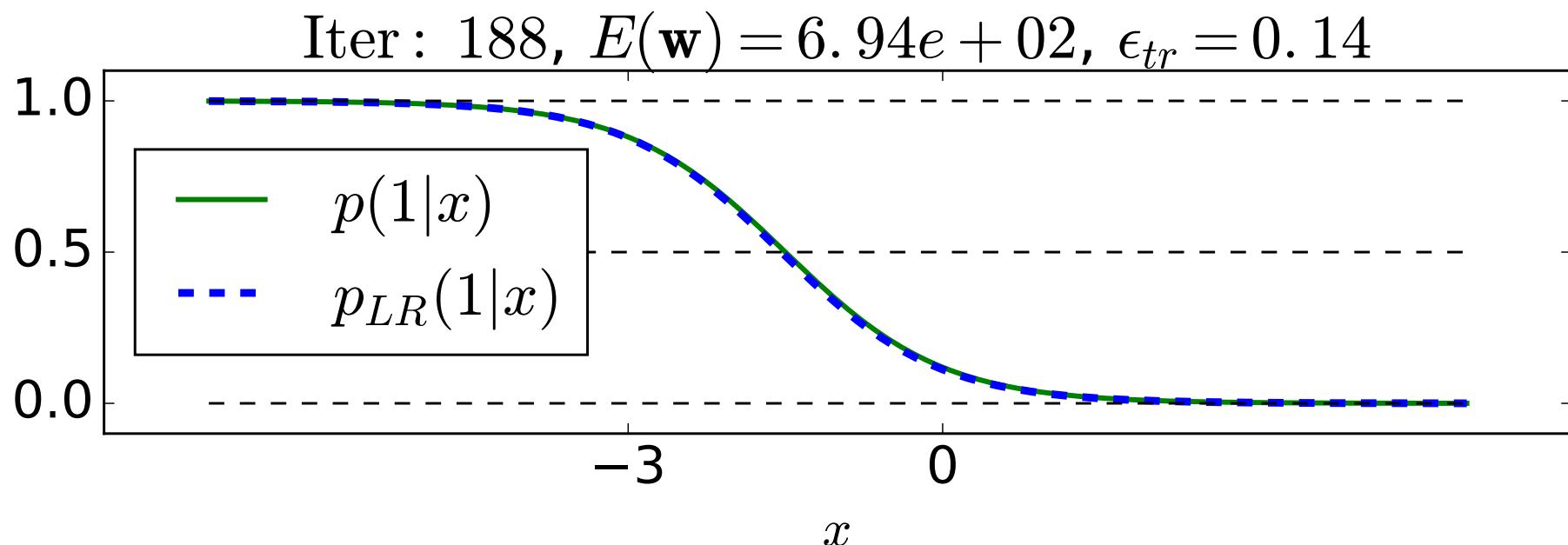
Example 1, Two Normal Distsibs with Equal Variance (7)

Things to note:

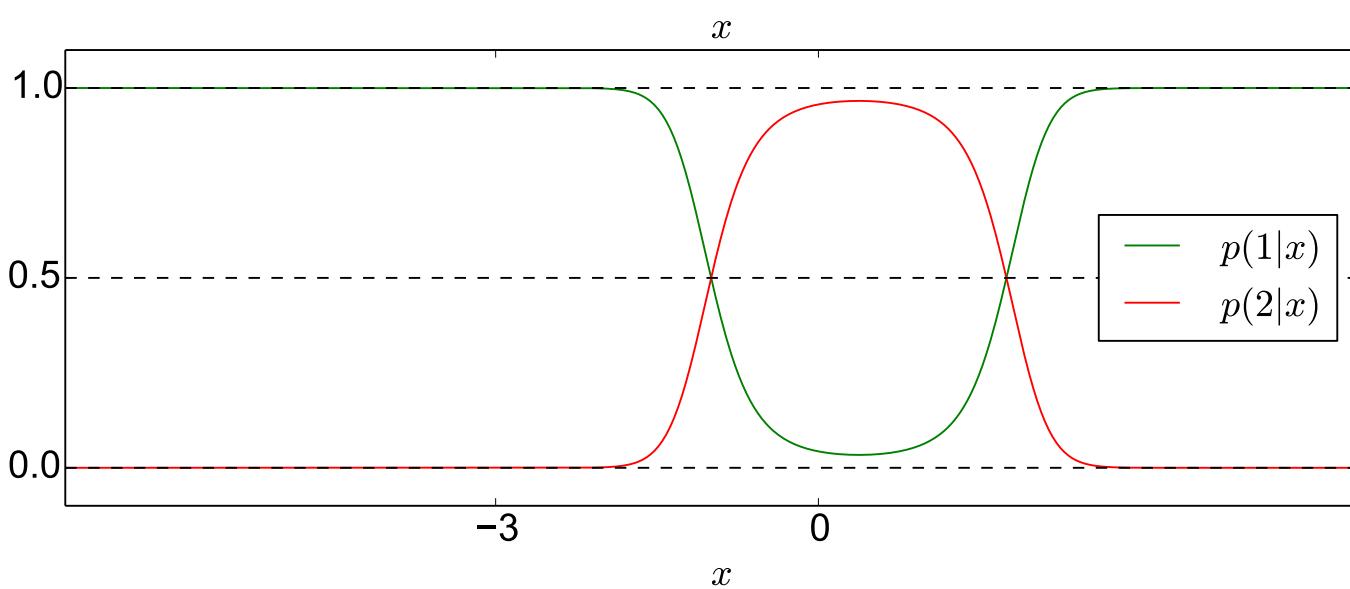
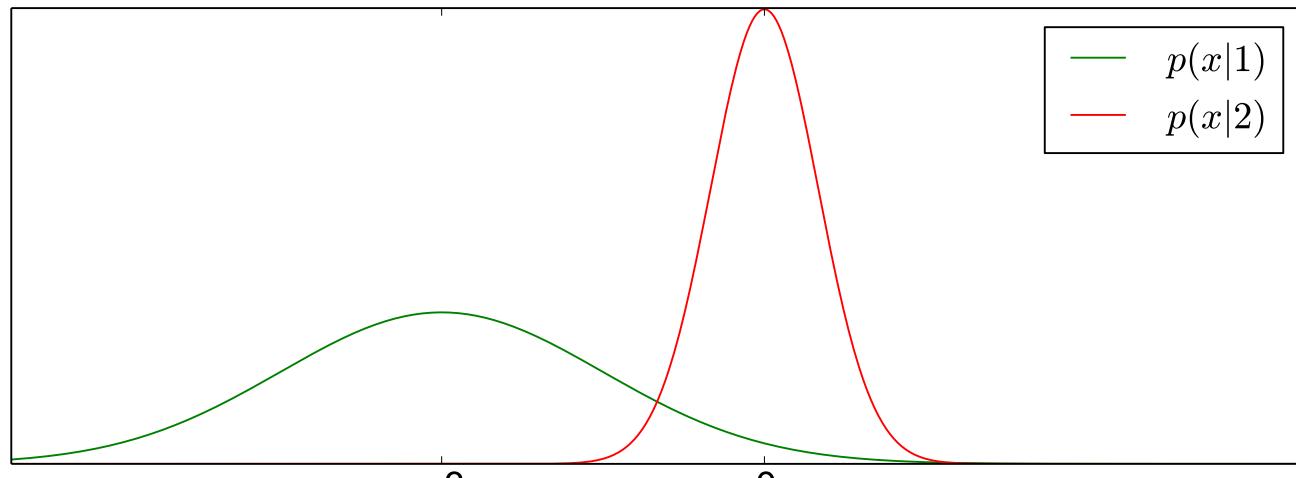
- ◆ ϵ_{tr} does not monotonically decrease with iterations. $E(\mathbf{w})$ does.
- ◆ Some intermediate ϵ_{tr} 's as well as the final one are lower than the Bayesian error $\epsilon_B = 0.16$. This is not a contradiction of the theory.

Converged state:

$$\mathbf{w} = [-2.07, -1.35]^\top.$$



Example 2, Non-Equal Variance but Different Mean (1)



$$p(x|1) = \mathcal{N}(x|\mu_1 = -3, \sigma_1^2 = 1.5)$$

$$p(x|2) = \mathcal{N}(x|\mu_2 = 0, \sigma_2^2 = 0.5)$$

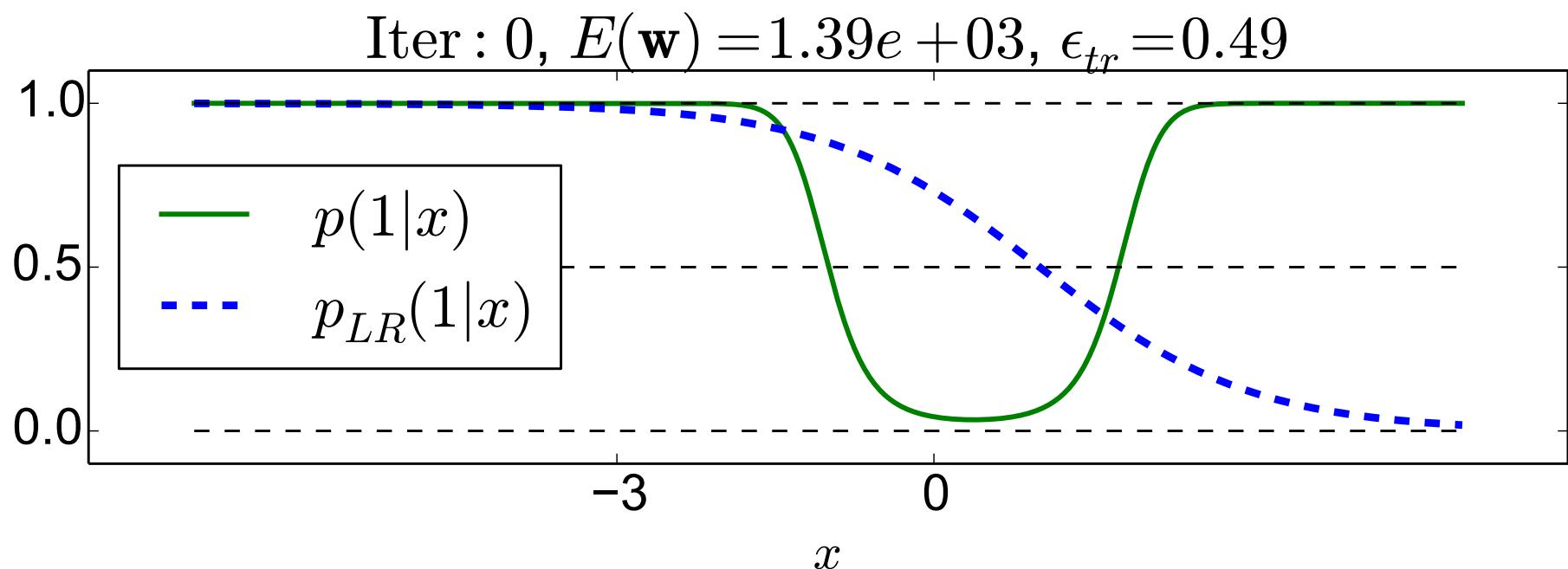
$p(1) = p(2) = 0.5$. **Bayesian error is** $\epsilon_B = 0.057$.

Example 2, Non-Equal Variance but Different Mean (2)

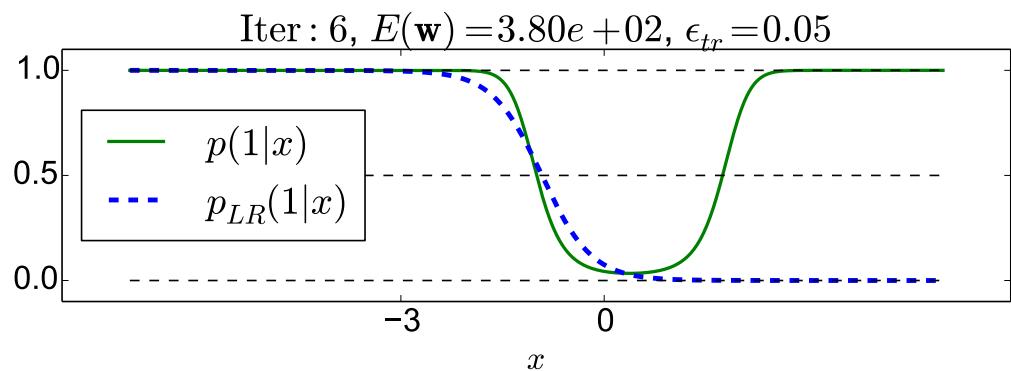
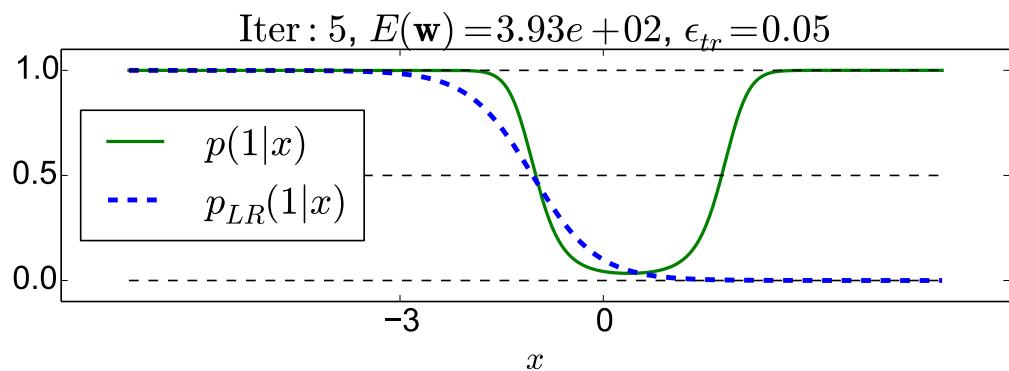
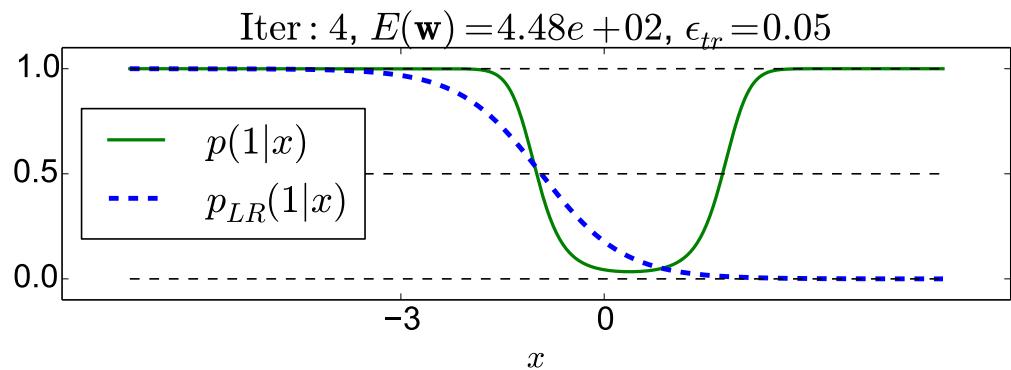
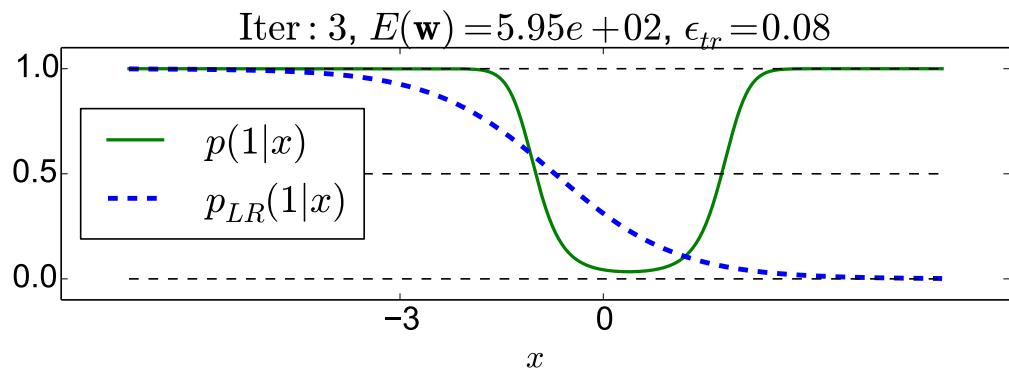
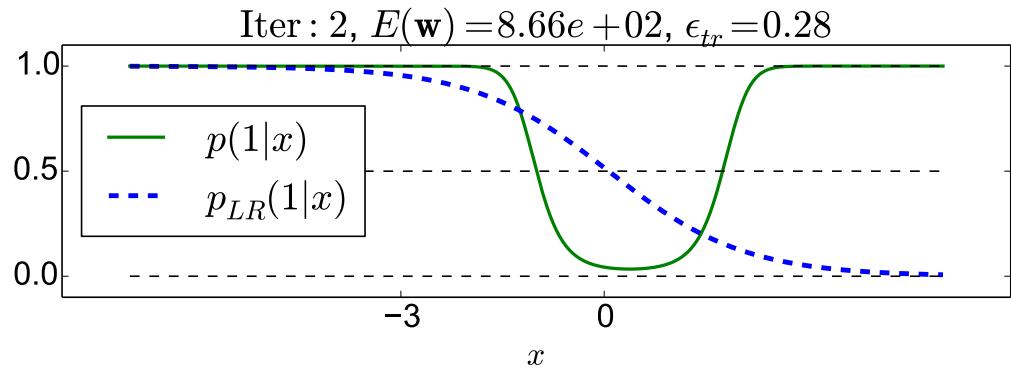
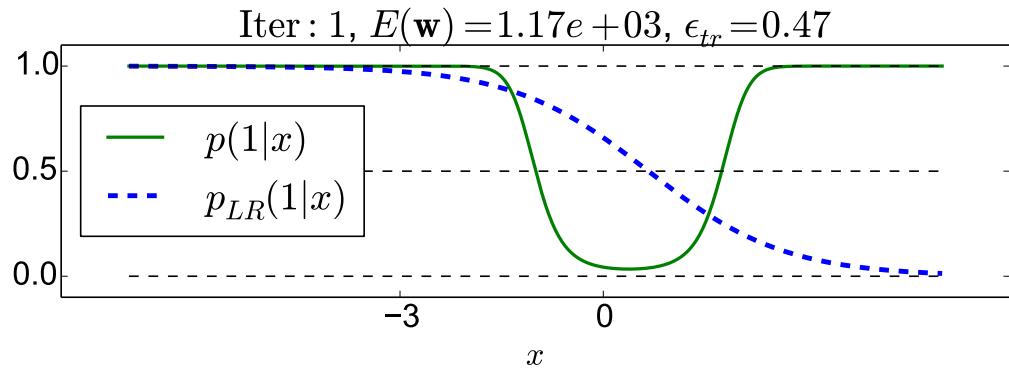
Initial state.

$$w = [1, -1]^\top.$$

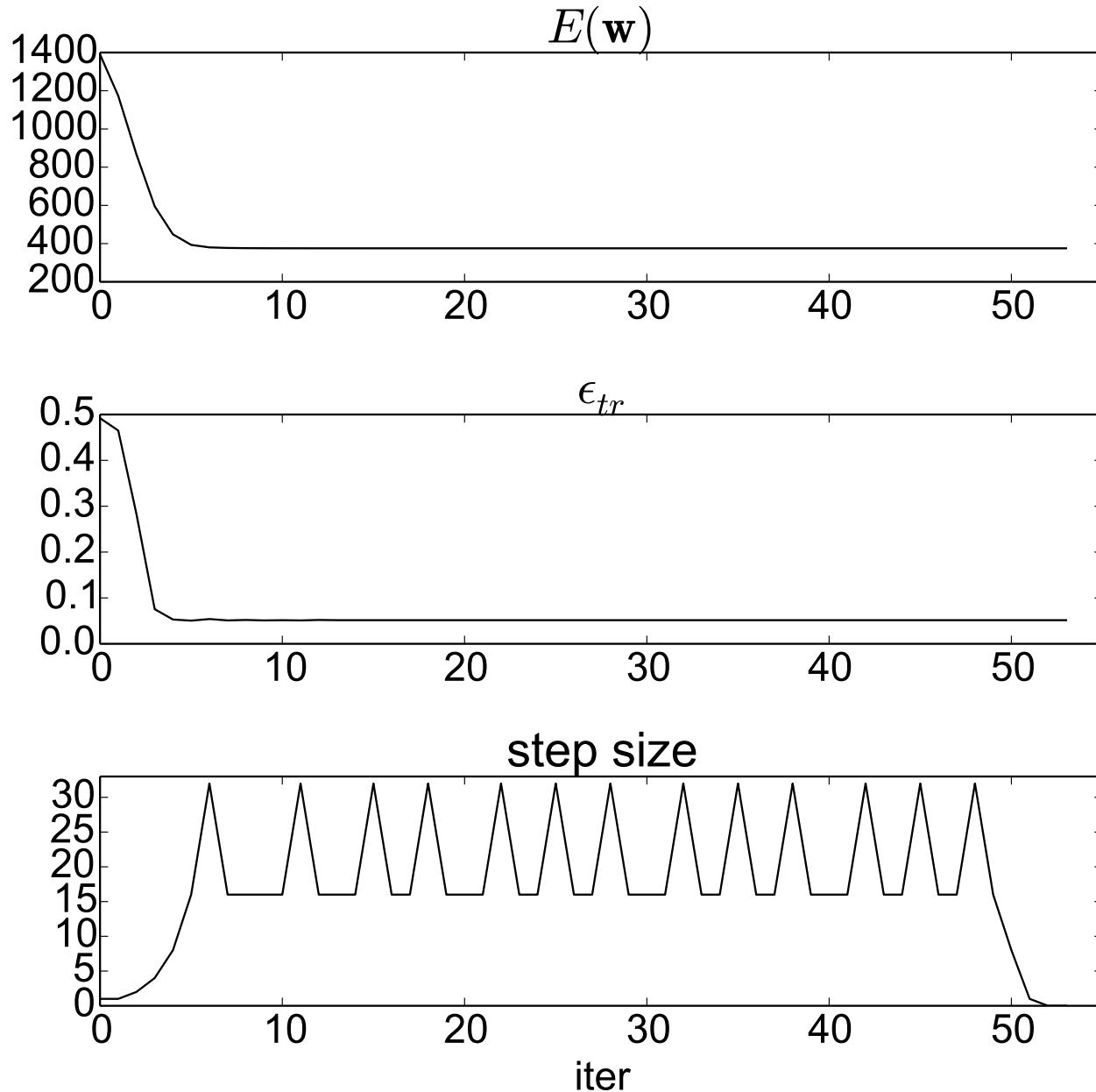
Training set: 1000 samples from each of the distributions.



Example 2, Non-Equal Variance but Different Mean (3)



Example 2, Non-Equal Variance but Different Mean (4)



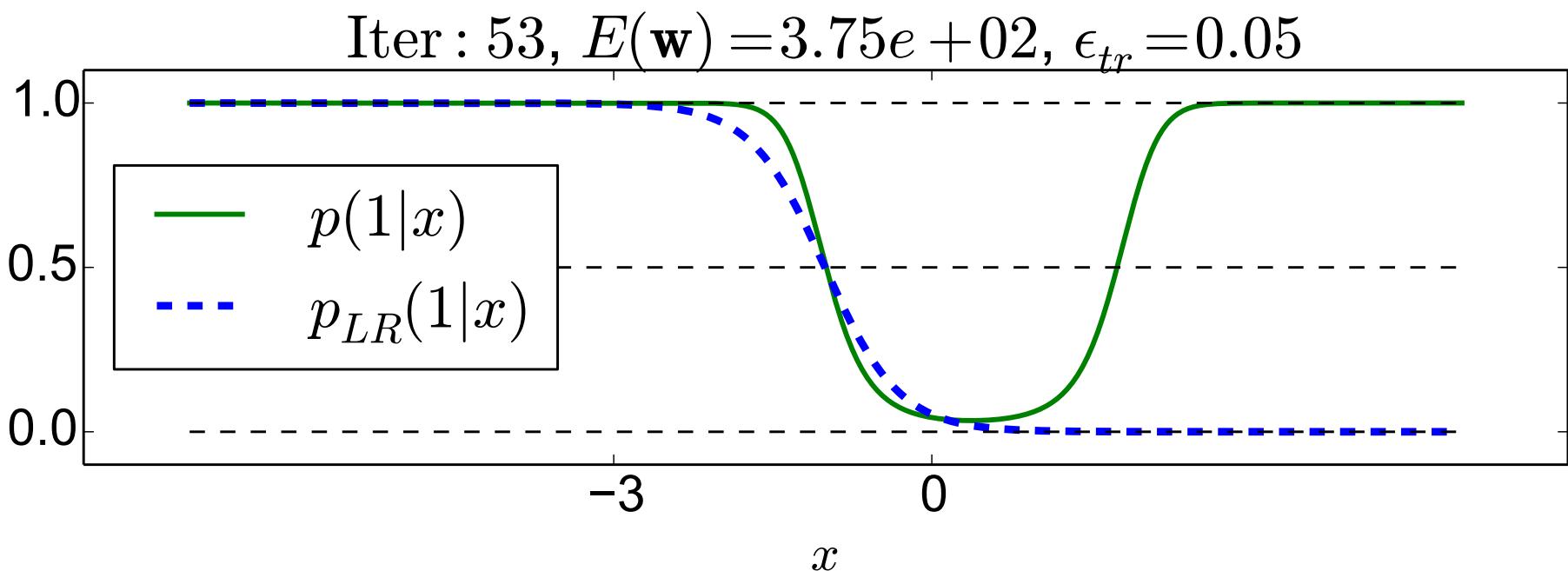
Example 2, Non-Equal Variance but Different Mean (5)

Things to note:

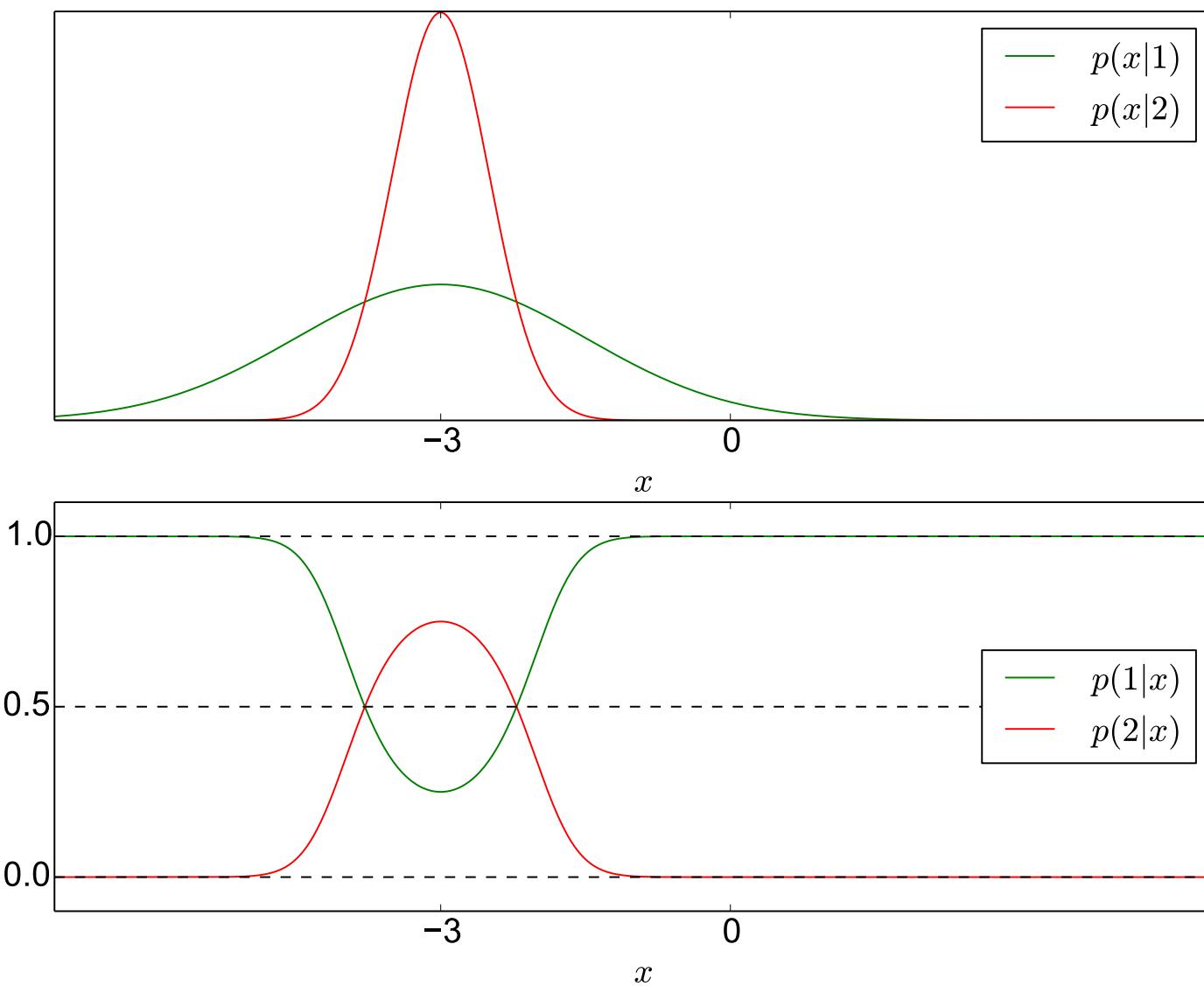
- The logistic regression cannot provide the two thresholds the optimal decision strategy requires. But it provides the one threshold which matters more in reducing the classification error (here the left one.)

Converged state.

$$w = [-2.88, -2.85]^\top.$$



Example 3, Non-Equal Variance and the Same Mean (1)



$$p(x|1) = \mathcal{N}(x|\mu_1 = -3, \sigma_1^2 = 1.5)$$

$$p(x|2) = \mathcal{N}(x|\mu_2 = -3, \sigma_2^2 = 0.5)$$

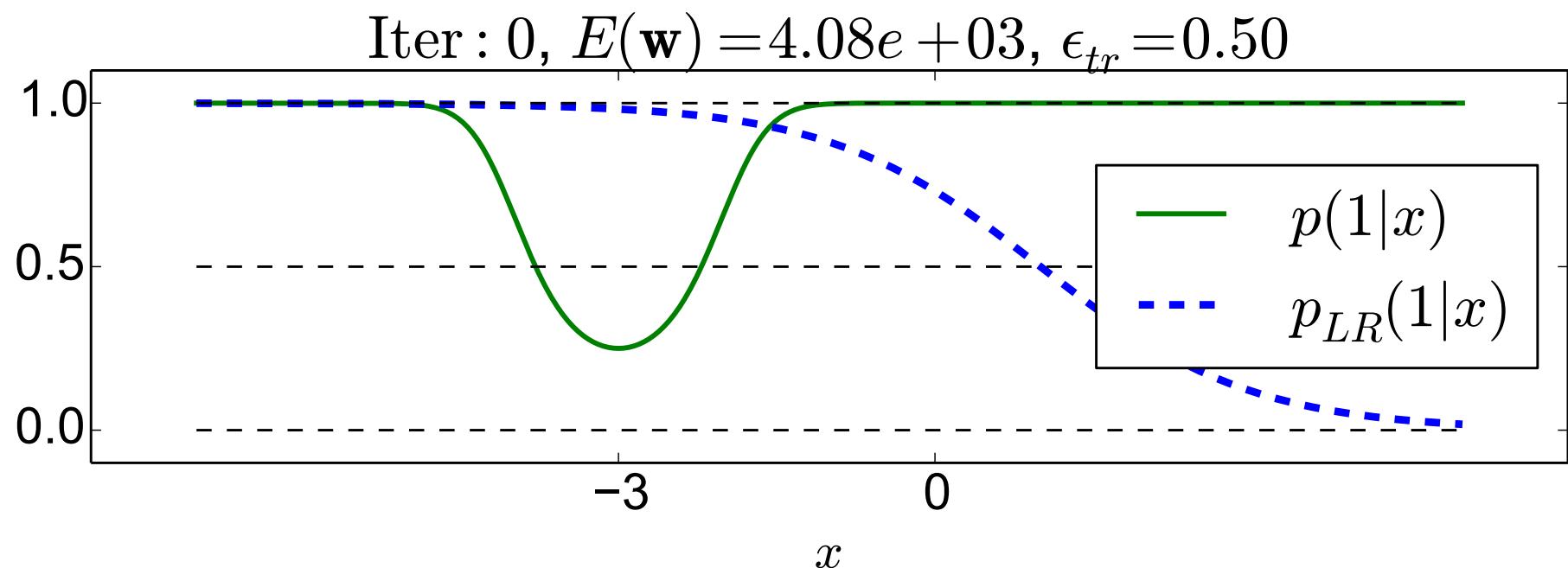
$p(1) = p(2) = 0.5$. **Bayesian error is** $\epsilon_B = 0.26$.

Example 3, Non-Equal Variance and the Same Mean (2)

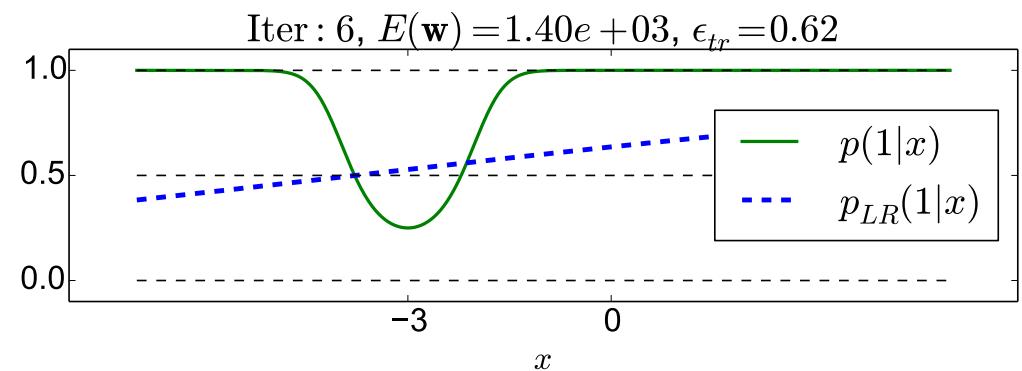
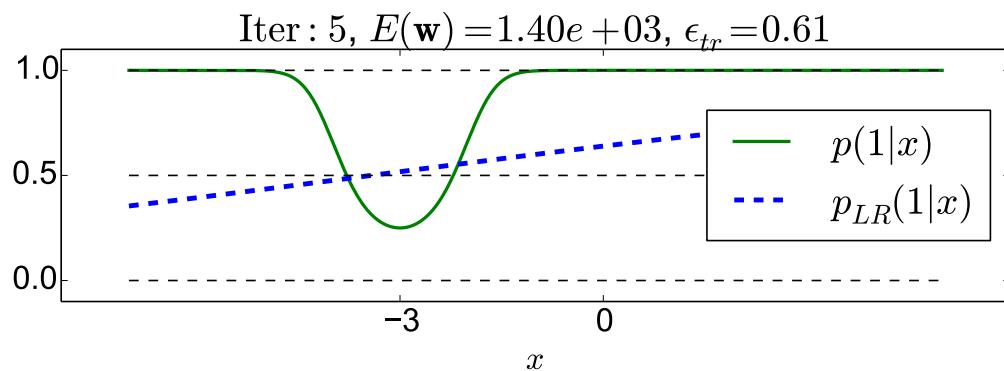
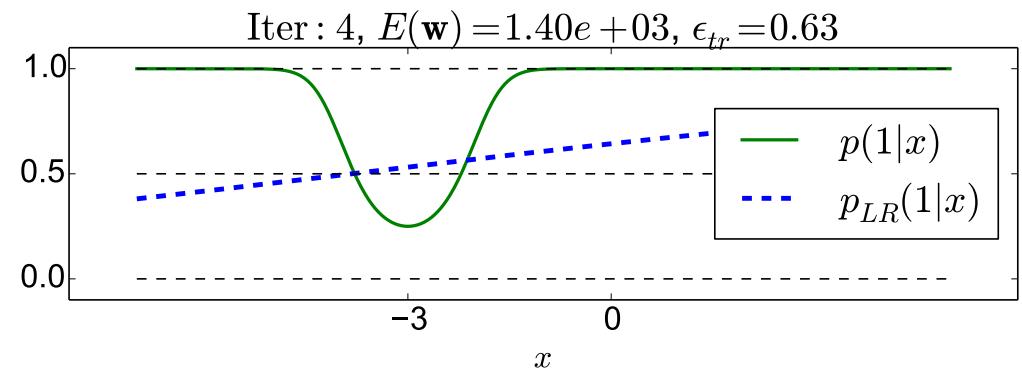
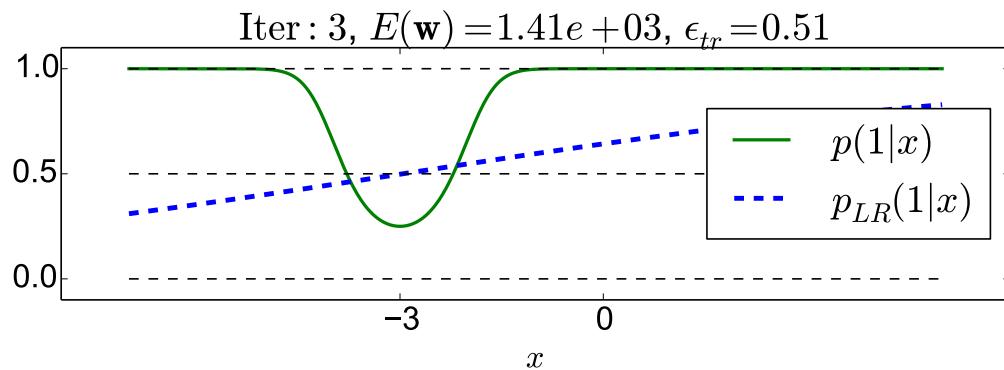
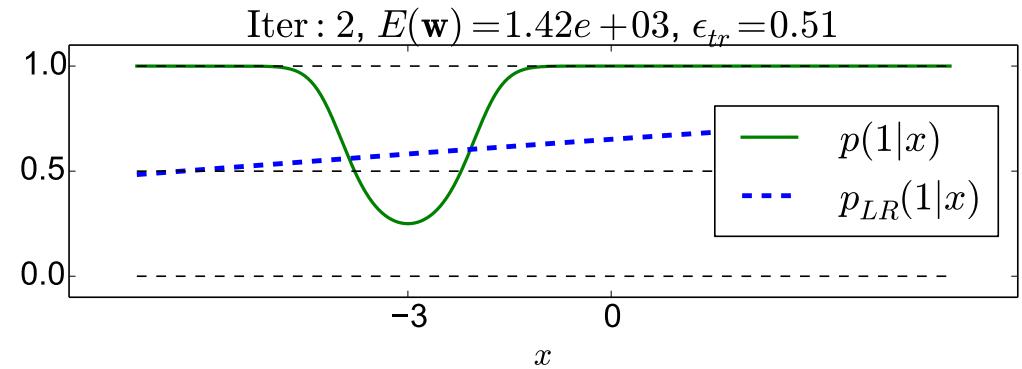
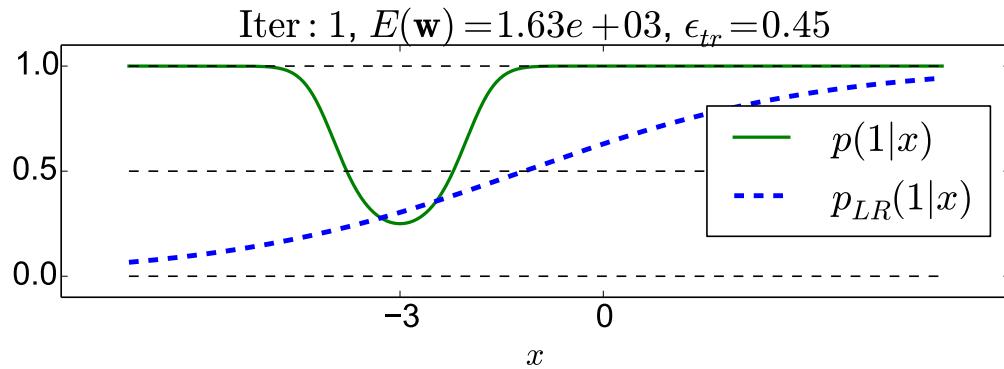
Initial state.

$$w = [1, -1]^\top.$$

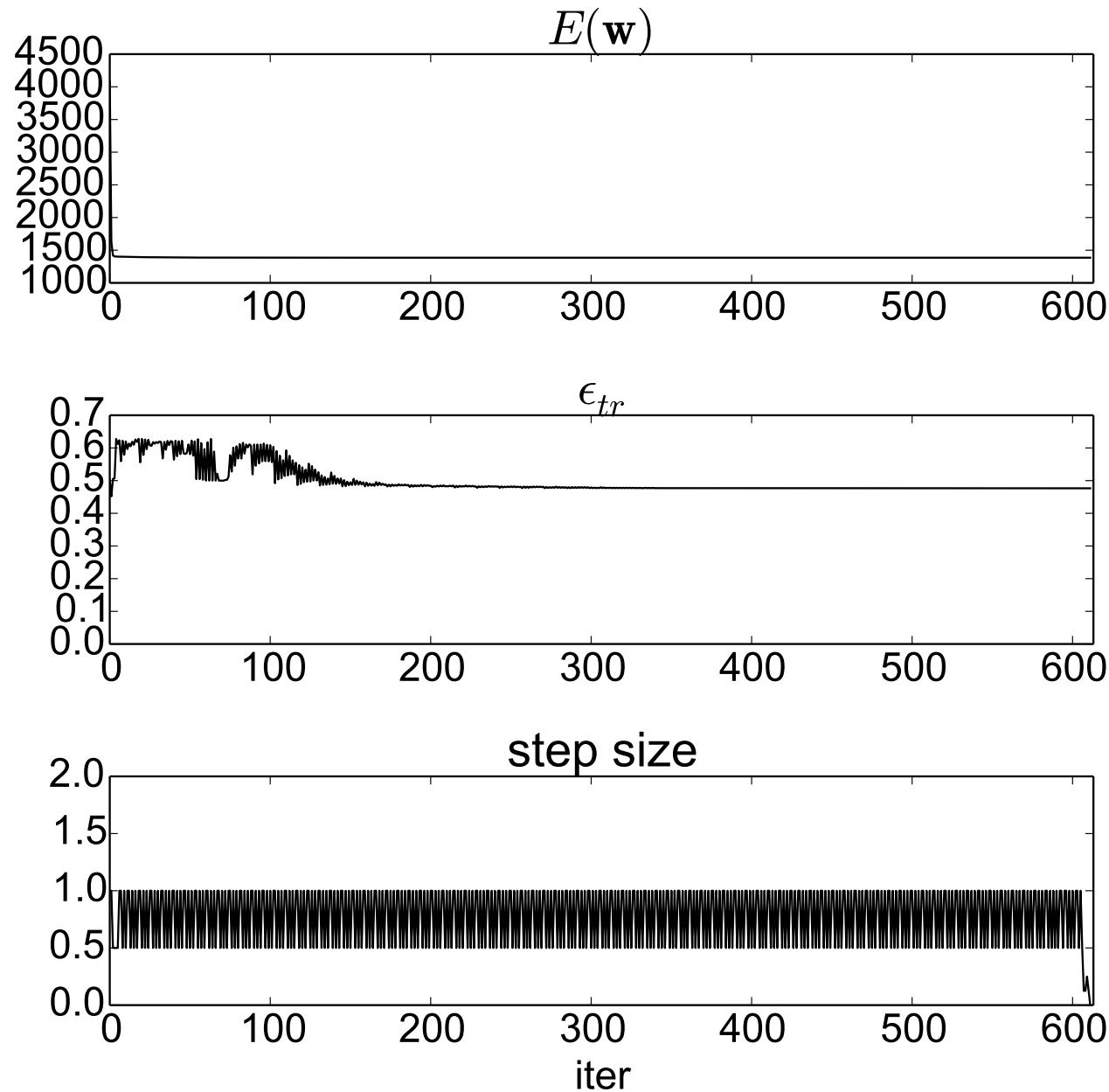
Training set: 1000 samples from each of the distributions.



Example 3, Non-Equal Variance and the Same Mean (3)



Example 3, Non-Equal Variance and the Same Mean (4)



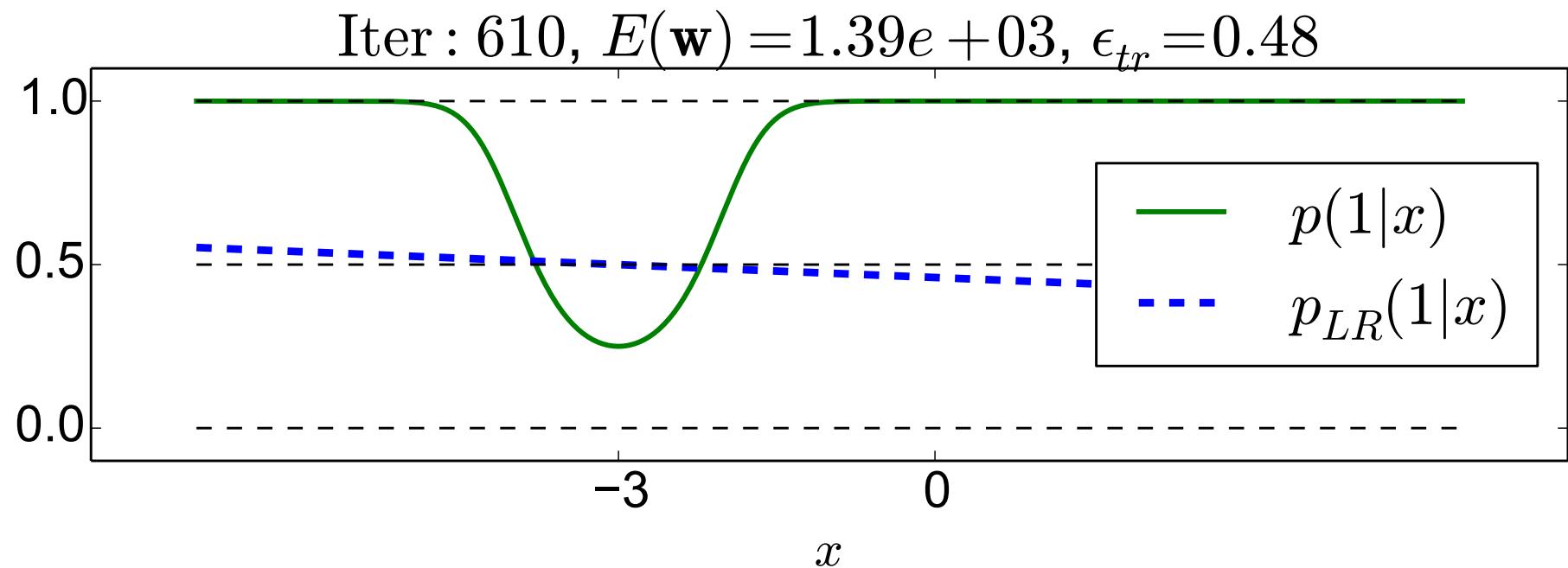
Example 3, Non-Equal Variance and the Same Mean (5)

Things to note:

- ◆ Failure case. The logistic regression cannot provide a good fit to the log odds in this case.

Final state:

$$w = [-0.161, -0.053]^\top.$$



Logistic Regression with Multiple Classes (1)

The logistic regression can be generalized to multiple classes as follows.

Each class $k \in \{1, 2, \dots, K\}$ has an associated weight vector \mathbf{w}_k .

The conditional probability for the k -th function is computed using the **softmax** function:

$$p(k|\mathbf{x}) = \frac{e^{\mathbf{w}_k \cdot \mathbf{x}}}{e^{\mathbf{w}_1 \cdot \mathbf{x}} + e^{\mathbf{w}_2 \cdot \mathbf{x}} + \dots + e^{\mathbf{w}_K \cdot \mathbf{x}}}. \quad (51)$$

Things to note:

- ◆ The above term sums to 1 (summing over k .)
- ◆ For two classes only, we get the same terms as previously, with $\mathbf{w} = \mathbf{w}_1 - \mathbf{w}_2$:

$$p(1|\mathbf{x}) = \frac{e^{\mathbf{w}_1 \cdot \mathbf{x}}}{e^{\mathbf{w}_1 \cdot \mathbf{x}} + e^{\mathbf{w}_2 \cdot \mathbf{x}}} = \frac{1}{1 + e^{-(\mathbf{w}_1 - \mathbf{w}_2) \cdot \mathbf{x}}} = \sigma((\mathbf{w}_1 - \mathbf{w}_2) \cdot \mathbf{x}) \quad (52)$$

$$p(2|\mathbf{x}) = \frac{e^{\mathbf{w}_2 \cdot \mathbf{x}}}{e^{\mathbf{w}_1 \cdot \mathbf{x}} + e^{\mathbf{w}_2 \cdot \mathbf{x}}} = \frac{1}{1 + e^{(\mathbf{w}_1 - \mathbf{w}_2) \cdot \mathbf{x}}} = \sigma(-(\mathbf{w}_1 - \mathbf{w}_2) \cdot \mathbf{x}) \quad (53)$$

Logistic Regression with Multiple Classes (2)

Conditional probabilities:

$$p(k|x) = \frac{e^{\mathbf{w}_k \cdot \mathbf{x}}}{e^{\mathbf{w}_1 \cdot \mathbf{x}} + e^{\mathbf{w}_2 \cdot \mathbf{x}} + \dots + e^{\mathbf{w}_K \cdot \mathbf{x}}}. \quad (54)$$

The training set $\mathcal{T} = \{(\mathbf{x}_1, k_1), (\mathbf{x}_2, k_2), \dots, (\mathbf{x}_N, k_N)\}$ (as before);
 The set of parameters to find: $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K]$;

The conditional log likelihood $l'(\mathbf{W})$:

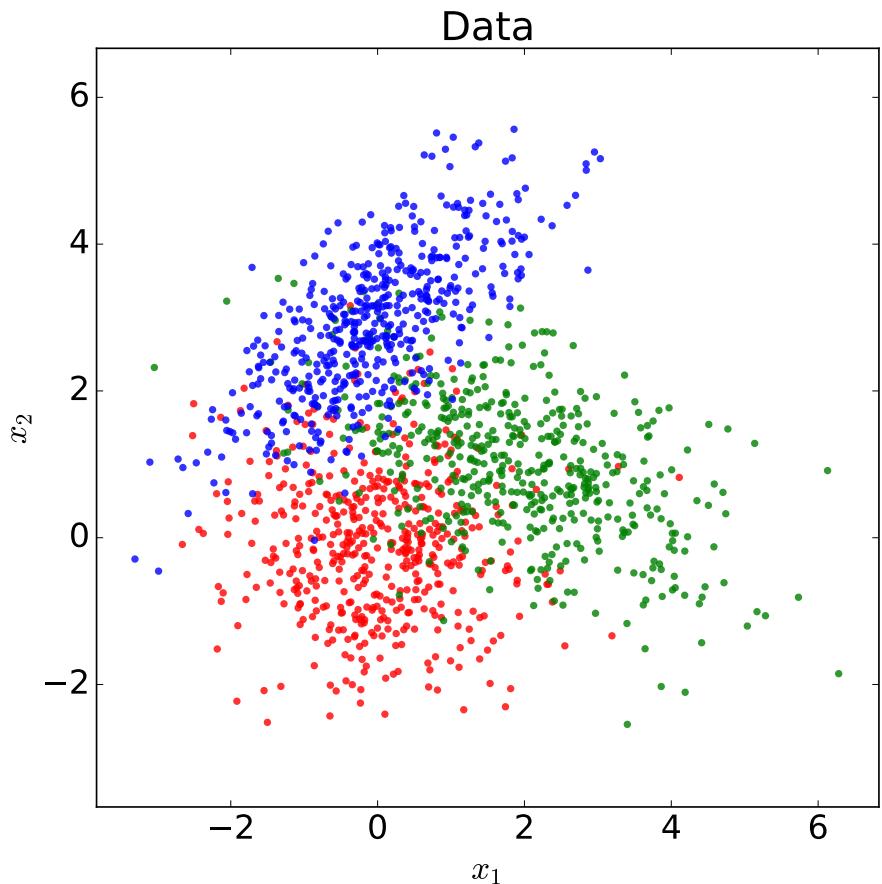
$$l'(\mathbf{W}) = \sum_{(\mathbf{x}, k) \in \mathcal{T}} \ln p(k|\mathbf{x}) = \sum_{(\mathbf{x}, k) \in \mathcal{T}} \mathbf{w}_k \cdot \mathbf{x} - \sum_{(\mathbf{x}, k) \in \mathcal{T}} \ln(e^{\mathbf{w}_1 \cdot \mathbf{x}} + e^{\mathbf{w}_2 \cdot \mathbf{x}} + \dots + e^{\mathbf{w}_K \cdot \mathbf{x}}) \quad (55)$$

Optimal parameters \mathbf{W}^* :

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmax}} l'(\mathbf{W}) = \underset{\mathbf{W}}{\operatorname{argmin}} E(\mathbf{W}) \quad (56)$$

($E(\mathbf{W})$ is cross entropy, as before)

Logistic Regression with Multiple Classes, Example (1)



Each class: 500 samples from multivariate normal distribution.

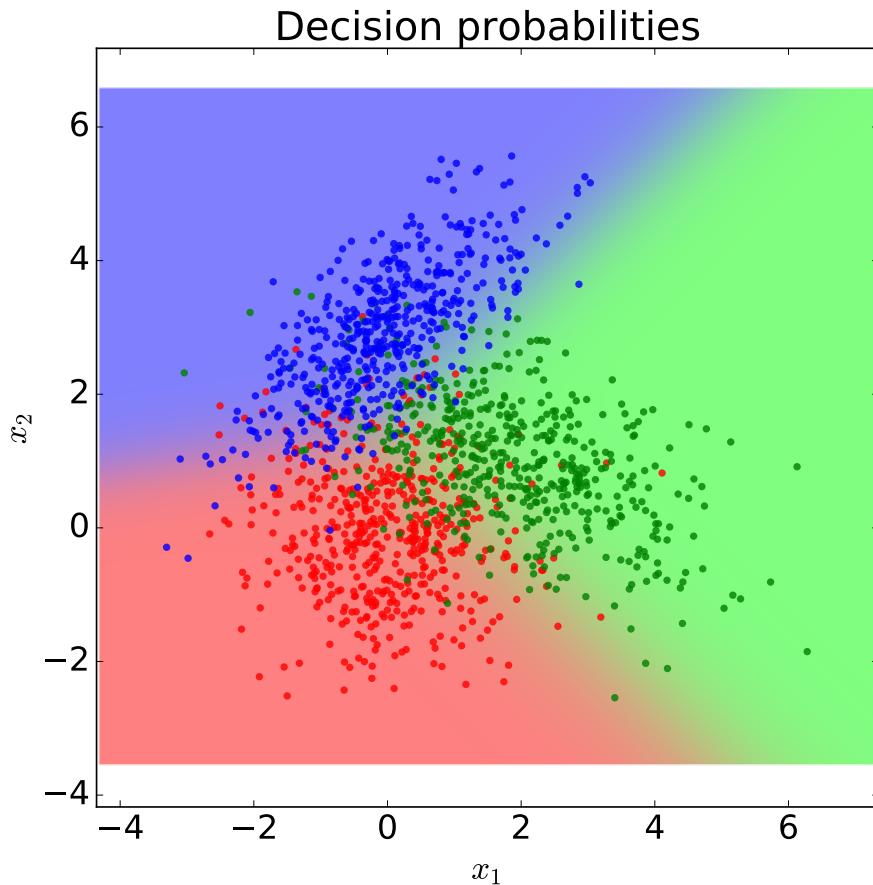
Centers:

$$(0,0), (2,1), (0,3).$$

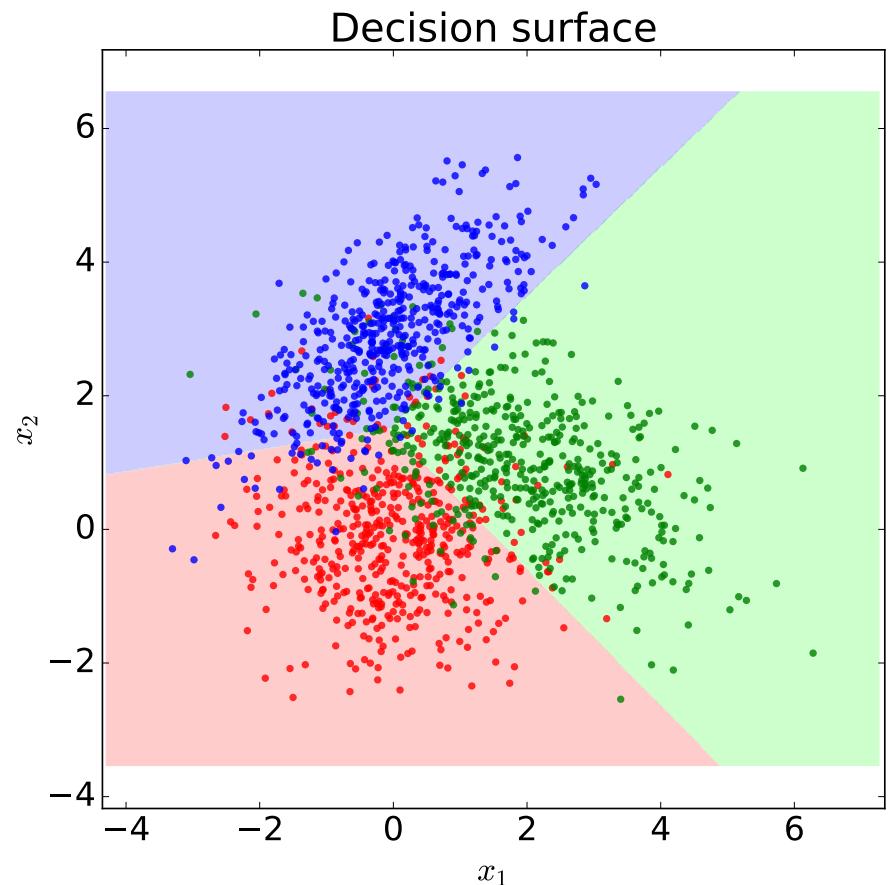
Cov. matrices:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 2 & -0.7 \\ -0.7 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}.$$

Logistic Regression with Multiple Classes, Example (2)



Conditional probabilities $p(k|\mathbf{x})$ (coded by color intensity)



Decisions, coded by class' colors. Note the linearity of decision boundaries.

Perceptron Classifier

Empirical vs Structural Risk Minimization

lecturer: Jiří Matas, matas@cmp.felk.cvut.cz

authors: V. Hlaváč, J. Matas, O. Drbohlav

Czech Technical University, Faculty of Electrical Engineering
Department of Cybernetics, Center for Machine Perception
121 35 Praha 2, Karlovo nám. 13, Czech Republic

<http://cmp.felk.cvut.cz>

Last update: 29/Oct/2020

LECTURE PLAN

- ◆ The problem of classifier design.
- ◆ Empirical and structural risk.
- ◆ Perceptron algorithms.
- ◆ Optimal separating plane with the Kozinec algorithm.

Classifier Design (Bayes)

The object of interest is characterised by observable properties $\mathbf{x} \in X$ and its class membership (unobservable, hidden state) $k \in K$, where X is the space of observations and K the set of hidden states.

The objective of classifier design is to find a strategy $q^*: X \rightarrow K$ that has some optimal properties.

Bayesian decision theory solves the problem of minimisation of risk

$$R(q) = \sum_{\mathbf{x}, k} p(\mathbf{x}, k) W(q(\mathbf{x}), k)$$

given the following quantities:

- ◆ $p(\mathbf{x}, k), \forall \mathbf{x} \in X, k \in K$ – the statistical model of the dependence of the observable properties (measurements) on class membership (almost always unknown)
- ◆ $W(q(\mathbf{x}), k)$ the loss of decision $q(\mathbf{x})$ if the true class is k

Classifier Design via Parameter Estimation

- ◆ Assume $p(\mathbf{x}, k)$ have a particular form, e.g. Gaussian (mixture), piece-wise constant, etc., with a finite (i.e. small) number of parameters θ_k .
- ◆ Estimate the parameters from the using training set \mathcal{T}
- ◆ Solve the classifier design problem (e.g. risk minimisation), substituting the estimated $\hat{p}(\mathbf{x}, k)$ for the true (and unknown) probabilities $p(\mathbf{x}, k)$

? : What estimation principle should be used?

- : There is no direct relationship between known properties of estimated $\hat{p}(\mathbf{x}, k)$ and the properties (typically the risk) of the obtained classifier $q'(\mathbf{x})$
- : If the true $p(\mathbf{x}, k)$ is not of the assumed form, $q'(\mathbf{x})$ may be arbitrarily bad, even for the asymptotic case $L \rightarrow \infty$.
- + : Implementation is often straightforward, especially if parameters θ_k for each class are assumed independent.
- + : Performance on test data can be predicted by crossvalidation.

Classifier Design via Risk Minimization

- ◆ How about constructing a ‘direct’ classifier (without estimating $p(\mathbf{x}, k)$)?
- ◆ Choose a class Q of decision functions (classifiers) $q : X \rightarrow K$.
- ◆ Find $q^* \in Q$ minimising some criterion function on the training set that approximates the risk $R(q)$ (true risk is unknown).
- ◆ Empirical risk R_{emp} (training set error) minimization. True risk approximated by

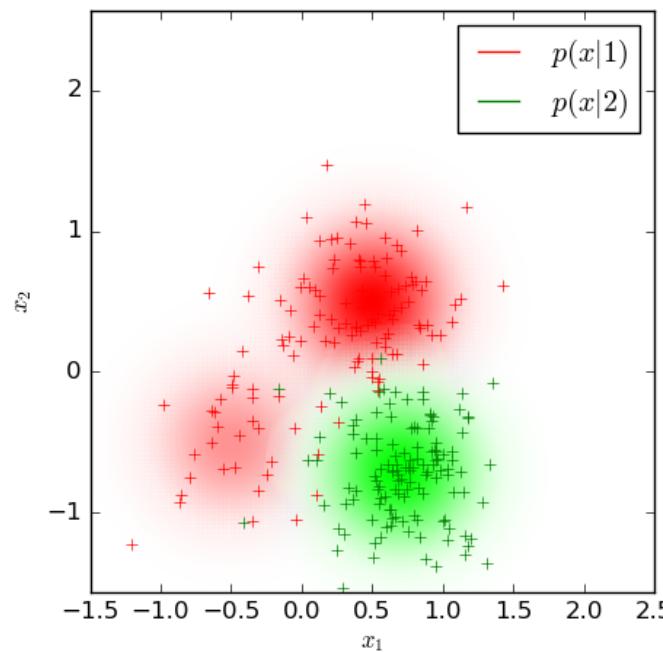
$$R_{emp}(q_\theta(\mathbf{x})) = \frac{1}{L} \sum_{i=1}^L W(q_\theta(\mathbf{x}_i), k_i),$$

$$\theta^* = \operatorname{argmin}_{\theta} R_{emp}(q_\theta(\mathbf{x}))$$

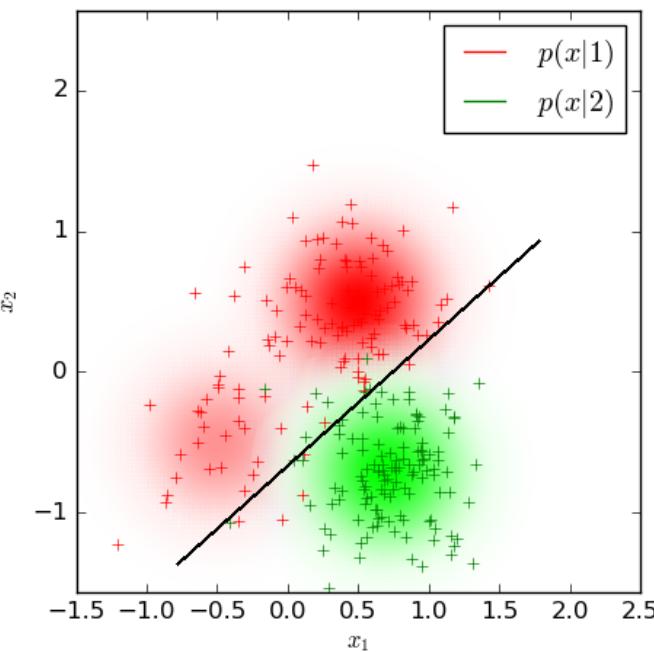
Examples: Perceptron, Neural nets (Back-propagation), etc.

Empirical Risk Minimization Can Lead To Overfitting

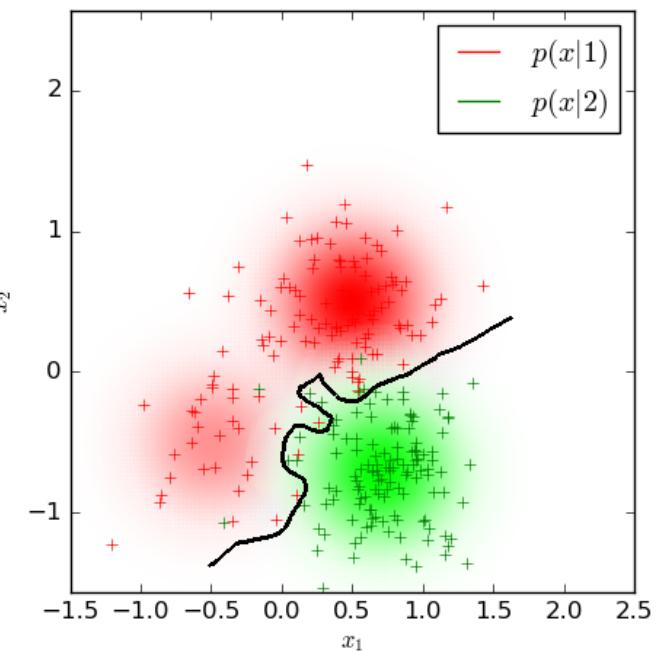
- ◆ How wide a class Q of classifiers $q_\theta(\mathbf{x})$ should be used?
- ◆ The problem of generalization is a key problem of pattern recognition: a small empirical risk R_{emp} need not imply a small true expected risk R .



class distributions
(equal prior probs)
and samples



underfit



overfit

As discussed previously, a suitable model can be selected e.g. using cross-validation.

Structural Risk Minimization Principle (1)

We would like to minimise the risk

$$R(q) = \sum_{x,k} p(\mathbf{x}, k) W(q_\theta(\mathbf{x}), k),$$

but $p(\mathbf{x}, k)$ is unknown.

Vapnik and Chervonenkis inequality:

$$\Pr \left(R(q) \leq R_{emp}(q) + \sqrt{\frac{1}{N} \left[h \left(\log \left(\frac{2N}{h} \right) + 1 \right) - \log \left(\frac{\eta}{4} \right) \right]} \right) = 1 - \eta, \quad (1)$$

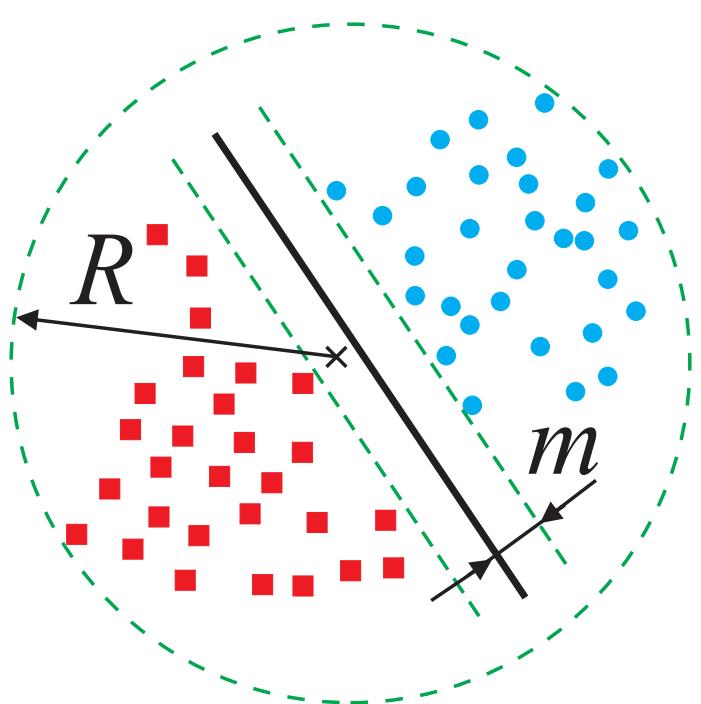
where N is the training set size, h is the VC dimension (capacity) of the class of strategies Q and $0 < \eta < 1$. Valid only for $h \ll N$.

Notes:

- + the term does not depend on the unknown $p(\mathbf{x}, k)$
- + VC dimension h known for some classes of Q , e.g. linear classifiers.

Structural Risk Minimization Principle (2)

- ◆ There are more types of upper bounds on the expected risk.
E.g. for linear discriminant functions



VC dimension (capacity)

$$h \leq \frac{R^2}{m^2} + 1$$

- ◆ Examples of learning algorithms: SVM or ε -Kozinec.

$$(\mathbf{w}^*, b^*) = \operatorname{argmax}_{\mathbf{w}, b} \min \left(\min_{x \in X_1} \frac{\mathbf{w} \cdot \mathbf{x} + b}{\|\mathbf{w}\|}, \min_{x \in X_2} -\frac{\mathbf{w} \cdot \mathbf{x} + b}{\|\mathbf{w}\|} \right).$$

Empirical Risk Minimisation, Notes

Is then empirical risk minimisation = minimisation of training set error, e.g. neural networks with backpropagation, useless? No, because:

- structural risk may be so large that the upper bound is useless.
- + Vapnik's theory justifies using empirical risk minimisation on classes of functions with finite VC dimension.
- + Vapnik suggests learning with progressively more complex classes Q .
- + Empirical risk minimisation is computationally hard (impossible for large L). Most classes of decision functions Q where empirical risk minimisation (at least local) can be efficiently organised are often useful.

Linear Classifiers

- ◆ For some statistical models, the Bayesian or non-Bayesian strategy is implemented by a linear discriminant function.
- ◆ Capacity (VC dimension) of linear strategies in an n -dimensional space is $n + 2$. Thus, the learning task is well-posed, i.e., strategy tuned on a finite training multiset does not differ much from correct strategy found for a statistical model.
- ◆ There are efficient learning algorithms for linear classifiers.
- ◆ Some non-linear discriminant functions can be implemented as linear after the feature space transformation.

Linear Separability (Two Classes)

Consider a dataset $\mathcal{T} = \{(\mathbf{x}_1, k_1), (\mathbf{x}_2, k_2), \dots, (\mathbf{x}_L, k_L)\}$, with $\mathbf{x}_i \in \mathbb{R}^D$ and $k_i \in \{-1, 1\}$ ($i = 1, 2, \dots, L.$)

The data are **linearly separable** if there exists a hyperplane which divides \mathbb{R}^D to two half-spaces such that the data of a given class are all in one half-space.

Formally, the data are linearly separable if

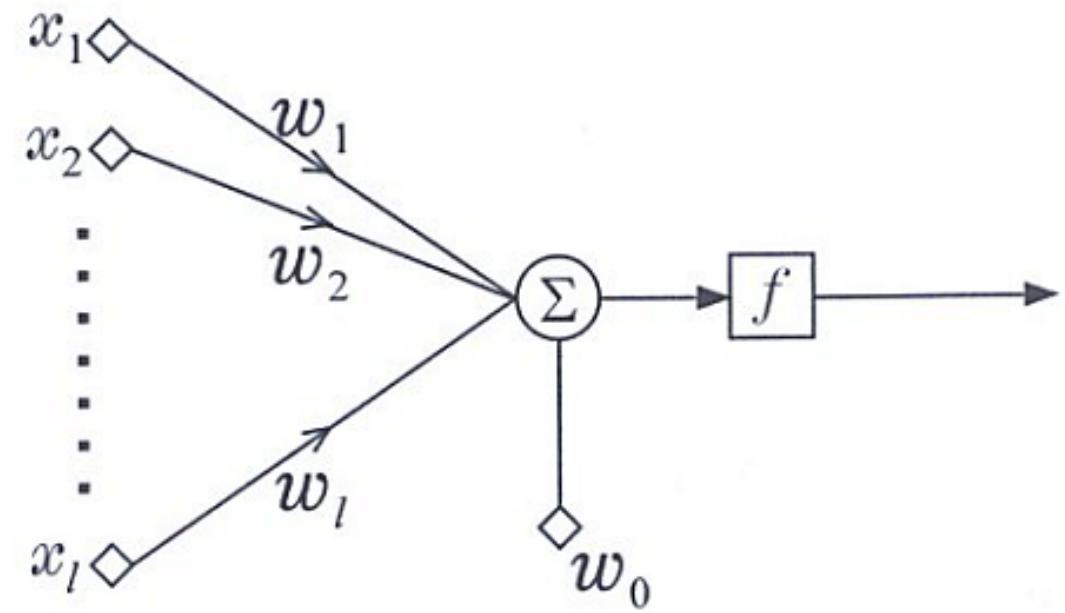
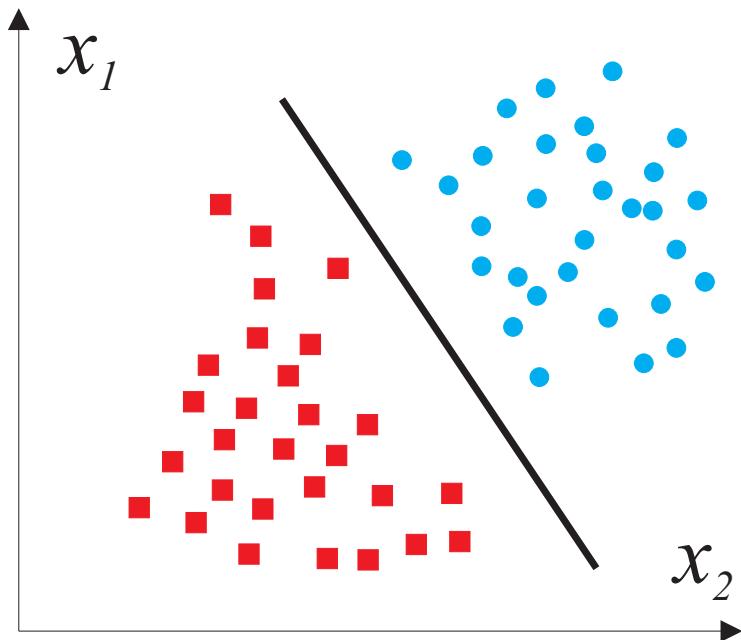
$$\exists \mathbf{w} \in \mathbb{R}^{D+1}: \text{sign} \left(\mathbf{w} \cdot \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix} \right) = k_i \quad \forall i = 1, 2, \dots, L. \quad (2)$$

Example of linearly separable data is on the next slide.

Dichotomy, Two Classes Only

$|K| = 2$, i.e. two hidden states (typically also classes)

$$q(x) = \begin{cases} k = 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + w_0 > 0, \\ k = -1, & \text{if } \mathbf{w} \cdot \mathbf{x} + w_0 < 0. \end{cases} \quad (3)$$



Perceptron Classifier

Input: $\mathcal{T} = \{(\mathbf{x}_1, k_1) \dots (\mathbf{x}_L, k_L)\}$, $k \in \{-1, 1\}$

Goal: Find a weight vector w and offset w_0 such that :

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_j + w_0 &> 0 \quad \text{if } k_j = 1 , \quad (\forall j \in \{1, 2, \dots, L\}) \\ \mathbf{w} \cdot \mathbf{x}_j + w_0 &< 0 \quad \text{if } k_j = -1 \end{aligned} \tag{4}$$

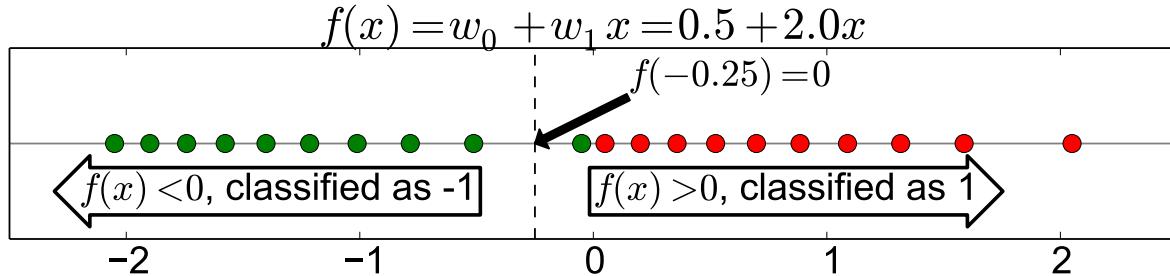
Equivalently, (as in the logistic regression lecture), with $\mathbf{x}' = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$ and $\mathbf{w}' = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$:

$$\begin{aligned} \mathbf{w}' \cdot \mathbf{x}'_j &> 0 \quad \text{if } k_j = 1 \quad (\forall j \in \{1, 2, \dots, L\}) , \\ \mathbf{w}' \cdot \mathbf{x}'_j &< 0 \quad \text{if } k_j = -1 , \end{aligned} \tag{5}$$

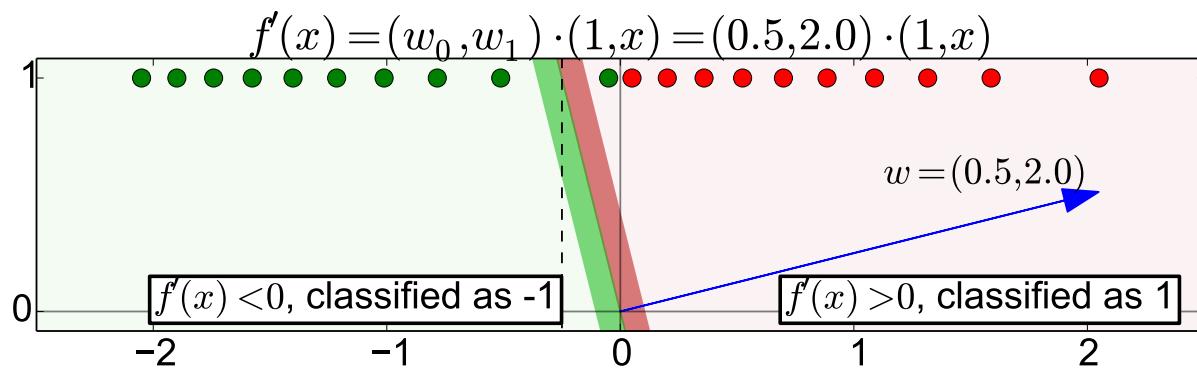
or, with $\mathbf{x}''_j = k_j \mathbf{x}'_j$,

$$\mathbf{w}' \cdot \mathbf{x}''_j > 0 \quad , \quad (\forall j \in \{1, 2, \dots, L\} .) \tag{6}$$

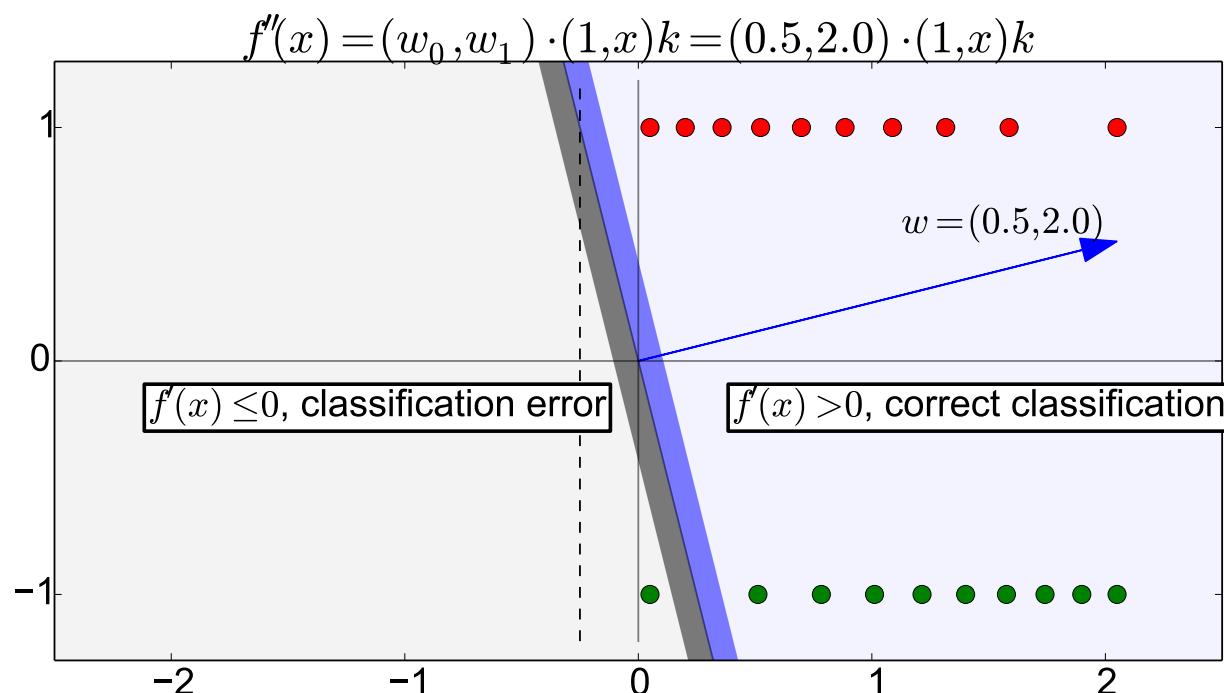
Perceptron Classifier Formulation, Example



● class 1, ● class -1
 Data points, $x \in \mathbb{R}$



Augmenting by 1's, $\mathbf{x}'_j \in \mathbb{R}^2$



Multiplying by k_j , $k_j \mathbf{x}''_j \in \mathbb{R}^2$

Perceptron Learning: Algorithm

We use the last representation ($\mathbf{x}_j'' = k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}$, $\mathbf{w}' = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$) and drop the dashes to avoid notation clutter.

Goal: Find a weight vector $\mathbf{w} \in \mathbb{R}^{D+1}$ (original feature space dimensionality is D) such that:

$$\mathbf{w} \cdot \mathbf{x}_j > 0 \quad (\forall j \in \{1, 2, \dots, L\}) \quad (7)$$

Perceptron algorithm, (Rosenblat 1962):

1. $t = 0$, $\mathbf{w}^{(t)} = 0$.
2. Find a wrongly classified observation \mathbf{x}_j :

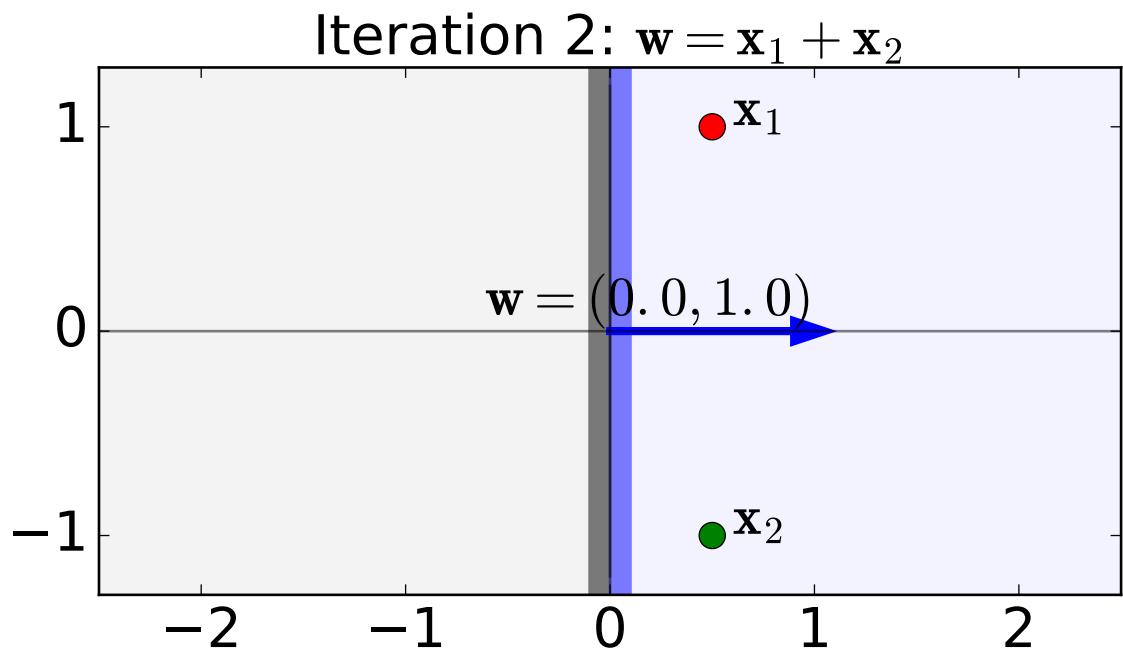
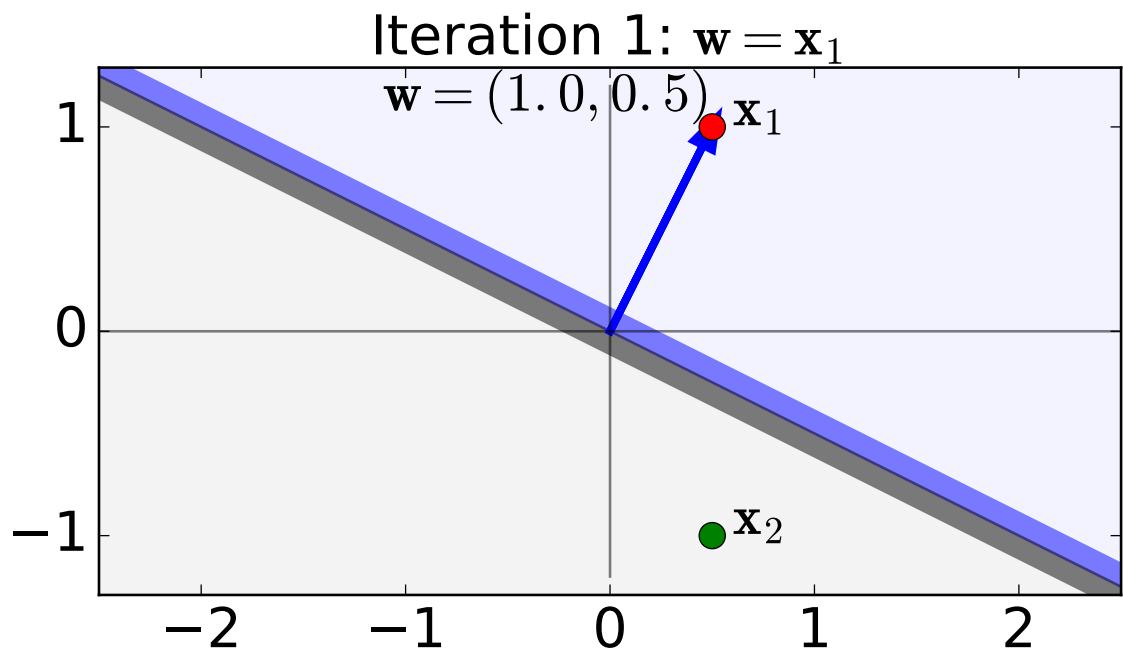
$$\mathbf{w}^{(t)} \cdot \mathbf{x}_j \leq 0, \quad (j \in \{1, 2, \dots, L\}).$$

3. If there is no misclassified observation then terminate. Otherwise,

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \mathbf{x}_j.$$

4. Goto 2.

Perceptron, Example 1

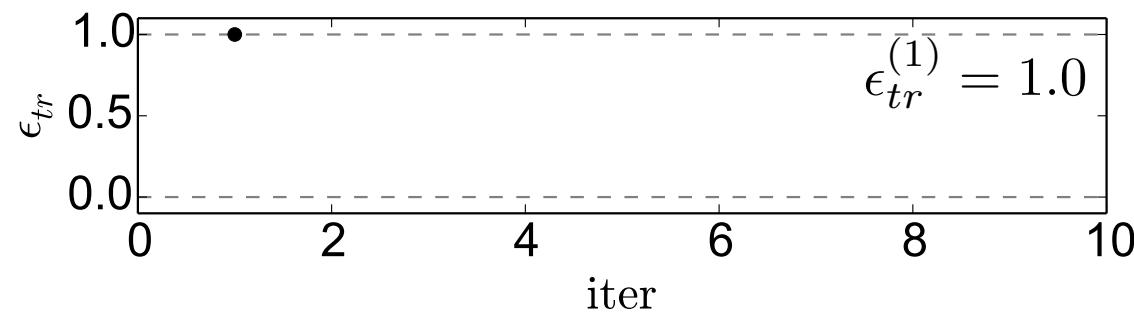
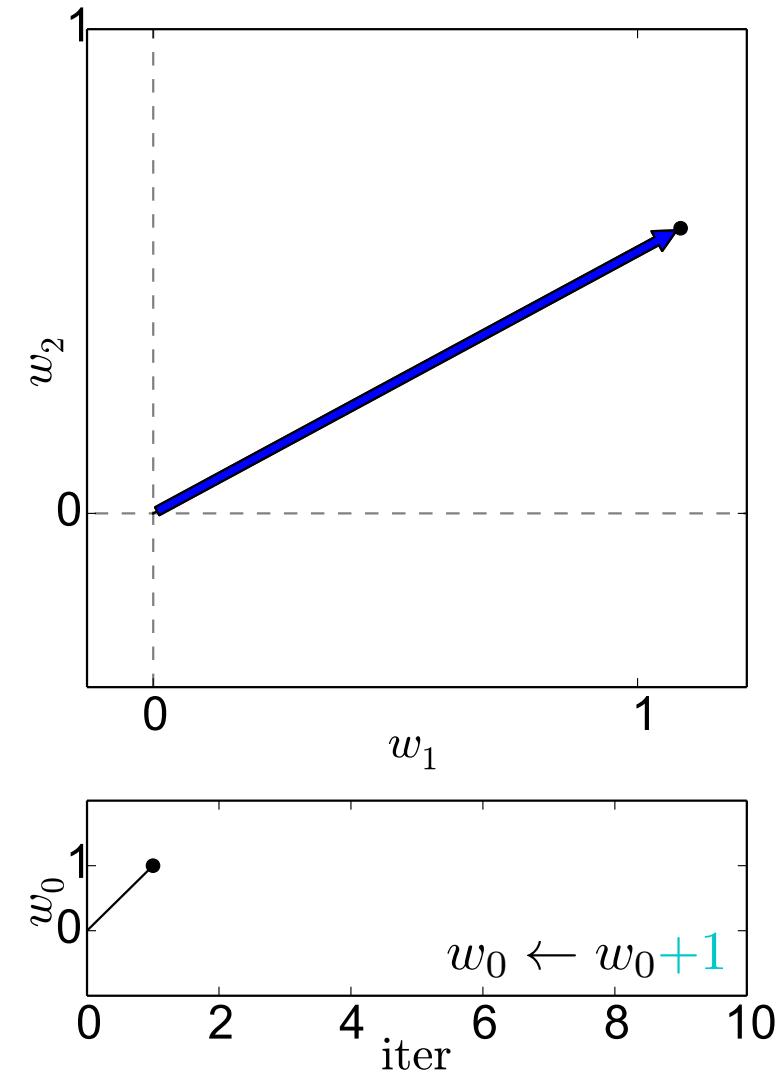
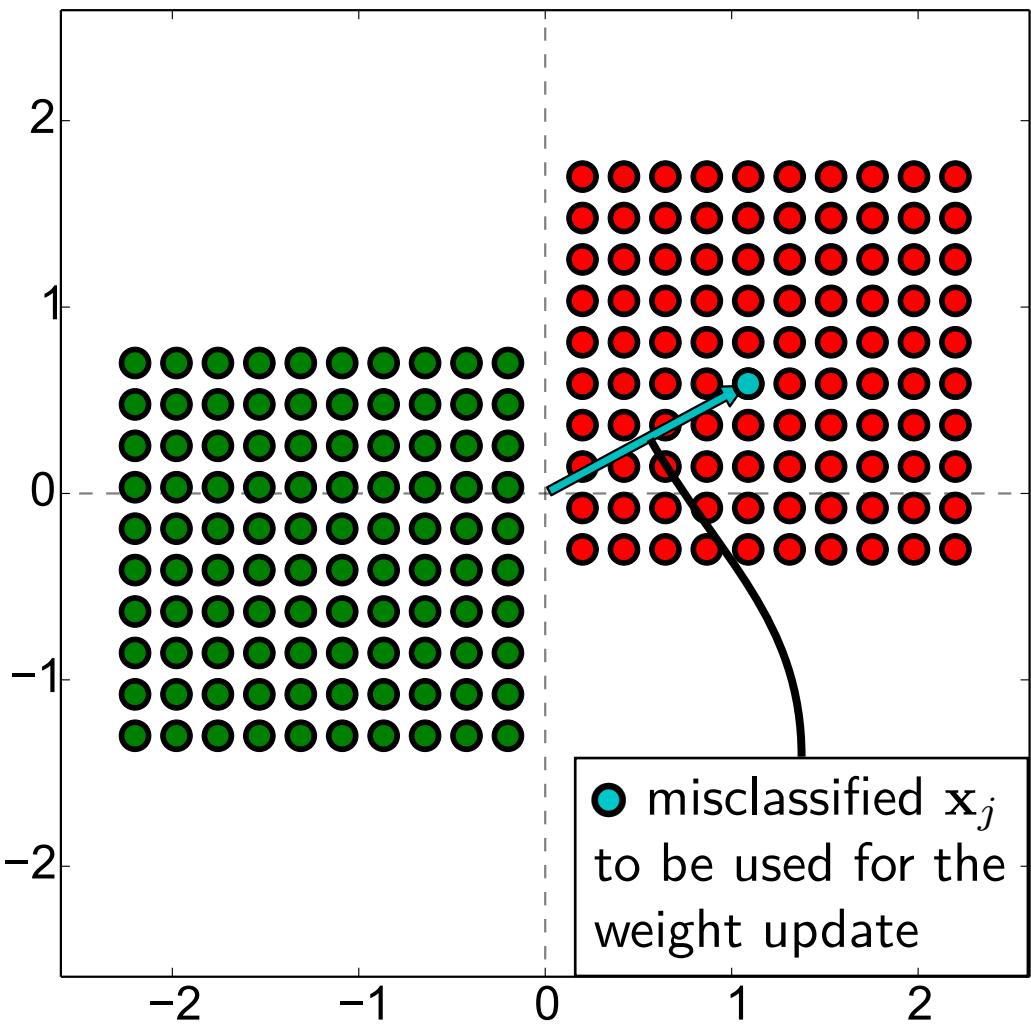


Consider this dataset with 2 points. As $w^{(0)} = 0$, all points are misclassified. Order the points randomly and go over this dataset. Find the first misclassified point. It is x_1 , say. Make the update of weight, $w^{(1)} \leftarrow w^{(0)} + x_1$. Note that x_2 is misclassified.

$w^{(2)} \leftarrow w^{(1)} + x_2$. The whole dataset is correctly classified. Done.

Perceptron, Example 2, Iter. 1

● class 1, ● class -1, ●/●=misclassified,



$$\mathbf{w}^{(1)} = \underbrace{\mathbf{w}^{(0)}}_0 + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}$$

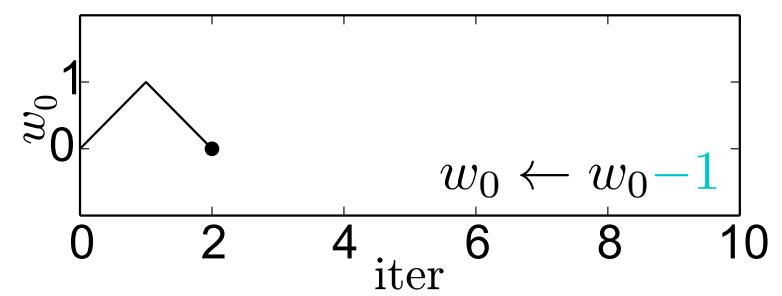
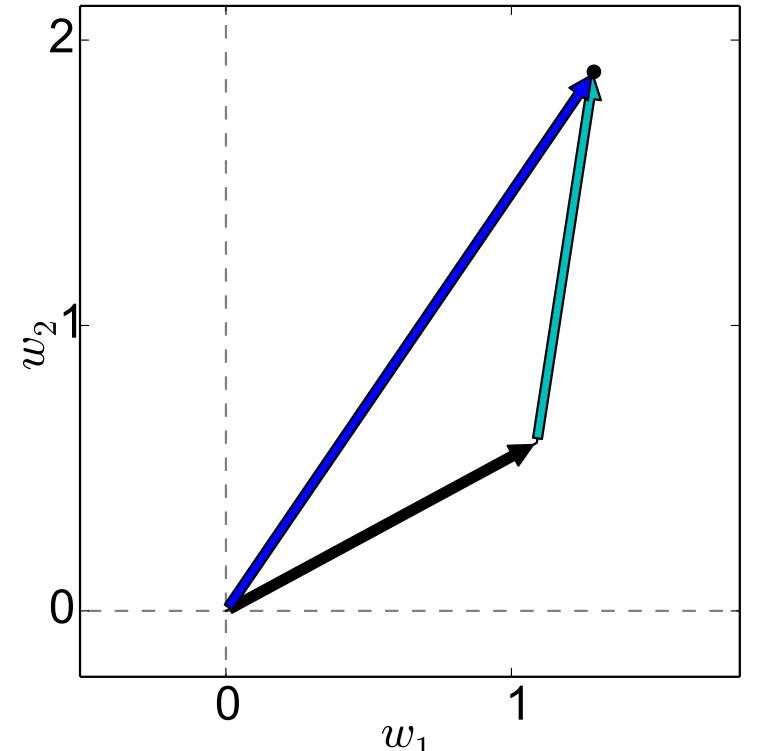
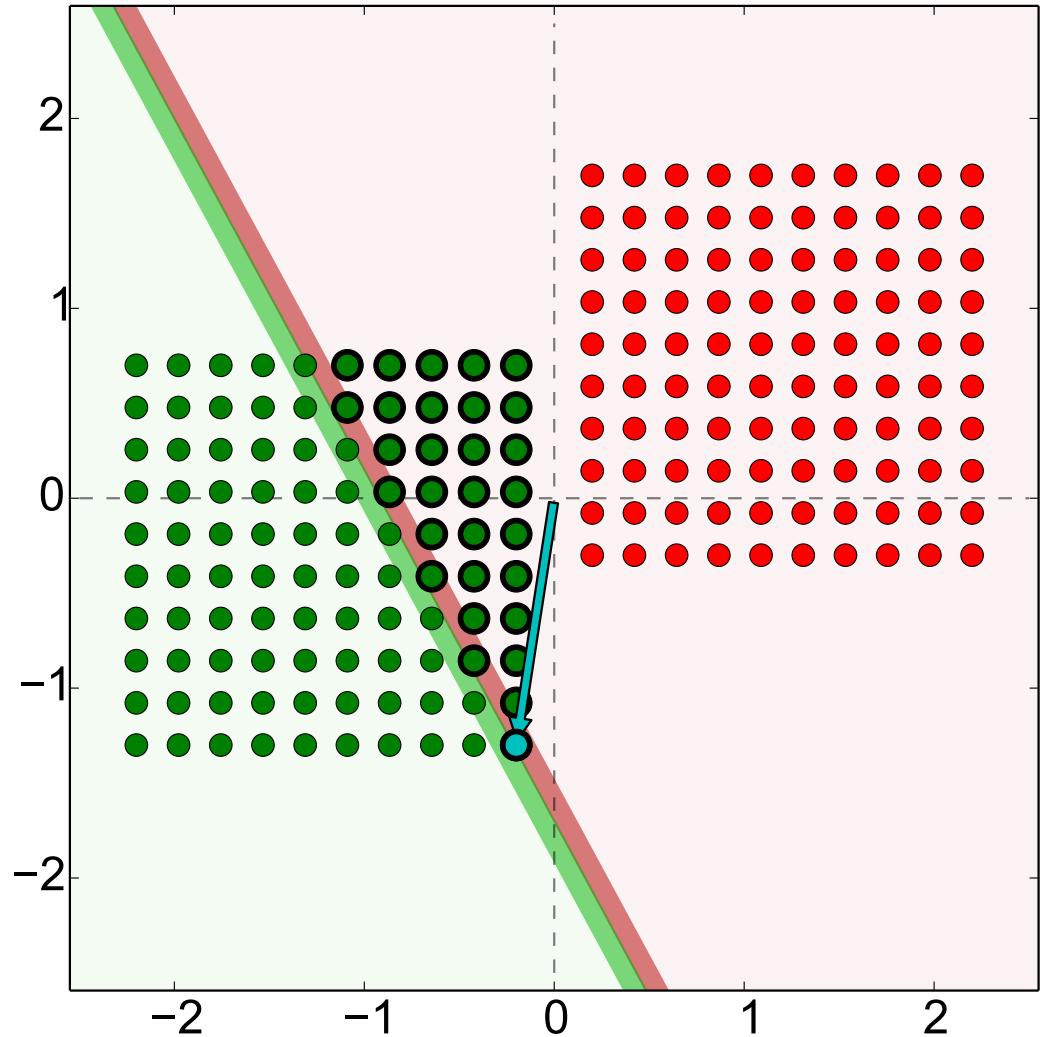
1 point visited so far.

All data are misclassified.

Perceptron, Example 2, Iter. 2

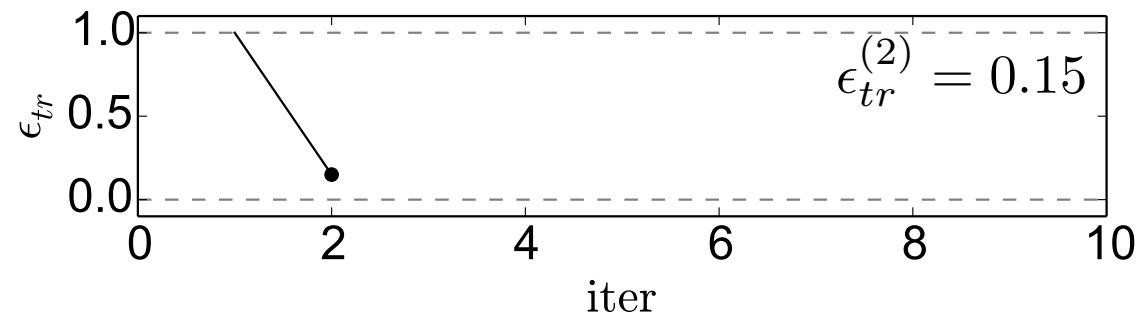
● class 1, ● class -1, ●/●=misclassified,

● the weight updater



$$\mathbf{w}^{(2)} = \mathbf{w}^{(1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix}$$

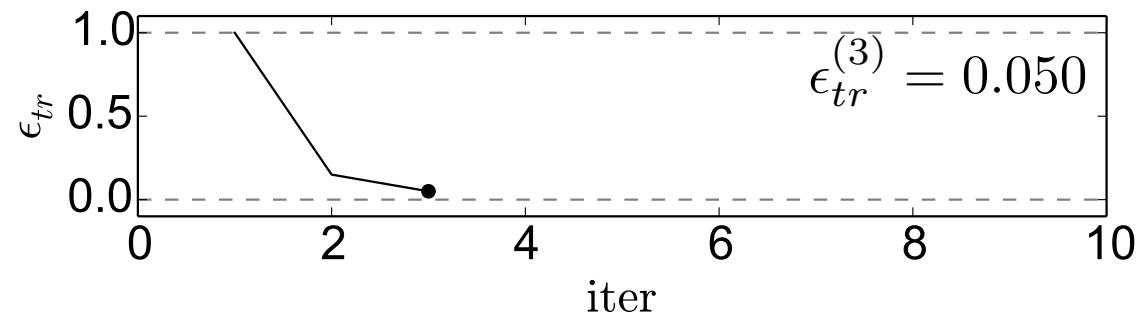
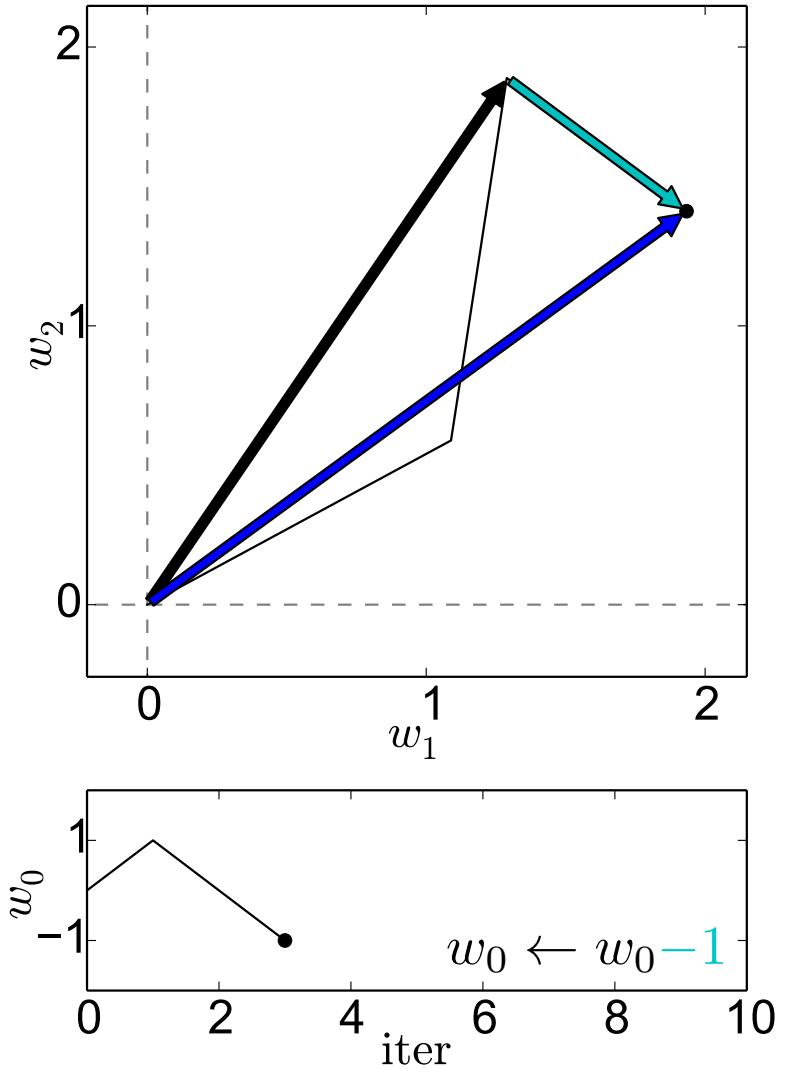
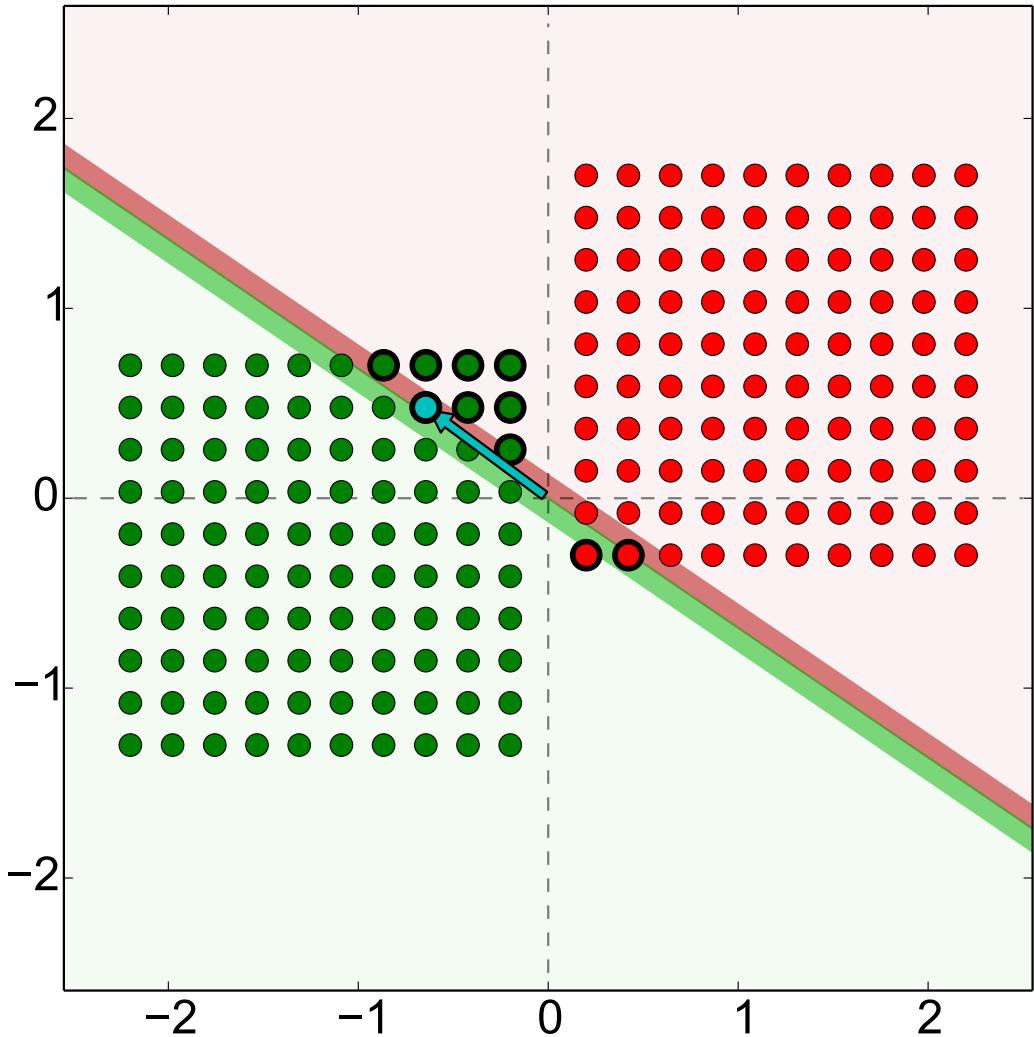
2 points visited so far.



Perceptron, Example 2, Iter. 3

● class 1, ● class -1, ●/●=misclassified,

● the weight updater



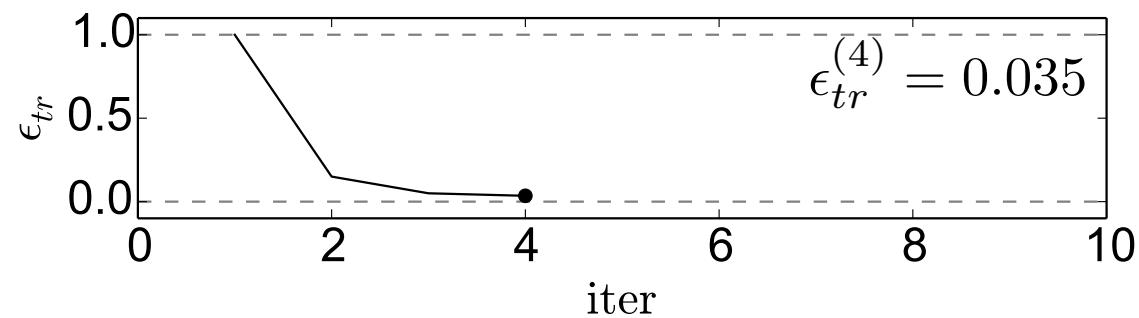
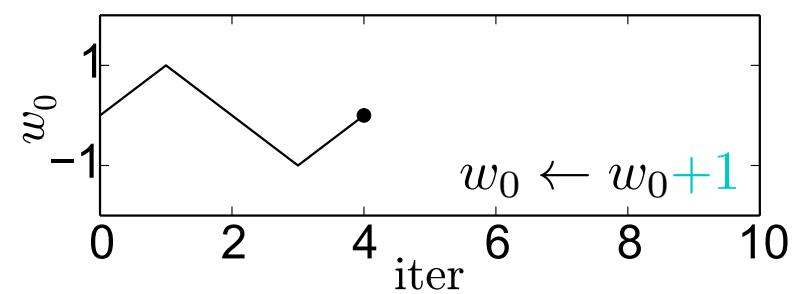
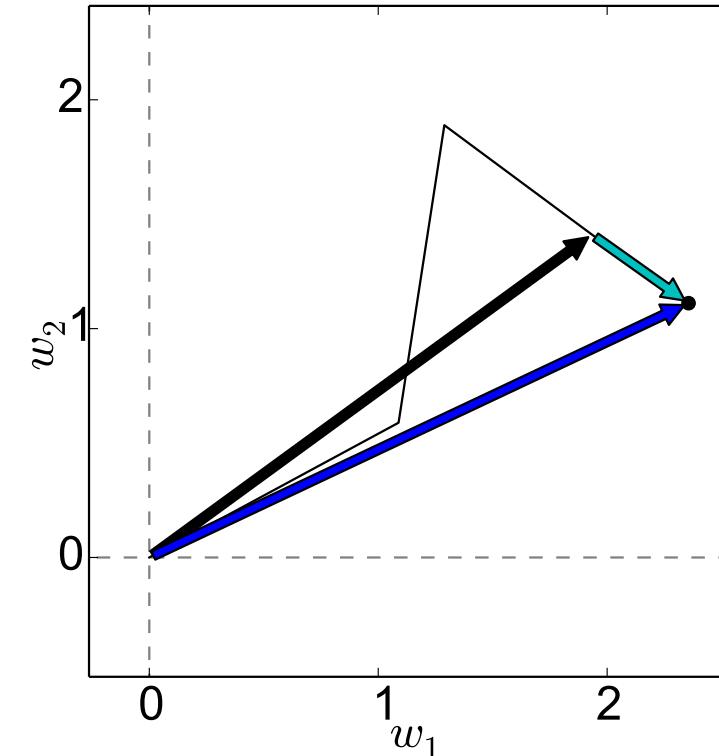
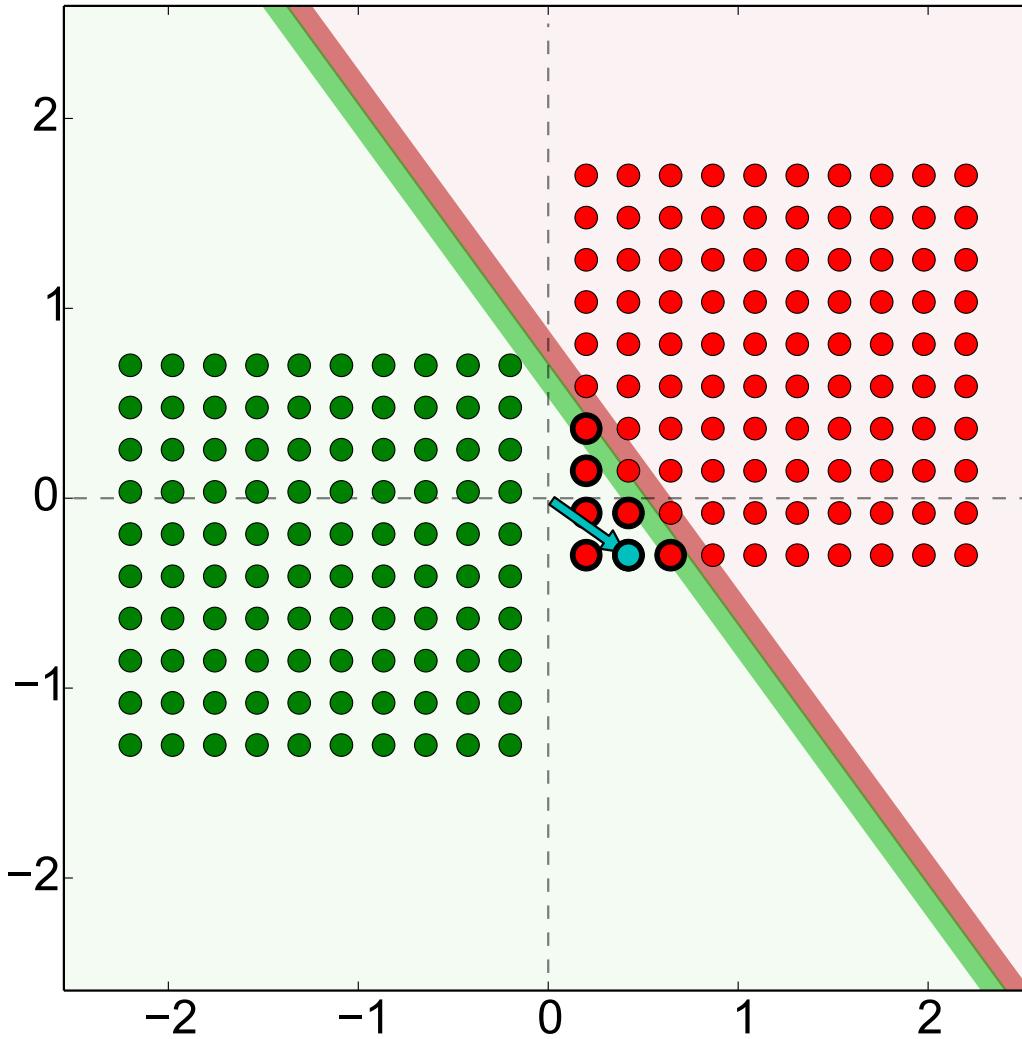
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 3)$$

15 points visited so far.

Perceptron, Example 2, Iter. 4

● class 1, ● class -1, ●/●=misclassified,

● the weight updater



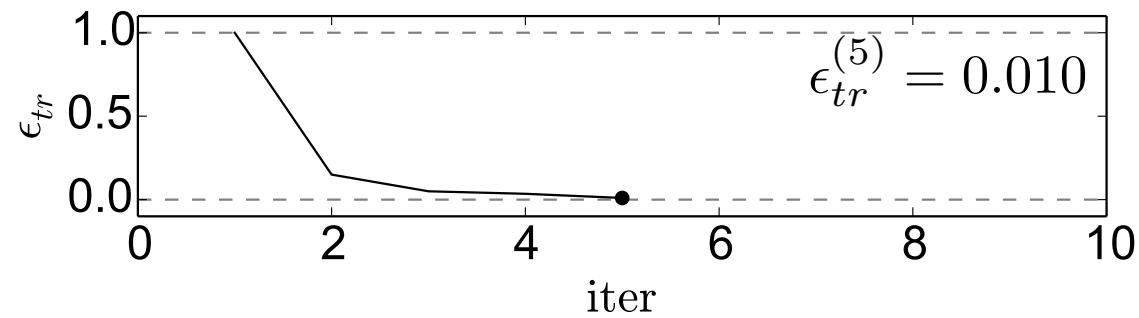
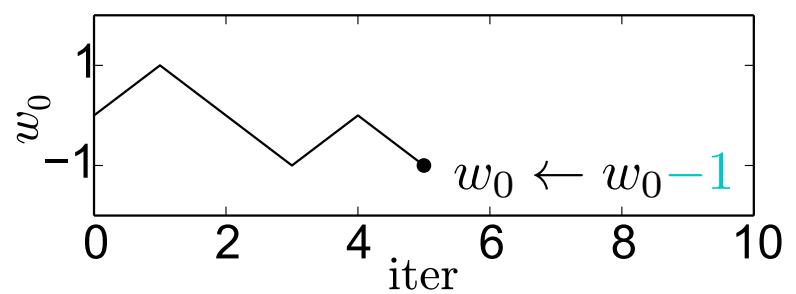
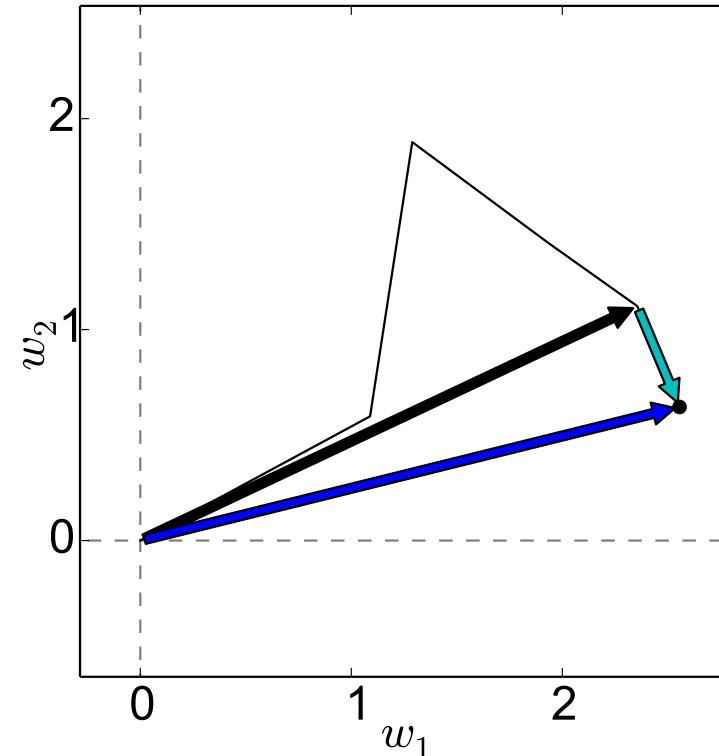
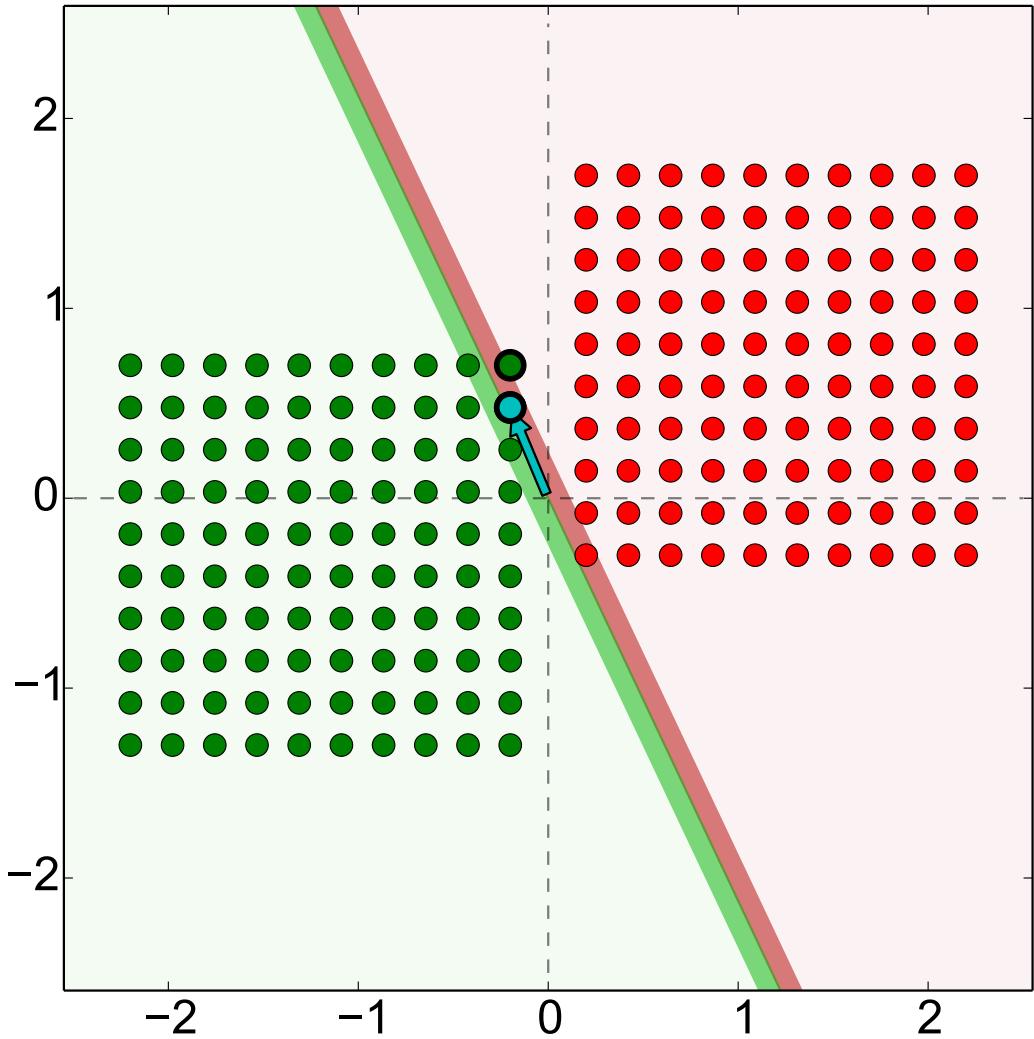
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 4)$$

19 points visited so far.

Perceptron, Example 2, Iter. 5

● class 1, ● class -1, ●/●=misclassified,

● the weight updater



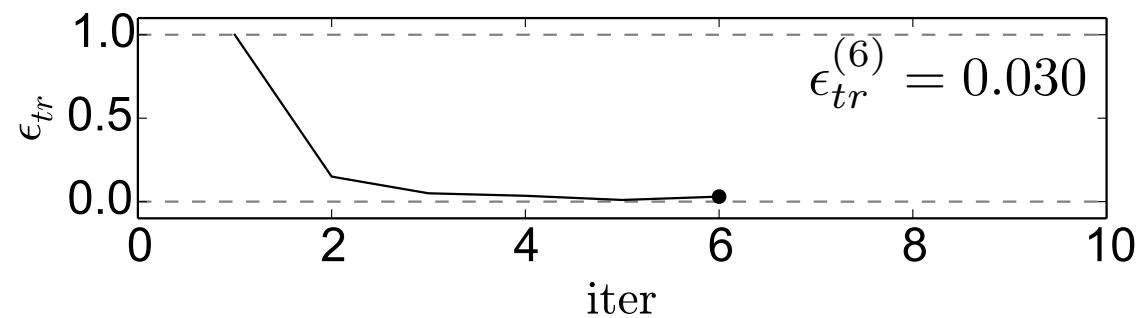
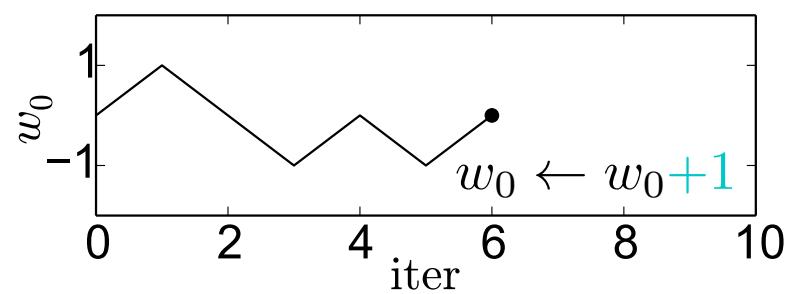
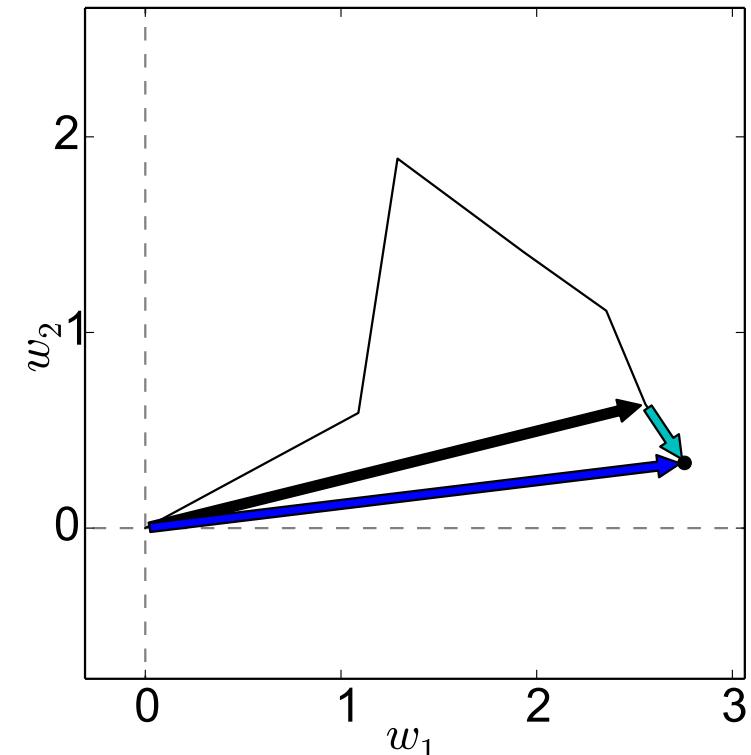
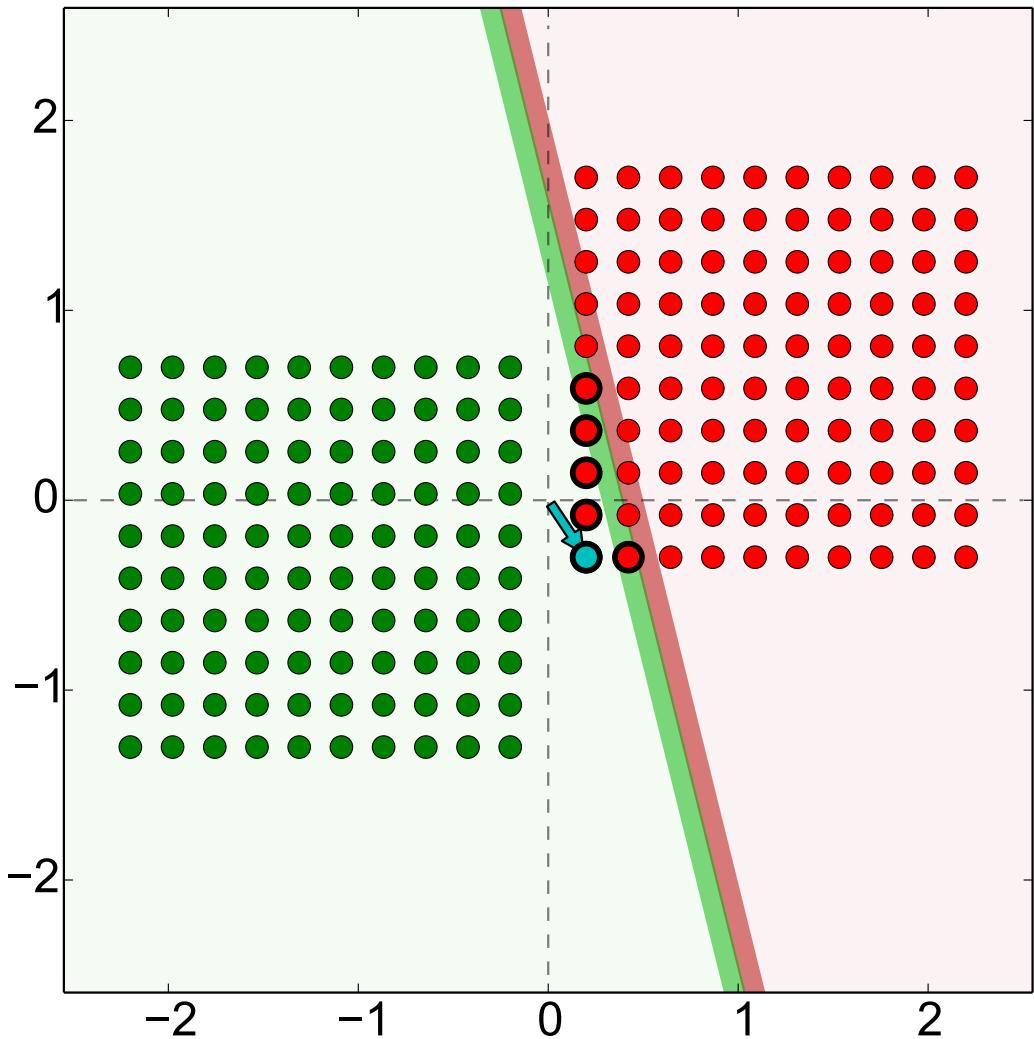
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 5)$$

114 points visited so far.

Perceptron, Example 2, Iter. 6

● class 1, ● class -1, ●/●=misclassified,

● the weight updater

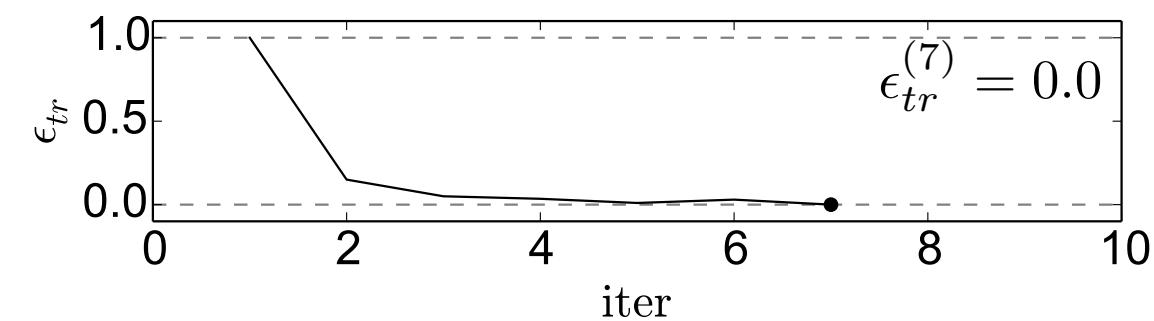
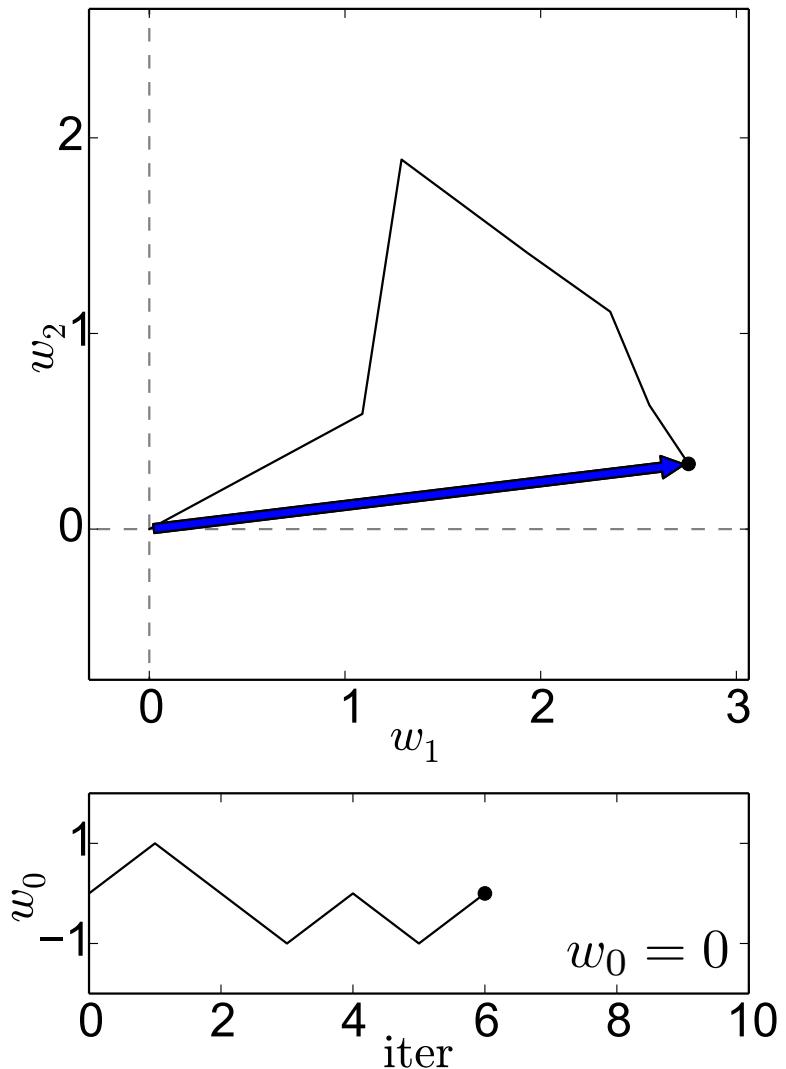
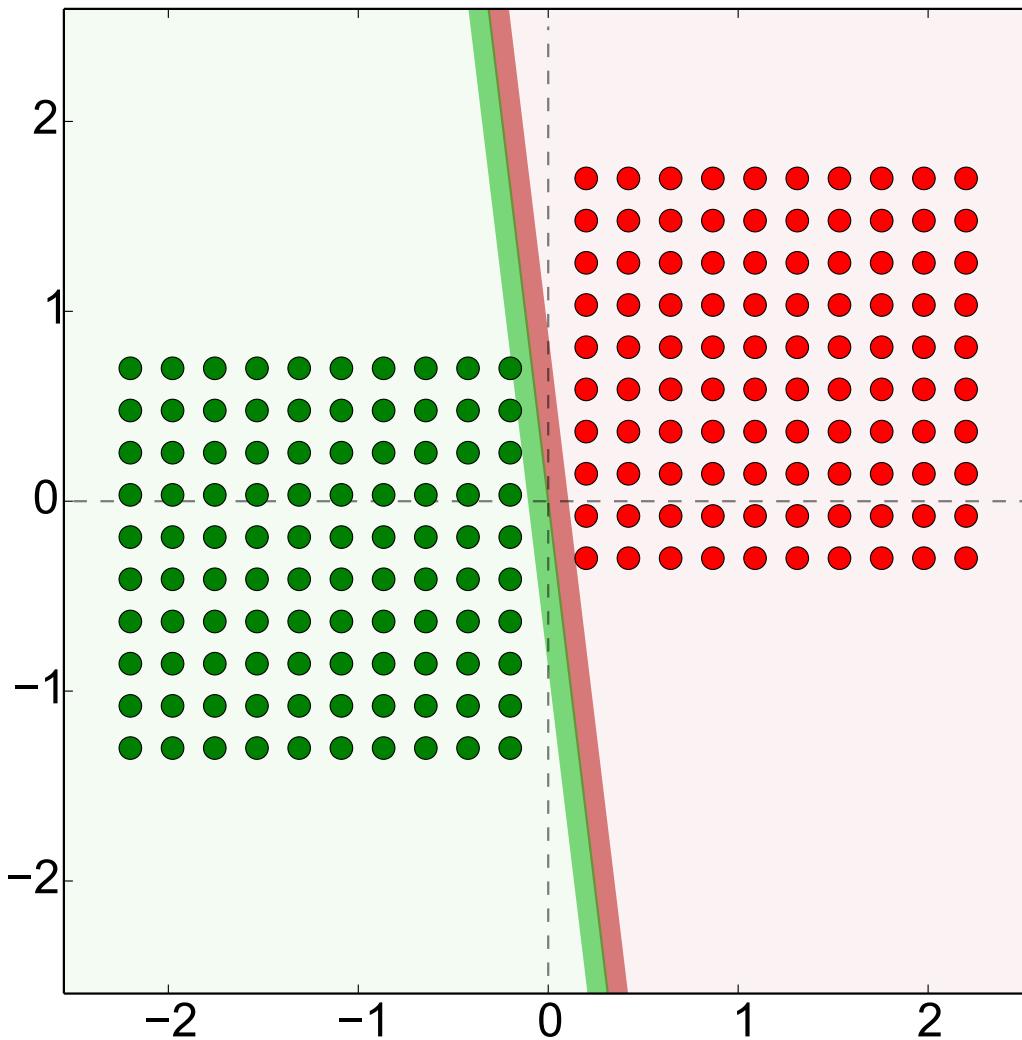


$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 6)$$

186 points visited so far.

Perceptron, Example 2, Iter. 7

● class 1, ● class -1, ●/●=misclassified,



400 points visited.
 All data classified correctly. Done.

Final weight:
 $\mathbf{w} = (0, 2.76, 0.33)^\top$

Novikoff Theorem

Let the data be linearly separable and let there be a unit vector \mathbf{u} and a scalar $\gamma \in \mathbb{R}^+$ such that

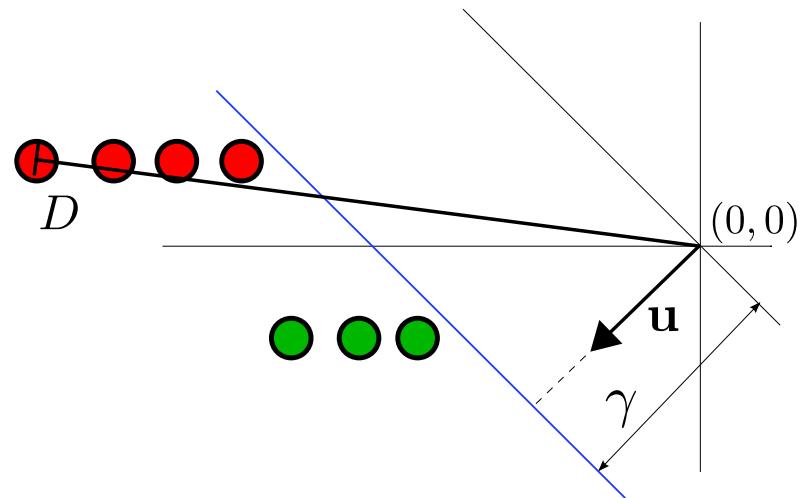
$$\mathbf{u} \cdot \mathbf{x}_j \geq \gamma \quad \forall j \in \{1, 2, \dots, L\} \quad (\|\mathbf{u}\| = 1) \quad (8)$$

Let the norm of the longest vector in the dataset be D :

$$D = \max_{x \in \mathcal{T}} \|\mathbf{x}\|. \quad (9)$$

Then the perceptron algorithm will **finish in a finite number of steps** t^* , with

$$t^* \leq \frac{D^2}{\gamma^2}. \quad (10)$$



- ? What if the data is not separable?
- ? How to terminate perceptron learning?

Novikoff Theorem, Proof (1)

Let $\mathbf{x}^{(t)}$ be the point which is incorrectly classified at time t , so

$$\mathbf{w}^{(t)} \cdot \mathbf{x}^{(t)} \leq 0. \quad (11)$$

Recall that the weight $\mathbf{w}^{(t+1)}$ is computed using this update $\mathbf{x}^{(t)}$ as

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \mathbf{x}^{(t)}. \quad (12)$$

For the squared norm of $\mathbf{w}^{(t+1)}$, we have

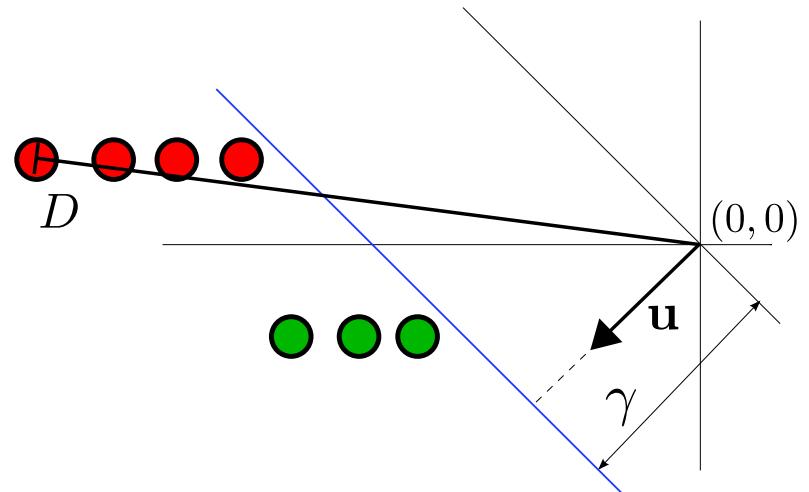
$$\|\mathbf{w}^{(t+1)}\|^2 = \mathbf{w}^{(t+1)} \cdot \mathbf{w}^{(t+1)} = (\mathbf{w}^{(t)} + \mathbf{x}^{(t)}) \cdot (\mathbf{w}^{(t)} + \mathbf{x}^{(t)}) \quad (13)$$

$$= \|\mathbf{w}^{(t)}\|^2 + 2\underbrace{\mathbf{w}^{(t)} \cdot \mathbf{x}^{(t)}}_{\leq 0} + \underbrace{\|\mathbf{x}^{(t)}\|^2}_{\leq D^2} \quad (14)$$

$$\leq \|\mathbf{w}^{(t)}\|^2 + D^2 \leq \|\mathbf{w}^{(t-1)}\|^2 + 2D^2 \quad (15)$$

$$\leq \|\mathbf{w}^{(t-2)}\|^2 + 3D^2 \leq \dots \leq \|\mathbf{w}^{(0)}\|^2 + (t+1)D^2 \quad (16)$$

$$\|\mathbf{w}^{(t+1)}\|^2 \leq (t+1)D^2 \quad (17)$$



Novikoff Theorem, Proof (2)

We also have that

$$\mathbf{w}^{(t+1)} \cdot \mathbf{u} = \mathbf{w}^{(t)} \cdot \mathbf{u} + \underbrace{\mathbf{x}^{(t)} \cdot \mathbf{u}}_{\geq \gamma} \quad (18)$$

$$\geq \mathbf{w}^{(t)} \cdot \mathbf{u} + \gamma \geq \mathbf{w}^{(t-1)} \cdot \mathbf{u} + 2\gamma \quad (19)$$

$$\geq \mathbf{w}^{(t-2)} \cdot \mathbf{u} + 3\gamma \geq \dots \quad (20)$$

$$\geq \mathbf{w}^{(0)} \cdot \mathbf{u} + (t+1)\gamma \quad (21)$$

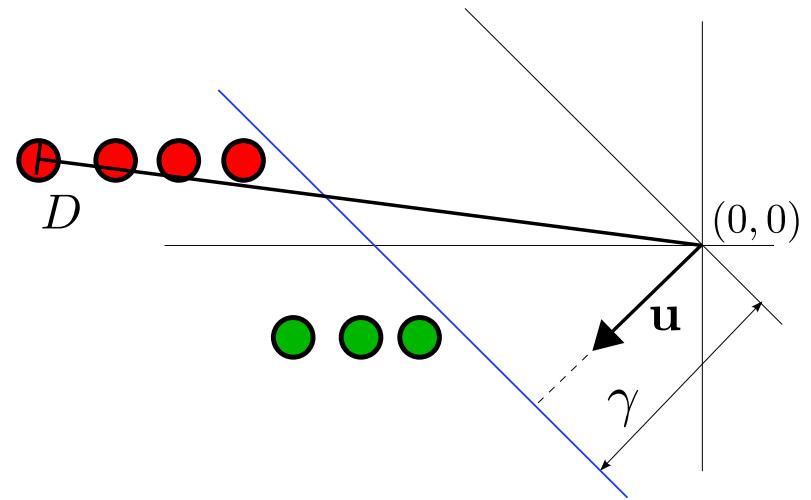
$$\mathbf{w}^{(t+1)} \cdot \mathbf{u} \geq (t+1)\gamma \quad (22)$$

We take the two inequalities together, to obtain

$$(t+1)D^2 \geq \|\mathbf{w}^{(t+1)}\|^2 \geq (\mathbf{w}^{(t+1)} \cdot \mathbf{u})^2 \geq (t+1)^2\gamma^2 \quad (\|\mathbf{u}\| = 1) \quad (23)$$

Therefore,

$$(t+1) \leq \frac{D^2}{\gamma^2}. \quad (24)$$



Perceptron Learning as an Optimisation Problem (1)

Perceptron algorithm, batch version, handling non-separability, another perspective:

Input: $\mathcal{T} = \{\mathbf{x}_1, \dots, \mathbf{x}_L\}$

Output: a weight vector \mathbf{w} minimising

$$J(\mathbf{w}) = |\{\mathbf{x} \in X : \mathbf{w}^{(t)} \cdot \mathbf{x} \leq 0\}| \quad (25)$$

or, equivalently

$$J(\mathbf{w}) = \sum_{\substack{\mathbf{x} \in X \\ \mathbf{w}^{(t)} \cdot \mathbf{x} \leq 0}} 1 \quad (26)$$

What would the most common optimisation method, i.e. gradient descent, perform?

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla J(\mathbf{w}) \quad (27)$$

The gradient of $J(\mathbf{w})$ is, however, either 0 or undefined. The gradient minimisation cannot proceed.

Perceptron Learning as an Optimisation Problem (2)

Let us redefine the cost function:

$$J_p(\mathbf{w}) = \sum_{\substack{\mathbf{x} \in X \\ \mathbf{w} \cdot \mathbf{x} \leq 0}} (-\mathbf{w} \cdot \mathbf{x}) \quad (28)$$

$$\nabla J_p(\mathbf{w}) = \frac{\partial J}{\partial \mathbf{w}} = \sum_{\substack{\mathbf{x} \in X \\ \mathbf{w} \cdot \mathbf{x} \leq 0}} (-\mathbf{x}) \quad (29)$$

- ◆ The Perceptron Algorithm is a gradient **descent** method for $J_p(\mathbf{w})$ (gradient for a single misclassified sample is $-\mathbf{x}$, so the weight update is \mathbf{x})
- ◆ Learning and empirical risk minimisation is just an instance of an **optimization problem**.
- ◆ Either gradient minimisation (backpropagation in neural networks) or convex (quadratic) minimisation (in mathematical literature called convex programming) is used.

Perceptron Learning: Non-Separable Case

Perceptron algorithm, batch version, handling non-separability:

Input: $\mathcal{T} = \{\mathbf{x}_1, \dots, \mathbf{x}_L\}$

Output: weight vector \mathbf{w}^*

1. $\mathbf{w}^{(0)} = 0, E = |T| = L, \mathbf{w}^* = 0$.
2. Find all mis-classified observations $X^- = \{\mathbf{x} \in X : \mathbf{w}^{(t)} \cdot \mathbf{x} \leq 0\}$.
3. if $|X^-| < E$ then $E = |X^-|; \mathbf{w}^* = \mathbf{w}^{(t)}$
4. if $\text{TermCond}(\mathbf{w}^*, t, t_{\text{lup}})$ then terminataate (t_{lup} is the time of the last update)
else:

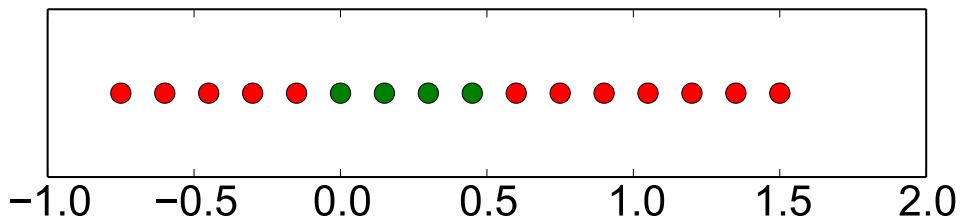
$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta_t \sum_{x \in X^-} x$$

5. Goto 2.
-

- ◆ The algorithm converges with probability 1 to the optimal solution.
- ◆ Convergence rate is not known.
- ◆ Termination condition $\text{TermCond}(\cdot)$ is a complex function of the quality of the best solution, time since last update $t - t_{\text{lup}}$ and requirements on the solution.

Dimension Lifting

Consider the data on the right. They are not linearly separable, because there is no $\mathbf{w} \in \mathbb{R}^2$ such that $\text{sign}(w_0 + w_1 x)$ would correctly classify the data.



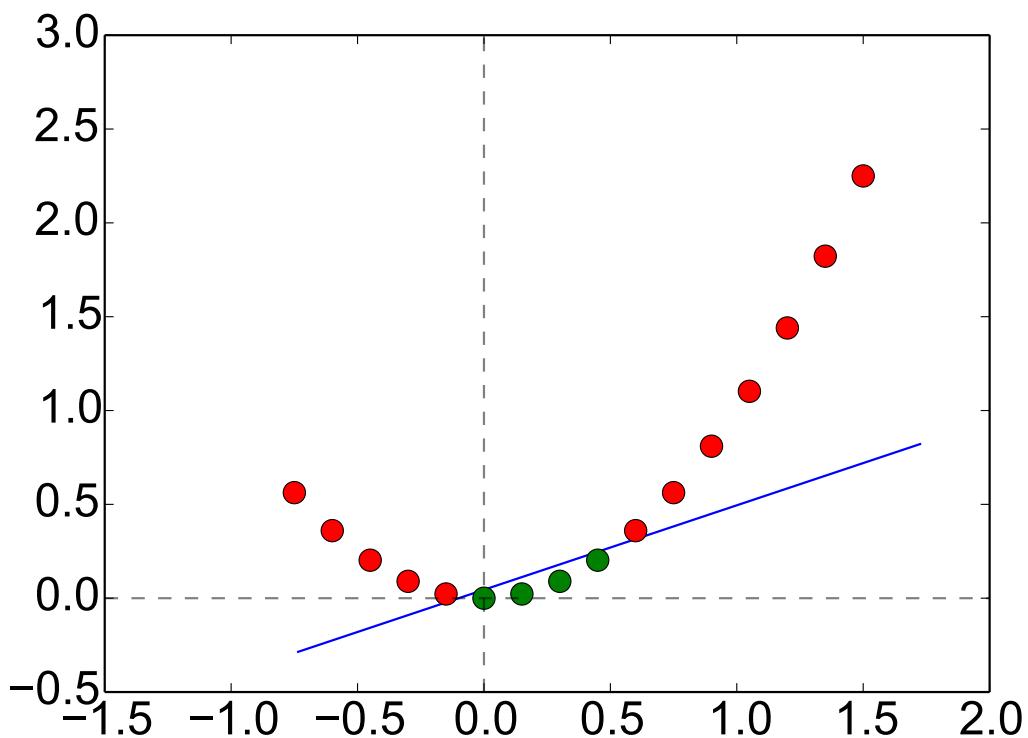
Let us artificially enlarge the dimensionality of the feature space by a mapping $\phi(x) : \mathbb{R} \rightarrow \mathbb{R}^2$:

$$\mathbf{x} \leftarrow \phi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix} \quad (30)$$

After such mapping, the data become linearly separable (the separator is shown on the right).

In general, lifting the feature space means adding D' dimensions and replacing the original feature vectors by

$$\mathbf{x} \leftarrow \phi(\mathbf{x}), \quad \phi(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^{D+D'}. \quad (31)$$



Lifting, Example 1

We will now apply the perceptron algorithm to the problem just shown. Note that instead of the lifting

$$\mathbf{x} \leftarrow \begin{bmatrix} x \\ x^2 \end{bmatrix}$$

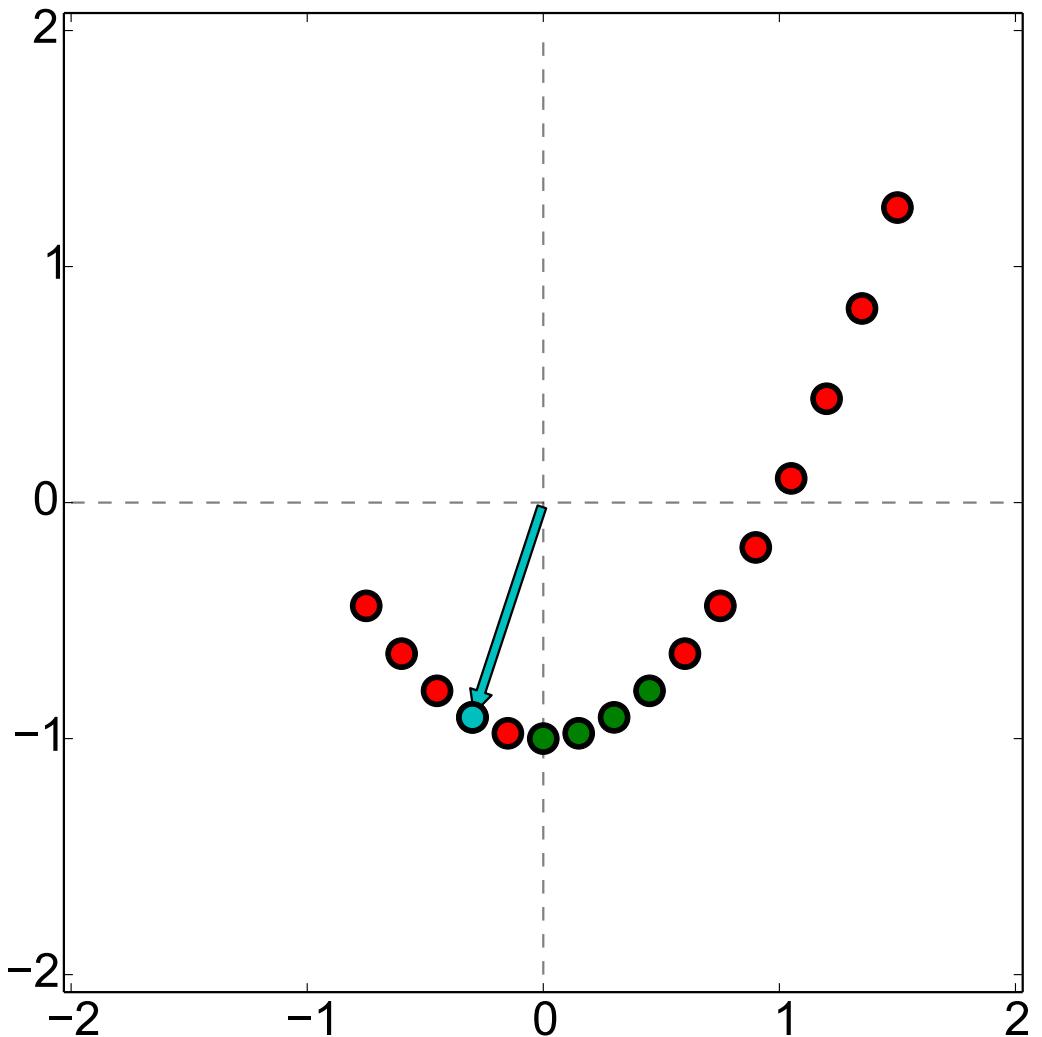
we will use

$$\mathbf{x} \leftarrow \begin{bmatrix} x \\ x^2 - 1 \end{bmatrix}$$

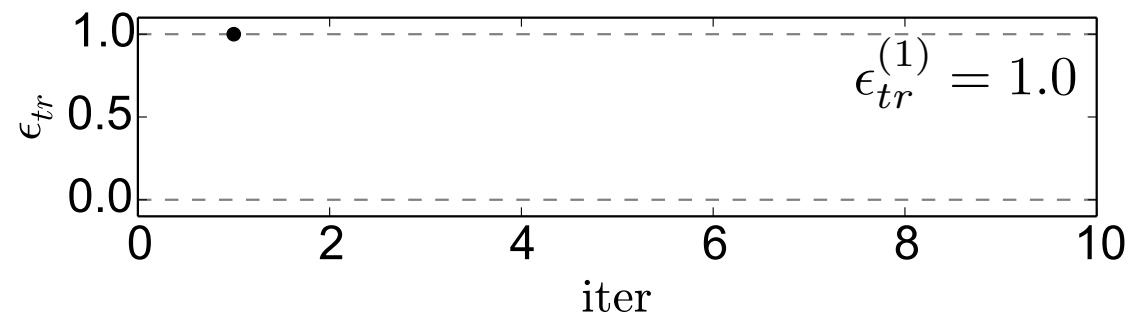
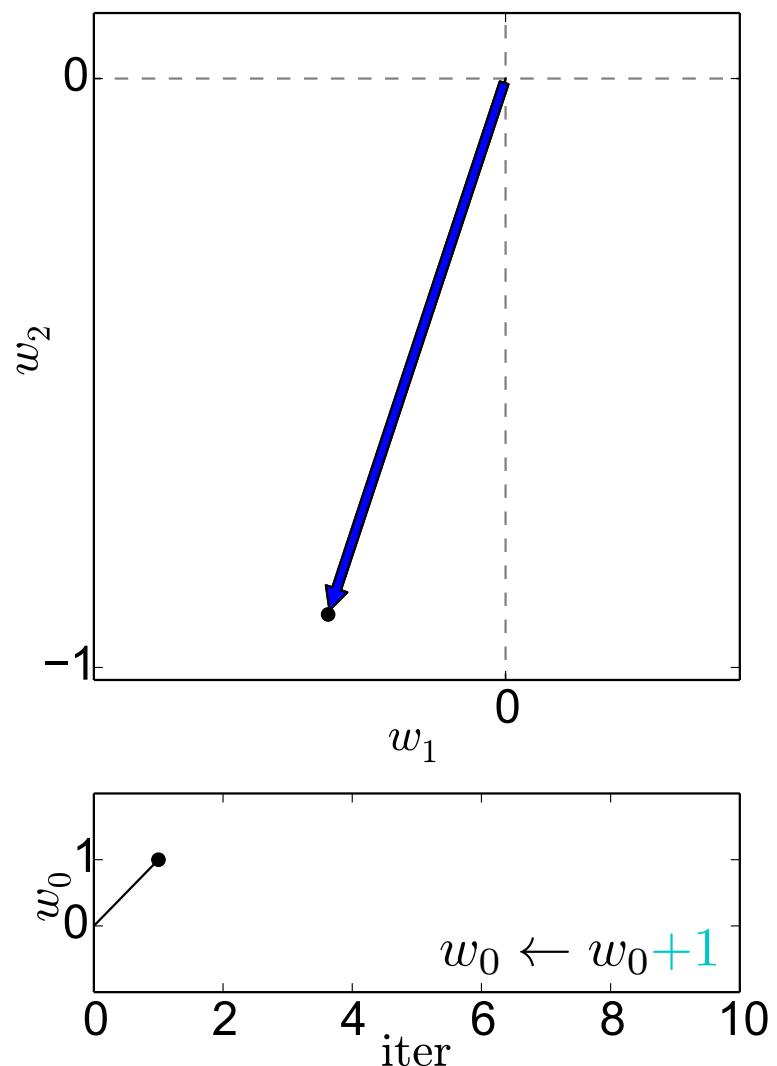
because perceptron then converges faster for our data (this is related to the Novikoff theorem.)

Lifting, Example 1, Iter. 1

● class 1, ● class -1, ●/●=misclassified,



● the weight updater

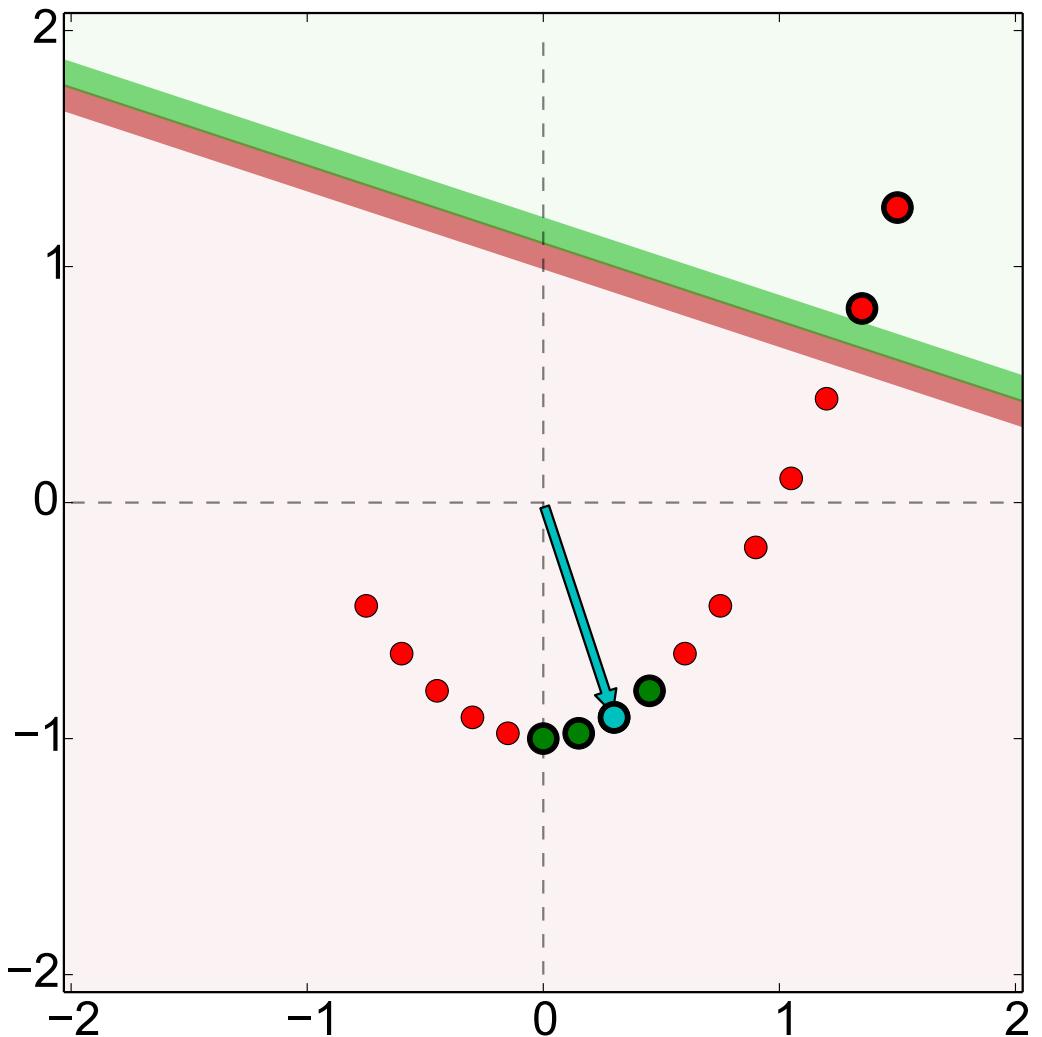


$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 1)$$

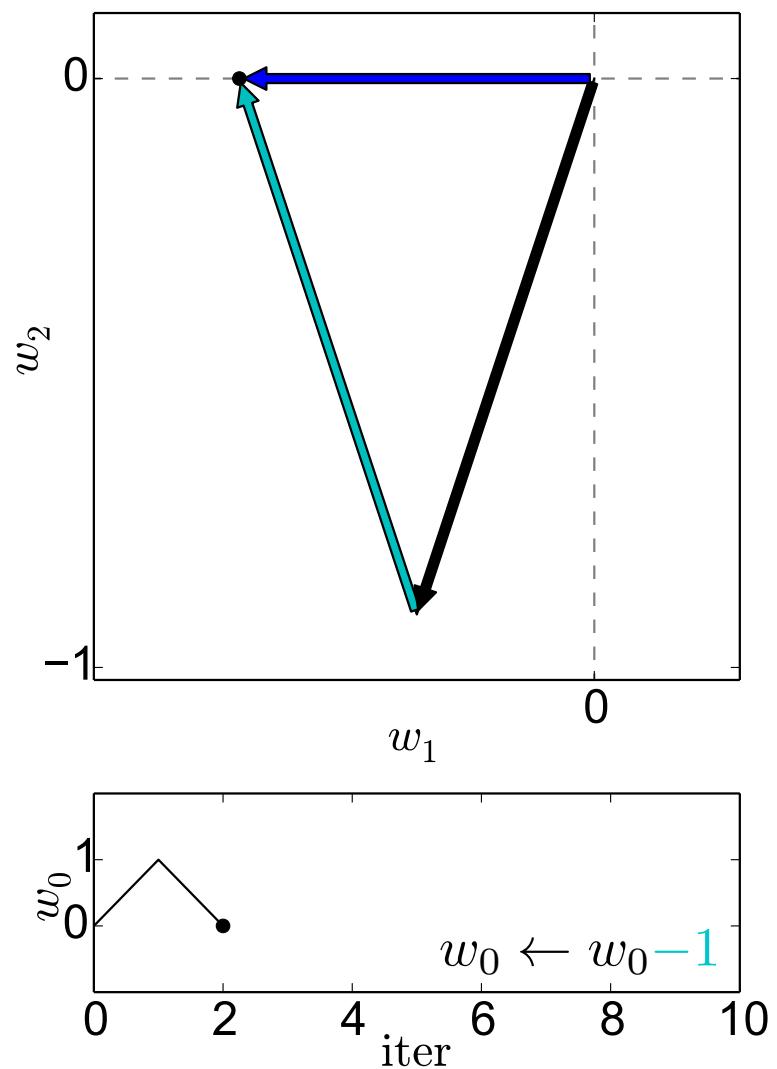
1 point visited so far.

Lifting, Example 1, Iter. 2

● class 1, ● class -1, ●/●=misclassified,

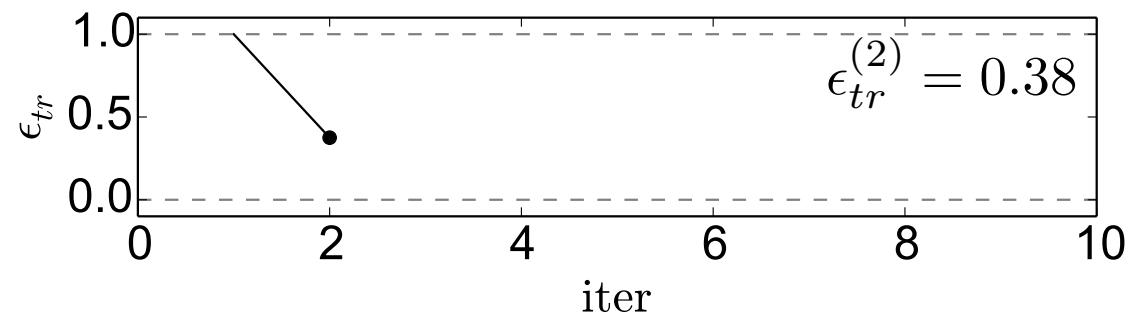


● the weight updater



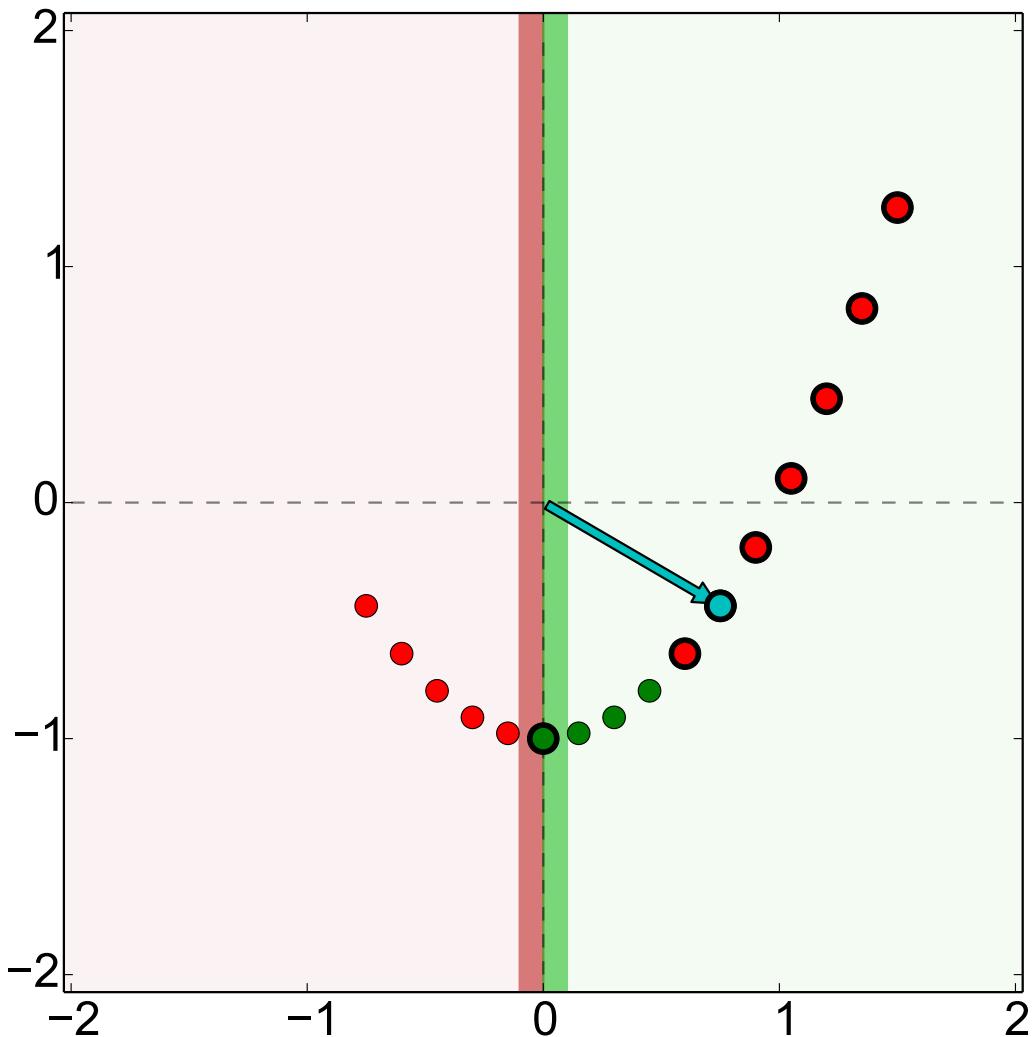
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 2)$$

3 points visited so far.

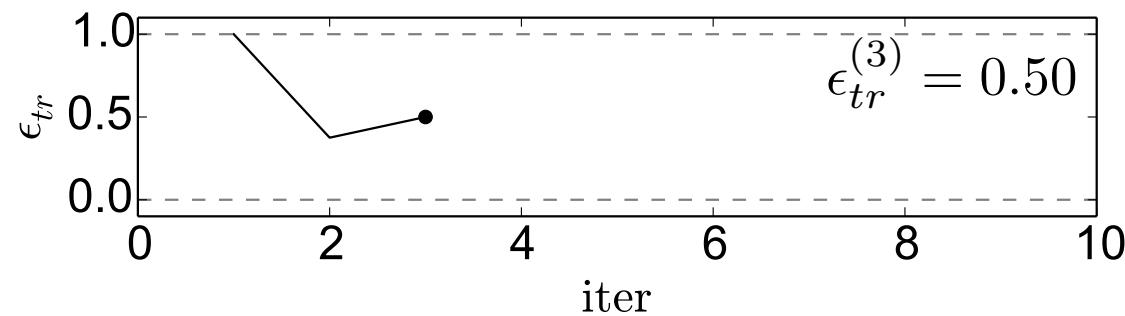
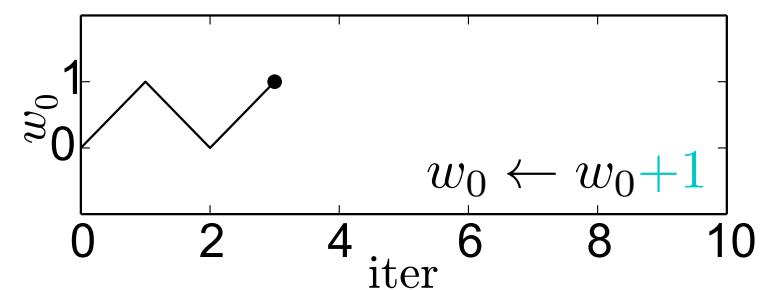
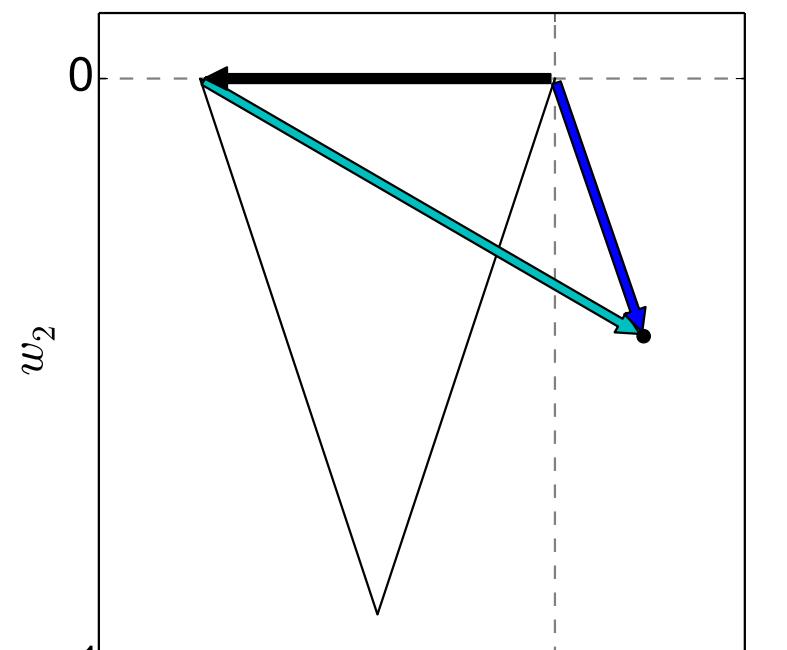


Lifting, Example 1, Iter. 3

● class 1, ● class -1, ●/●=misclassified,



● the weight updater



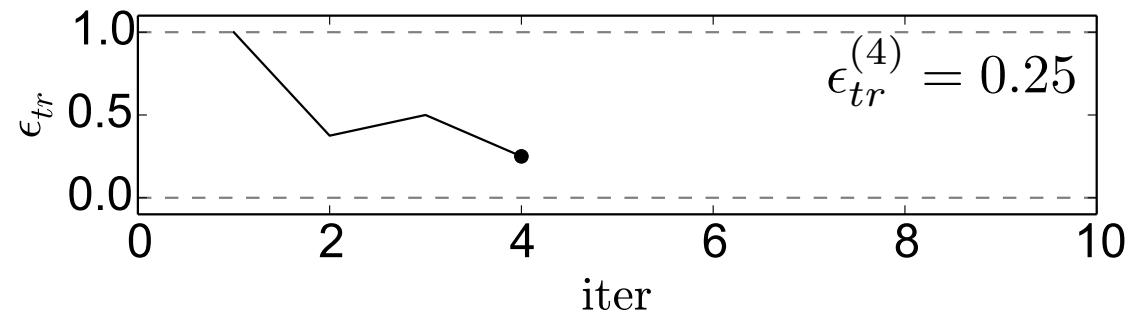
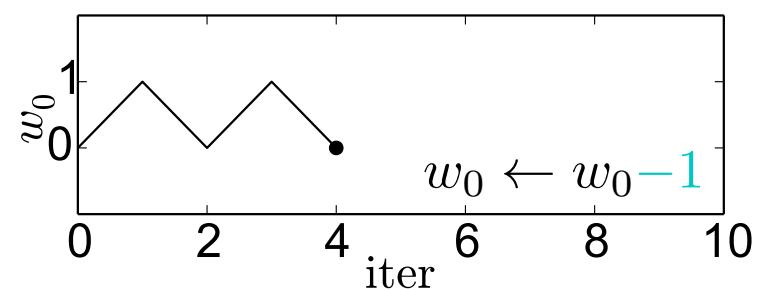
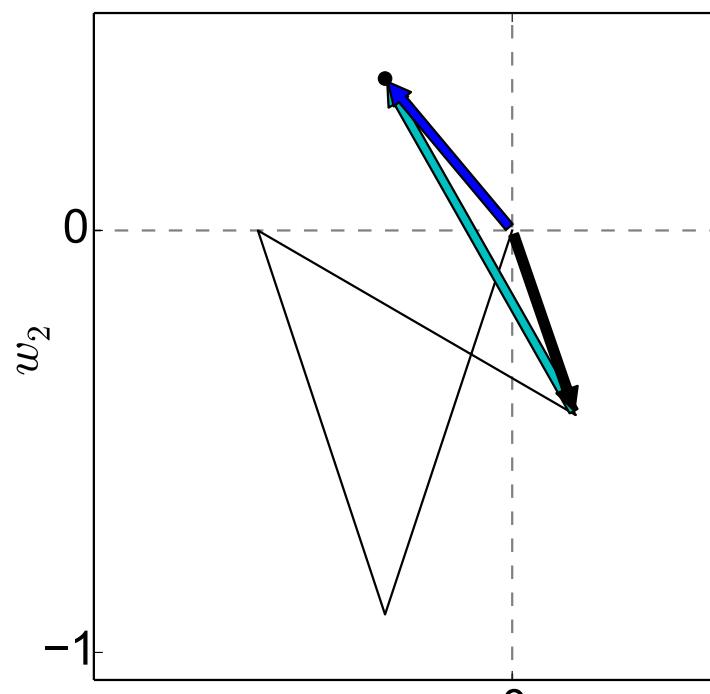
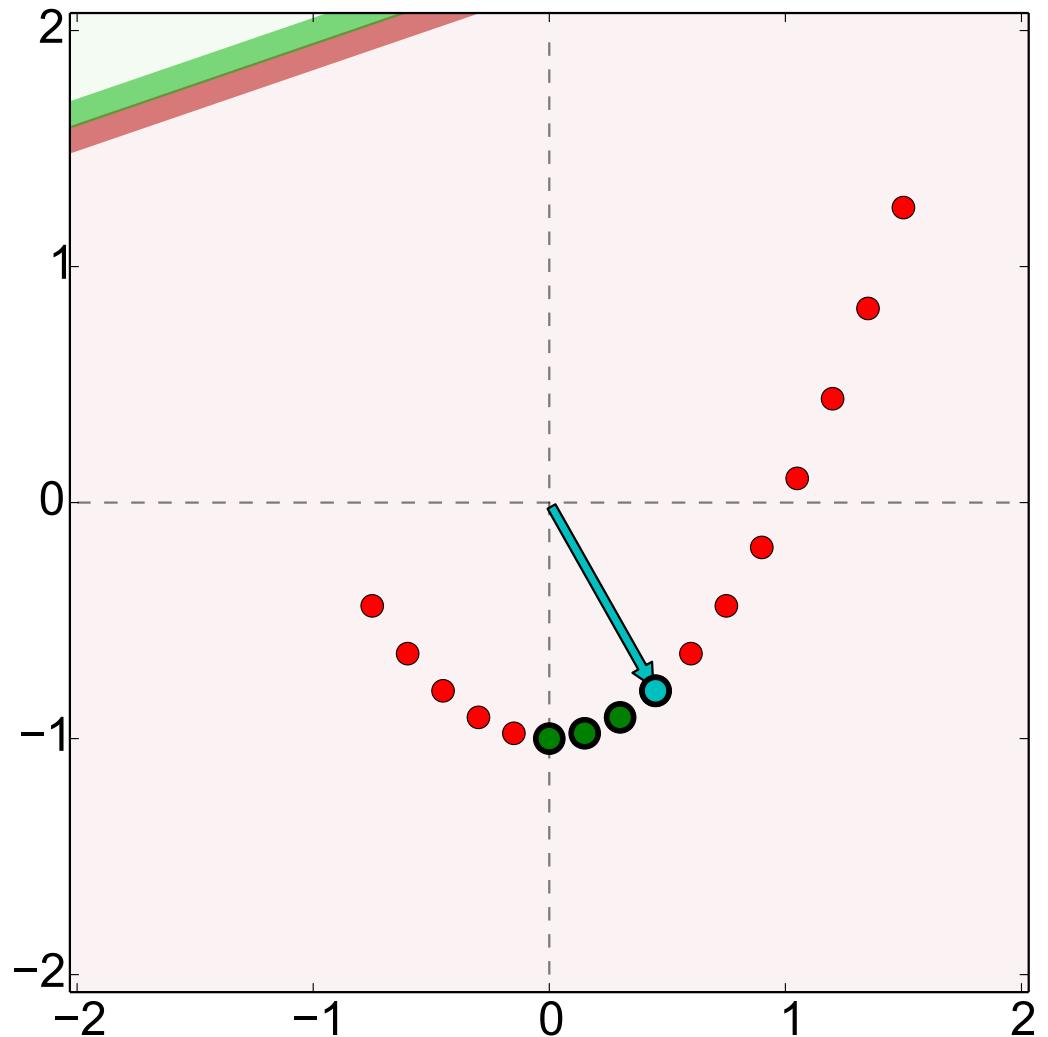
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 3)$$

6 points visited so far.

Lifting, Example 1, Iter. 4

● class 1, ● class -1, ●/●=misclassified,

● the weight updater

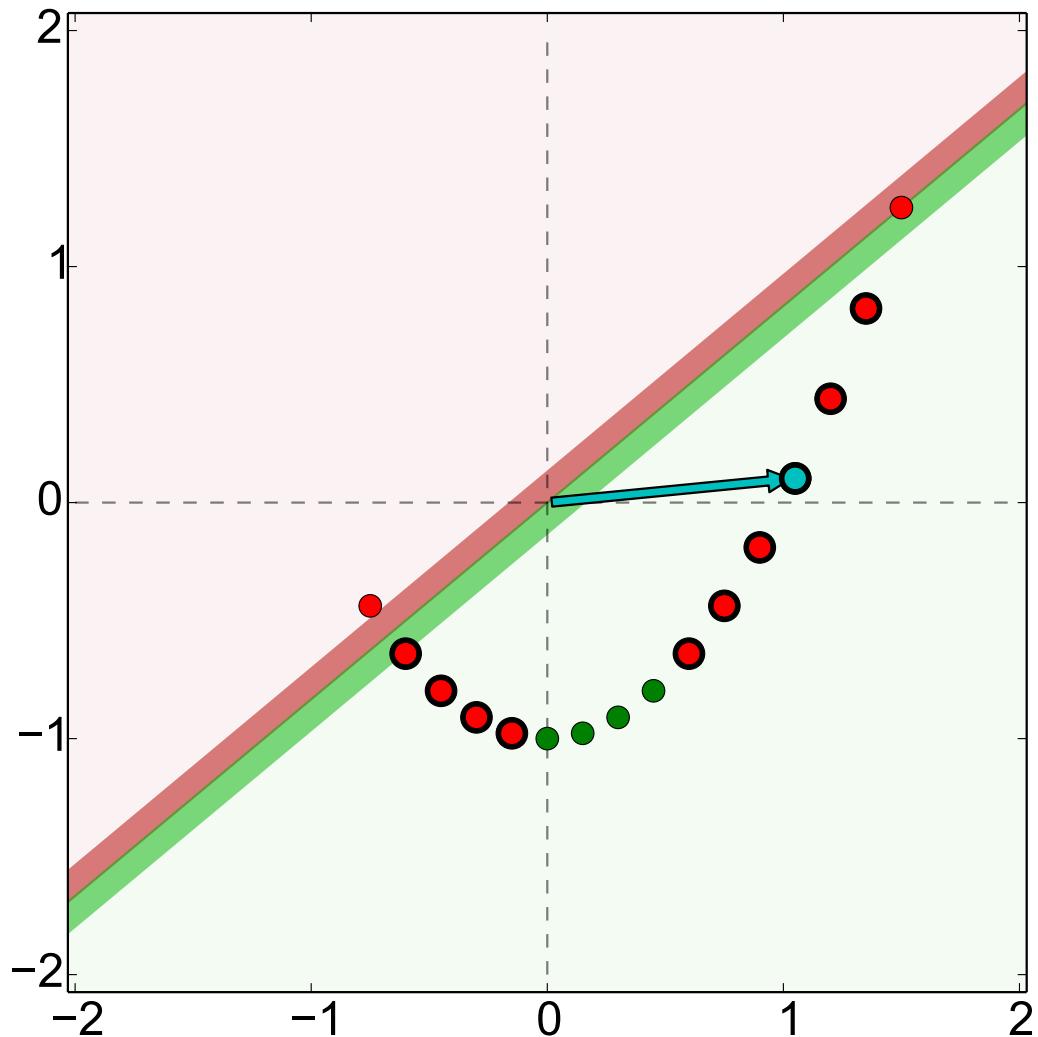


$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 4)$$

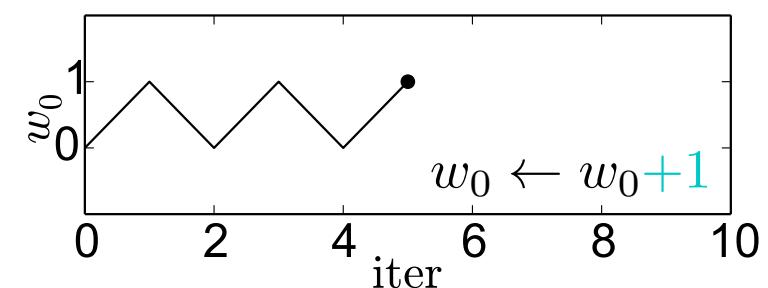
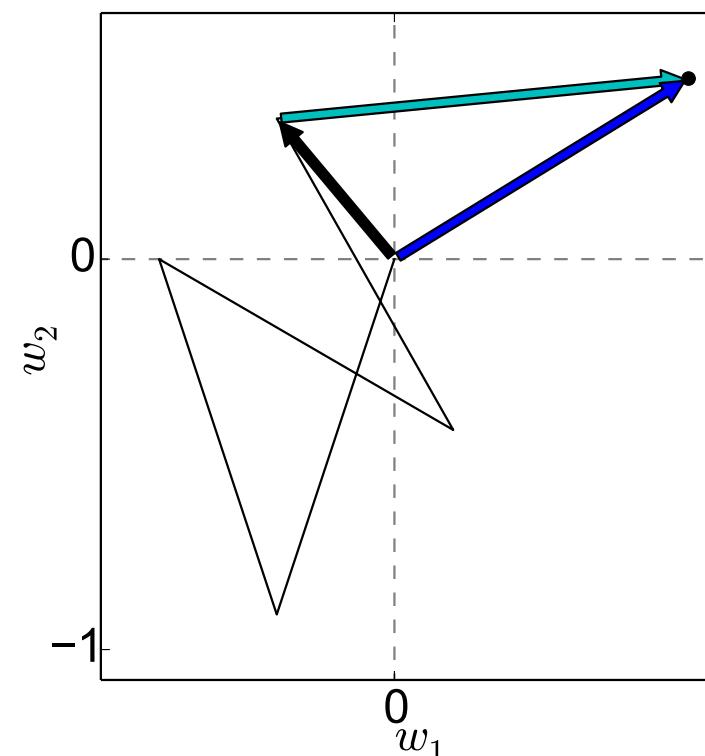
13 points visited so far.

Lifting, Example 1, Iter. 5

● class 1, ● class -1, ●/●=misclassified,

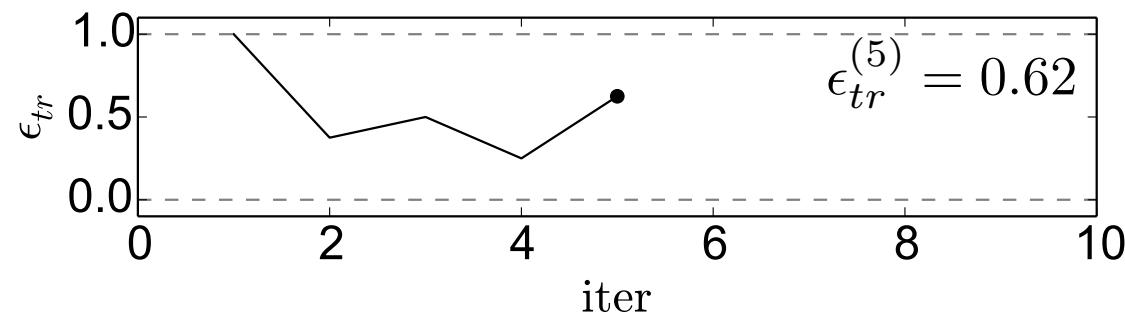


● the weight updater



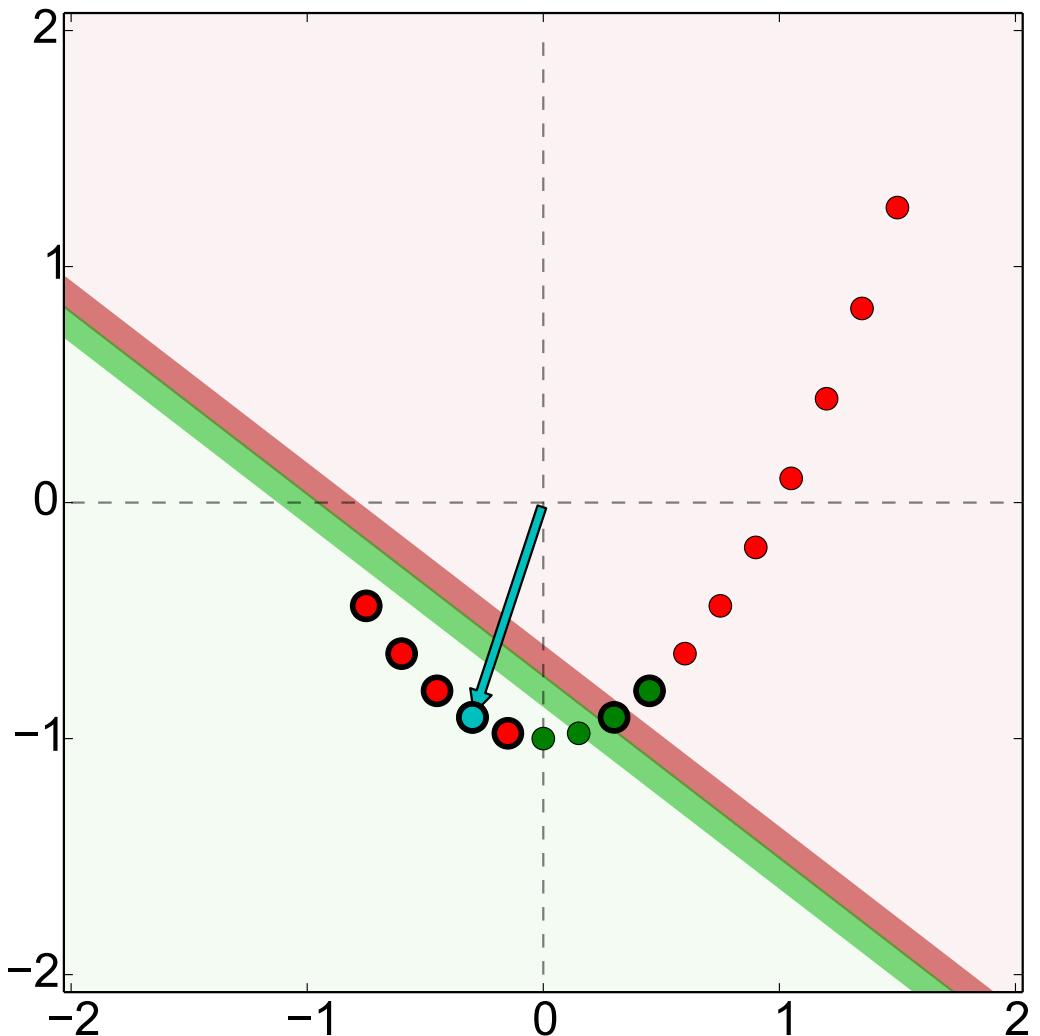
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 5)$$

14 points visited so far.

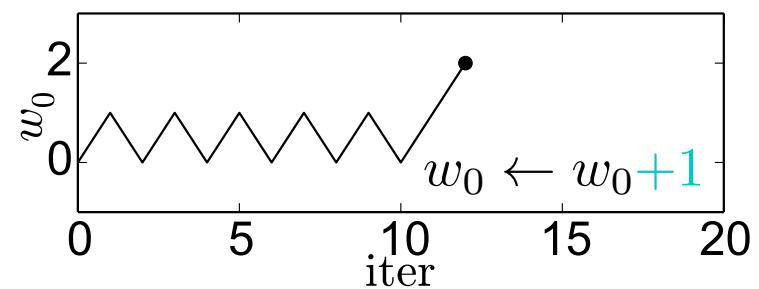
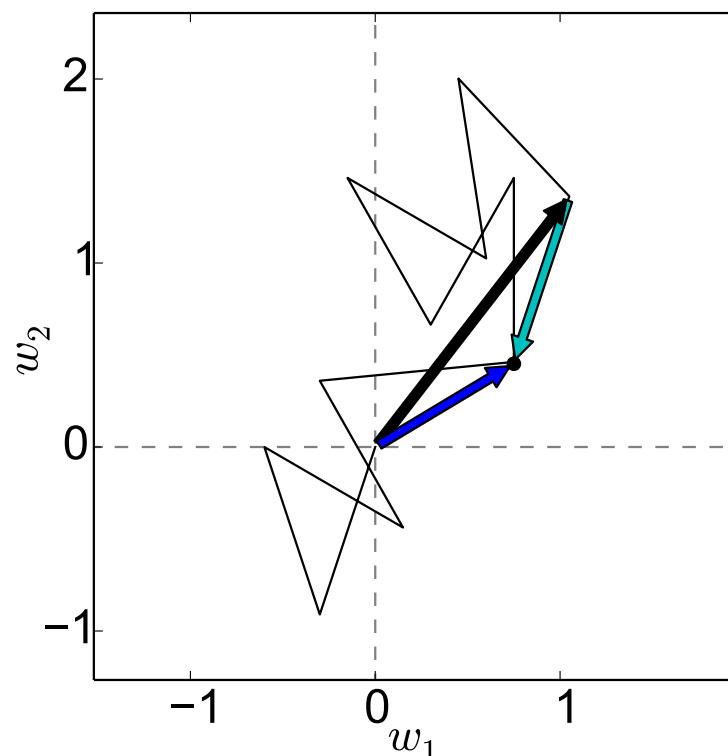


Lifting, Example 1, Iter. 12

● class 1, ● class -1, ●/●=misclassified,

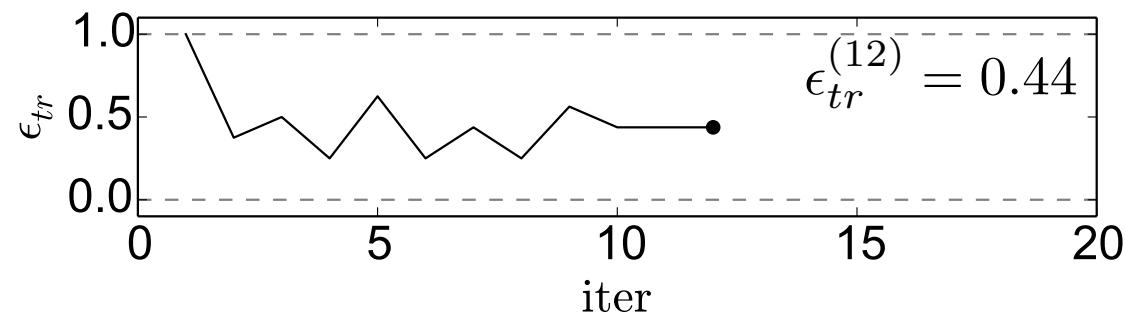


● the weight updater



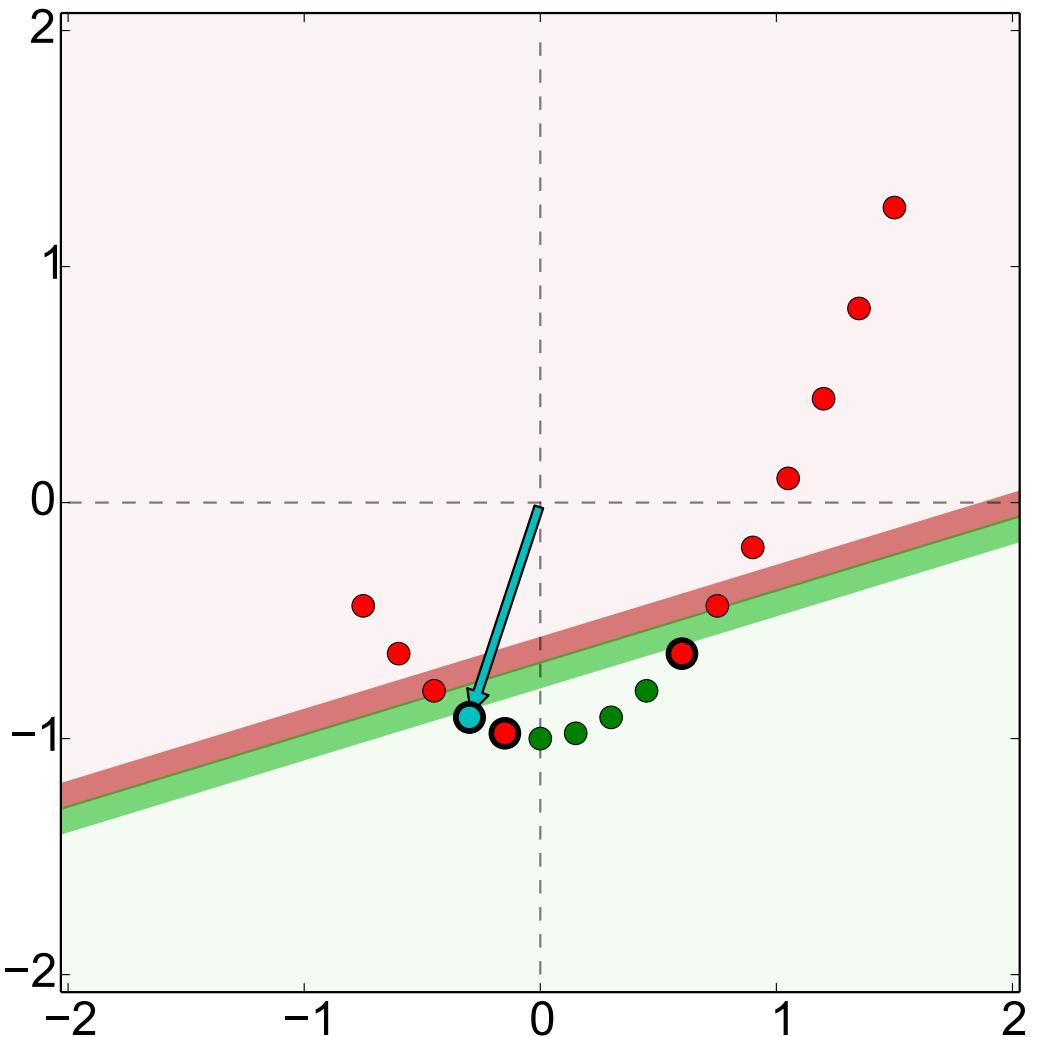
$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 12)$$

36 points visited so far.

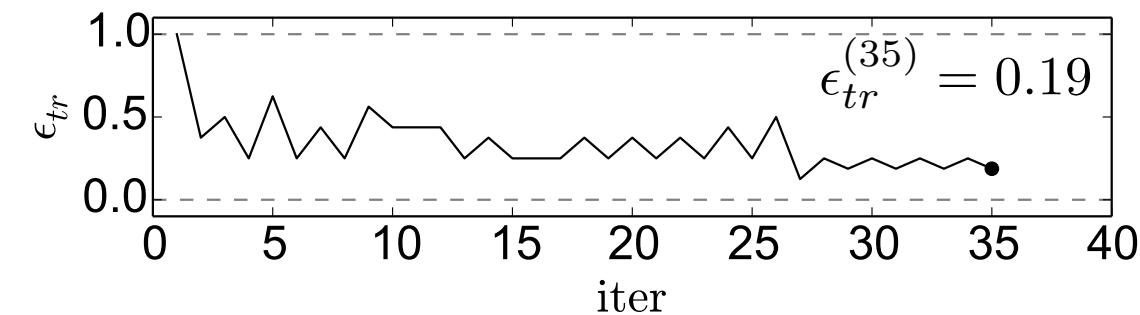
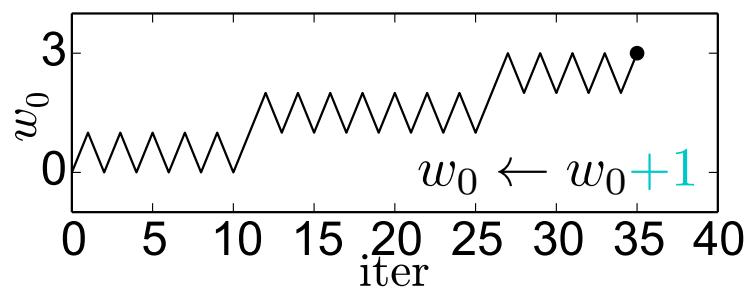
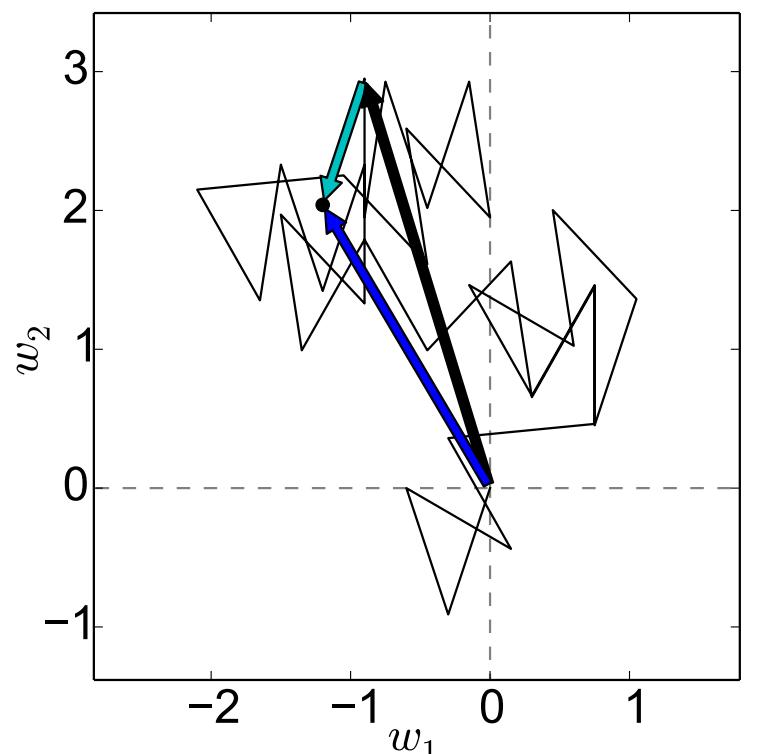


Lifting, Example 1, Iter. 35

● class 1, ● class -1, ●/●=misclassified,



● the weight updater

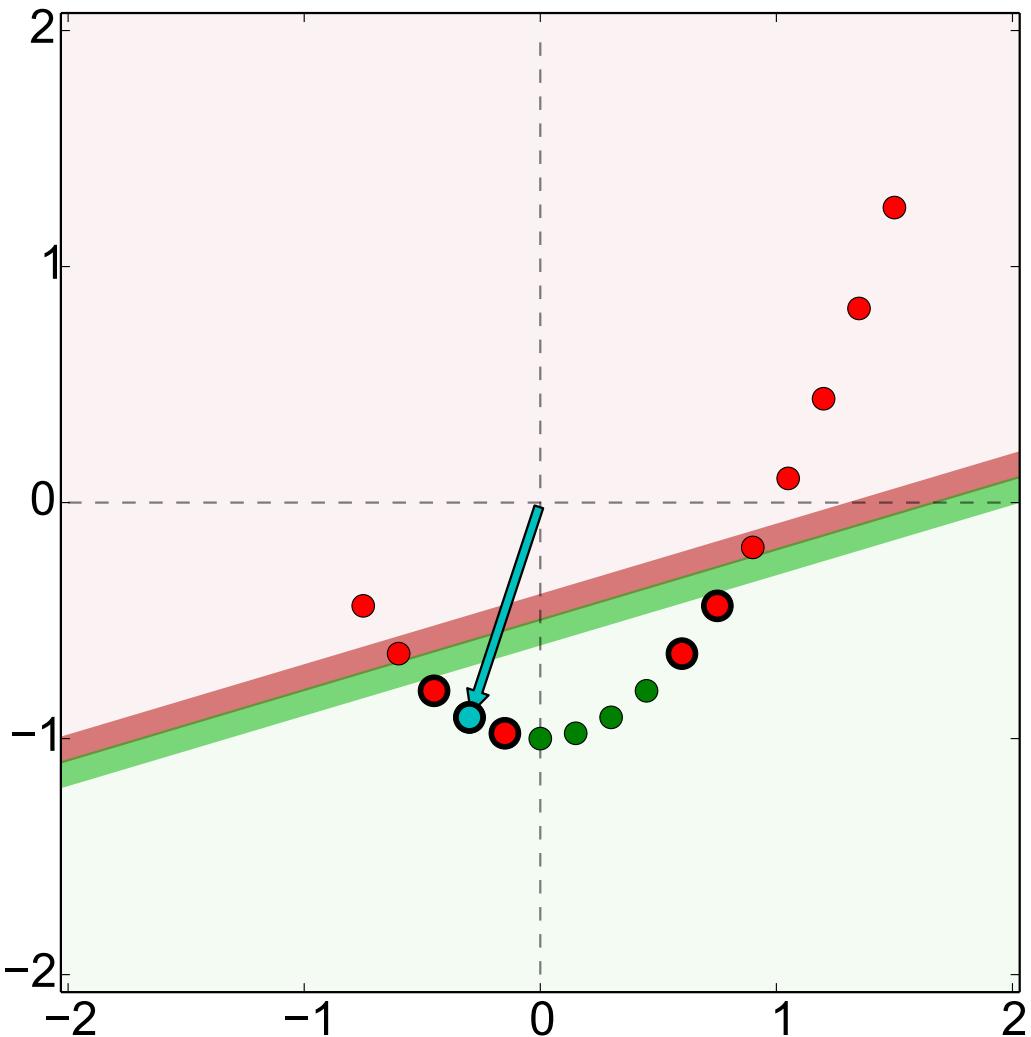


$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 35)$$

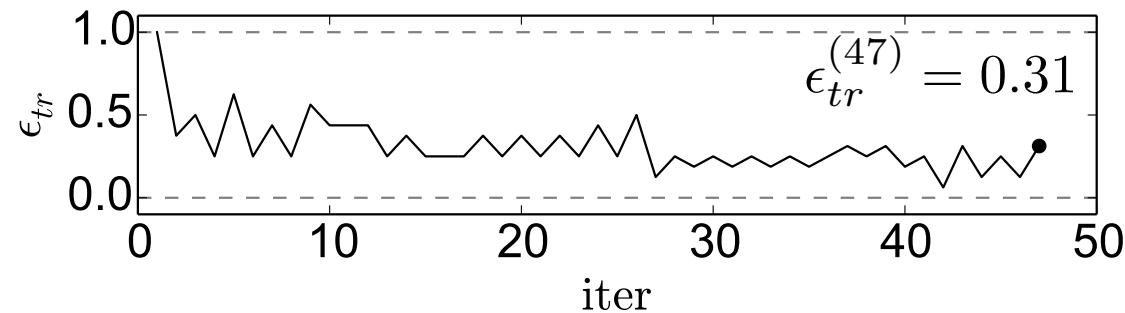
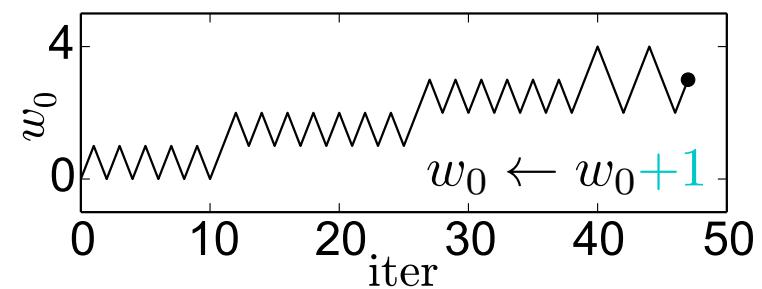
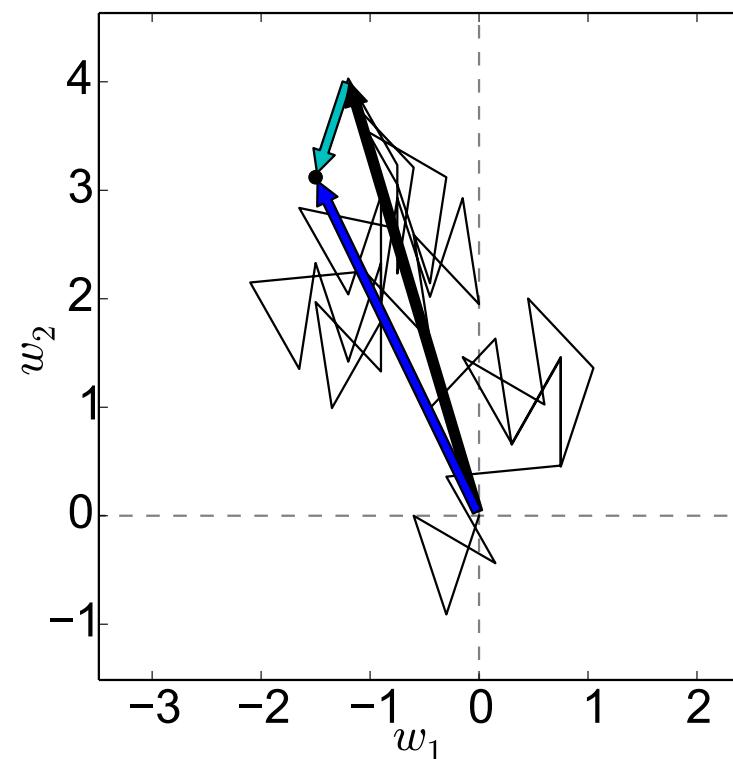
97 points visited so far.

Lifting, Example 1, Iter. 47

● class 1, ● class -1, ●/●=misclassified,



● the weight updater

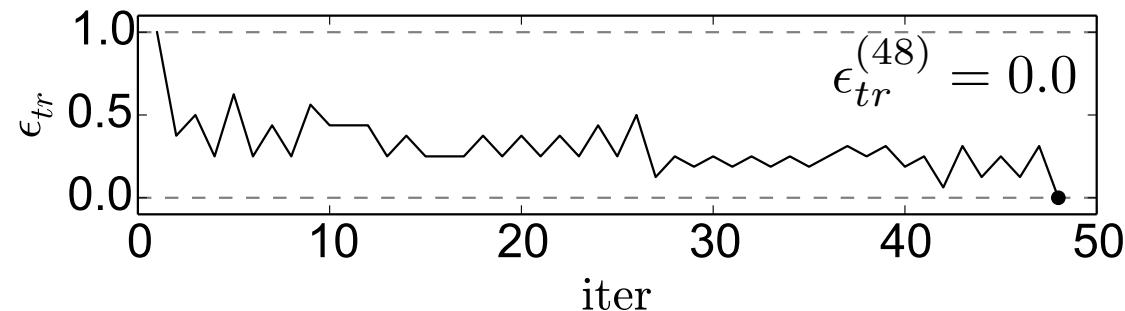
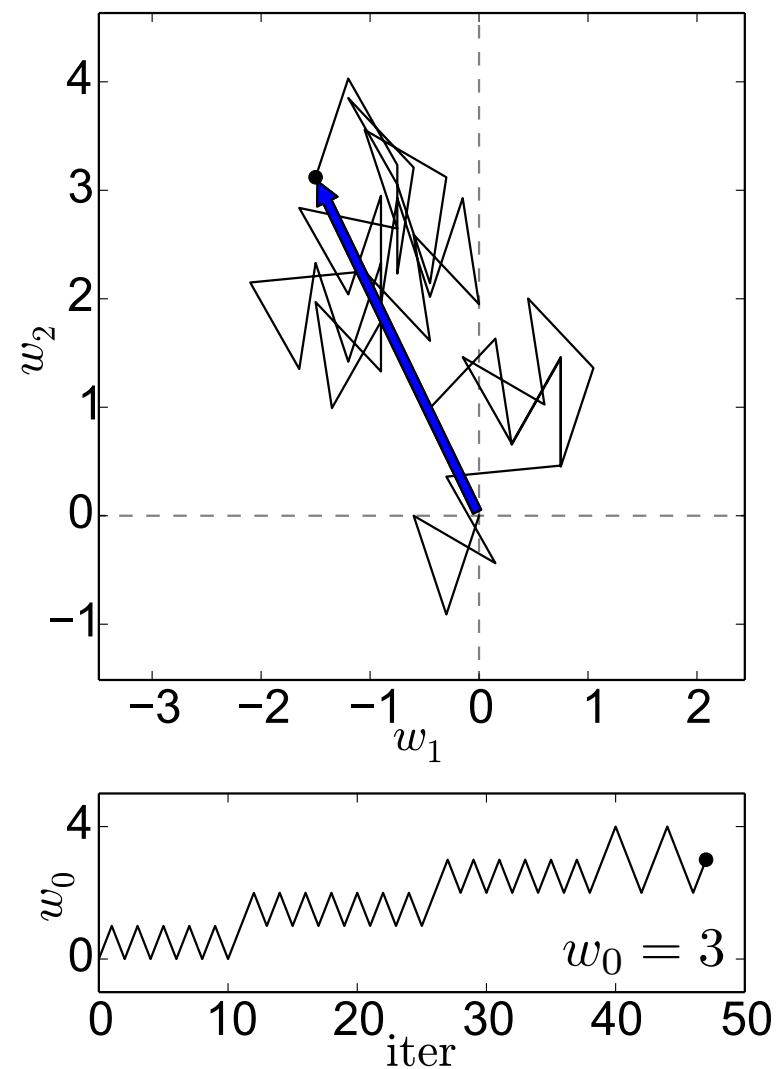
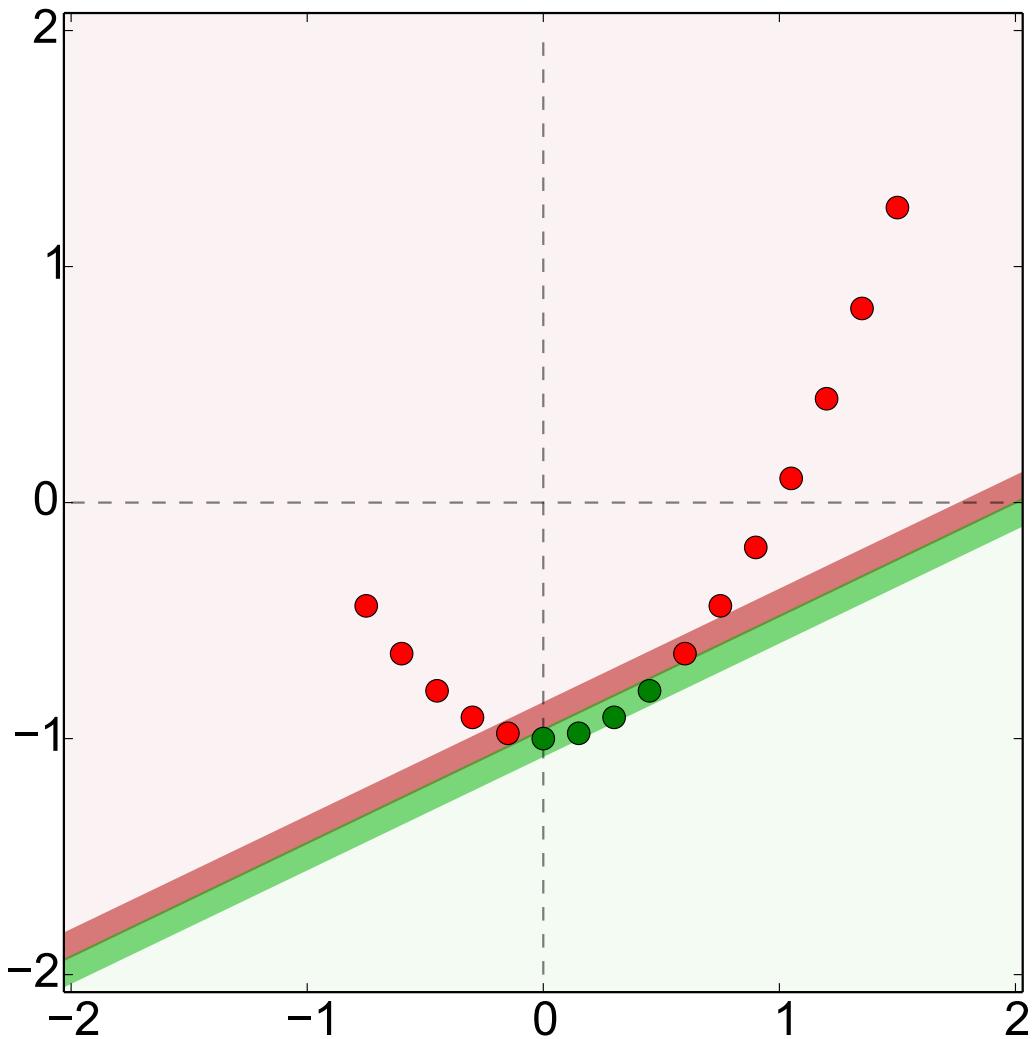


$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + k_j \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \quad (t = 47)$$

139 points visited so far.

Lifting, Example 1, Iter. 48

● class 1, ● class -1, ●/●=misclassified,



160 points visited.
All data classified correctly. Done.

Final weight:
 $\mathbf{w} = (3, -1.5, 3.12)^\top$

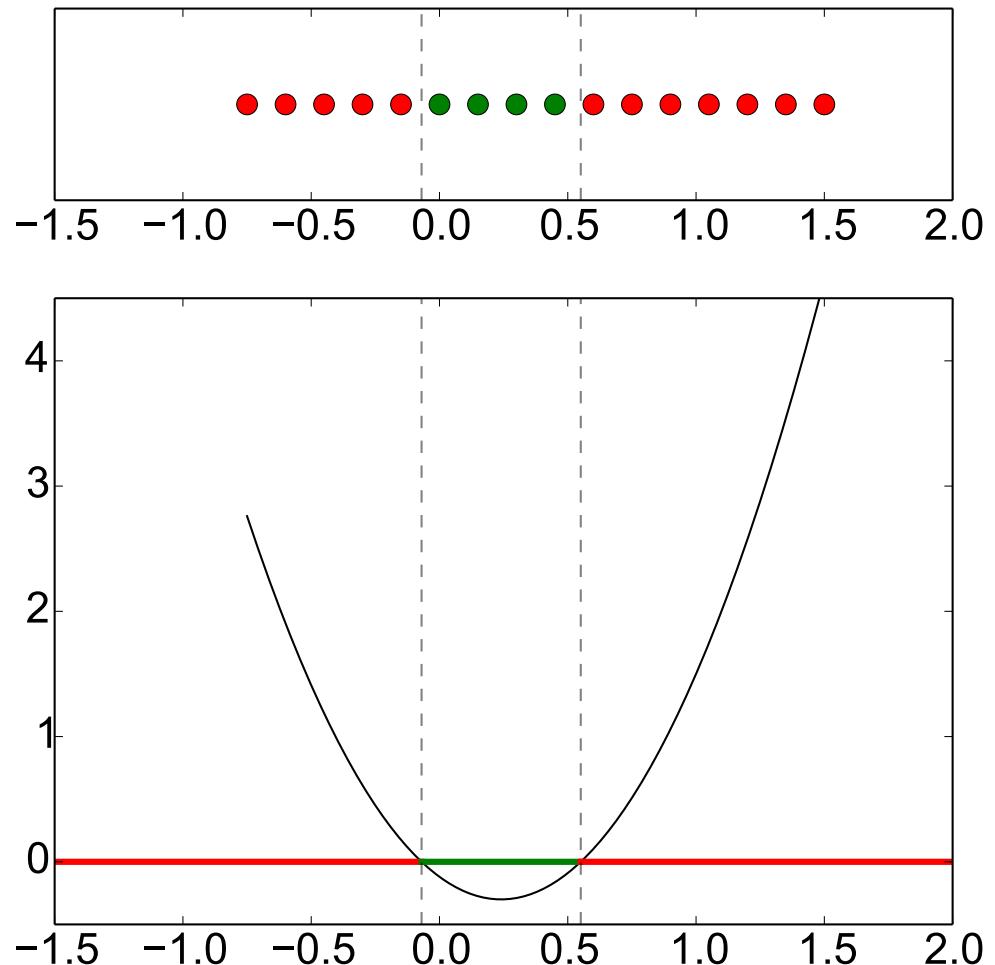
Lifting, Example 1, Result

The final weight vector for the dimensionality-lifted dataset is $\mathbf{w} = (3, -1.5, 3.12)^\top$.

The resulting discriminant function is:

$$f(x) = 3 - 1.5x + 3.12(x^2 - 1) \quad (32)$$

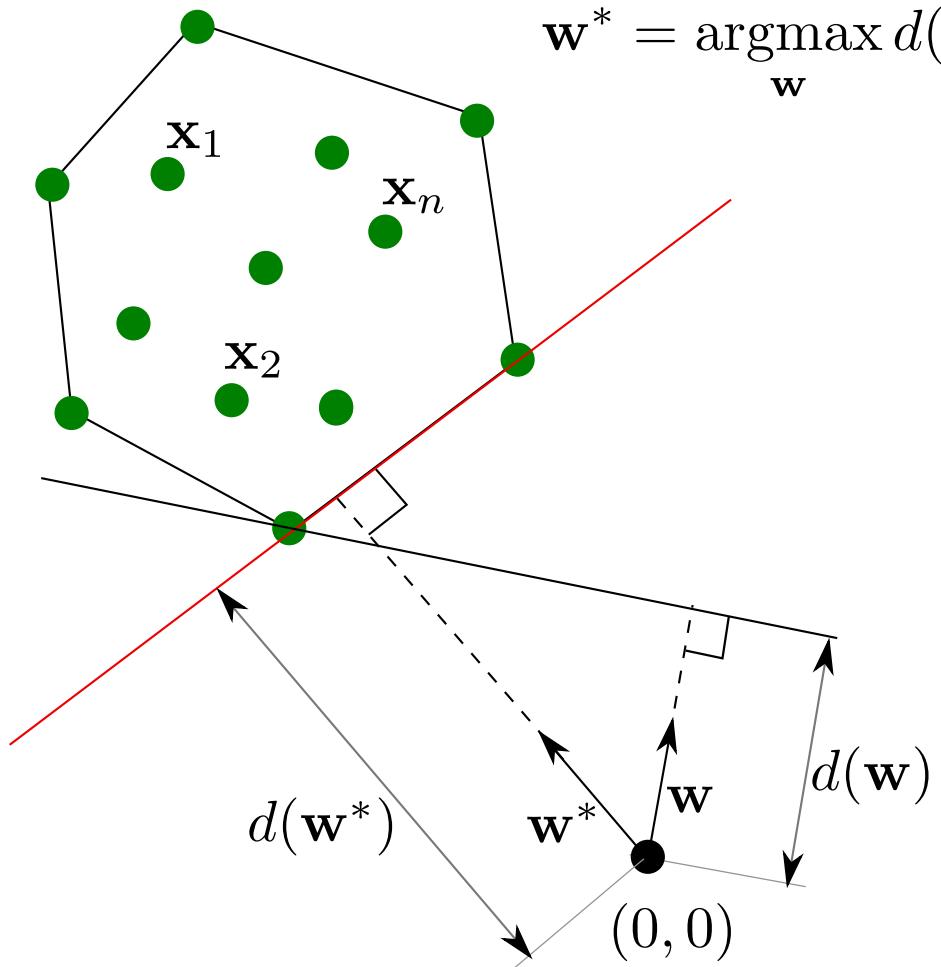
$$= -0.12 - 1.5x + 3.12x^2. \quad (33)$$



Optimal Separating Plane and The Closest Point To The Convex Hull

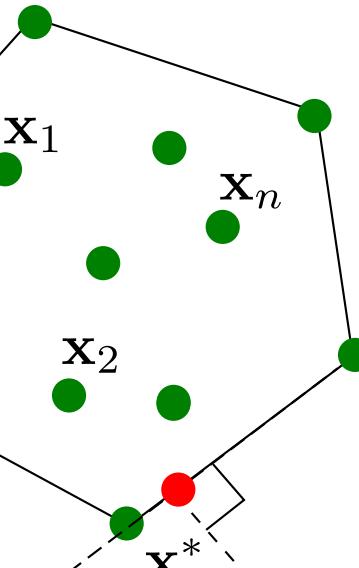
$$d(\mathbf{w}) = \min_j \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}_j$$

$$\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} d(\mathbf{w})$$



\overline{X} : convex hull of $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$

$$\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x} \in \overline{X}} \|\mathbf{x}\|$$

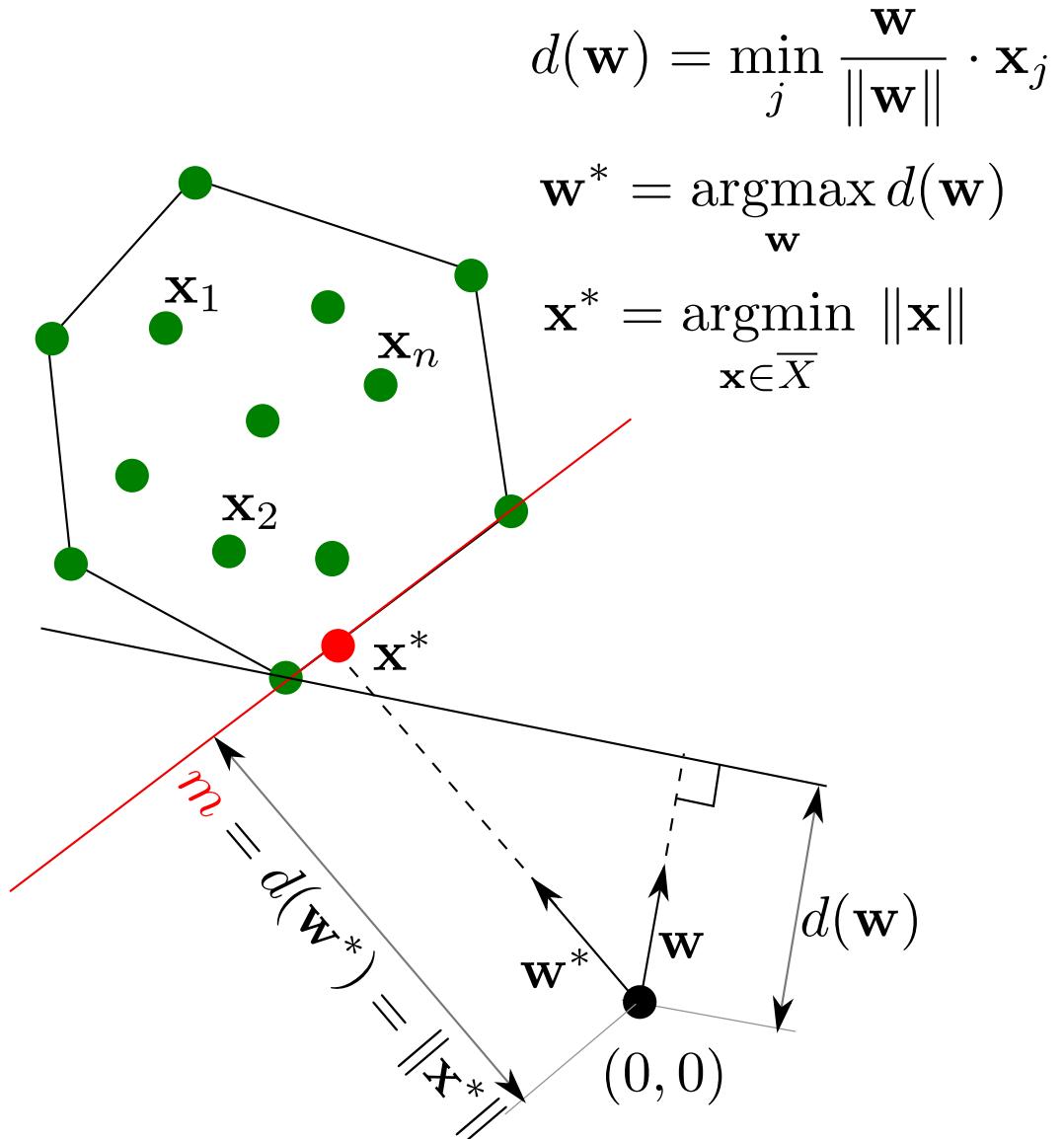


$\|\mathbf{x}^*\|$

$(0, 0)$

The problem of finding optimal separating hyperplane (left) is equivalent to finding closest point in the convex hull (right). Recall that the classifier that maximises separation minimises the structural risk.

Bounds For Margin m



$$\min_j \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}_j \right) \leq m \leq \|\mathbf{w}\|, \mathbf{w} \in \overline{X} \quad (34)$$

ε -Solution

- ◆ The aim is to speed up the algorithm.
- ◆ The allowed uncertainty ε is introduced.

$$\|\mathbf{w}\| - \min_j \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot \mathbf{x}_j \right) \leq \varepsilon$$

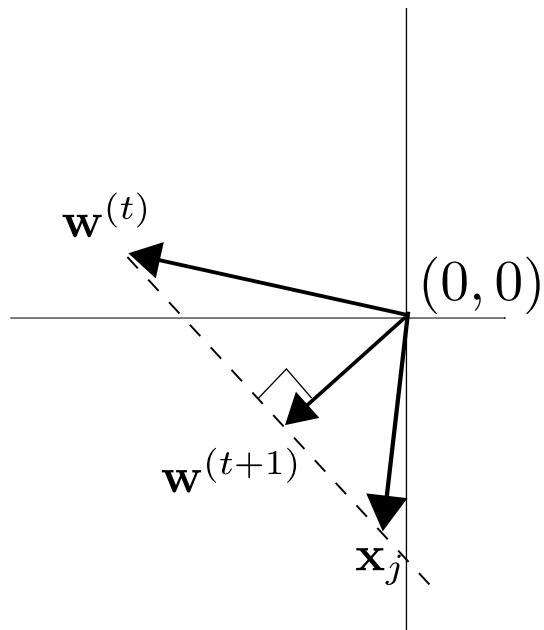
Training Algorithm 2 – Kozinec (1973)

1. $\mathbf{w}^{(0)} = \mathbf{x}_j$, i.e. any observation.
2. A wrongly classified observation \mathbf{x}_j is sought, i.e., $\mathbf{w}^{(t)} \cdot \mathbf{x}_j \leq 0$, $j \in J$.
3. If there is no wrongly classified observation then the algorithm finishes otherwise

$$\mathbf{w}^{(t+1)} = (1 - \kappa^*) \mathbf{w}^{(t)} + \kappa^* \mathbf{x}_j,$$

$$\kappa^* = \operatorname{argmin}_{\kappa \in (0,1)} \|(1 - \kappa) \mathbf{w}^{(t)} + \kappa \mathbf{x}_j\|$$

4. Goto 2.

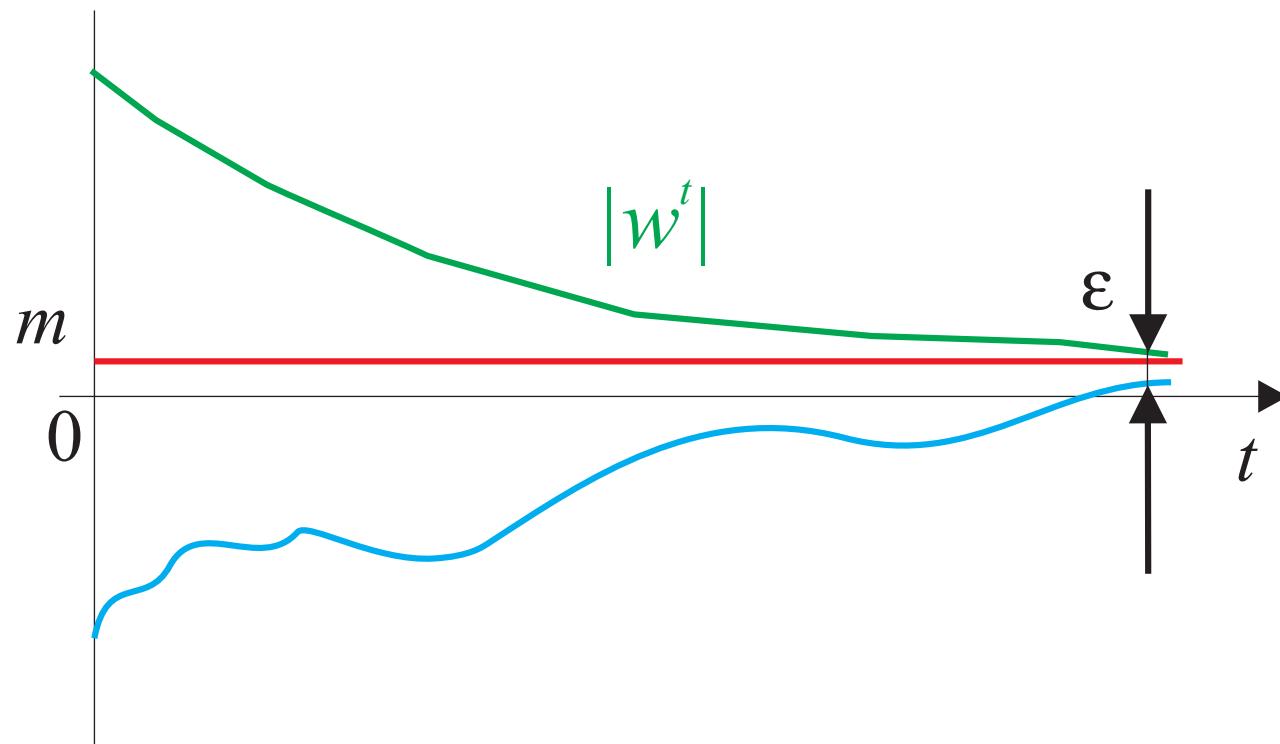


Kozinec and ε -Solution

The second step of Kozinec algorithm is modified to:

A wrongly classified observation \mathbf{x}_j is sought for which

$$|\mathbf{w}^{(t)}| - \min_j \left(\frac{\mathbf{w}^{(t)}}{|\mathbf{w}^{(t)}|} \cdot \mathbf{x}_j \right) \geq \varepsilon$$



Support Vector Machines

Lecturer:
Jiří Matas

Authors:
Ondřej Drbohlav, Jiří Matas

Centre for Machine Perception
Czech Technical University, Prague
<http://cmp.felk.cvut.cz>

Slide credits:
Alexander Apartsin

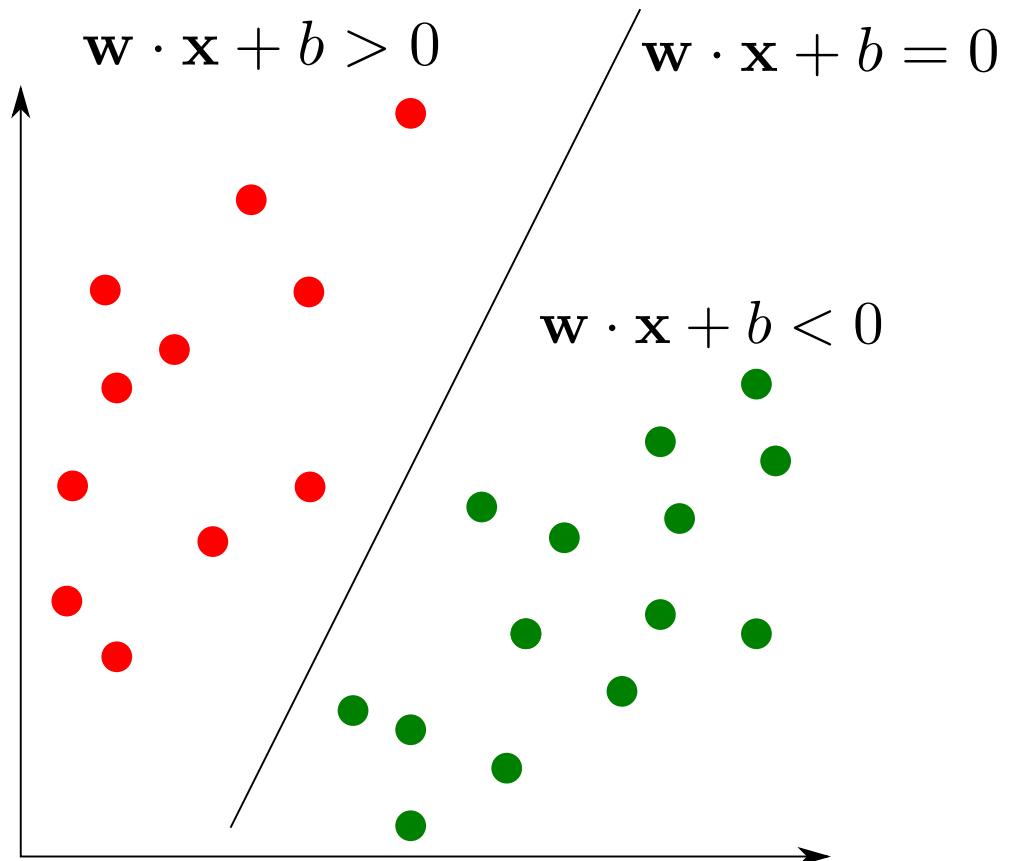
12.11.2018

A Linear Classifier

Classification according to signum of an affine function of \mathbf{x} :

$$q(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) \quad (1)$$

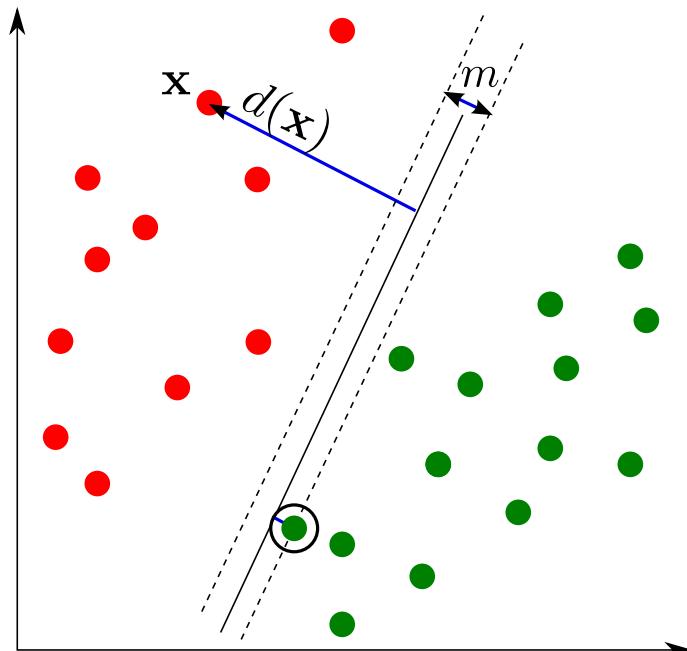
A solution for $\{\mathbf{w}, b\}$ correctly classifying the training set:



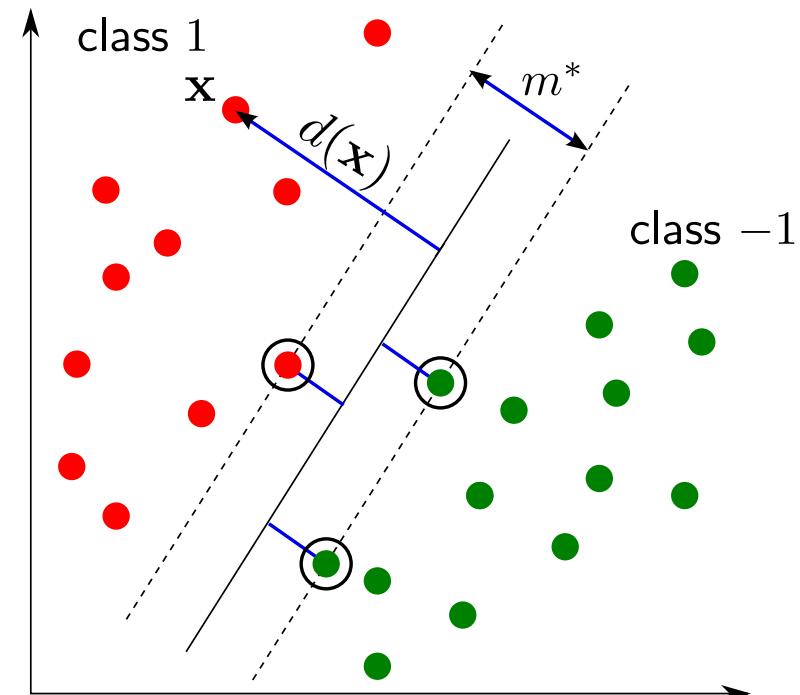
Maximum Margin Linear Classifier

- ◆ Let $d(\mathbf{x})$ denote the distance of a point $\mathbf{x} \in \mathcal{T}$ from the training set \mathcal{T} to the decision boundary of a linear classifier given by parameters (\mathbf{w}, b) .
- ◆ The margin m of a linear classifier (\mathbf{w}, b) is defined as follows:
 - If the classifier classifies all data correctly then $m = 2 \min_{\mathbf{x} \in \mathcal{T}} d(\mathbf{x})$.
Points $\mathbf{x} \in \mathcal{T}$ satisfying $m = 2d(\mathbf{x})$ are called **support vectors**.
 - If the classifier has non-zero error on \mathcal{T} then $m = 0$.
- ◆ **Goal:** Find the classifier (\mathbf{w}^*, b^*) maximizing the margin. Vapnik justifies the use of maximum margin from the viewpoint of Structural Risk Minimization.

Margin of a classifier (\mathbf{w}, b) :



Maximum margin classifier (\mathbf{w}^*, b^*) :

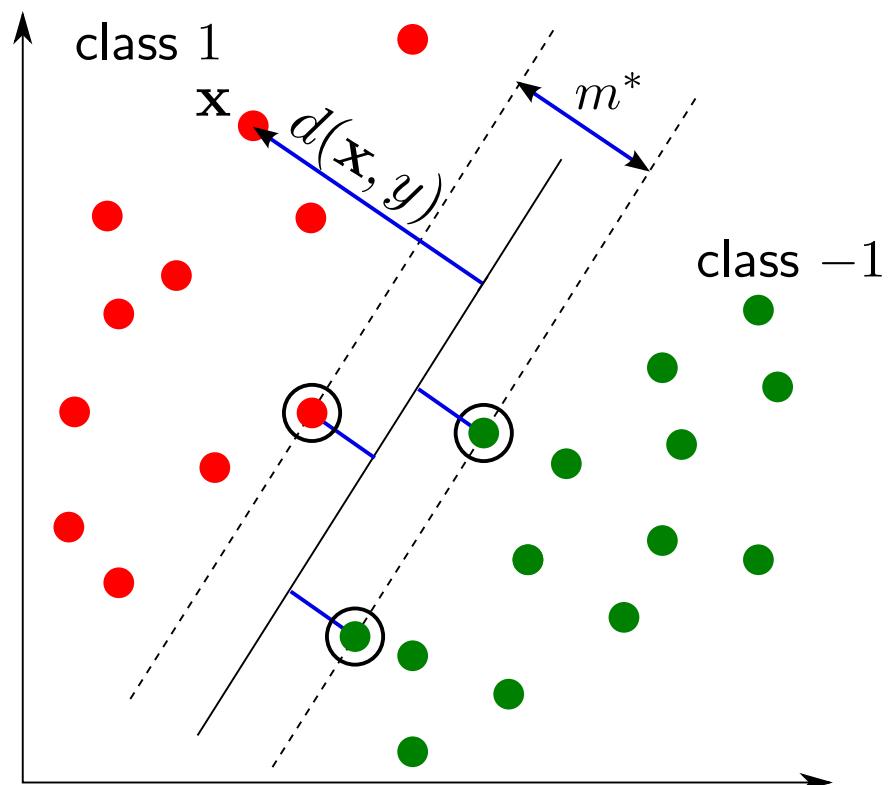


Maximizing Margin, Formulation

- Let us define signed distance $d(\mathbf{x}, y)$ of a point \mathbf{x} belonging to class $y \in \{1, -1\}$ to the decision boundary of classifier (\mathbf{w}, b) :

$$d(\mathbf{x}, y) = \frac{y(\mathbf{w} \cdot \mathbf{x} + b)}{\|\mathbf{w}\|} \quad (2)$$

- We search for (\mathbf{w}, b) such that $d(\mathbf{x}, y) > 0$ for all training data (all training points are in their class' half-space). This is equivalent to $y(\mathbf{w} \cdot \mathbf{x} + b) > 0$.



Optimization task:

$$(\mathbf{w}^*, b^*) = \operatorname{argmax}_{\mathbf{w}, b} \min_{(\mathbf{x}, y) \in \mathcal{T}} 2d(\mathbf{x}, y)$$

subject to:

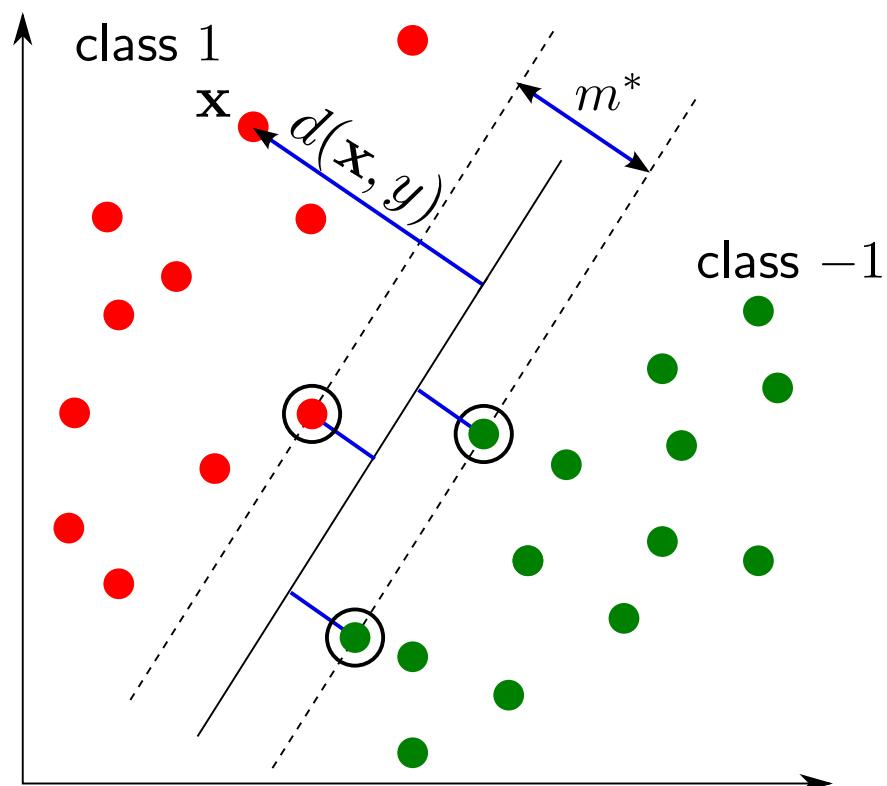
$$y(\mathbf{w} \cdot \mathbf{x} + b) > 0, \forall (\mathbf{x}, y) \in \mathcal{T} \quad (C)$$

Maximizing Margin, Scale Ambiguity

- ◆ There is a scale ambiguity in the parameters (\mathbf{w}, b) . Any feasible (\mathbf{w}, b) (that is, satisfying Eq. (C)) can be multiplied by a positive constant $(\mathbf{w}, b) \rightarrow (\sigma\mathbf{w}, \sigma b)$, and:
 - feasibility does not change, as

$$y(\sigma\mathbf{w} \cdot \mathbf{x} + \sigma b) = \sigma y(\mathbf{w} \cdot \mathbf{x} + b) > 0 \Leftrightarrow y(\mathbf{w} \cdot \mathbf{x} + b) > 0, \text{ and} \quad (3)$$

- signed distances do not change, as



$$d(\mathbf{x}, y) = \frac{y(\sigma\mathbf{w} \cdot \mathbf{x} + \sigma b)}{\|\sigma\mathbf{w}\|} = \frac{y(\mathbf{w} \cdot \mathbf{x} + b)}{\|\mathbf{w}\|}. \quad (4)$$

Optimization task:

$$(\mathbf{w}^*, b^*) = \operatorname{argmax}_{\mathbf{w}, b} \min_{(\mathbf{x}, y) \in \mathcal{T}} 2d(\mathbf{x}, y)$$

subject to:

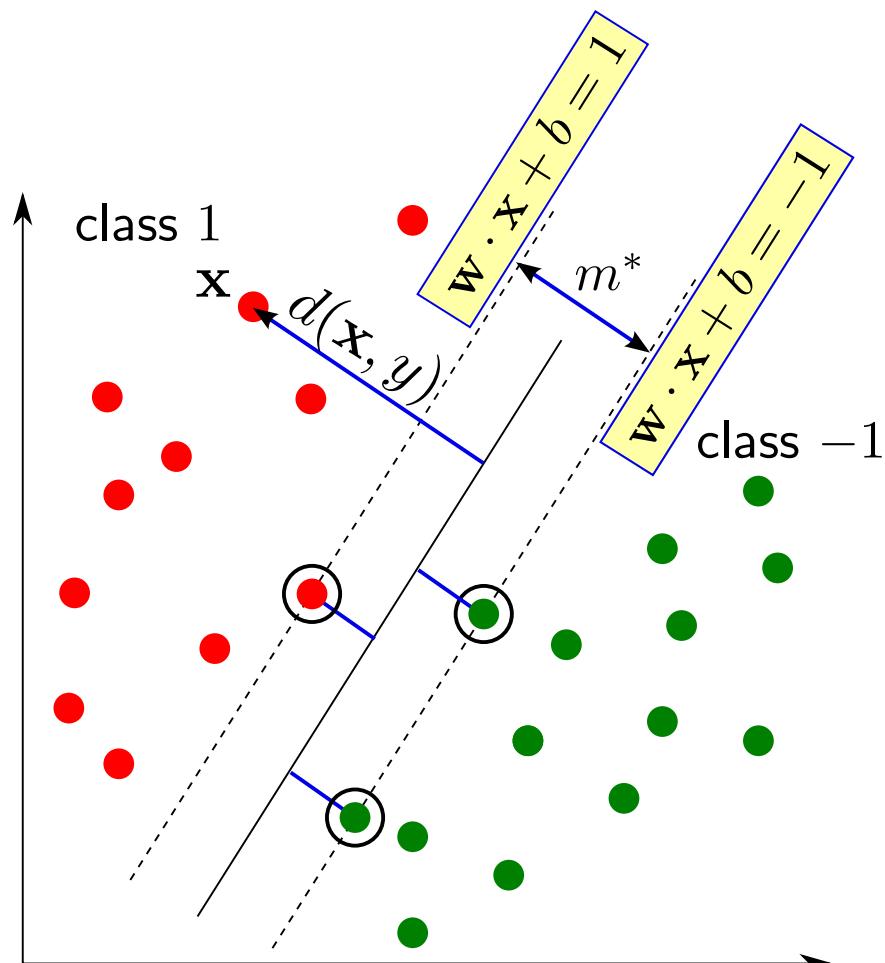
$$y(\mathbf{w} \cdot \mathbf{x} + b) > 0, \forall (\mathbf{x}, y) \in \mathcal{T} \quad (C)$$

Maximizing Margin, Fixing Scale

- ◆ Constraints $y(\mathbf{w} \cdot \mathbf{x} + b) > 0$ are equivalent to $y(\mathbf{w} \cdot \mathbf{x} + b) \geq \epsilon$ (with $\epsilon > 0$)
- ◆ Break the scale ambiguity by setting $\epsilon = 1$:

$$(\mathbf{w}^*, b^*) = \operatorname{argmax}_{\mathbf{w}, b} \min_{(\mathbf{x}, y) \in \mathcal{T}} 2d(\mathbf{x}, y)$$

subject to: $y(\mathbf{w} \cdot \mathbf{x} + b) \geq 1, \forall (\mathbf{x}, y) \in \mathcal{T}$ (5)



Optimization task (original):

$$(\mathbf{w}^*, b^*) = \operatorname{argmax}_{\mathbf{w}, b} \min_{(\mathbf{x}, y) \in \mathcal{T}} 2d(\mathbf{x}, y)$$

subject to:

$$y(\mathbf{w} \cdot \mathbf{x} + b) > 0, \forall (\mathbf{x}, y) \in \mathcal{T} \quad (C)$$

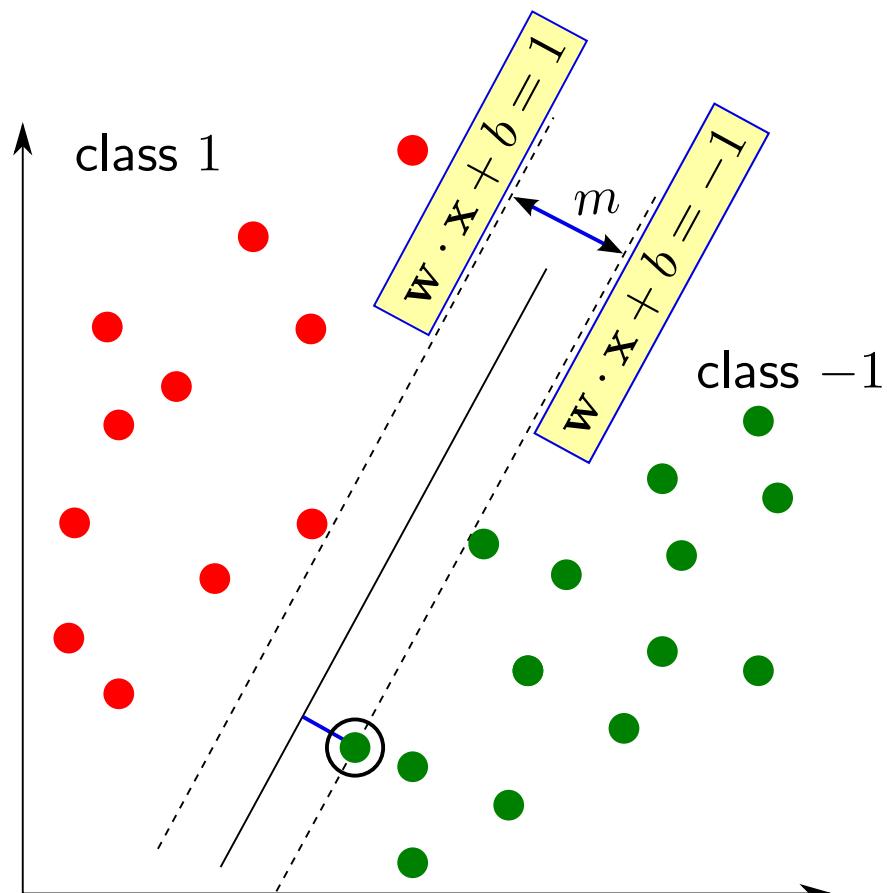
$$d(\mathbf{x}, y) = \frac{y(\mathbf{w} \cdot \mathbf{x} + b)}{\|\mathbf{w}\|}$$

Maximizing Margin, Final Optimization Formulation (1)

- That is, all points must be outside the strip delineated by the two lines $\mathbf{w} \cdot \mathbf{x} + b = 1$ and $\mathbf{w} \cdot \mathbf{x} + b = -1$. The width of this strip is $\frac{2}{\|\mathbf{w}\|}$. It follows that the maximum margin m^* is

$$m^* = \max_{\mathbf{w}, b} \min_{(\mathbf{x}, y) \in \mathcal{T}} 2d(\mathbf{x}, y) = \max_{\mathbf{w}, b} \frac{2}{\|\mathbf{w}\|}$$

subject to: $y(\mathbf{w} \cdot \mathbf{x} + b) \geq 1, \forall (\mathbf{x}, y) \in \mathcal{T}$ (6)



Optimization task (original):

$$(\mathbf{w}^*, b^*) = \operatorname{argmax}_{\mathbf{w}, b} \min_{(\mathbf{x}, y) \in \mathcal{T}} 2d(\mathbf{x}, y)$$

subject to:

$$y(\mathbf{w} \cdot \mathbf{x} + b) > 0, \forall (\mathbf{x}, y) \in \mathcal{T} \quad (C)$$

$$d(\mathbf{x}, y) = \frac{y(\mathbf{w} \cdot \mathbf{x} + b)}{\|\mathbf{w}\|}$$

Maximizing Margin, Final Optimization Formulation (2)

- That is, all points must be outside the strip delineated by the two lines $\mathbf{w} \cdot \mathbf{x} + b = 1$ and $\mathbf{w} \cdot \mathbf{x} + b = -1$. The width of this strip is $\frac{2}{\|\mathbf{w}\|}$. It follows that the maximum margin m^* is

$$m^* = \max_{\mathbf{w}, b} \min_{(\mathbf{x}, y) \in \mathcal{T}} 2d(\mathbf{x}, y) = \max_{\mathbf{w}, b} \frac{2}{\|\mathbf{w}\|}$$

subject to: $y(\mathbf{w} \cdot \mathbf{x} + b) \geq 1, \forall (\mathbf{x}, y) \in \mathcal{T}$ (7)

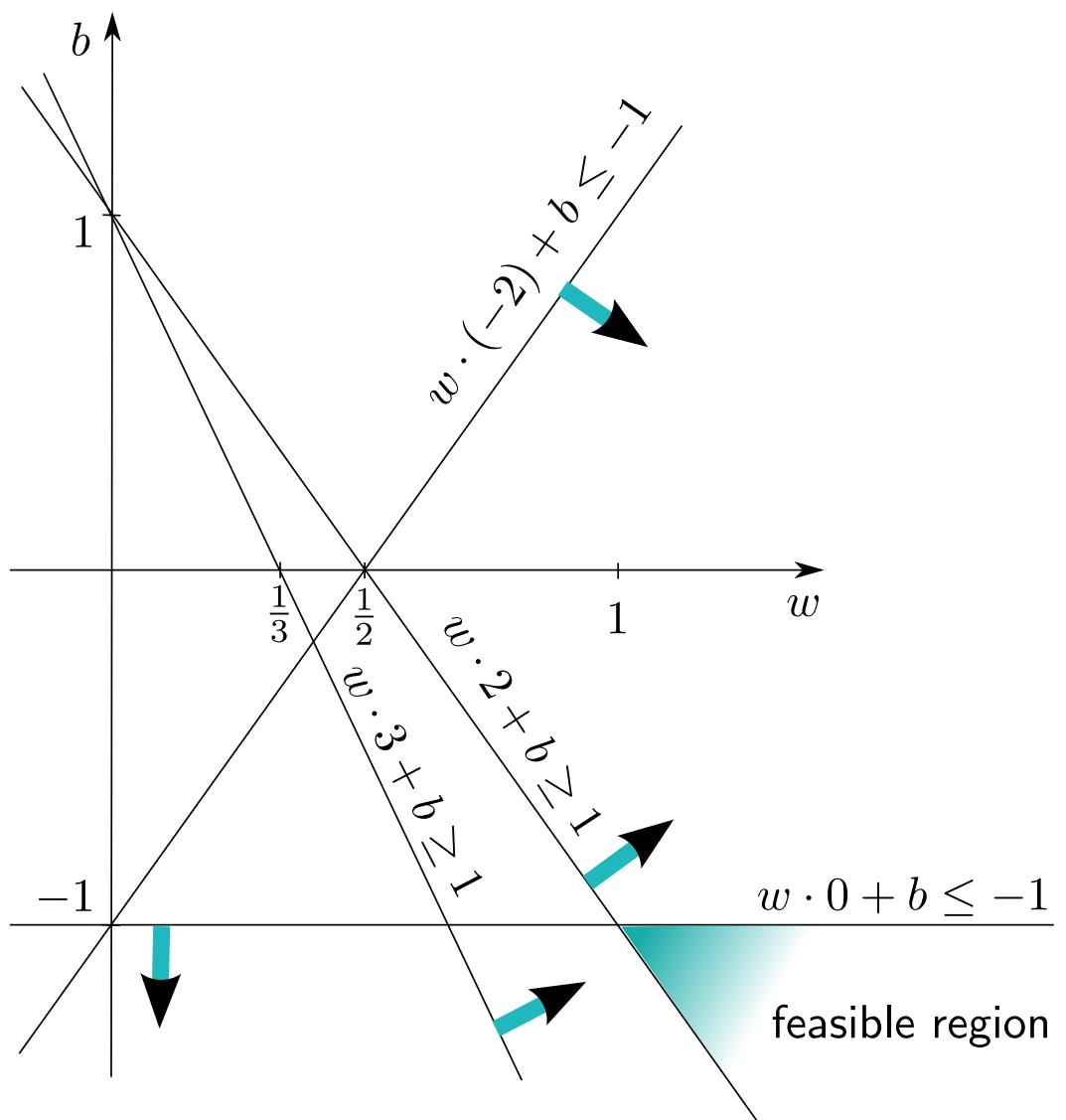
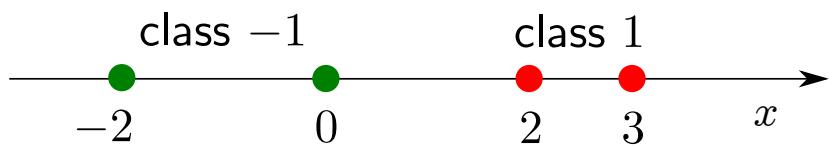
- There holds: $\underset{\mathbf{w}}{\operatorname{argmax}} \frac{2}{\|\mathbf{w}\|} = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{w}\| = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{w}\|^2$. Therefore, the (\mathbf{w}^*, b^*) maximizing the margin are:

$$(\mathbf{w}^*, b^*) = \underset{(\mathbf{w}, b)}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{w}\|^2$$

subject to: $y(\mathbf{w} \cdot \mathbf{x} + b) \geq 1, \forall (\mathbf{x}, y) \in \mathcal{T}$ (8)

- This is a Quadratic Programming (QP) problem (more generally, it is minimization of a convex function on a convex domain.)

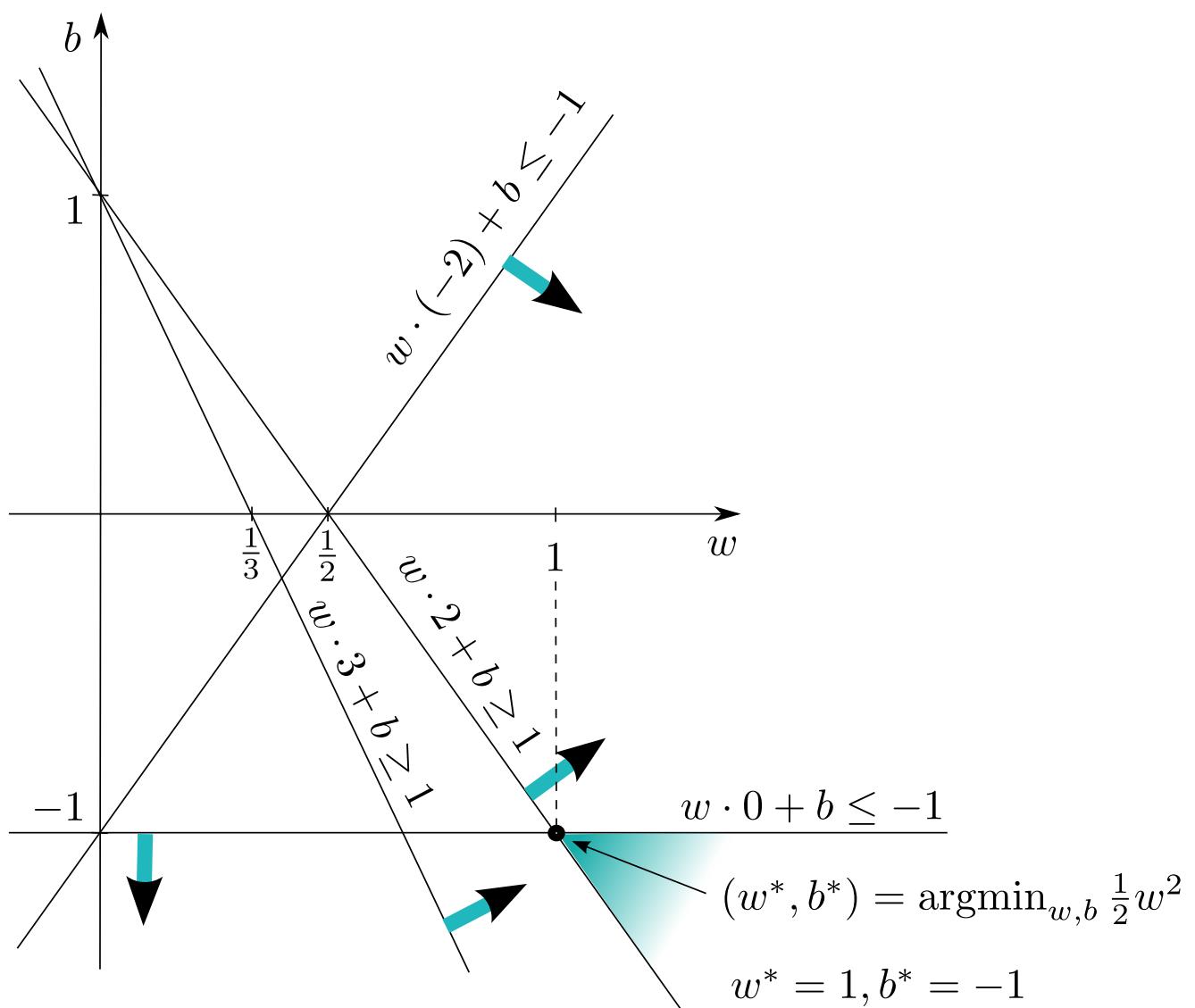
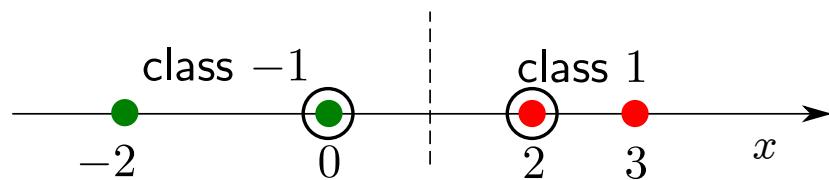
SVM, Example (1D)



SVM, Example (1D), Result

10/38

$$wx + b = x - 1 = 0$$



SVM, Primal Problem

The derived optimization problem for \mathbf{w} and b is

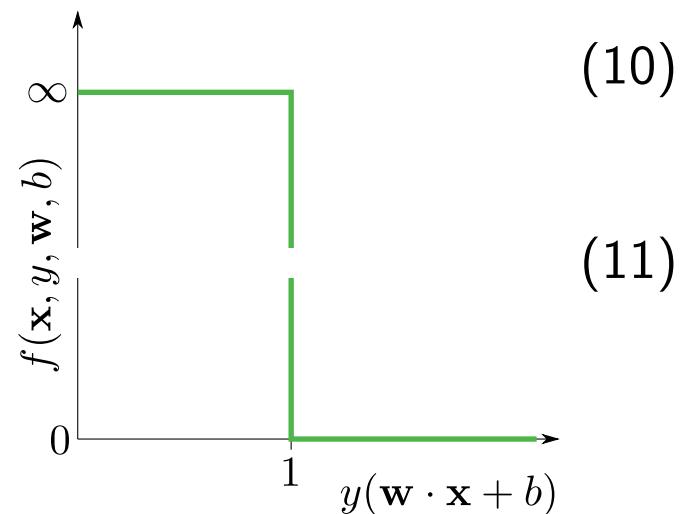
$$(\mathbf{w}^*, b^*) = \underset{(\mathbf{w}, b)}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{w}\|^2$$

subject to: $y(\mathbf{w} \cdot \mathbf{x} + b) \geq 1, \forall (\mathbf{x}, y) \in \mathcal{T}$ (9)

It is called *primal* problem. We will also soon derive the *dual* problem. For now, note that the above optimization task can be equivalently regarded as solving an unconstrained problem (this observation will become handy when deriving the dual problem):

$$(\mathbf{w}^*, b^*) = \underset{(\mathbf{w}, b)}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{(\mathbf{x}, y) \in \mathcal{T}} f(\mathbf{x}, y, \mathbf{w}, b) \right\}, \text{ where } (10)$$

$$f(\mathbf{x}, y, \mathbf{w}, b) = \begin{cases} 0 & \text{if } y(\mathbf{w} \cdot \mathbf{x} + b) \geq 1, \\ \infty & \text{otherwise} \end{cases}$$



Note that $f(\mathbf{x}, y, \mathbf{w}, b)$ for a given (\mathbf{x}, y) is a convex function of \mathbf{w}, b .

The Dual Formulation (1)

Start with just discussed primal formulation. Let $\mathcal{T} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$ be the training set. We want to solve

$$(\mathbf{w}^*, b^*) = \operatorname{argmin}_{(\mathbf{w}, b)} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^N f(\mathbf{x}_i, y_i, \mathbf{w}, b) \right\}, \text{ where}$$

$$f(\mathbf{x}_i, y_i, \mathbf{w}, b) = \begin{cases} 0 & \text{if } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \\ \infty & \text{otherwise} \end{cases}. \quad (12)$$

This is the same as (α_i 's are non-negative multipliers):

$$(\mathbf{w}^*, b^*) = \operatorname{argmin}_{\mathbf{w}, b} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + \max_{\substack{\{\alpha_i\} \\ \alpha_i \geq 0 \\ i \in \{1, \dots, N\}}} \left(- \sum_{i=1}^N \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \right) \right\}. \quad (13)$$

because

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 1 \Rightarrow \max_{\alpha_i} (-\alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1]) = 0 \text{ for } \alpha_i = 0, \quad (14)$$

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) < 1 \Rightarrow \max_{\alpha_i} (-\alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1]) = \infty \text{ for } \alpha_i = \infty, \quad (15)$$

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1 \Rightarrow \max_{\alpha_i} (-\alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1]) = 0 \text{ for any } \alpha_i \geq 0. \quad (16)$$

The Dual Formulation (2)

This is in turn the same as

$$(\mathbf{w}^*, b^*) = \operatorname{argmin}_{\mathbf{w}, b} \max_{\substack{\{\alpha_i\} \\ \alpha_i \geq 0 \\ i \in \{1, \dots, N\}}} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \right\}. \quad (17)$$

There holds, in full generality, that $\max_p \min_q f(p, q) \leq \min_q \max_p f(p, q)$. For our case,

$$\begin{aligned} & \min_{\mathbf{w}, b} \max_{\substack{\{\alpha_i\} \\ \alpha_i \geq 0 \\ i \in \{1, \dots, N\}}} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \right\} \geq \\ & \geq \max_{\substack{\{\alpha_i\} \\ \alpha_i \geq 0 \\ i \in \{1, \dots, N\}}} \min_{\mathbf{w}, b} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \right\} \end{aligned} \quad (18)$$

This is the essence of converting the primal problem to the dual one. And, our case is even better: strong duality holds, and the two terms are equal (duality gap is zero). Denote the inner term by $L(\mathbf{w}, b, \alpha)$ (corresponds to what's commonly known as the Lagrangian):

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \quad (19)$$

The Dual Formulation (3)

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \alpha_i [y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \quad (20)$$

We want to find $\operatorname{argmax}_{\alpha \geq 0} \min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha)$. First, for fixed α , find $\min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha)$:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i \quad (21)$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^N \alpha_i y_i = 0 \quad (22)$$

Put this to Lagrangian:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \alpha_i [y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = \quad (23)$$

$$= \frac{1}{2} \|\mathbf{w}\|^2 - \left(\sum_{i=1}^N \alpha_i y_i \mathbf{x}_i \right) \cdot \mathbf{w} - \sum_{i=1}^N \alpha_i y_i b + \sum_{i=1}^N \alpha_i \quad (24)$$

$$= -\frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^N \alpha_i = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \quad (25)$$

The Dual Formulation, Result and Insights

The dual optimization problem:

$$\alpha = \operatorname{argmax}_{\alpha} \left(\min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha) \right) = \operatorname{argmax}_{\alpha} \left\{ \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \right\} \quad (26)$$

subject to: $\sum_i \alpha_i y_i = 0; \quad \alpha_i \geq 0, \quad \forall i \in \{1, 2, \dots, N\}$ (27)

- ◆ Number of optimization variables α_i 's is N (the number of training data). But at the solution, all α_i 's but those of support vectors are zero.
- ◆ Once the solution is obtained, the primal variables can be computed as

$$\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i \quad \text{only support vectors } (\alpha_i > 0) \text{ contribute} \quad (28)$$

$$y^S [\mathbf{w} \cdot \mathbf{x}^S + b] = 1 \text{ for any support vector } (\mathbf{x}^S, y^S) \Rightarrow b = y^S - \mathbf{w} \cdot \mathbf{x}^S \quad (29)$$

- ◆ The discriminant function $\mathbf{w} \cdot \mathbf{x} + b$ thus takes the form (\mathcal{P} are indices of all support vectors):

$$\mathbf{w} \cdot \mathbf{x} + b = \sum_{i \in \mathcal{P}} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + \underbrace{y^S - \sum_{i \in \mathcal{P}} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}^S)}_{\text{constant, independent of } \mathbf{x}} \quad (30)$$

- ◆ Both the dual classification problem and the discriminant function involve data points **only** in the form of **dot products**.

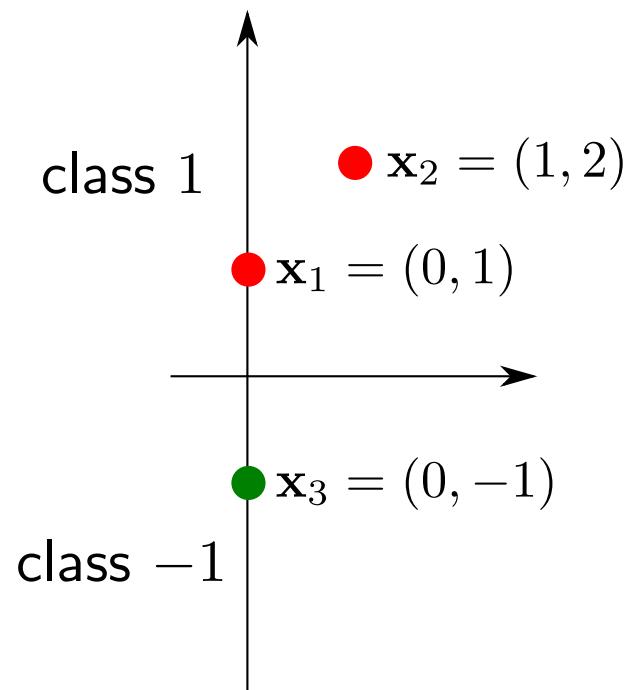
The Dual Problem, Example (1)

Consider the 3 points as below

Objective: maximize

$$\alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}^T \begin{bmatrix} y_1 y_1 \mathbf{x}_1 \cdot \mathbf{x}_1 & y_1 y_2 \mathbf{x}_1 \cdot \mathbf{x}_2 & y_1 y_3 \mathbf{x}_1 \cdot \mathbf{x}_3 \\ y_2 y_1 \mathbf{x}_2 \cdot \mathbf{x}_1 & y_2 y_2 \mathbf{x}_2 \cdot \mathbf{x}_2 & y_2 y_3 \mathbf{x}_2 \cdot \mathbf{x}_3 \\ y_3 y_1 \mathbf{x}_3 \cdot \mathbf{x}_1 & y_3 y_2 \mathbf{x}_3 \cdot \mathbf{x}_2 & y_3 y_3 \mathbf{x}_3 \cdot \mathbf{x}_3 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

subject to: $\alpha_1, \alpha_2, \alpha_3 \geq 0$; $\alpha_1 + \alpha_2 - \alpha_3 = 0$



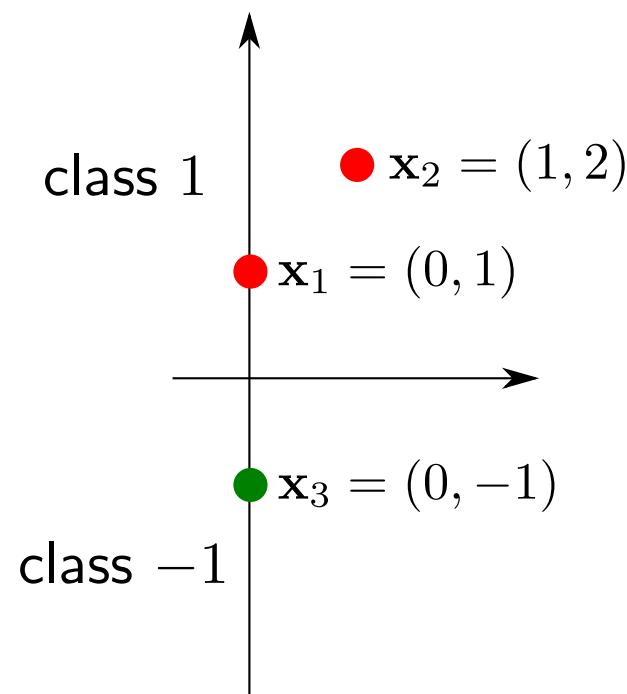
The Dual Problem, Example (2)

Consider the 3 points as below

Objective: maximize

$$\alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}^T \begin{bmatrix} 1 & 2 & 1 \\ 2 & 5 & 2 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

subject to: $\alpha_1, \alpha_2, \alpha_3 \geq 0$; $\alpha_1 + \alpha_2 - \alpha_3 = 0$

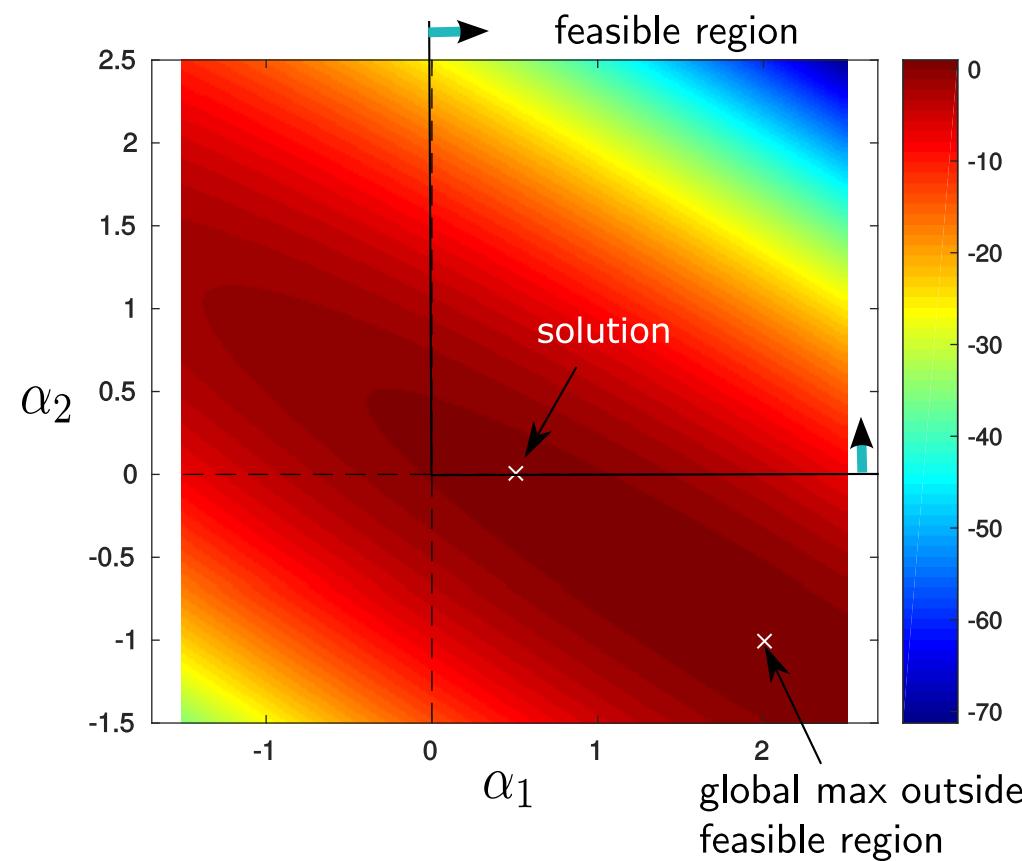
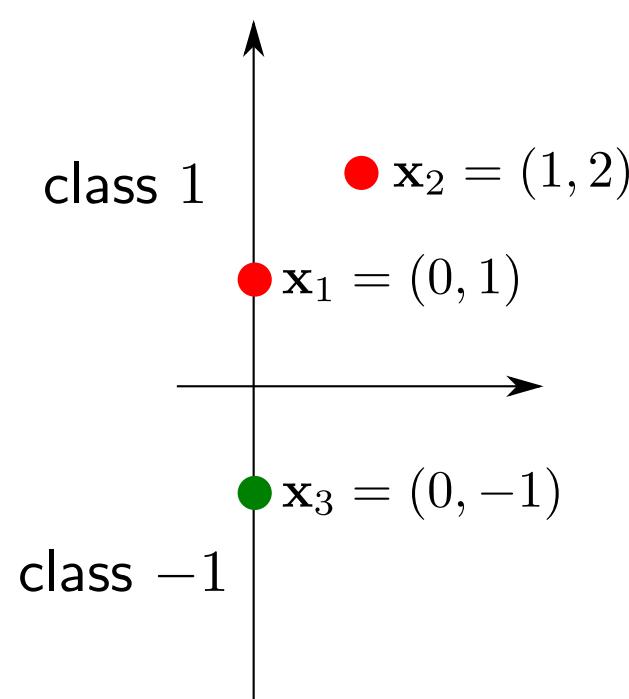


The Dual Problem, Example (3)

Substitute $\alpha_3 = \alpha_1 + \alpha_2$ and search for solution as a problem in α_1, α_2 . After some straightforward computation, the original problem turns to:

$$\text{maximize } 2(\alpha_1 + \alpha_2) - \frac{1}{2} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}^T \begin{bmatrix} 4 & 6 \\ 6 & 10 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}$$

subject to: $\alpha_1, \alpha_2 \geq 0$. **Solution:** $(\alpha_1, \alpha_2) = (\frac{1}{2}, 0)$, $\alpha_3 = \frac{1}{2} + 0 = \frac{1}{2}$.



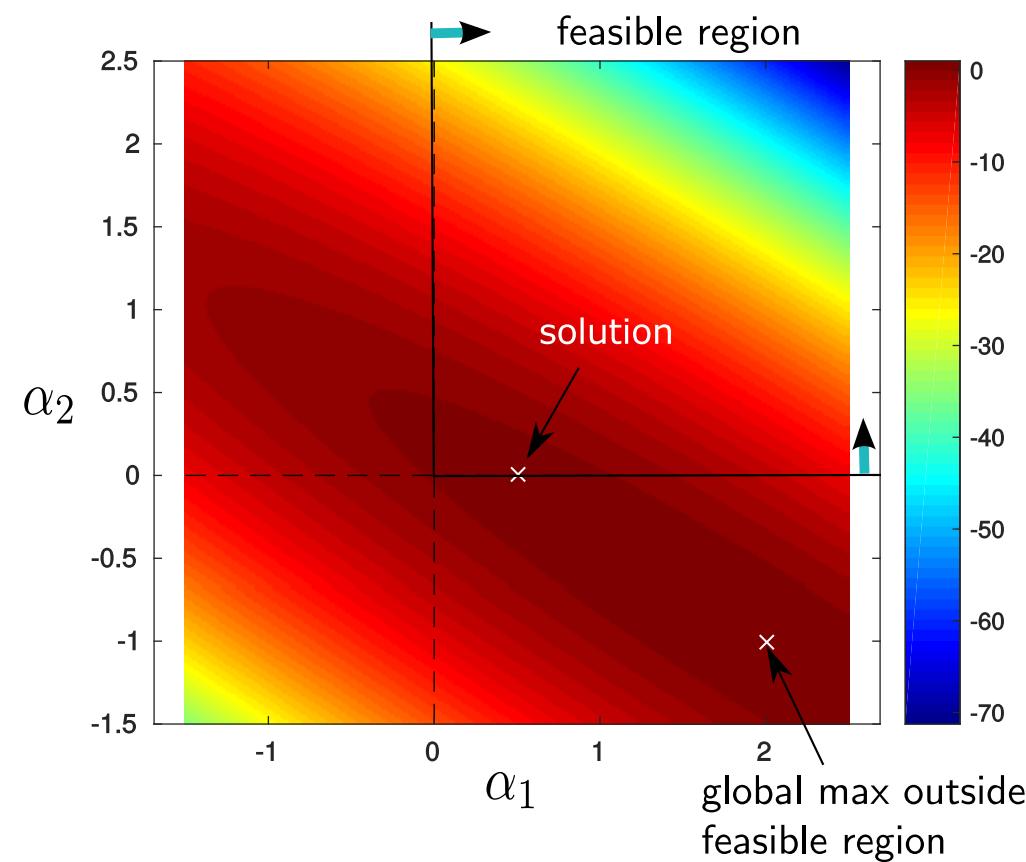
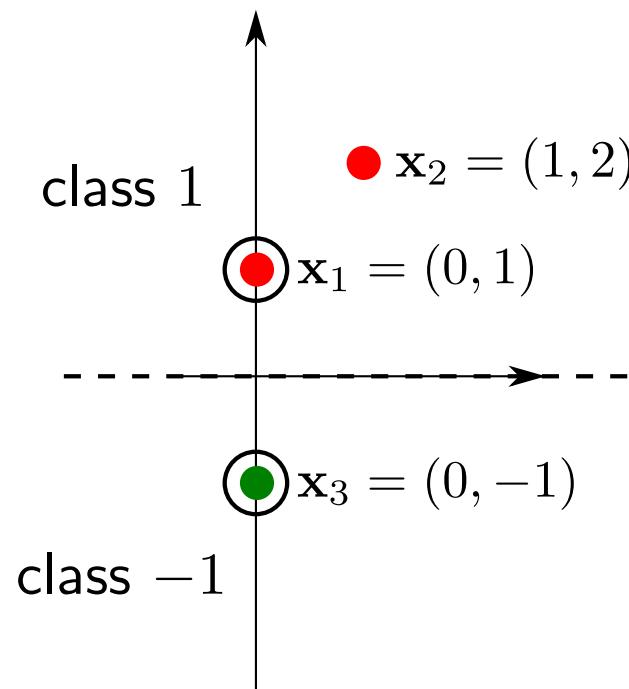
The Dual Problem, Example, Result

Result: $(\alpha_1, \alpha_2, \alpha_3) = (\frac{1}{2}, 0, \frac{1}{2})$. The support vectors are \mathbf{x}_1 and \mathbf{x}_3 because their $\alpha_i > 0$.

Vector $\mathbf{w} = \sum_{i=\{1,3\}} \alpha_i y_i \mathbf{x}_i = \frac{1}{2}(0, 1) - \frac{1}{2}(0, -1) = (0, 1)$.

Offset $b = y^S - \mathbf{w}\mathbf{x}^S = 1 - \mathbf{w}\mathbf{x}_1 = -1 - \mathbf{w}\mathbf{x}_3 = 0$.

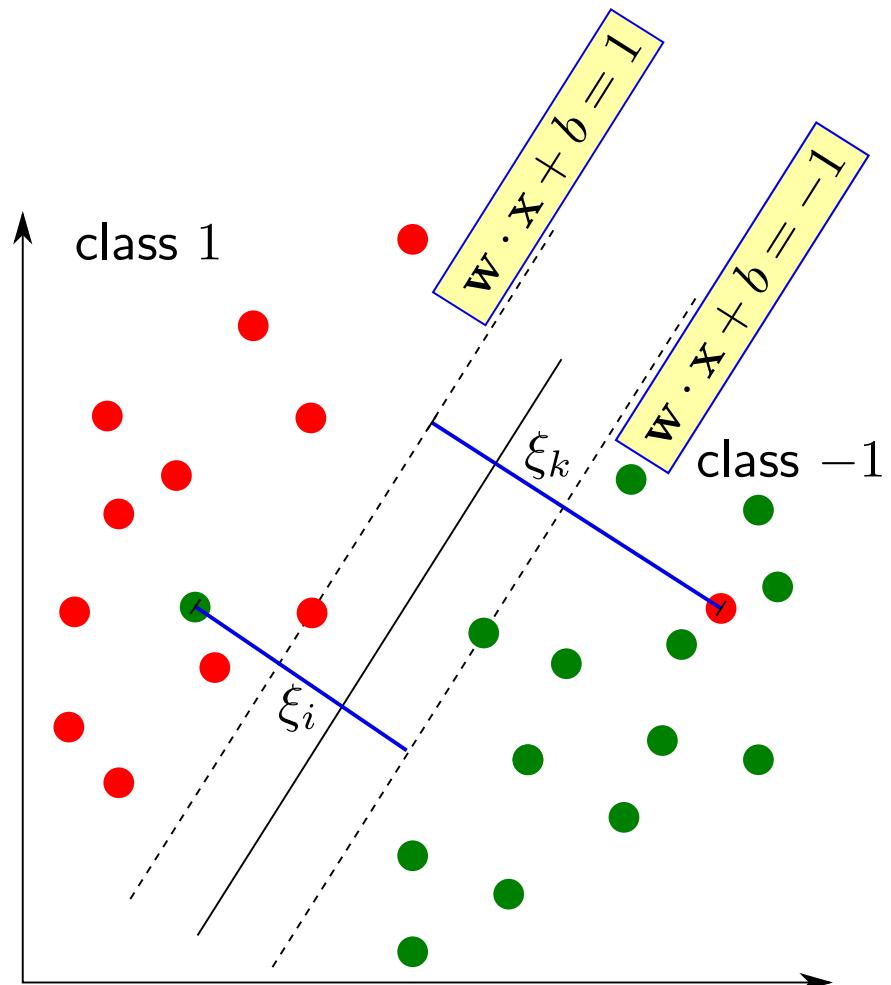
Decision boundary $(0, 1)^T \cdot \mathbf{x} = 0$.



Soft Margin SVM

If the data are not linearly separable, *slack variables* ξ_i need to be introduced.

- ◆ Position and size of margin is implied by \mathbf{w} and b , as before.
- ◆ If a point (\mathbf{x}, y) fulfills the condition $y(\mathbf{w} \cdot \mathbf{x} + b) \geq 1$ then no penalty is paid.
- ◆ Otherwise, the condition is relaxed to $y(\mathbf{w} \cdot \mathbf{x} + b) \geq 1 - \xi$ and penalty $C \cdot \xi$ is paid



$$(\mathbf{w}^*, b^*) = \underset{(\mathbf{w}, b)}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \quad (31)$$

subject to:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad (32)$$

$$\xi_i \geq 0, \quad (33)$$

$$\forall i = 1, \dots, N$$

Soft Margin SVM

The primal problem

$$(\mathbf{w}^*, b^*) = \underset{(\mathbf{w}, b)}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i$$

$$\text{subject to: } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i = 1, \dots, N \quad (34)$$

$$\xi_i \geq 0, \quad \forall i = 1, \dots, N \quad (35)$$

The dual problem:

$$\alpha = \underset{\alpha}{\operatorname{argmax}} \left\{ \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \right\} \quad (36)$$

$$\text{subject to: } \sum_i \alpha_i y_i = 0 \quad (37)$$

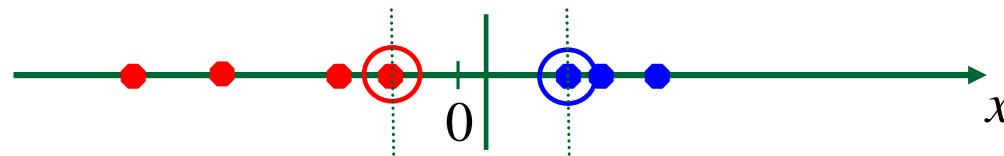
$$0 \leq \alpha_i \leq C, \quad \forall i \in \{1, 2, \dots, N\} \quad (38)$$

Linear SVMs: Overview

- The classifier is a *separating hyperplane*.
- Most “important” training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points \mathbf{x}_i are support vectors with non-zero Lagrangian multipliers b_i .
- Both in the dual formulation of the problem and in the solution training points appear only inside inner-products.

Who really need linear classifiers

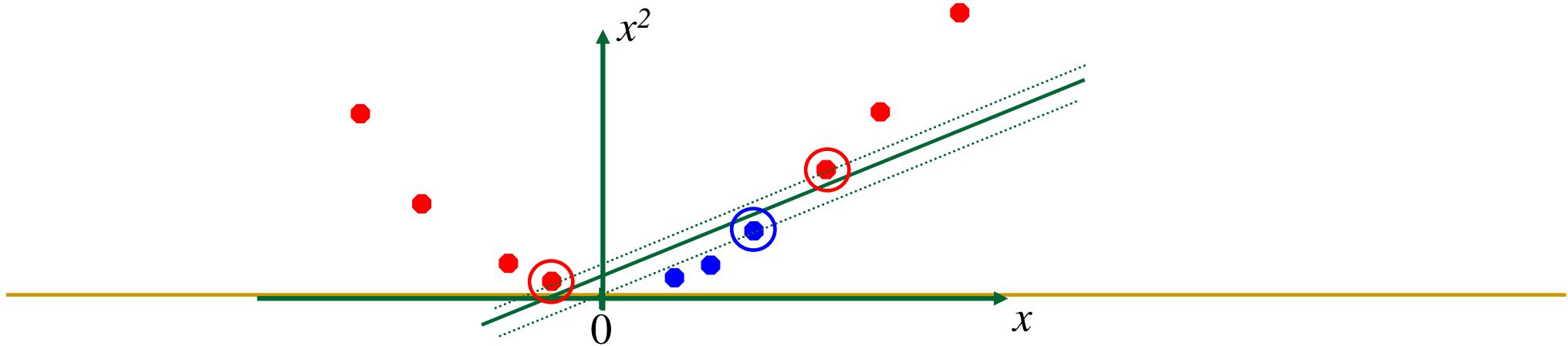
- Datasets that are linearly separable with some noise, linear SVM work well:



- But if the dataset is non-linearly separable?

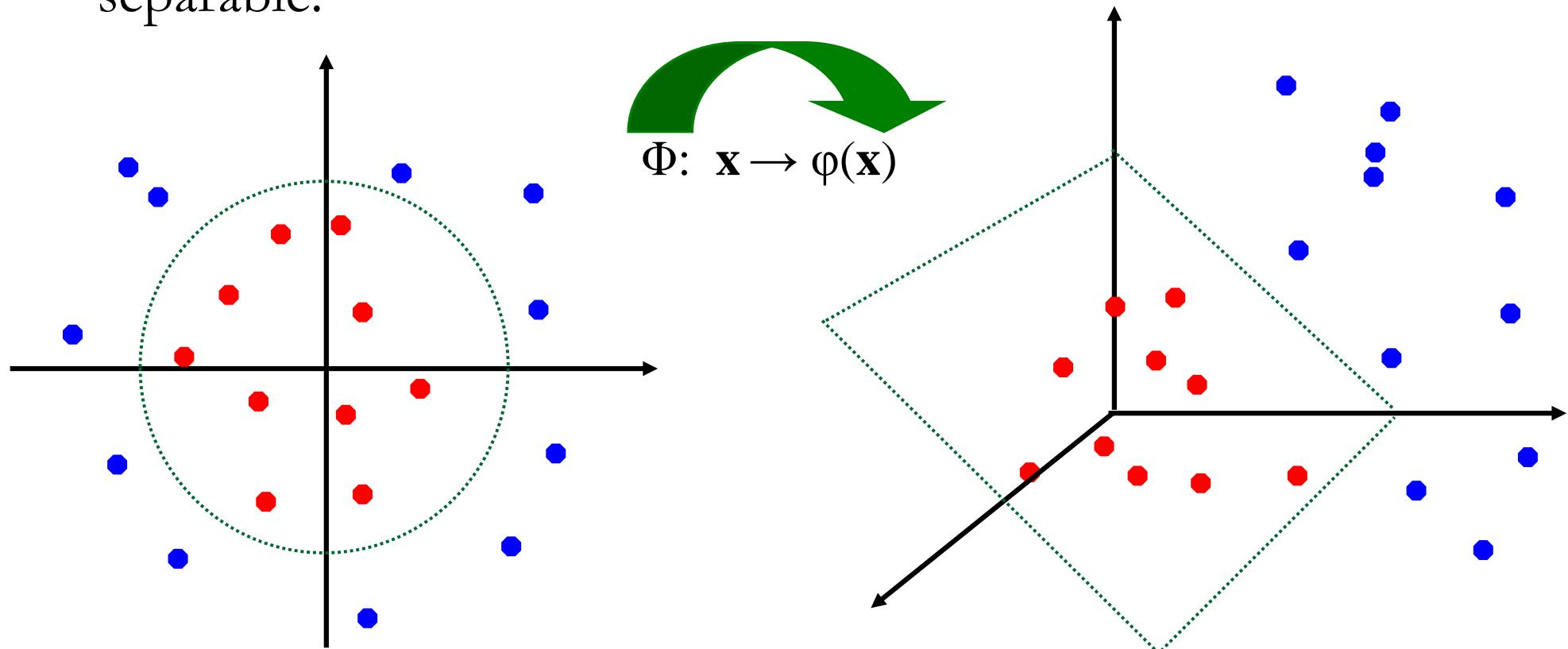


- How about... mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original space can always be mapped to some higher-dimensional feature space where the training set becomes separable:



The “Kernel Trick”

- The SVM only relies on the inner-product between vectors $\mathbf{x}_i \cdot \mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation $\Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner-product becomes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$$

- $K(\mathbf{x}_i, \mathbf{x}_j)$ is called the kernel function.
- For SVM, we only need specify the kernel $K(\mathbf{x}_i, \mathbf{x}_j)$, without need to know the corresponding non-linear mapping, $\varphi(\mathbf{x})$.

Non-linear SVMs

- The dual problem:

$$\begin{aligned}
 & \text{Maximizing: } L(\mathbf{h}) = \sum_{i=1}^N h_i - \frac{1}{2} \mathbf{h} \cdot \mathbf{D} \cdot \mathbf{h} \\
 & \text{Subject to: } \mathbf{h} \cdot \mathbf{y} = 0 \\
 & \quad 0 \leq \mathbf{h} \leq \mathbf{C} \\
 & \text{where } D_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)
 \end{aligned}$$

- Optimization techniques for finding h_i 's remain the same!
- The solution is:

$$\begin{aligned}
 \mathbf{w}^* &= \sum_{i \in SV} h_i y_i \varphi(\mathbf{x}_i) \\
 f(\mathbf{x}) &= \mathbf{w}^* \cdot \varphi(\mathbf{x}) + b^* \\
 &= \sum_{i \in SV} h_i y_i K(\mathbf{x}_i, \mathbf{x}) + b^*
 \end{aligned}$$

Examples of Kernel Trick (1)

- For the example in the previous figure:
 - The non-linear mapping

$$x \rightarrow \varphi(x) = (x, x^2)$$

- The kernel

$$\begin{aligned}\varphi(x_i) &= (x_i, x_i^2), \quad \varphi(x_j) = (x_j, x_j^2) \\ K(x_i, x_j) &= \varphi(x_i) \cdot \varphi(x_j) \\ &= x_i x_j (1 + x_i x_j)\end{aligned}$$

- Where is the benefit?

Examples of Kernel Trick (2)

- Polynomial kernel of degree 2 in 2 variables
 - The non-linear mapping:

$$\mathbf{x} = (x_1, x_2)$$

$$\varphi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

- The kernel

$$\varphi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\varphi(\mathbf{y}) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

$$K(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x}) \cdot \varphi(\mathbf{y})$$

$$= (1 + \mathbf{x} \cdot \mathbf{y})^2$$

Examples of kernel trick (3)

- Gaussian kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$$

- The mapping is of infinite dimension:

$$\varphi(\mathbf{x}) = (\dots, \varphi_\omega(\mathbf{x}), \dots), \quad \text{for } \omega \in R^d$$

$$\varphi_\omega(\mathbf{x}) = A e^{-B\omega^2} e^{-i\mathbf{w}\mathbf{x}}$$

$$K(\mathbf{x}, \mathbf{y}) = \int \varphi_\omega(\mathbf{x}) \varphi^*_\omega(\mathbf{y}) d\omega$$

- The moral: very high-dimensional and complicated non-linear mapping can be achieved by using a simple kernel!

What Functions are Kernels?

- For some functions $K(\mathbf{x}_i, \mathbf{x}_j)$ checking that $K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$ can be cumbersome.
- Mercer's theorem:

Every semi-positive definite symmetric function is a kernel

Examples of Kernel Functions

- Linear kernel:
$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$$
- Polynomial kernel of power p :
$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^p$$
- Gaussian kernel:
$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$$
 - In the form, equivalent to RBFNN, but has the advantage of that the center of basis functions, i.e., support vectors, are optimized in a supervised.
- Two-layer perceptron:
$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\alpha \mathbf{x}_i \cdot \mathbf{x}_j + \beta)$$

Lifting Dimension by Polynomial Mapping of Degree d

Let $d \in \mathbb{N}$ and $\mathbf{x} = [x_1, x_2, \dots, x_D]^\top \in \mathbb{R}^D$.

Let $\phi_d(\mathbf{x})$ denote the mapping which lifts \mathbf{x} to the space containing all monomials of degree d' , $1 \leq d' \leq d$ in the components of \mathbf{x} :

For example, when $\mathbf{x} = [x_1, x_2]^\top \in \mathbb{R}^2$,

$$\phi_1(\mathbf{x}) = [x_1, x_2]^\top, \quad (39)$$

$$\phi_2(\mathbf{x}) = [x_1, x_2, x_1^2, x_1 x_2, x_2^2]^\top, \quad (40)$$

$$\phi_3(\mathbf{x}) = [x_1, x_2, x_1^2, x_1 x_2, x_2^2, x_1^3, x_1^2 x_2, x_1 x_2^2, x_2^3]^\top. \quad (41)$$

The number of monomials of degree d' of $\mathbf{x} \in \mathbb{R}^D$ is $\binom{d'+D-1}{d'}$. The dimensionality L of the output space of $\phi_d(\mathbf{x})$ is thus

$$L = \sum_{d'=1}^d \binom{d'+D-1}{d'}. \quad (42)$$

Lifting Dimension by Polynomial Mapping of Degree d

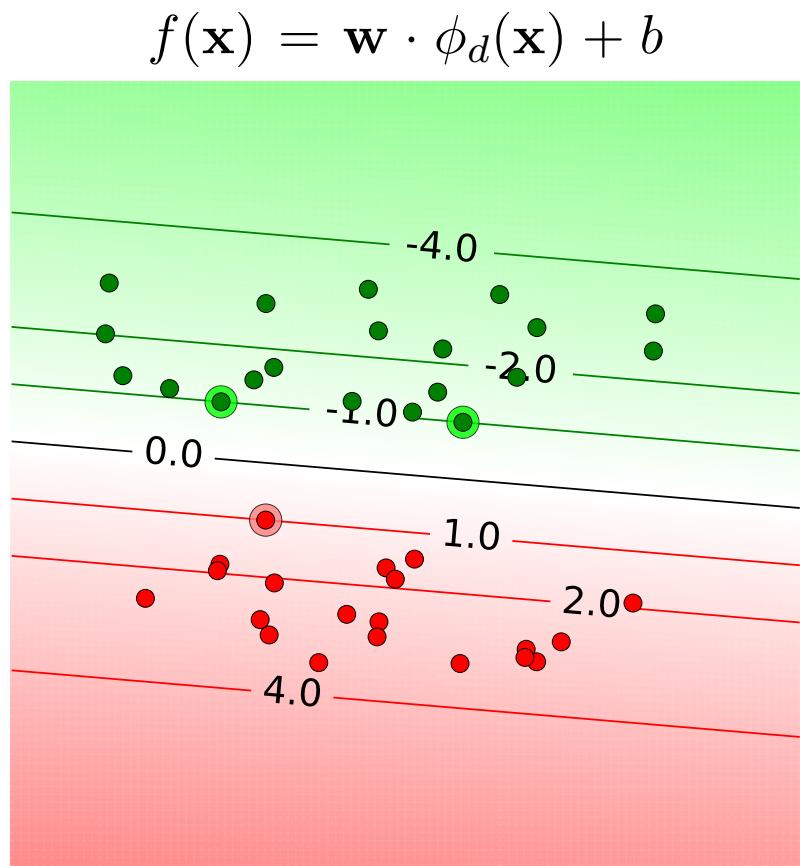
Feature space dimensionality D , lifting by $\phi_d(\mathbf{x})$

dimensionality of feature space after lifting (L)

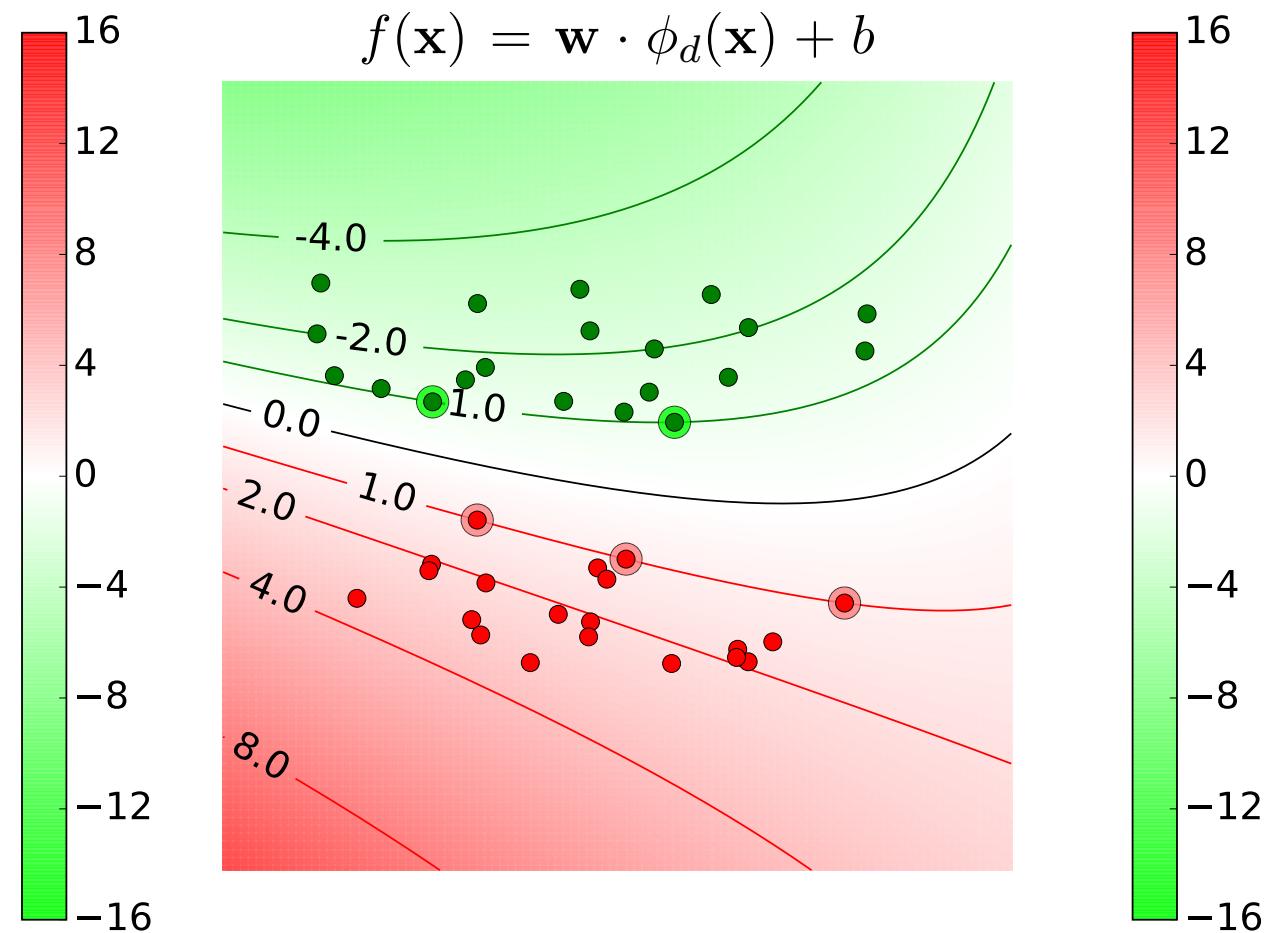
$D \backslash d$	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	2	5	9	14	20	27	35	44
3	3	9	19	34	55	83	119	164
4	4	14	34	69	125	209	329	494
5	5	20	55	125	251	461	791	1286
6	6	27	83	209	461	923	1715	3002
7	7	35	119	329	791	1715	3431	6434
8	8	44	164	494	1286	3002	6434	12869

Lifting by Polynomial Mapping of Degree d , Example

$d = 1, \dim(\phi_d(\mathbf{x})) = 2$
 support vectors : 3

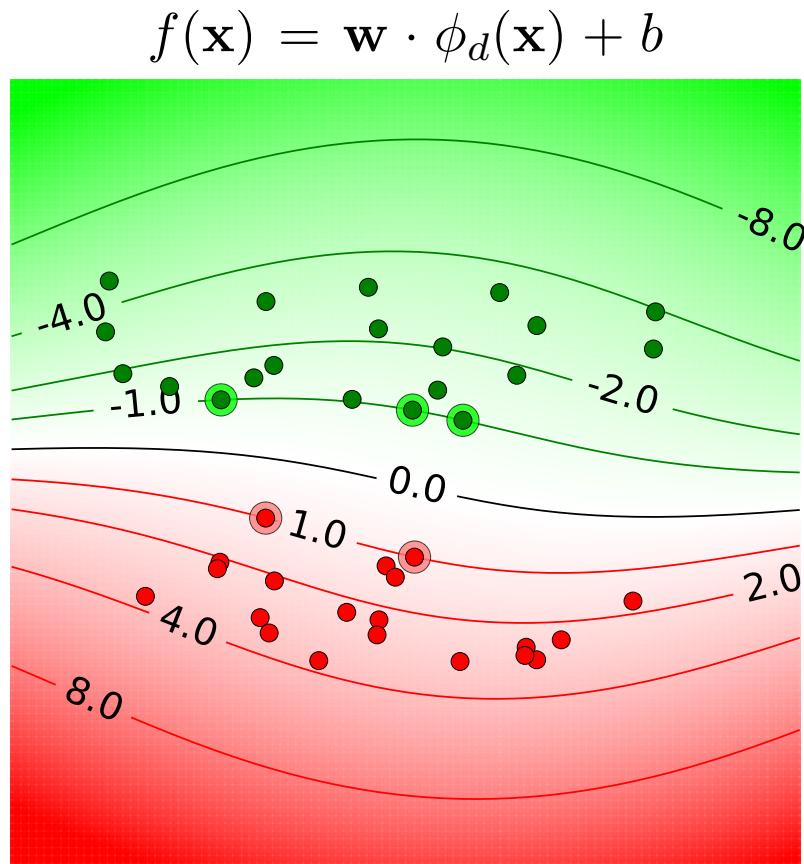


$d = 2, \dim(\phi_d(\mathbf{x})) = 5$
 support vectors : 5

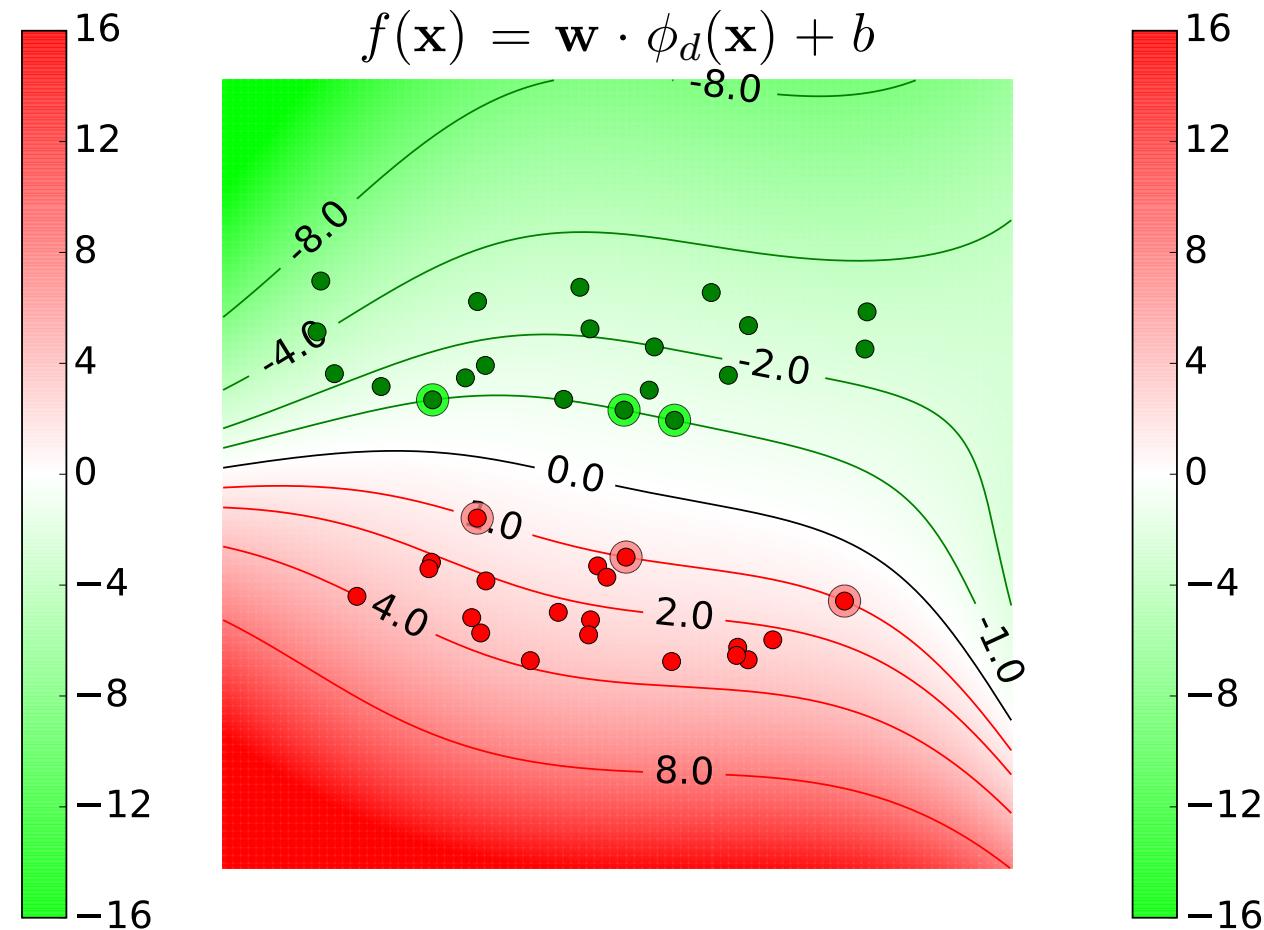


Lifting by Polynomial Mapping of Degree d , Example

$d = 3, \dim(\phi_d(\mathbf{x})) = 9$
 support vectors : 5



$d = 4, \dim(\phi_d(\mathbf{x})) = 14$
 support vectors : 6



SVM Overviews

- Main features:
 - By using the kernel trick, data is mapped into a high-dimensional feature space, without introducing much computational effort;
 - Maximizing the margin achieves better generation performance;
 - Soft-margin accommodates noisy data;
 - Not too many parameters need to be tuned.
- Demos(<http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml>)

SVM so far

- SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.
- SVMs are currently among the best performers for many benchmark datasets.
- SVM techniques have been extended to a number of tasks such as regression [Vapnik *et al.* '97].
- Most popular optimization algorithms for SVMs are SMO [Platt '99] and SVM^{light} [Joachims' 99], both use *decomposition* to handle large size datasets.
- It seems the kernel trick is the most attracting site of SVMs. This idea has now been applied to many other learning models where the inner-product is concerned, and they are called ‘kernel’ methods.
- Tuning SVMs remains to be the main research focus: how to an optimal kernel? Kernel should match the smooth structure of data.

Appendix

Online demo: <http://cs.stanford.edu/people/karpathy/svmjs/demo/>

AdaBoost

Lecturer:
Jiří Matas

Authors:
Jan Šochman, Jiří Matas,
Jana Kostlivá, Ondřej Drbohlav

Centre for Machine Perception
Czech Technical University, Prague
<http://cmp.felk.cvut.cz>

Last update: 16.11.2017



AdaBoost

Presentation outline

- ◆ AdaBoost algorithm
 - Why is it of interest?
 - How it works?
 - Why it works?
- ◆ AdaBoost variants

History

- ◆ 1990 – Boost-by-majority algorithm (Freund)
- ◆ 1995 – AdaBoost (Freund & Schapire)
- ◆ 1997 – Generalized version of AdaBoost (Schapire & Singer)
- ◆ 2001 – AdaBoost in Face Detection (Viola & Jones)

What is Discrete AdaBoost?

AdaBoost is an algorithm for designing a *strong* classifier $H(\mathbf{x})$ from *weak* classifiers $h_t(\mathbf{x})$ ($t = 1, \dots, T$) selected from the weak classifier set \mathcal{B} . The strong classifier $H(\mathbf{x})$ is constructed as:

$$H(\mathbf{x}) = \text{sign}(f(\mathbf{x})),$$

where

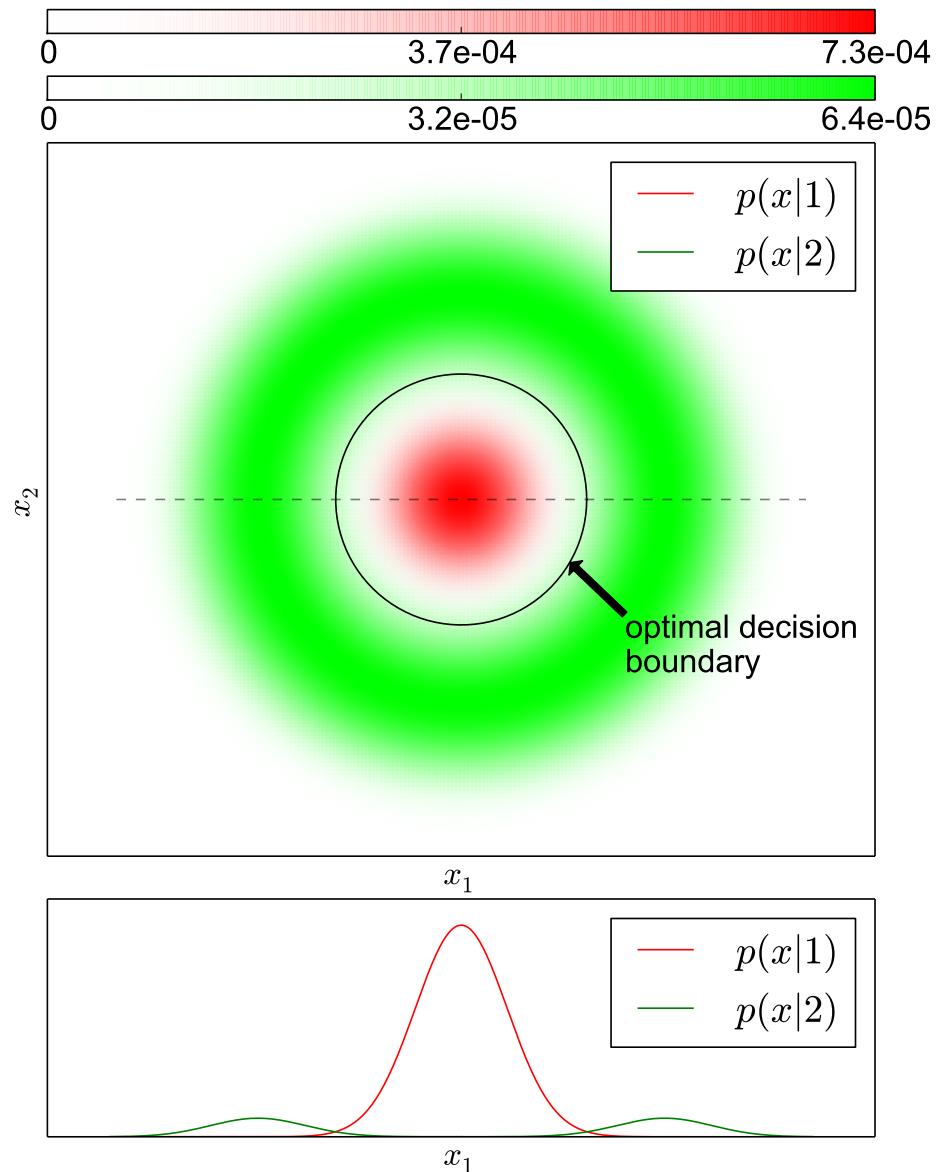
$$f(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$$

is a linear combination of weak classifiers $h_t(\mathbf{x})$ with positive weights $\alpha_t > 0$. Every weak classifier h_t is a binary classifier which outputs -1 or 1 .

Adaboost deals both with the selection of $h_t(\mathbf{x}) \in \mathcal{B}$, and with choosing α_t , for gradually increasing t .

The set of weak classifiers $\mathcal{B} = \{h(\mathbf{x})\}$ can be finite or infinite.

Example 1 – Training Set and Weak Classifier Set



the profile of the distributions along the shown line

Training set:

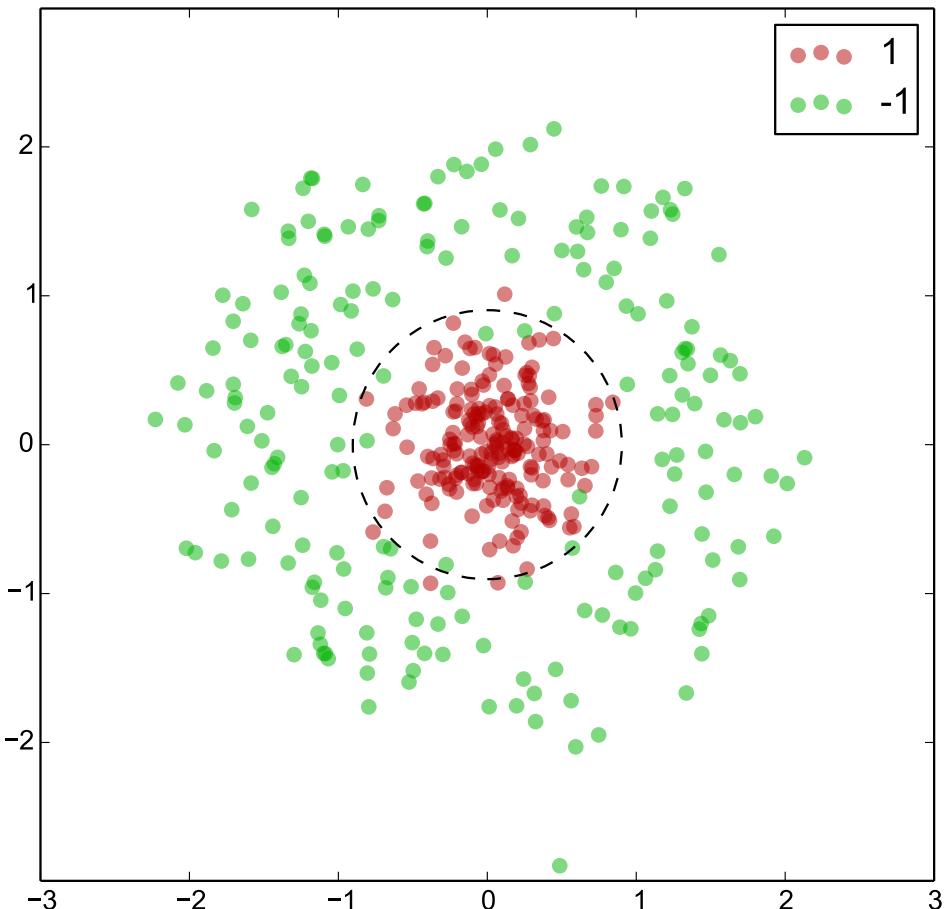
Samples generated from the two distributions shown, with

$$p(1) = p(2) = 0.5 \quad (1)$$

The Bayes error is 2.6%.

In the slides to follow, the classes are renamed from $(1, 2)$ to $(1, -1)$.

Example 1 – Training Set and Weak Classifier Set

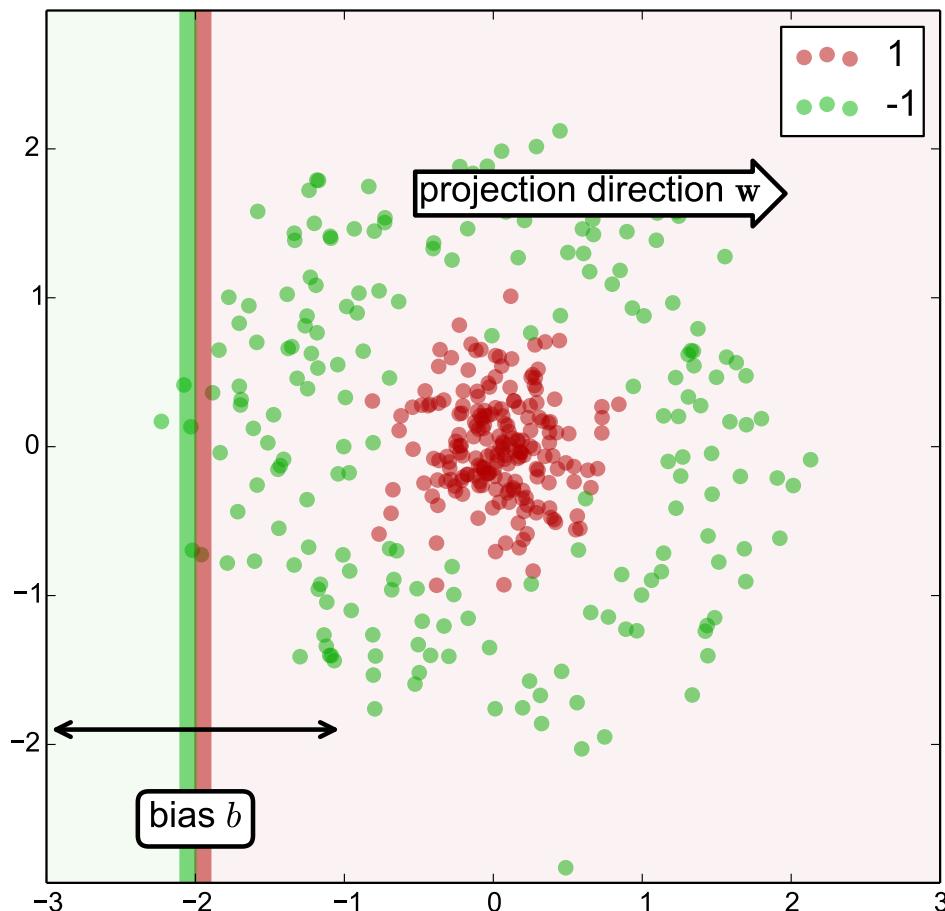


Training set:

$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_L, y_L)$, where $\mathbf{x}_i \in \mathbb{R}^2$ and $y_i \in \{-1, 1\}$, generated from the two distributions, $N = 200$ points from each.

The class distributions are not known to AdaBoost.

Example 1 – Training Set and Weak Classifier Set



Training set:

$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_L, y_L)$, where $\mathbf{x}_i \in \mathbb{R}^2$ and $y_i \in \{-1, 1\}$, generated from the two distributions, $N = 200$ points from each.

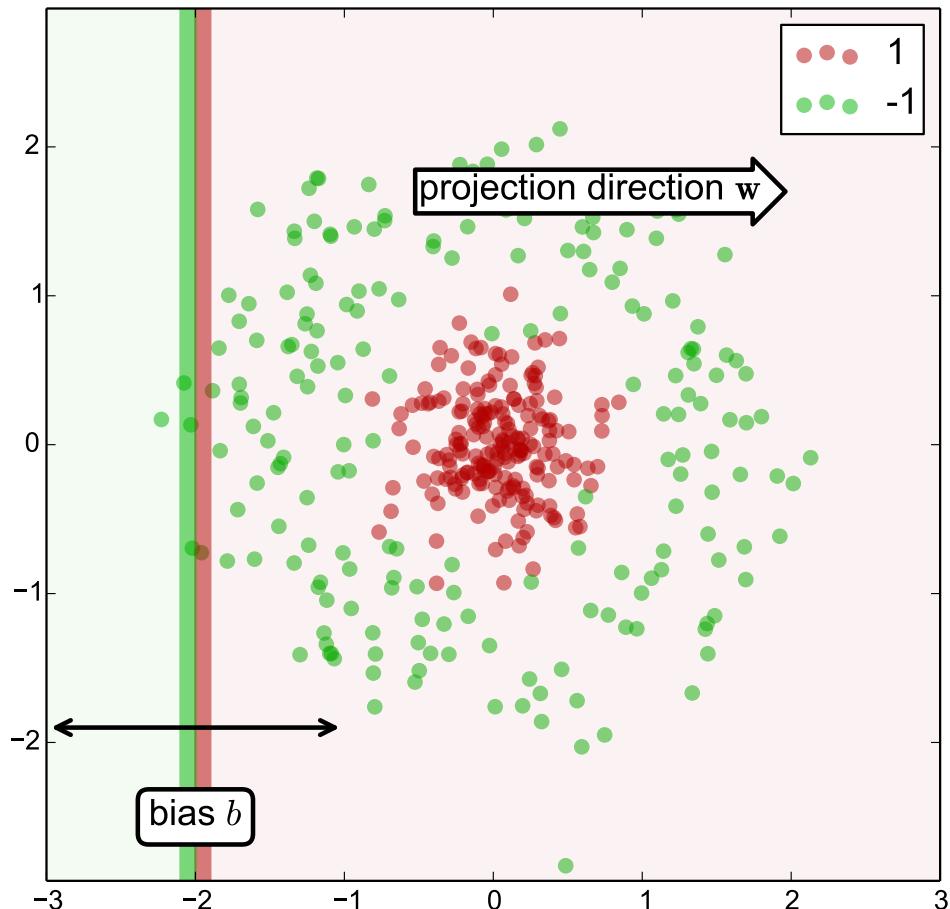
The class distributions are not known to AdaBoost.

Weak classifier: a linear classifier

$$h_{\mathbf{w}, b}(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b),$$

where \mathbf{w} is the projection direction vector and b is the bias.

Example 1 – Training Set and Weak Classifier Set



Training set:

$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_L, y_L)$, where $\mathbf{x}_i \in \mathbb{R}^2$ and $y_i \in \{-1, 1\}$, generated from the two distributions, $N = 200$ points from each.

The class distributions are not known to AdaBoost.

Weak classifier: a linear classifier

$$h_{\mathbf{w}, b}(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b),$$

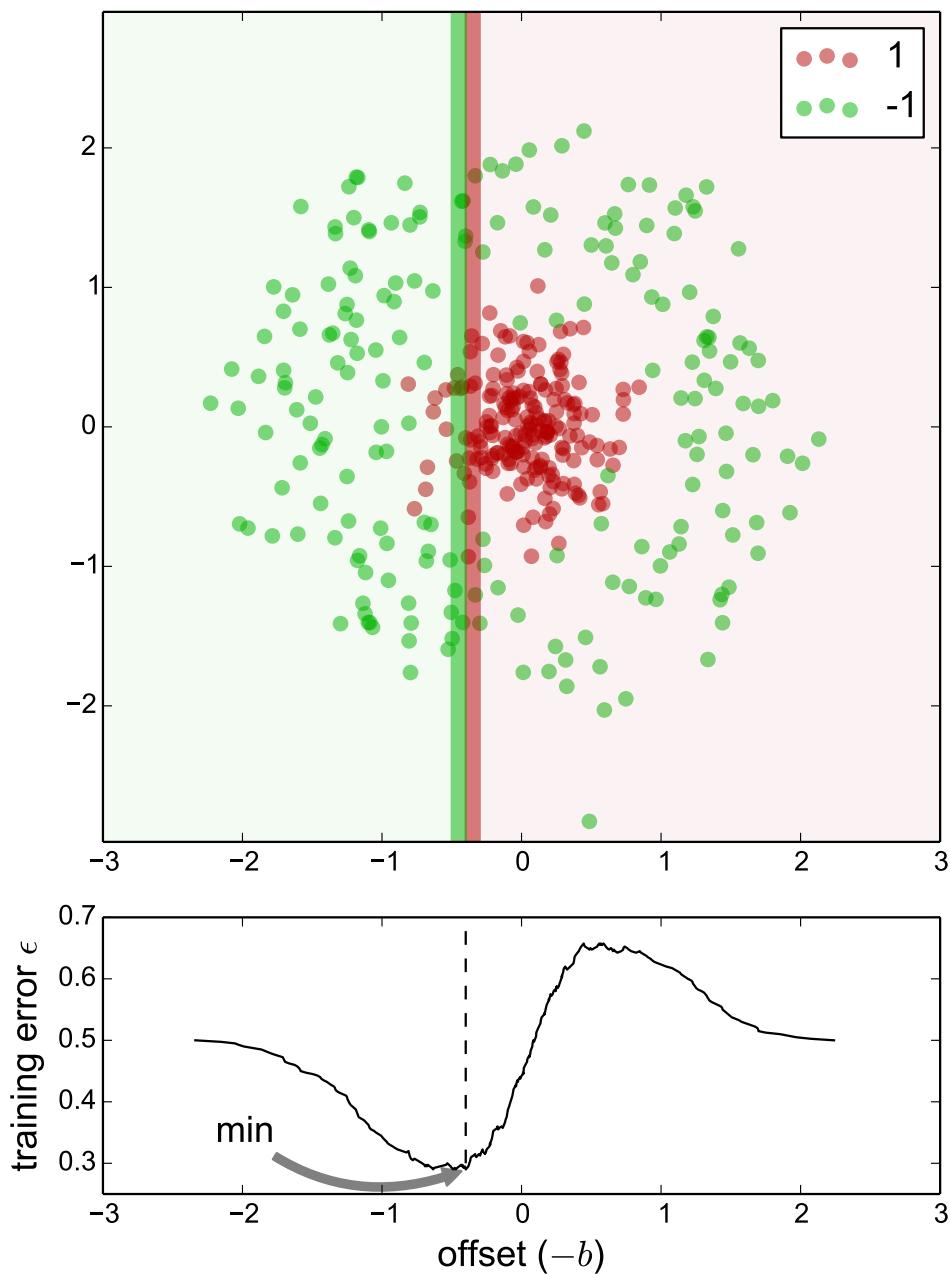
where \mathbf{w} is the projection direction vector and b is the bias.

Weak classifier set \mathcal{B} :

$$\{h_{\mathbf{w}, b} \mid \mathbf{w} \in \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_N\}, b \in \mathbb{R}\}$$

- ◆ N is the number of projection directions used

Example 1 – Training Set and Weak Classifier Set



Training set:

$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_L, y_L)$, where $\mathbf{x}_i \in \mathbb{R}^2$ and $y_i \in \{-1, 1\}$, generated from the two distributions, $N = 200$ points from each.

The class distributions are not known to AdaBoost.

Weak classifier: a linear classifier

$$h_{\mathbf{w}, b}(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b),$$

where \mathbf{w} is the projection direction vector and b is the bias.

Weak classifier set \mathcal{B} :

$$\{h_{\mathbf{w}, b} \mid \mathbf{w} \in \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_N\}, b \in \mathbb{R}\}$$

- ◆ N is the number of projection directions used
- ◆ for each projection direction \mathbf{w} , varying bias b results in different training errors ϵ .

AdaBoost Algorithm – Singer & Schapire (1997)

Input: $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_L, y_L)$, where $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \{-1, 1\}$

Initialize data weights $D_1(i) = 1/L$.

For $t = 1, \dots, T$:

- ◆ Find $h_t = \arg \min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_t = \sum_{i=1}^L D_t(i) \llbracket y_i \neq h(\mathbf{x}_i) \rrbracket$ (*WeakLearn*)
 $\llbracket \text{true} \rrbracket \stackrel{\text{def}}{=} 1, \quad \llbracket \text{false} \rrbracket \stackrel{\text{def}}{=} 0$
- ◆ If $\epsilon_t \geq \frac{1}{2}$ then stop
- ◆ Set $\alpha_t = \frac{1}{2} \log \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$ **Note:** $\alpha_t > 0$ because $\epsilon_t < \frac{1}{2}$
- ◆ Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}, \quad Z_t = \sum_{i=1}^L D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)},$$

where Z_t is a normalization factor chosen so that D_{t+1} is a distribution.

Output the final classifier:

$$H(\mathbf{x}) = \text{sign}(f(\mathbf{x})), \quad f(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$$

Note: Data Reweighting

$D_t(i)$: previous data weights

$D_{t+1}(i)$: new data weights

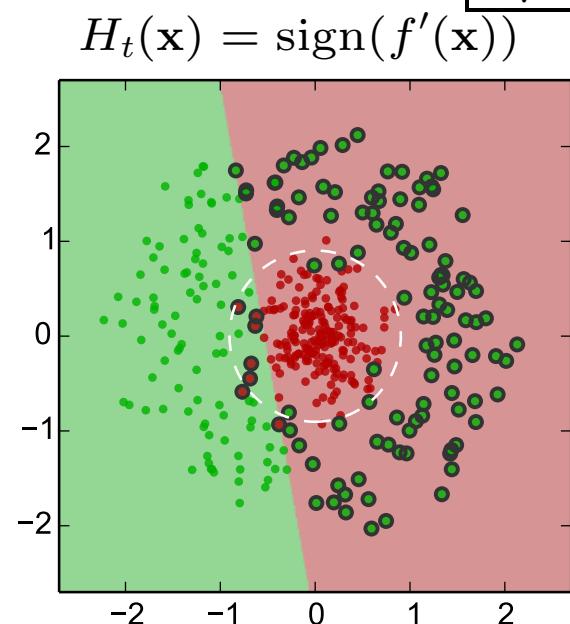
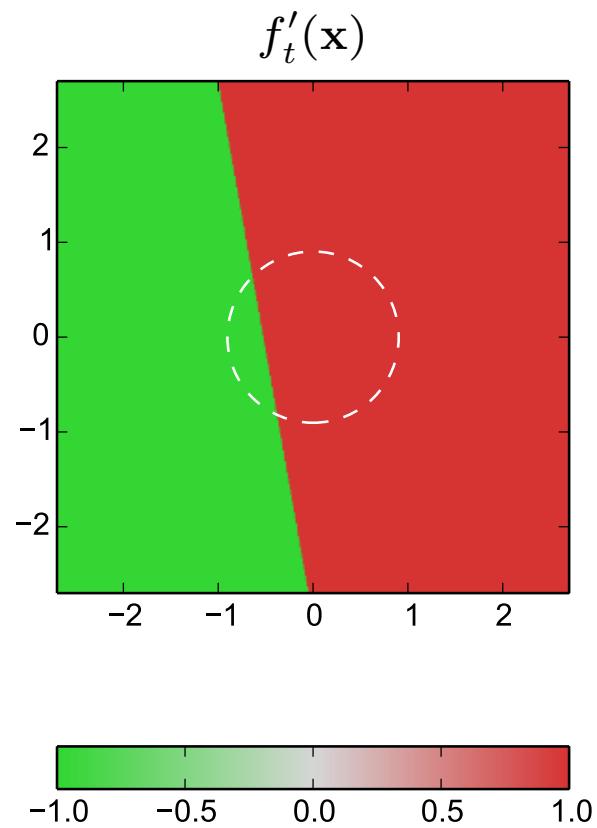
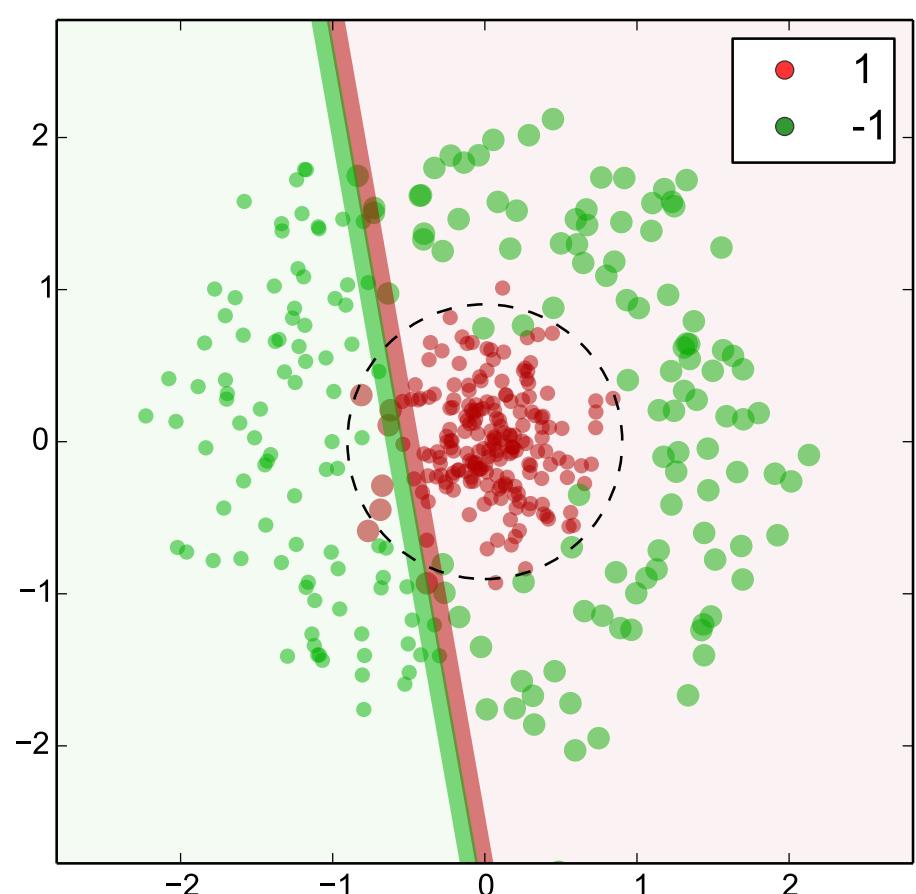
Weight update (recall that $\alpha_t > 0$):

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t} \quad (2)$$

$$e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = \begin{cases} e^{-\alpha_t} & < 1, \quad \text{if } y_i = h_t(\mathbf{x}_i) \\ e^{\alpha_t} & > 1, \quad \text{if } y_i \neq h_t(\mathbf{x}_i) \end{cases} \quad (3)$$

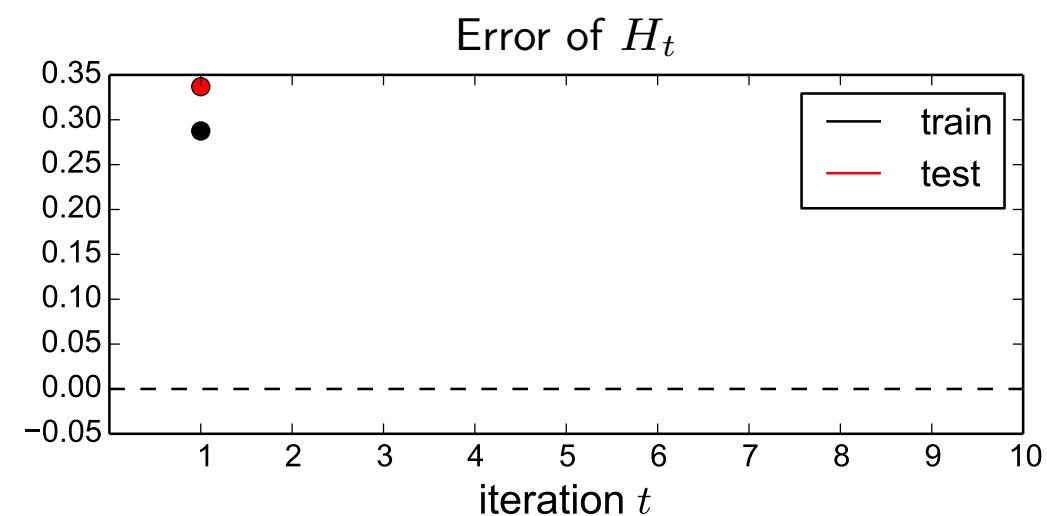
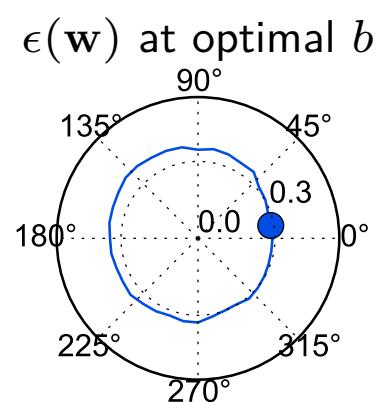
⇒ Increase (decrease) weight of wrongly (correctly) classified examples.

Example 1 – iteration 1



$$\epsilon_t = 28.8\%$$

$$\alpha_t = 0.454$$

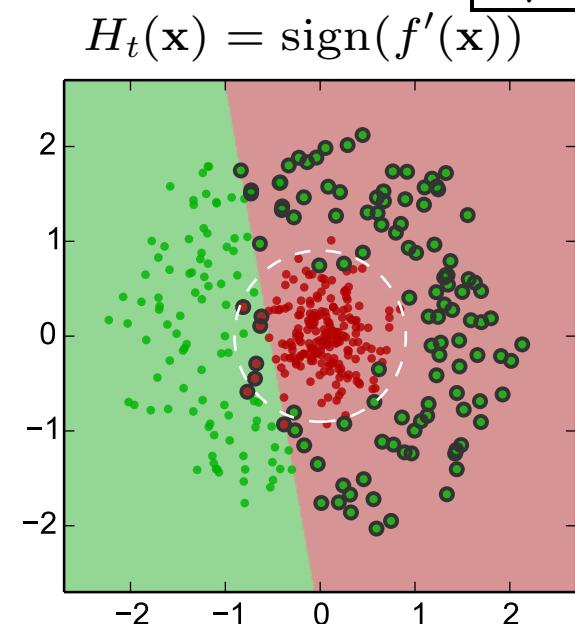
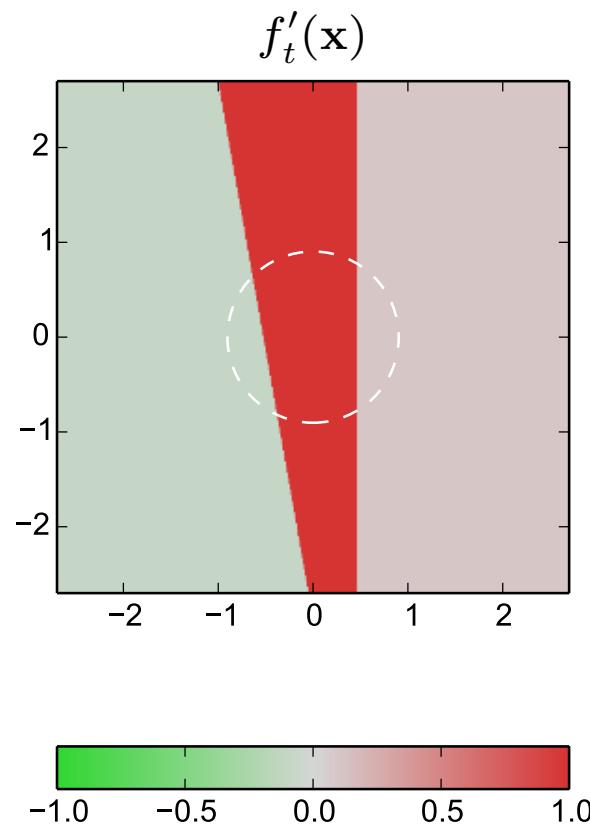
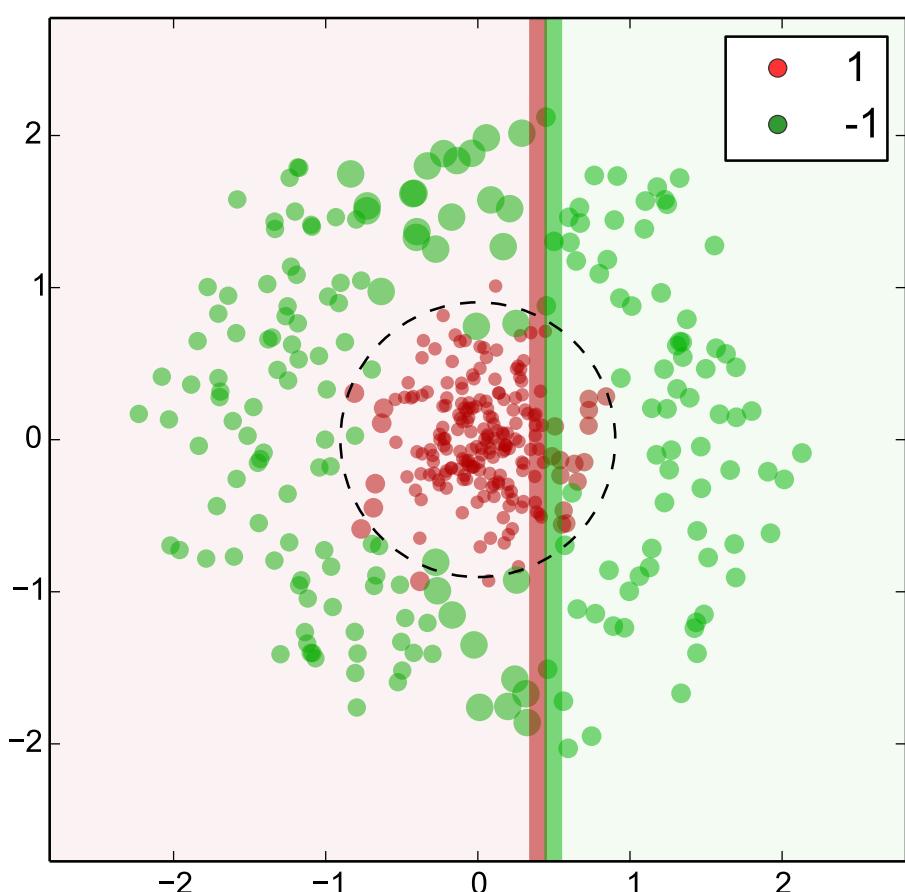


$$\epsilon_{H_t}^{\text{train}} = 28.7\%$$

$$\epsilon_{H_t}^{\text{test}} = 33.7\%$$

$$Z_t = 0.905$$

Example 1 – iteration 2



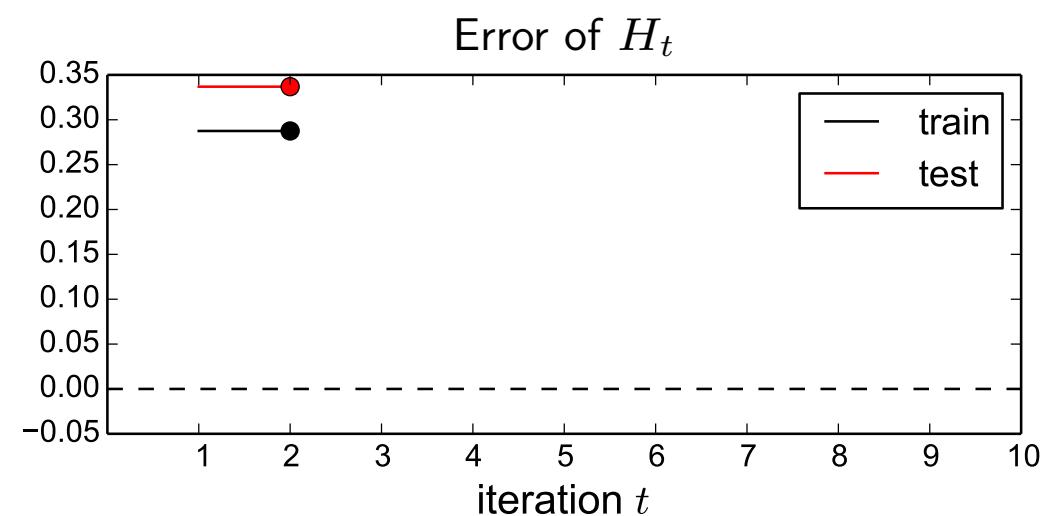
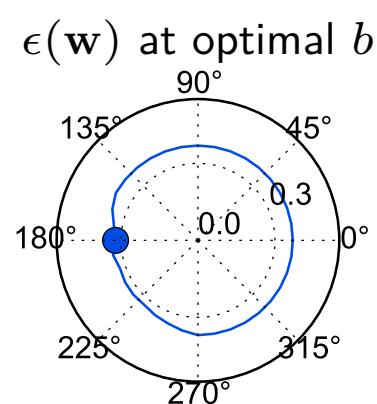
$$\epsilon_t = 32.1\%$$

$$\alpha_t = 0.375$$

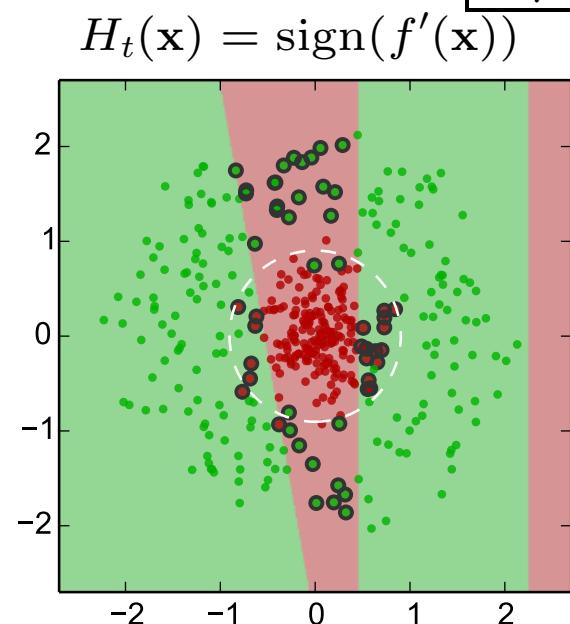
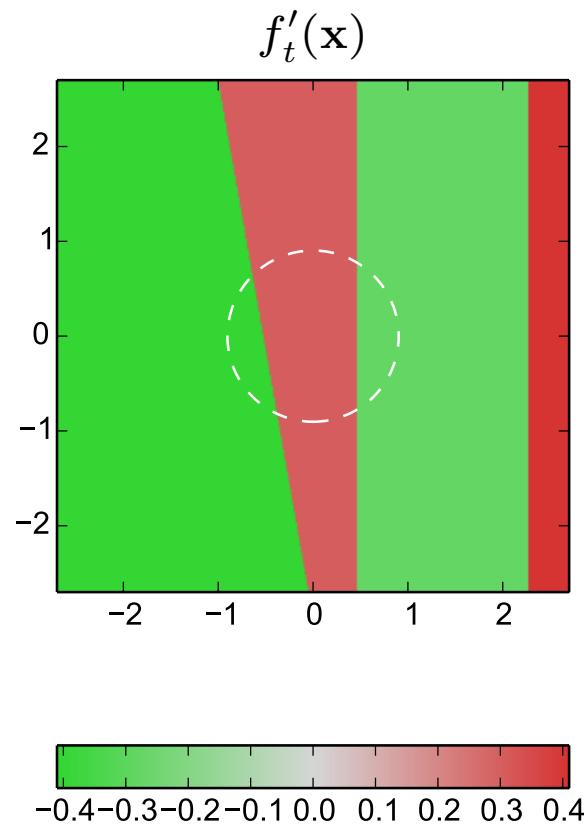
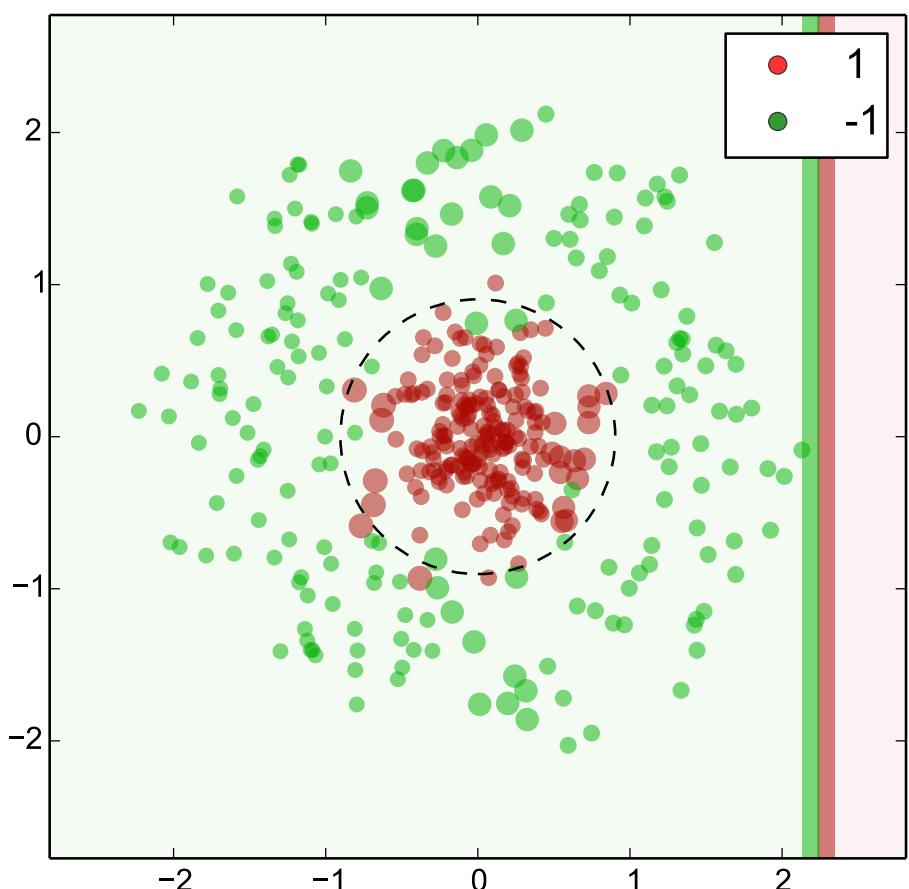
$$\epsilon_{H_t}^{\text{train}} = 28.7\%$$

$$\epsilon_{H_t}^{\text{test}} = 33.7\%$$

$$Z_t = 0.934$$

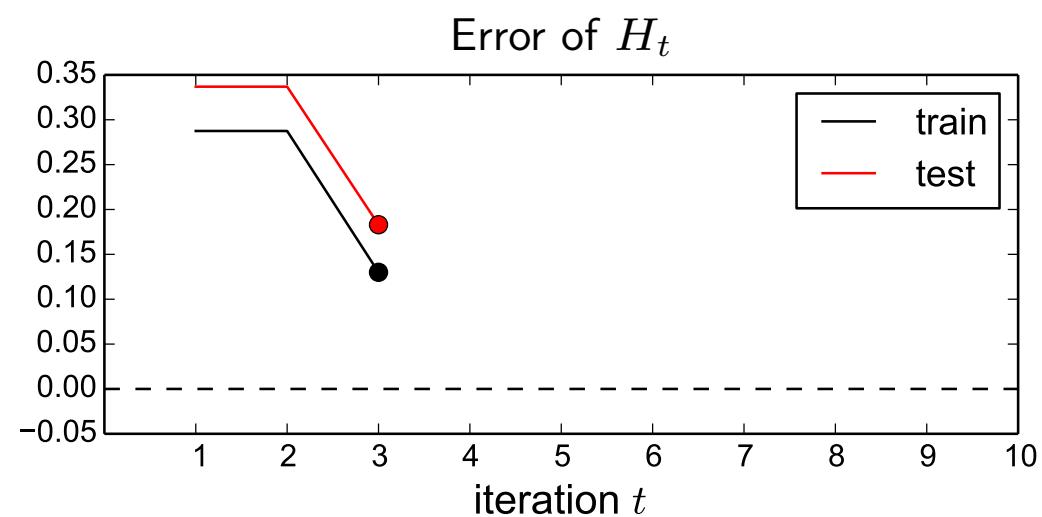
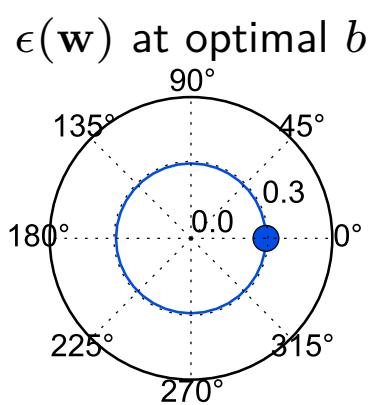


Example 1 – iteration 3



$$\epsilon_t = 29.2\%$$

$$\alpha_t = 0.443$$

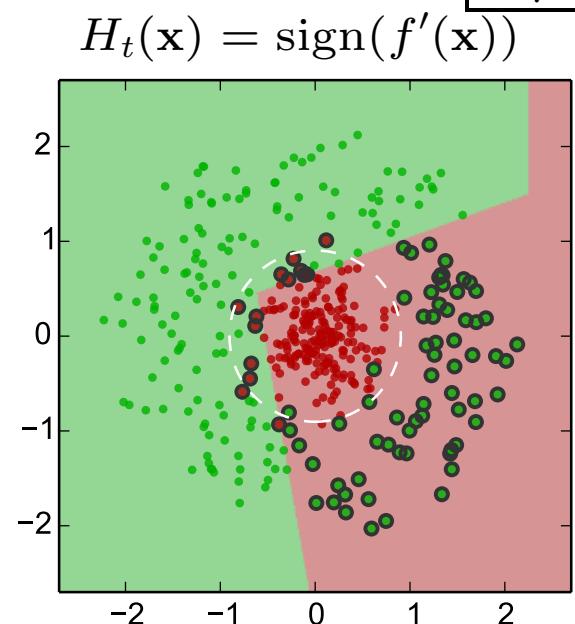
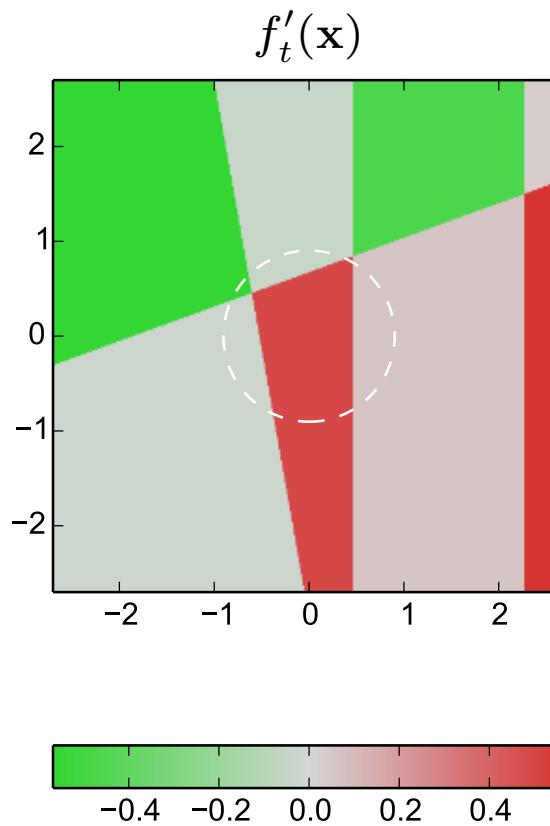
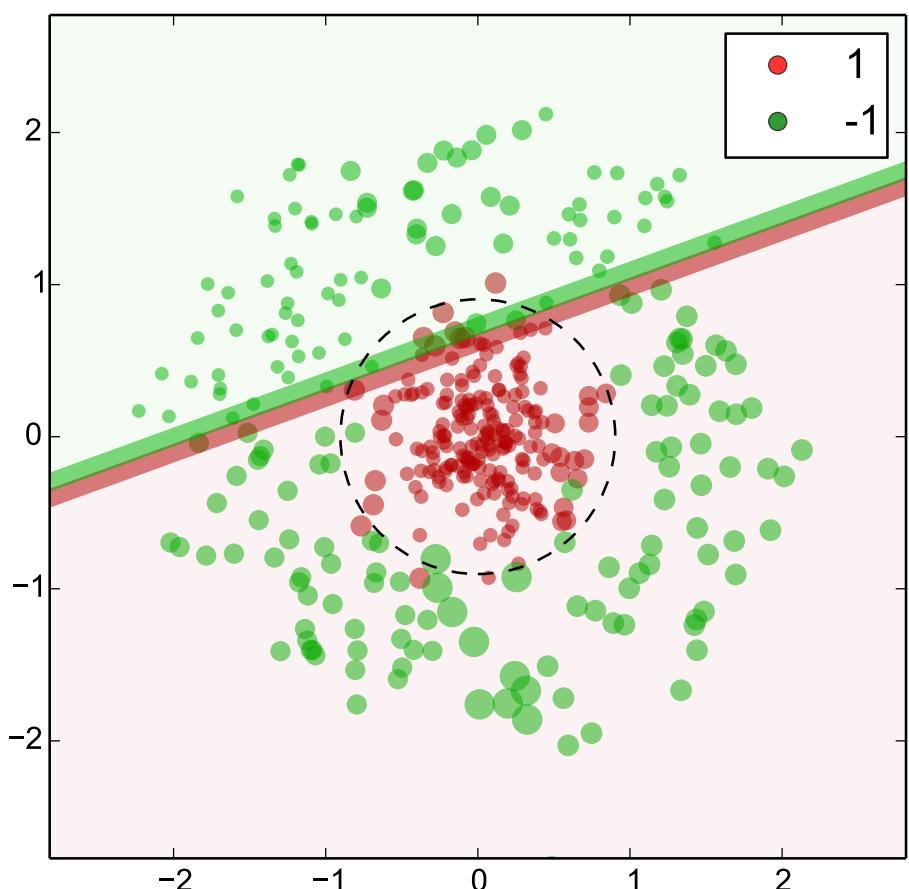


$$\epsilon_{H_t}^{\text{train}} = 13.0\%$$

$$\epsilon_{H_t}^{\text{test}} = 18.3\%$$

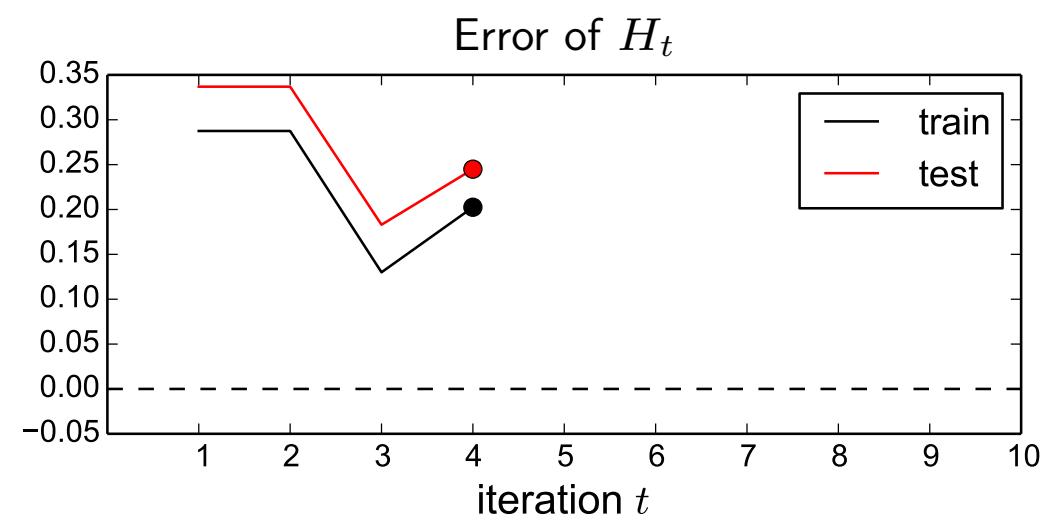
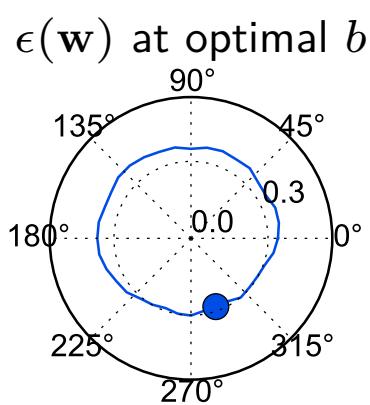
$$Z_t = 0.909$$

Example 1 – iteration 4



$$\epsilon_t = 28.3\%$$

$$\alpha_t = 0.465$$

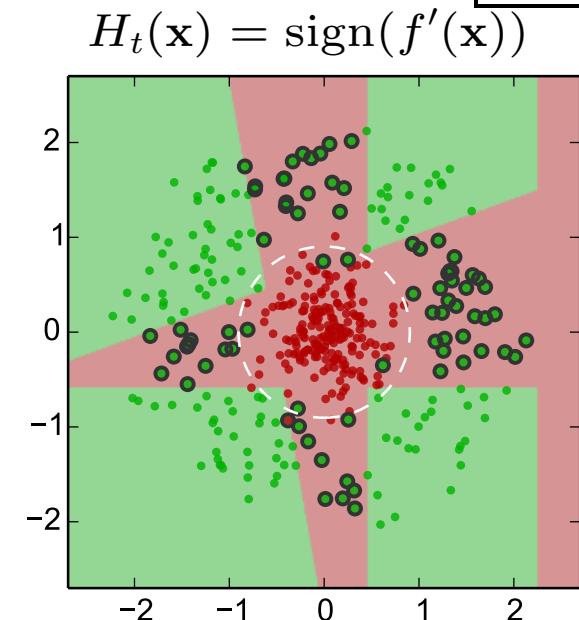
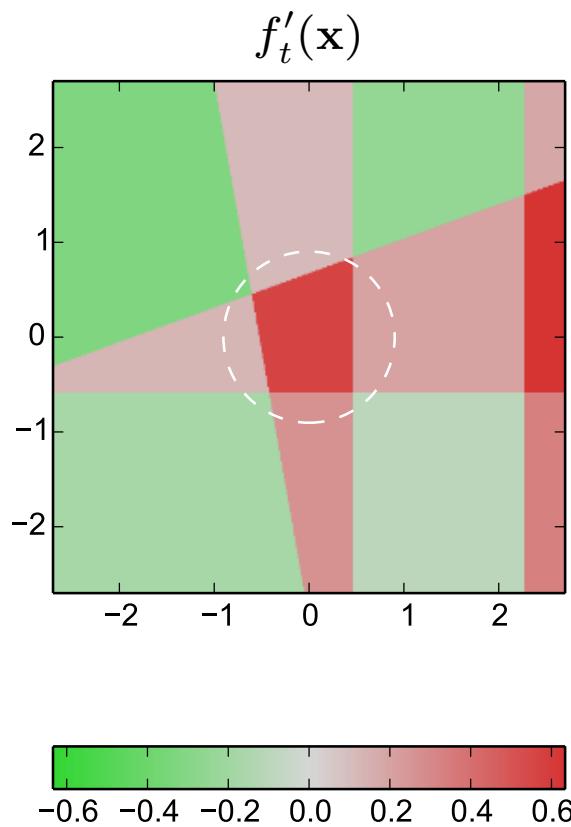
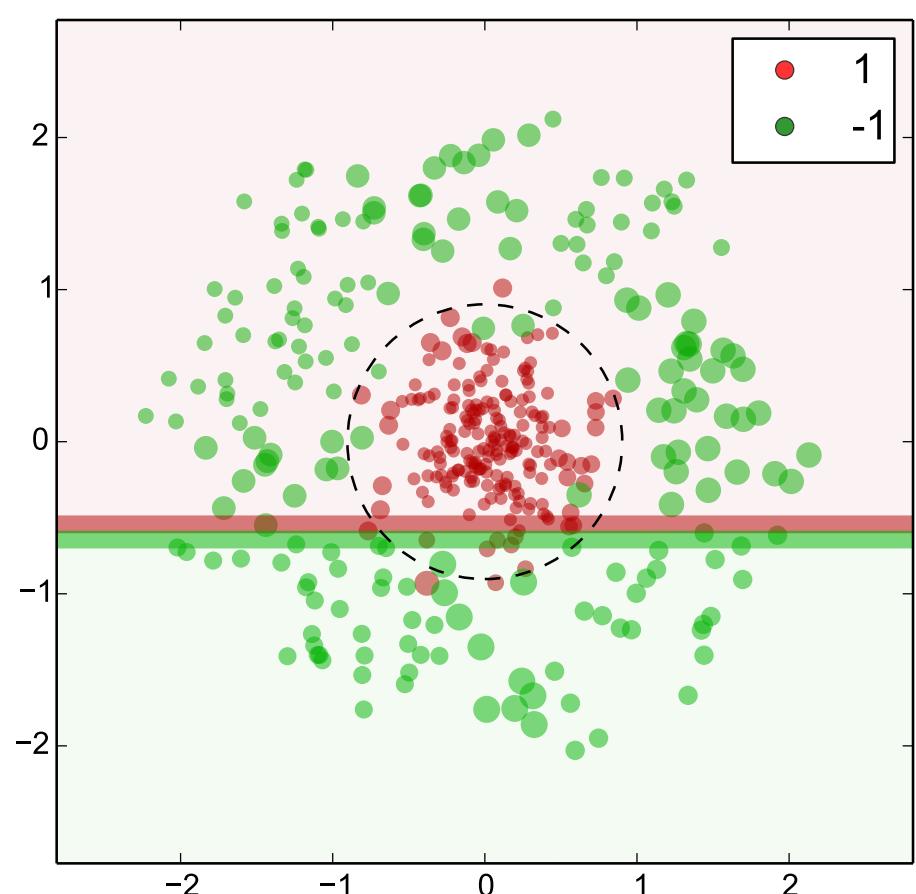


$$\epsilon_{H_t}^{\text{train}} = 20.2\%$$

$$\epsilon_{H_t}^{\text{test}} = 24.5\%$$

$$Z_t = 0.901$$

Example 1 – iteration 5



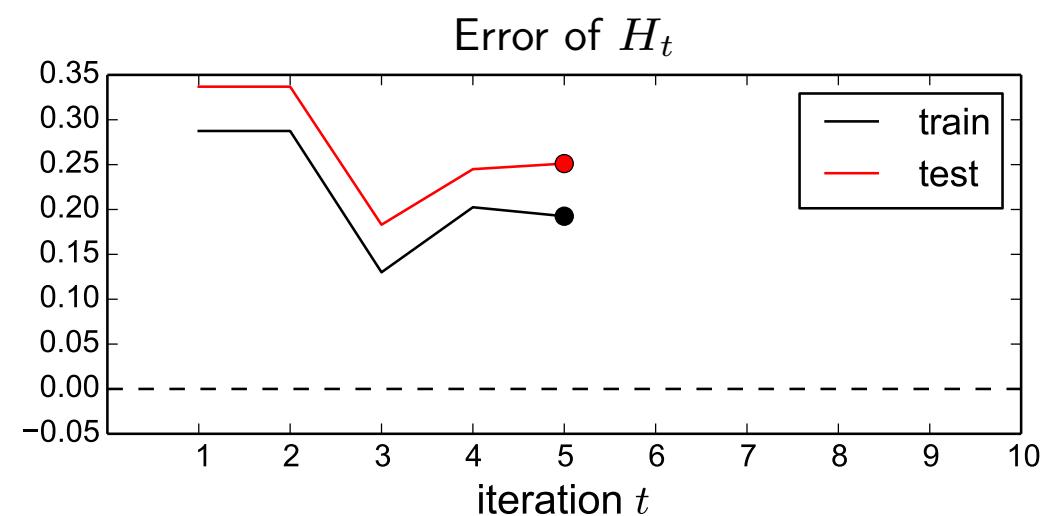
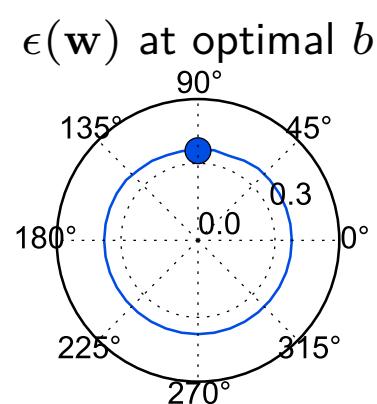
$$\epsilon_t = 34.9\%$$

$$\alpha_t = 0.312$$

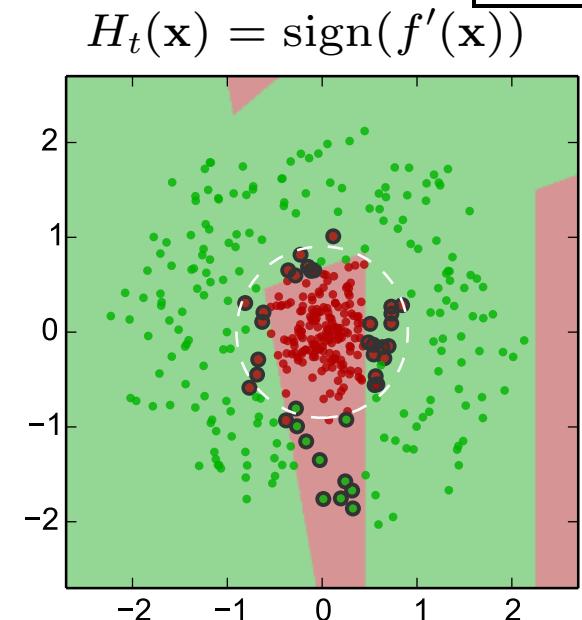
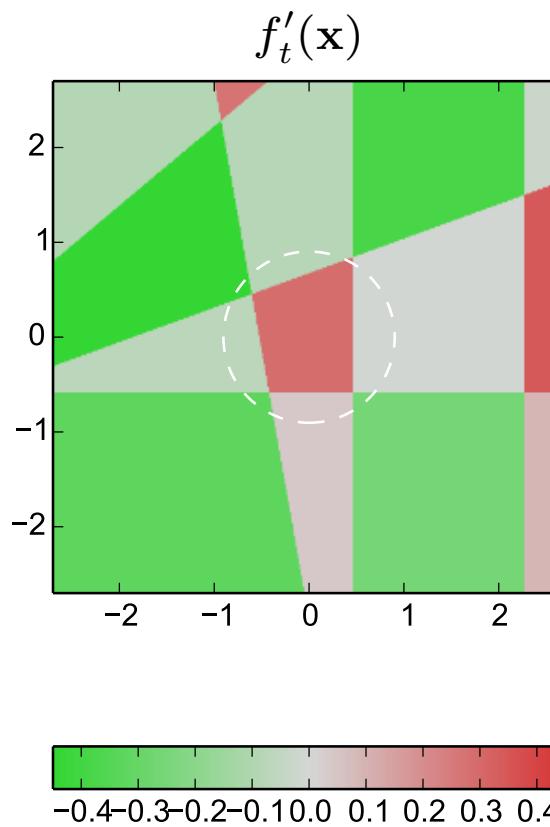
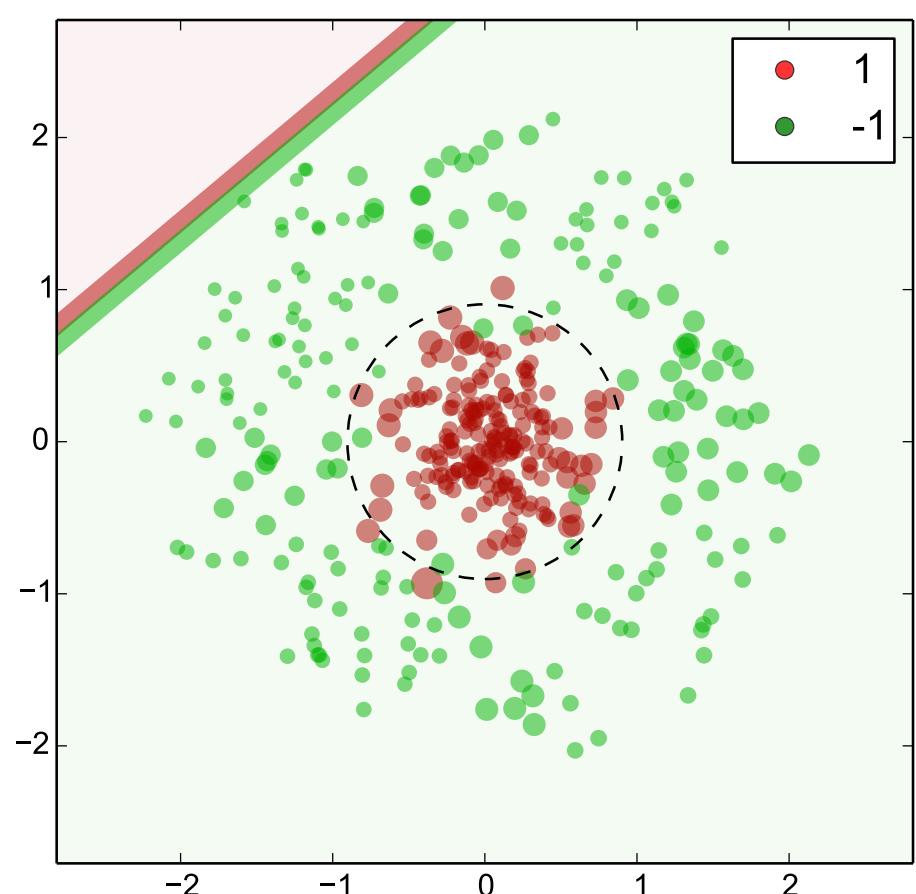
$$\epsilon_{H_t}^{\text{train}} = 19.2\%$$

$$\epsilon_{H_t}^{\text{test}} = 25.1\%$$

$$Z_t = 0.953$$

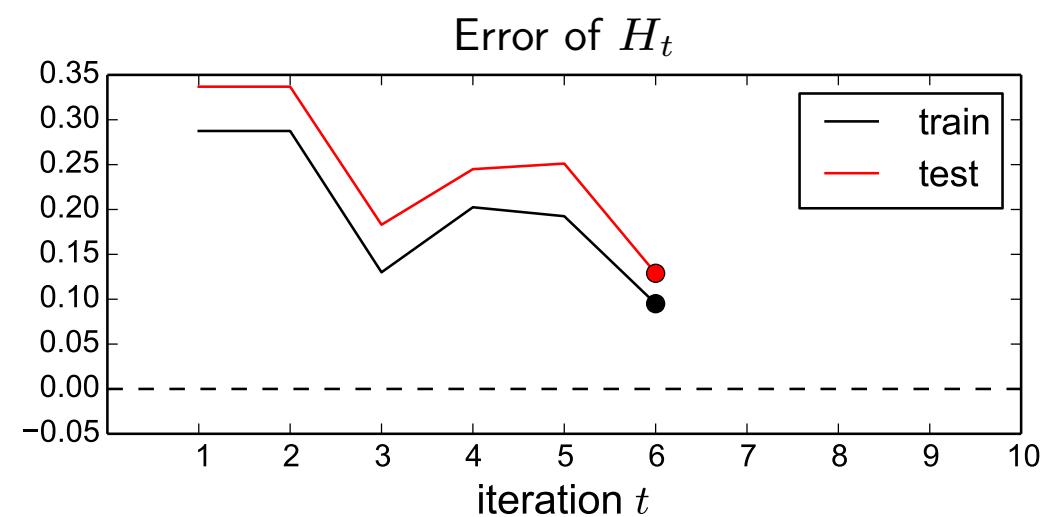
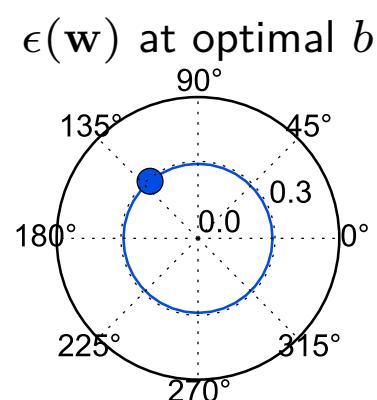


Example 1 – iteration 6



$$\epsilon_t = 29.0\%$$

$$\alpha_t = 0.447$$

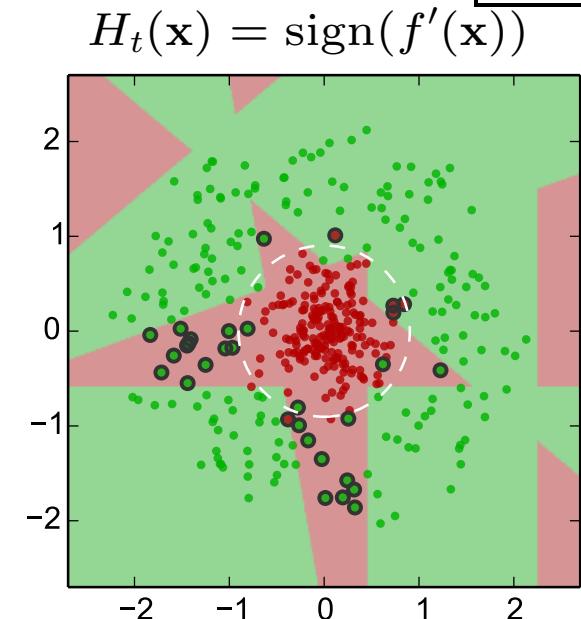
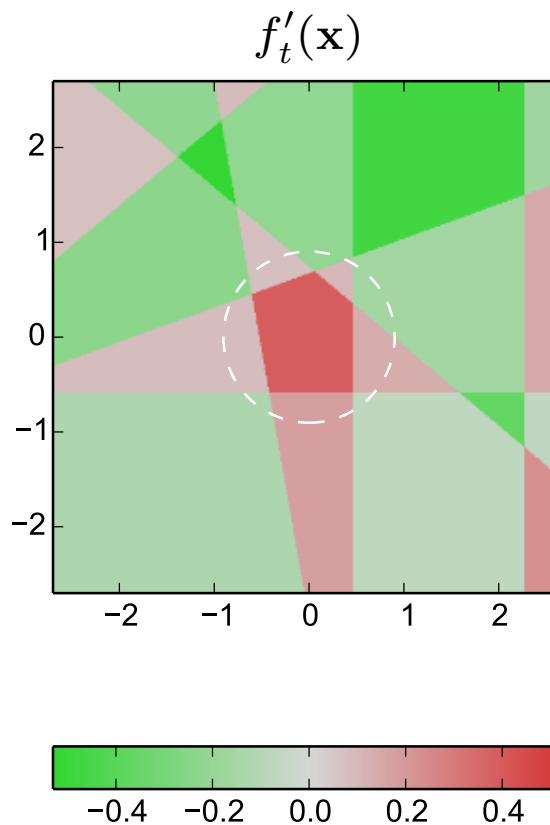
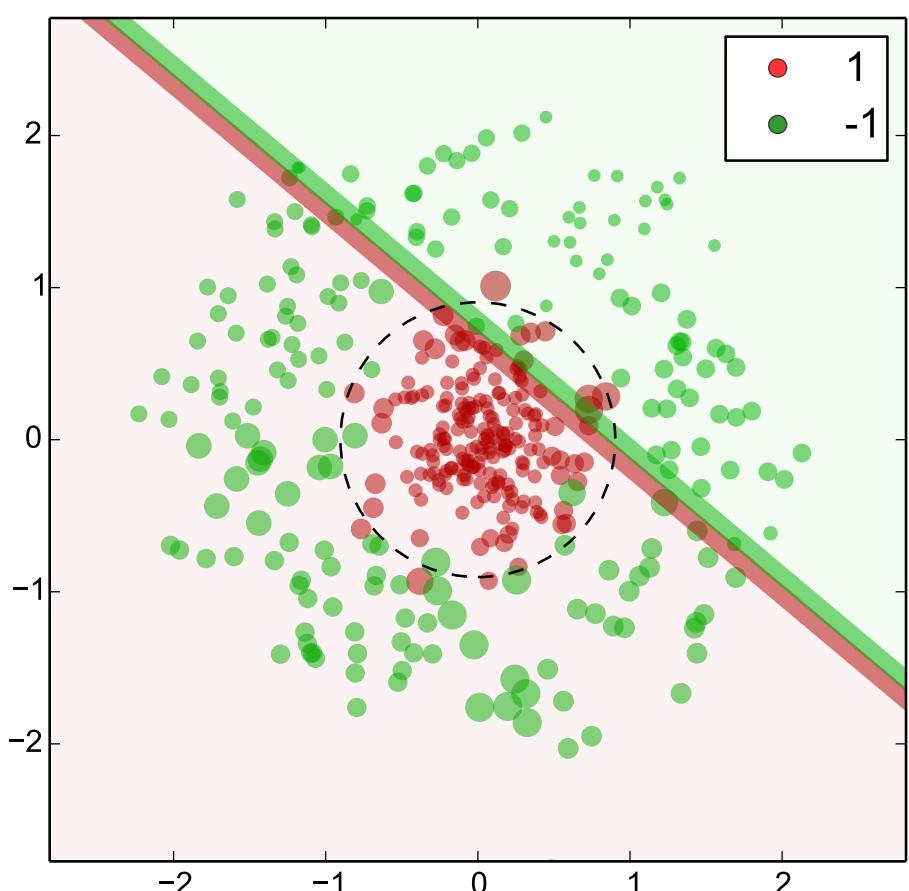


$$\epsilon_{H_t}^{\text{train}} = 9.50\%$$

$$\epsilon_{H_t}^{\text{test}} = 12.9\%$$

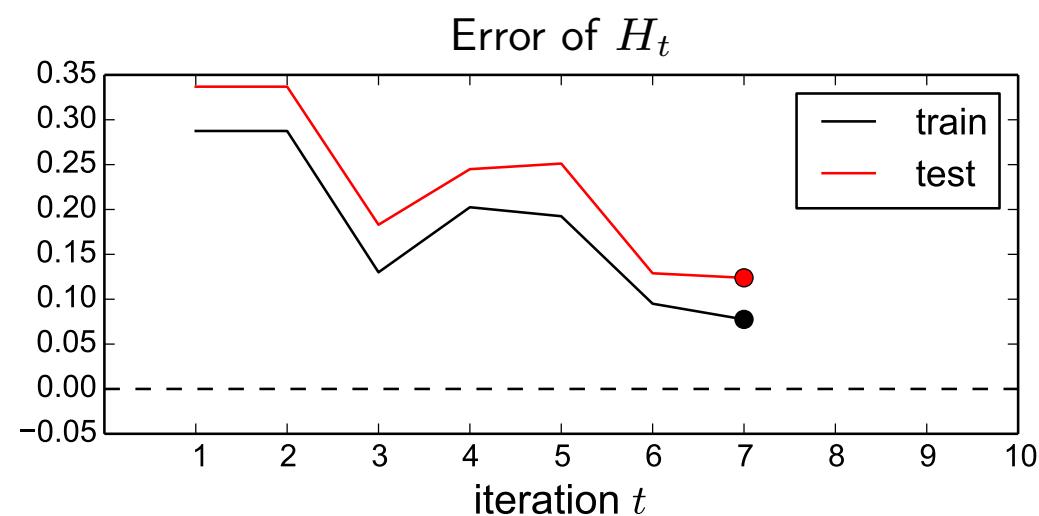
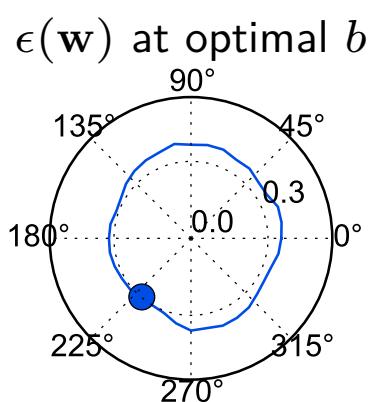
$$Z_t = 0.908$$

Example 1 – iteration 7



$$\epsilon_t = 29.8\%$$

$$\alpha_t = 0.429$$

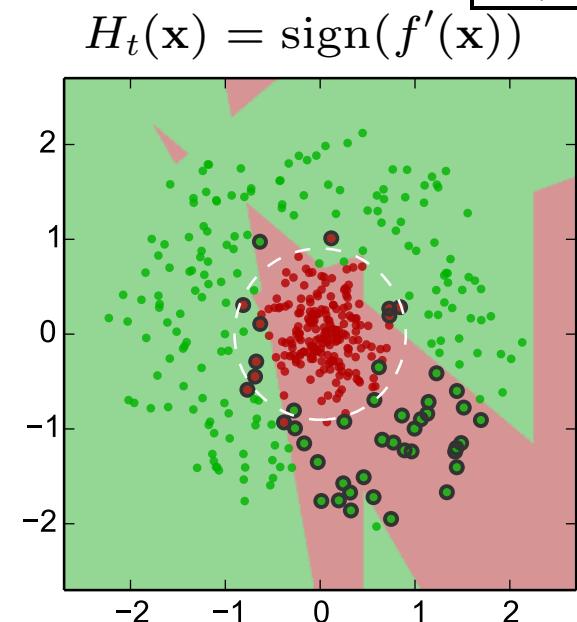
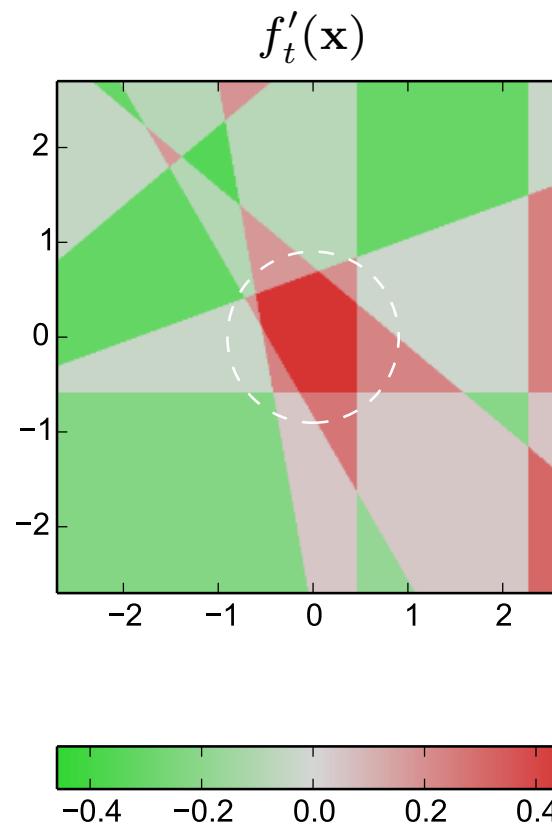
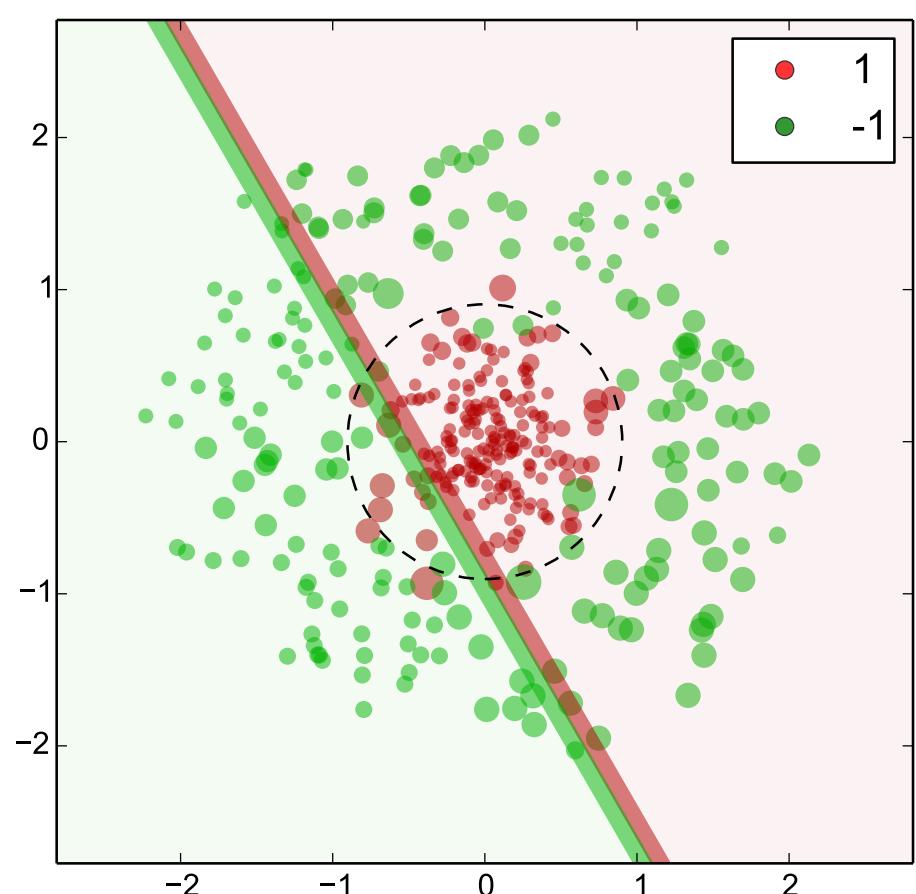


$$\epsilon_{H_t}^{\text{train}} = 7.75\%$$

$$\epsilon_{H_t}^{\text{test}} = 12.4\%$$

$$Z_t = 0.915$$

Example 1 – iteration 8



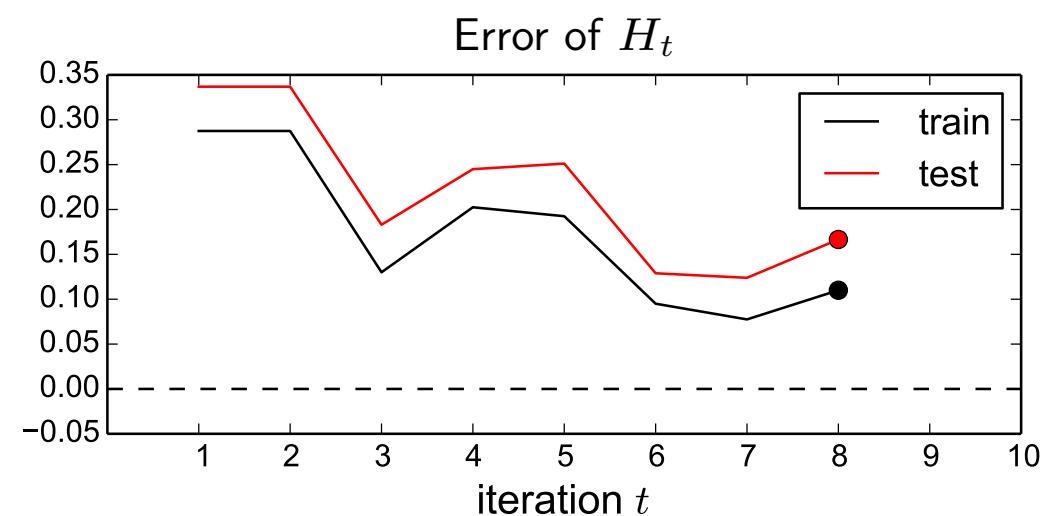
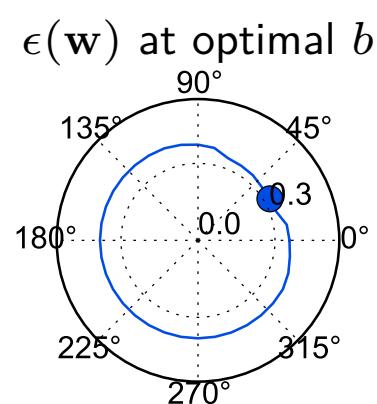
$$\epsilon_t = 32.3\%$$

$$\alpha_t = 0.369$$

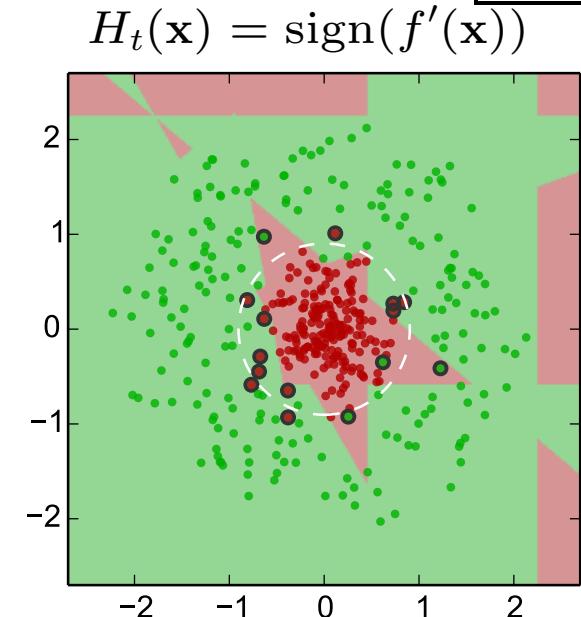
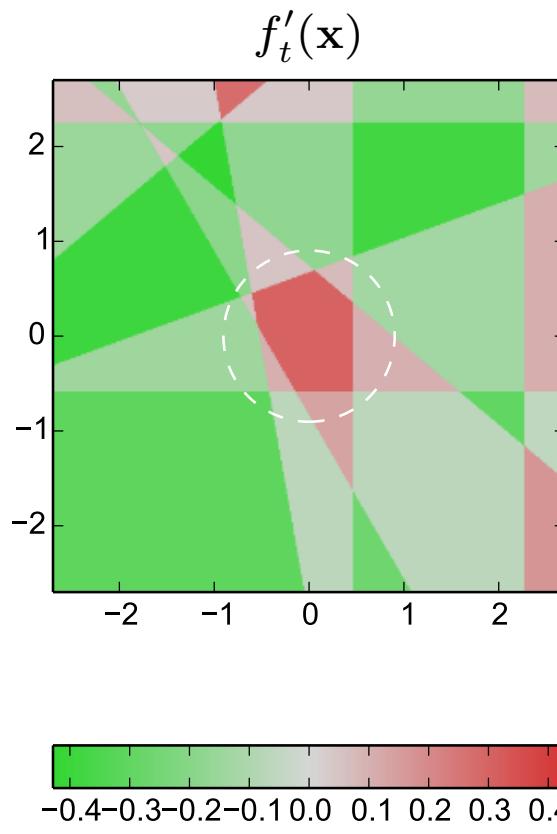
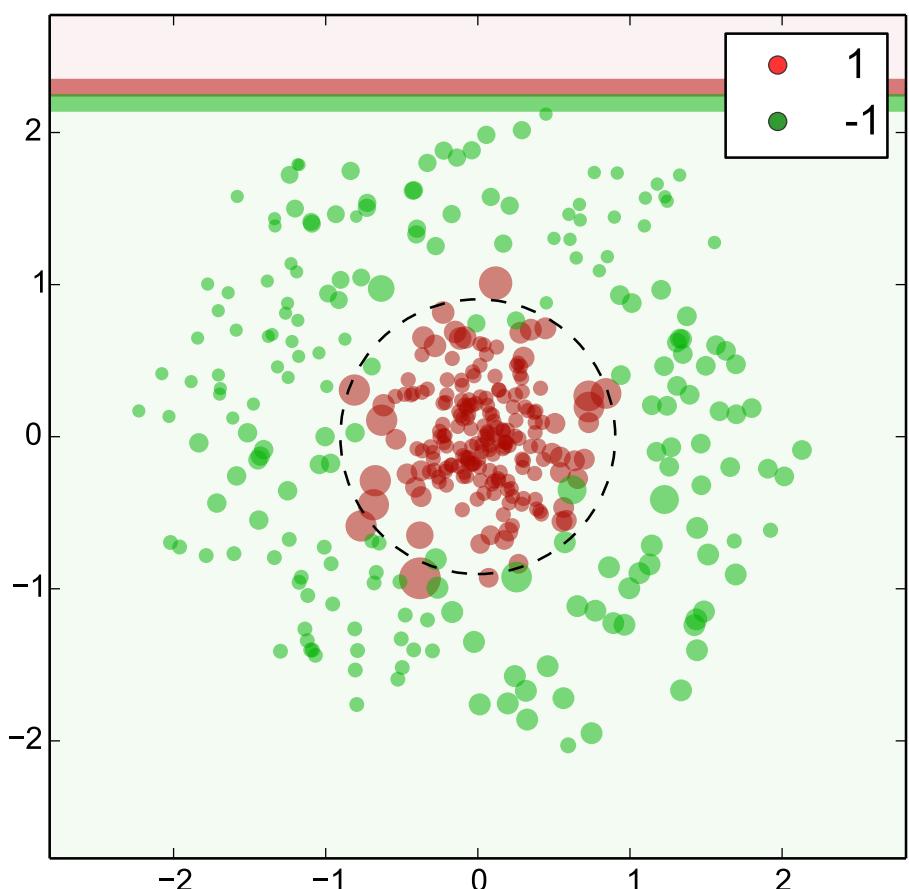
$$\epsilon_{H_t}^{\text{train}} = 11.0\%$$

$$\epsilon_{H_t}^{\text{test}} = 16.7\%$$

$$Z_t = 0.935$$



Example 1 – iteration 9



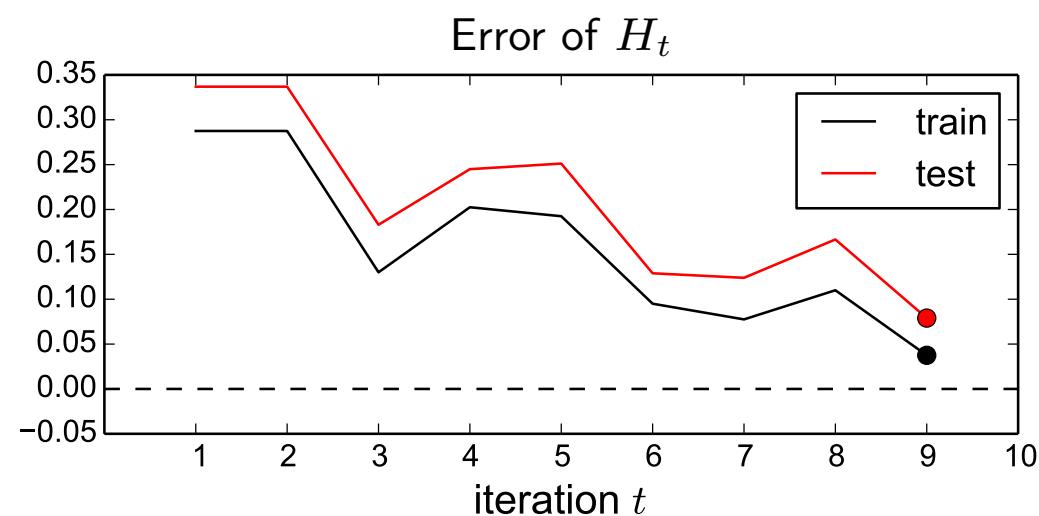
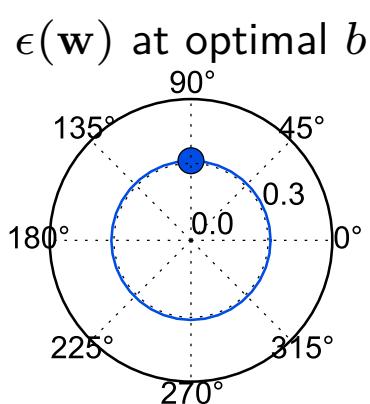
$$\epsilon_t = 31.0\%$$

$$\alpha_t = 0.400$$

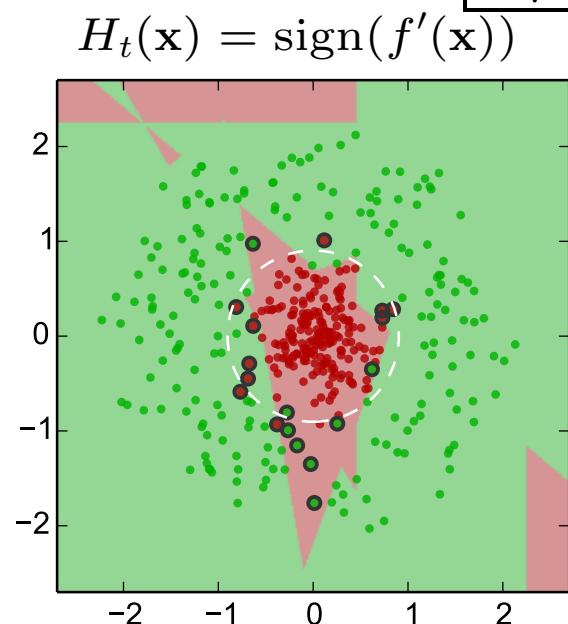
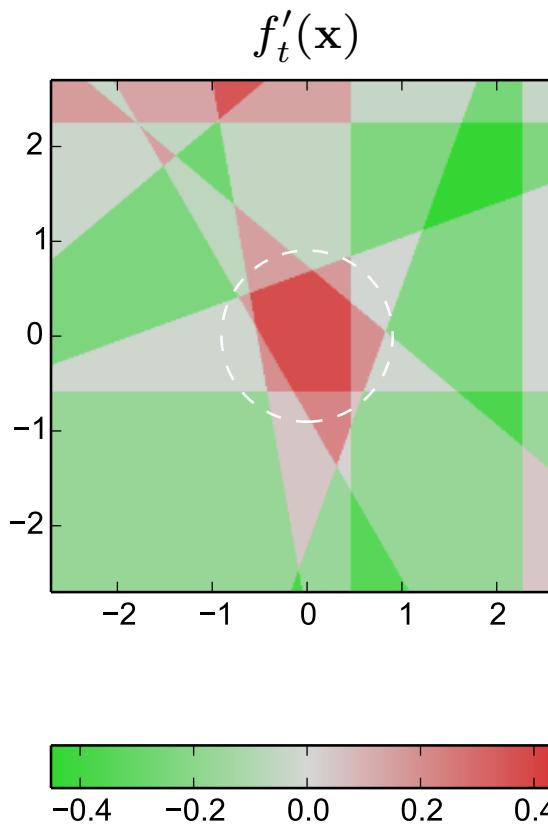
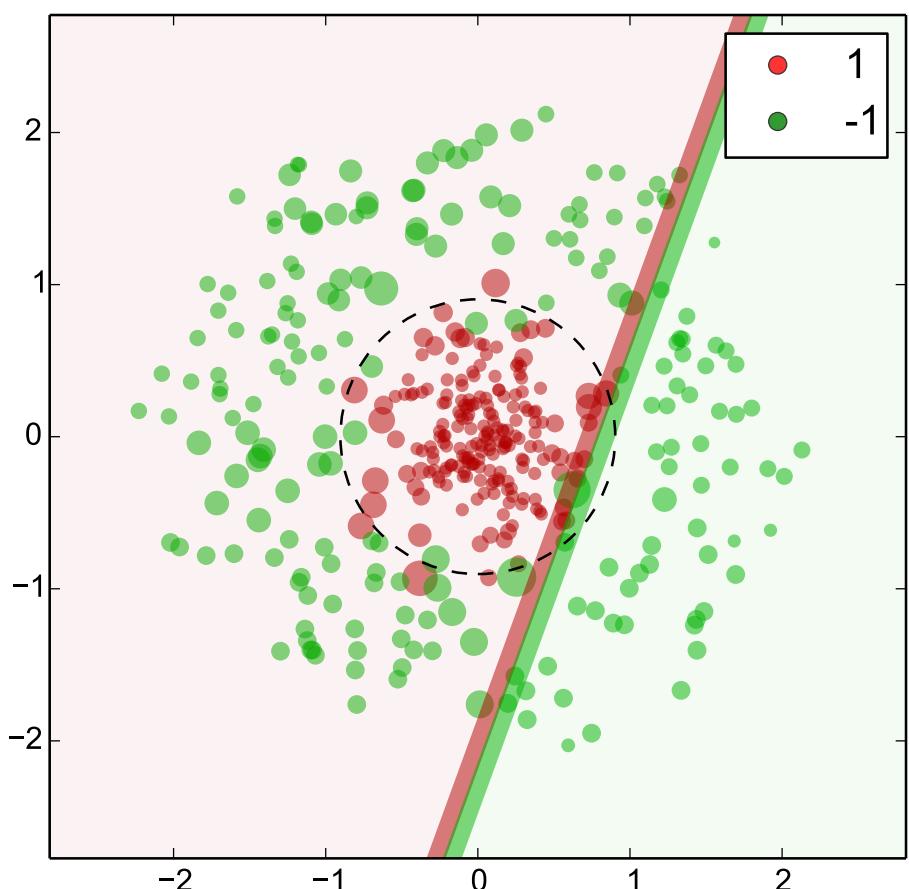
$$\epsilon_{H_t}^{\text{train}} = 3.75\%$$

$$\epsilon_{H_t}^{\text{test}} = 7.90\%$$

$$Z_t = 0.925$$

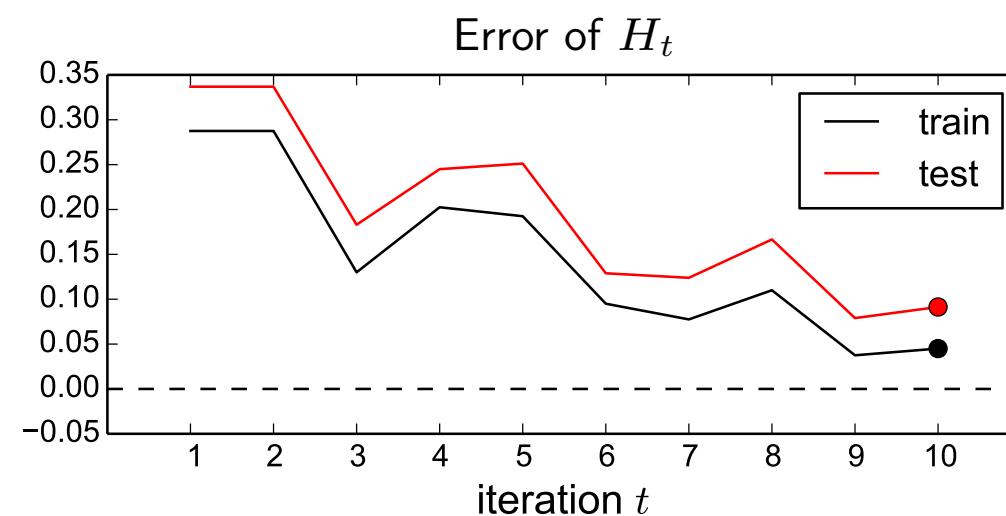
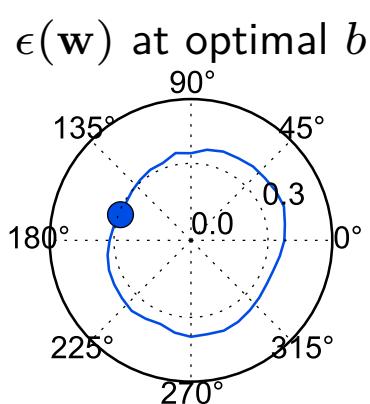


Example 1 – iteration 10



$$\epsilon_t = 29.2\%$$

$$\alpha_t = 0.443$$

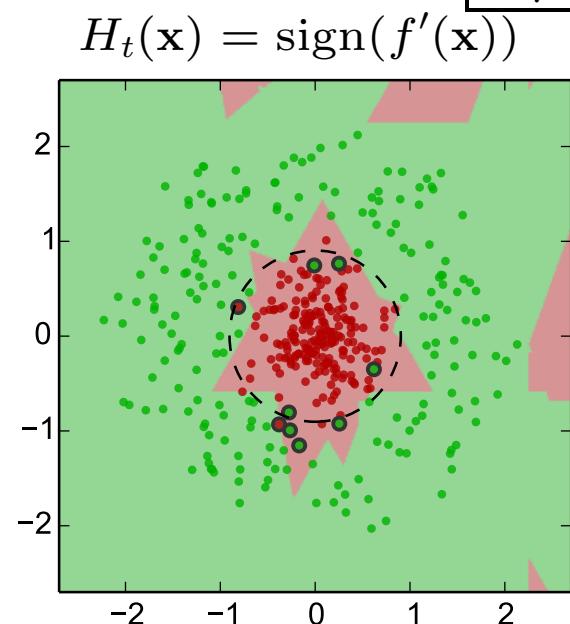
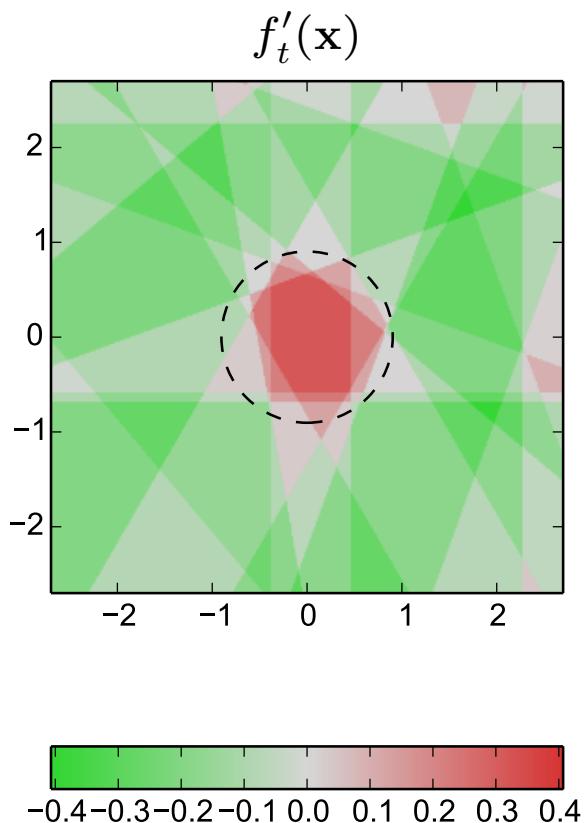
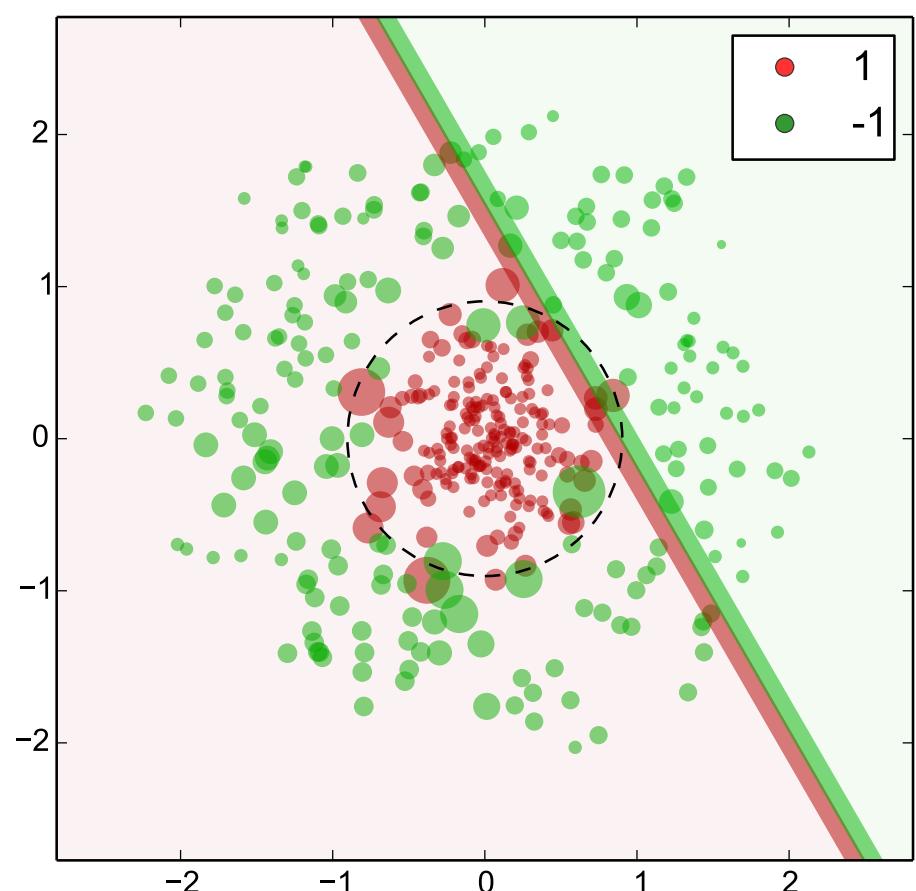


$$\epsilon_{H_t}^{\text{train}} = 4.50\%$$

$$\epsilon_{H_t}^{\text{test}} = 9.13\%$$

$$Z_t = 0.909$$

Example 1 – iteration 20



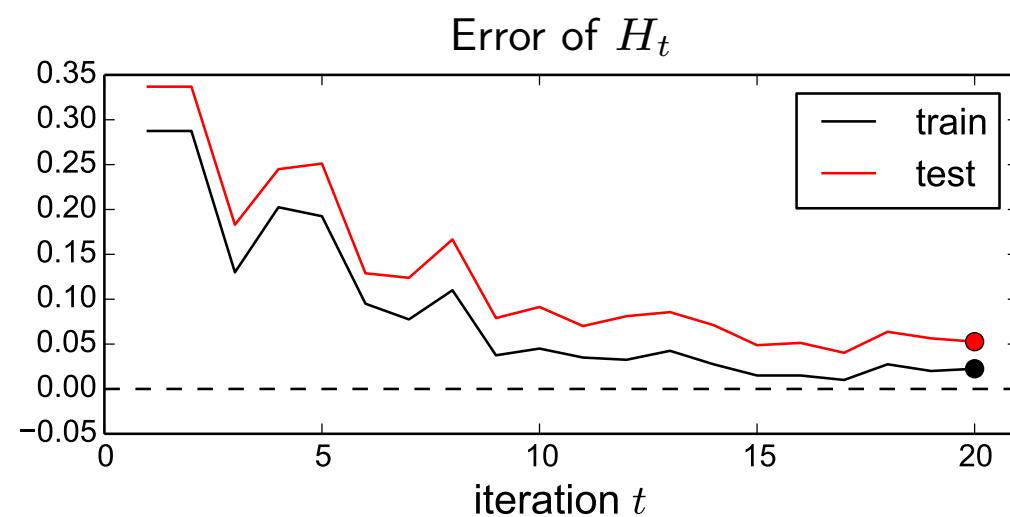
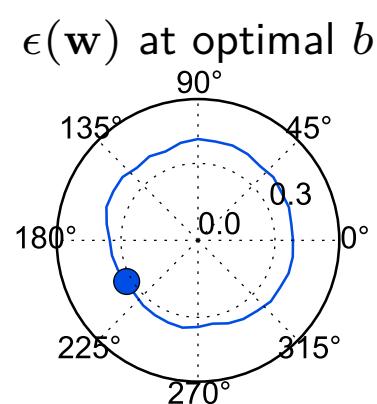
$$\epsilon_t = 32.0\%$$

$$\alpha_t = 0.376$$

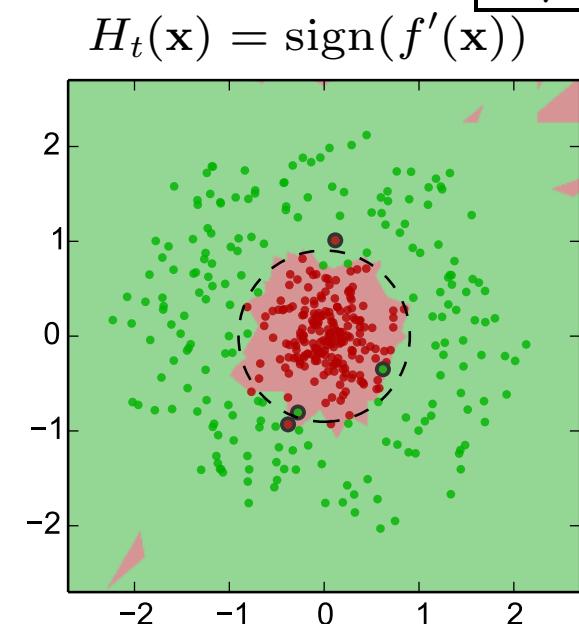
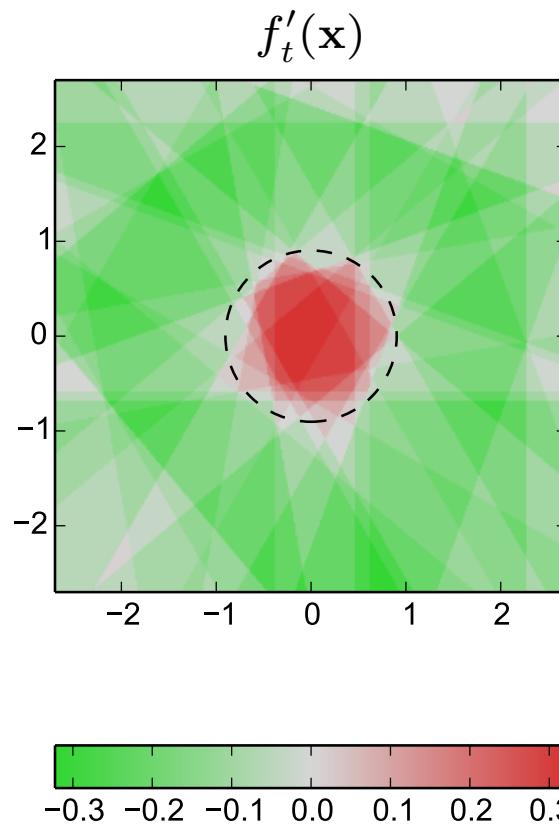
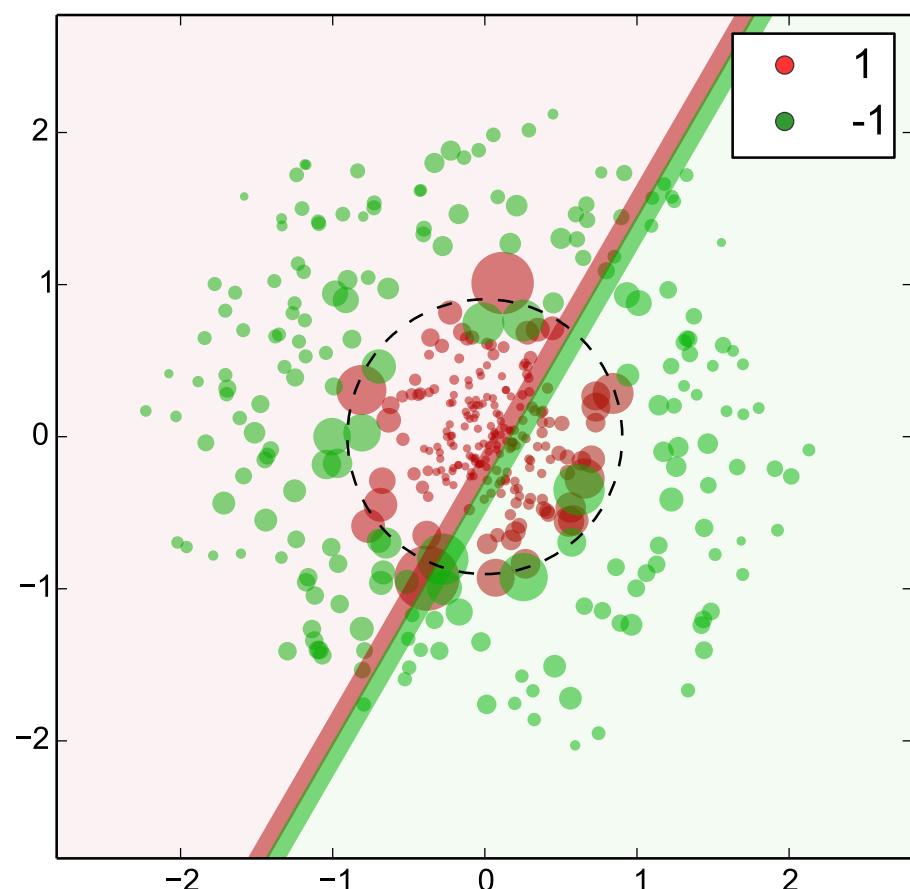
$$\epsilon_{H_t}^{\text{train}} = 2.25\%$$

$$\epsilon_{H_t}^{\text{test}} = 5.27\%$$

$$Z_t = 0.933$$



Example 1 – iteration 40



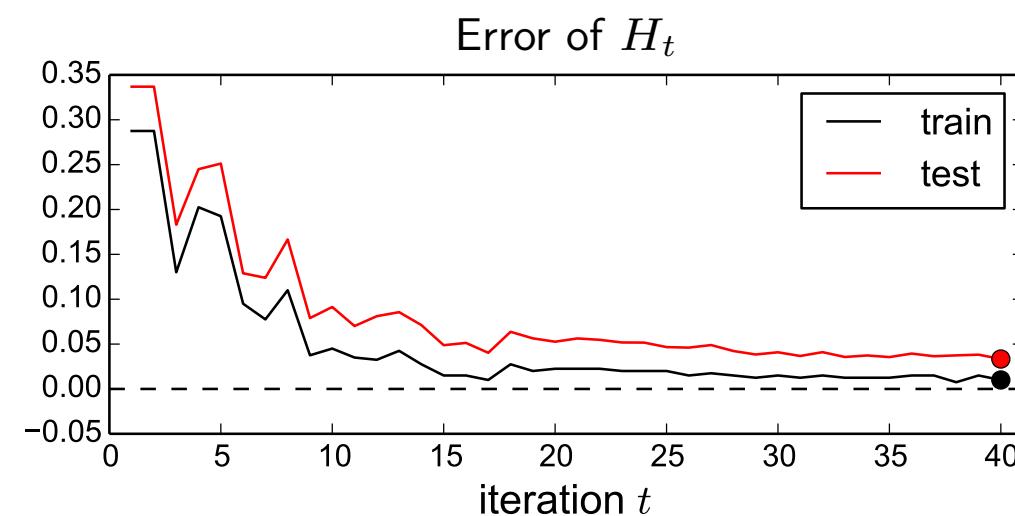
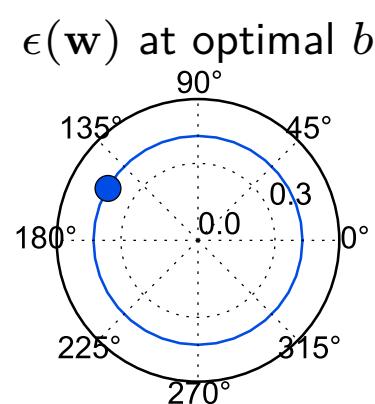
$$\epsilon_t = 40.4\%$$

$$\alpha_t = 0.194$$

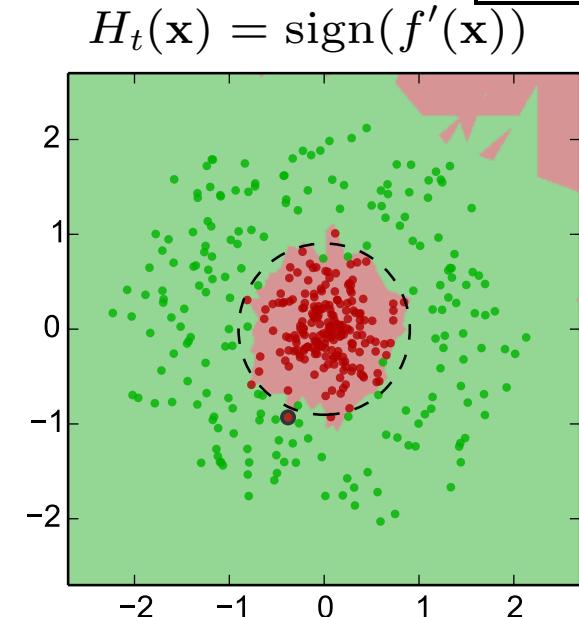
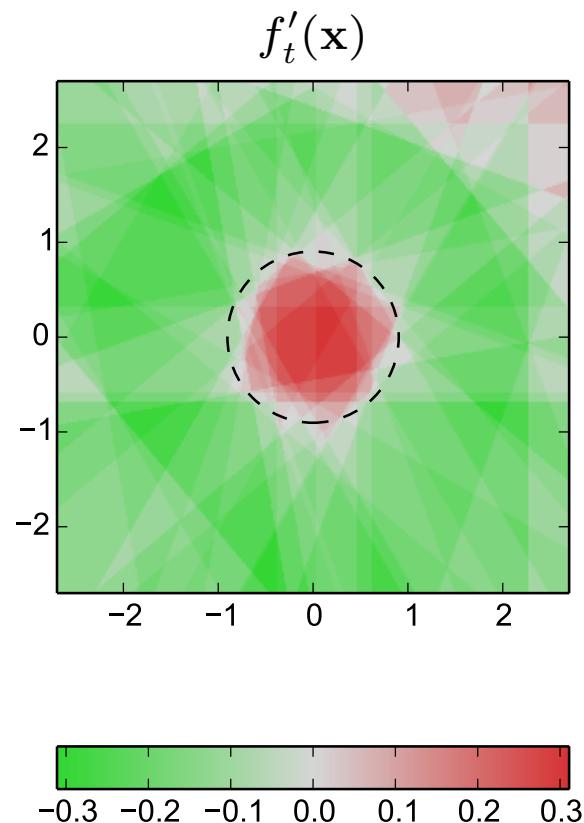
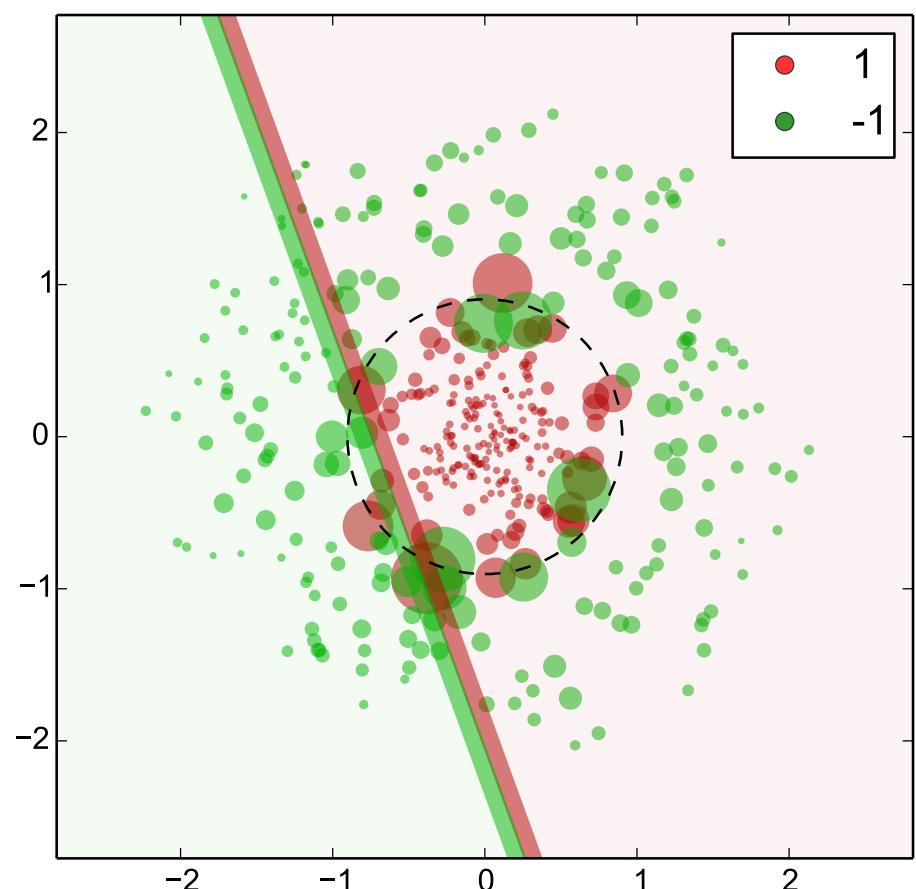
$$\epsilon_{H_t}^{\text{train}} = 1.00\%$$

$$\epsilon_{H_t}^{\text{test}} = 3.34\%$$

$$Z_t = 0.982$$



Example 1 – iteration 60



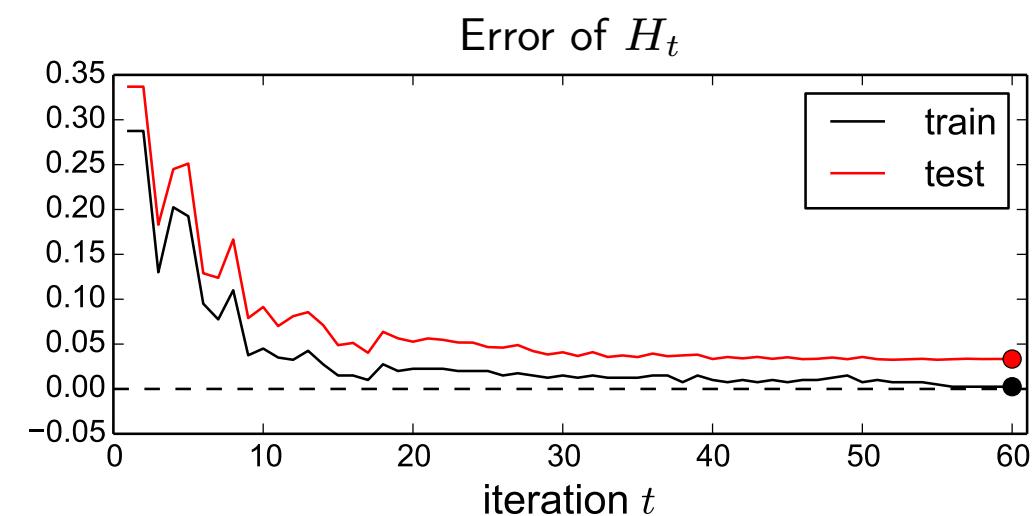
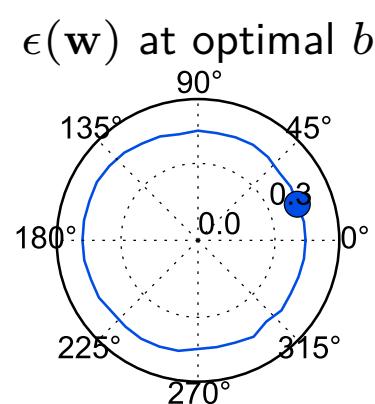
$$\epsilon_t = 41.1\%$$

$$\alpha_t = 0.179$$

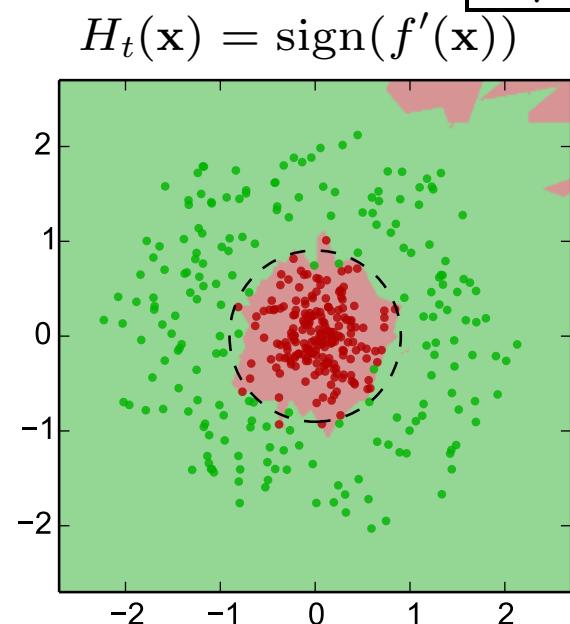
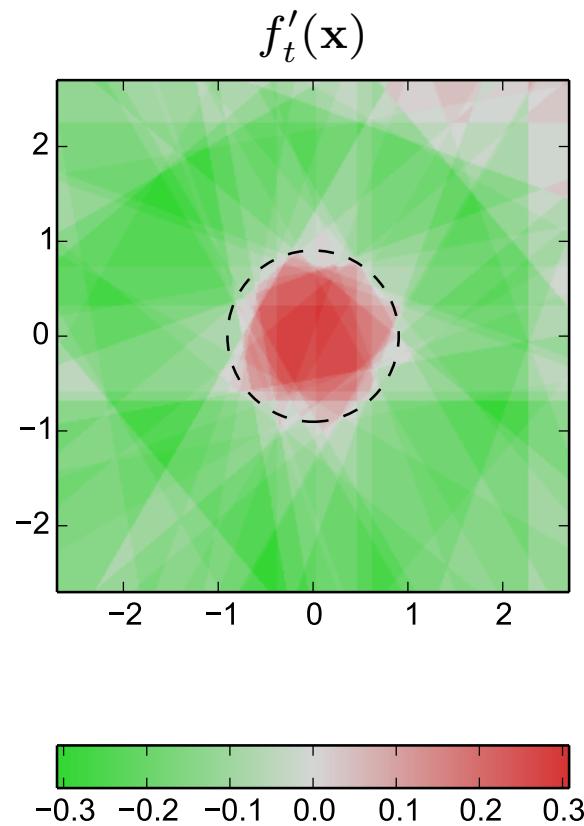
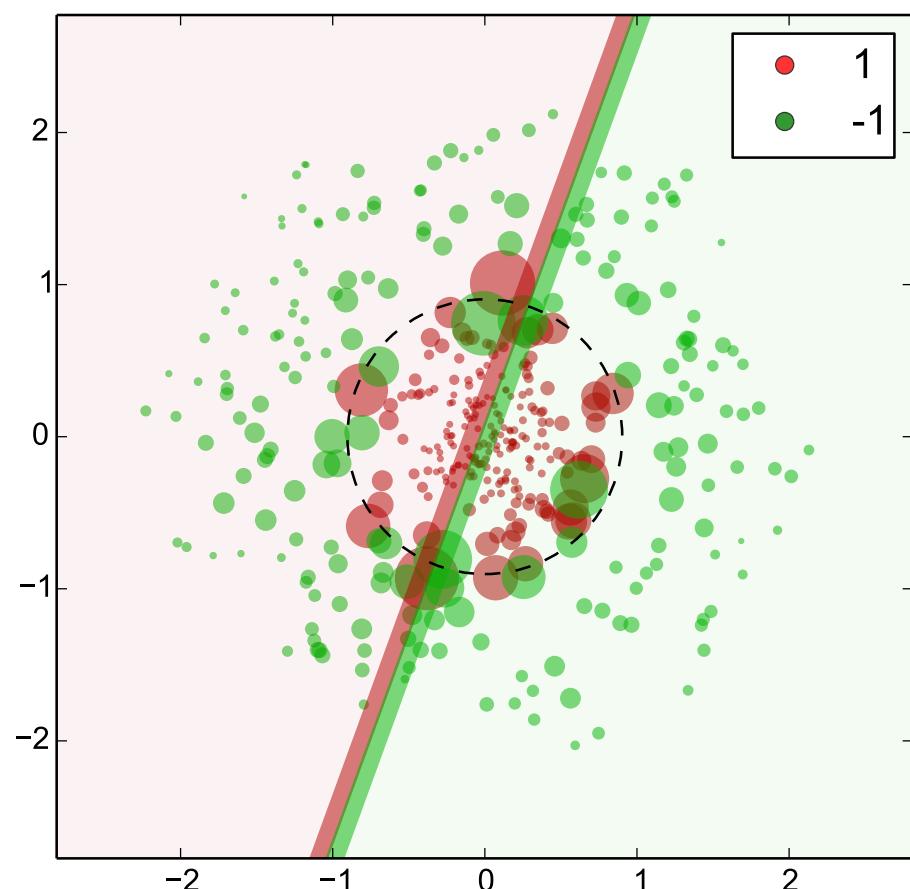
$$\epsilon_{H_t}^{\text{train}} = 0.250\%$$

$$\epsilon_{H_t}^{\text{test}} = 3.33\%$$

$$Z_t = 0.984$$



Example 1 – iteration 68



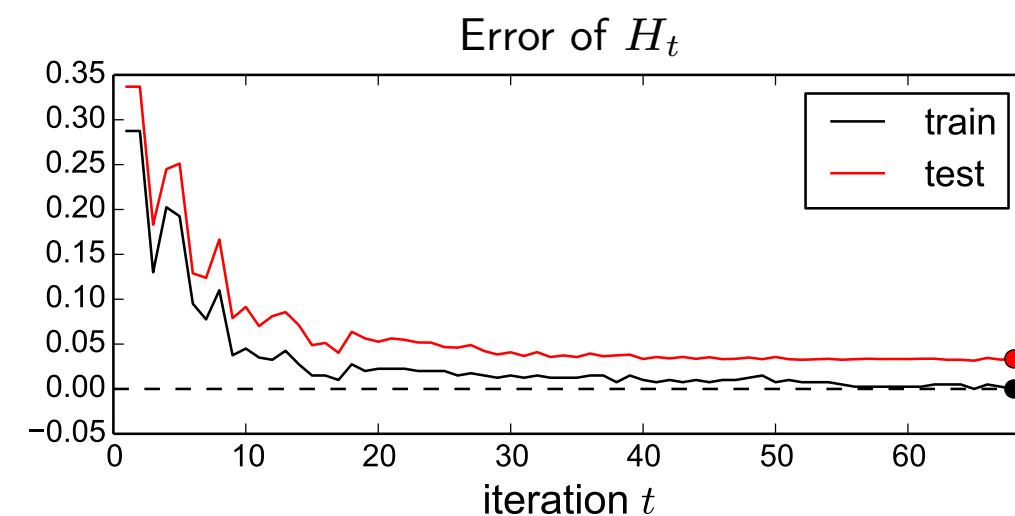
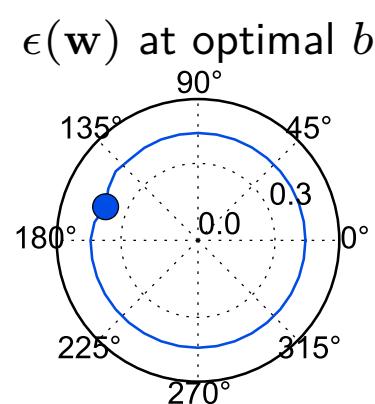
$$\epsilon_t = 38.3\%$$

$$\alpha_t = 0.239$$

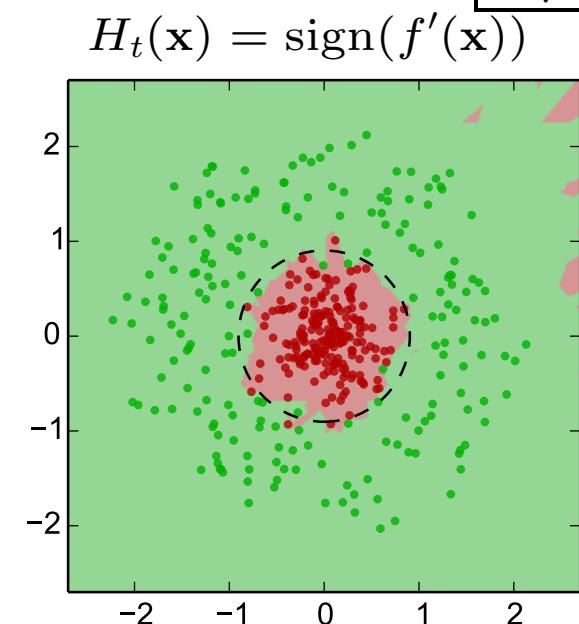
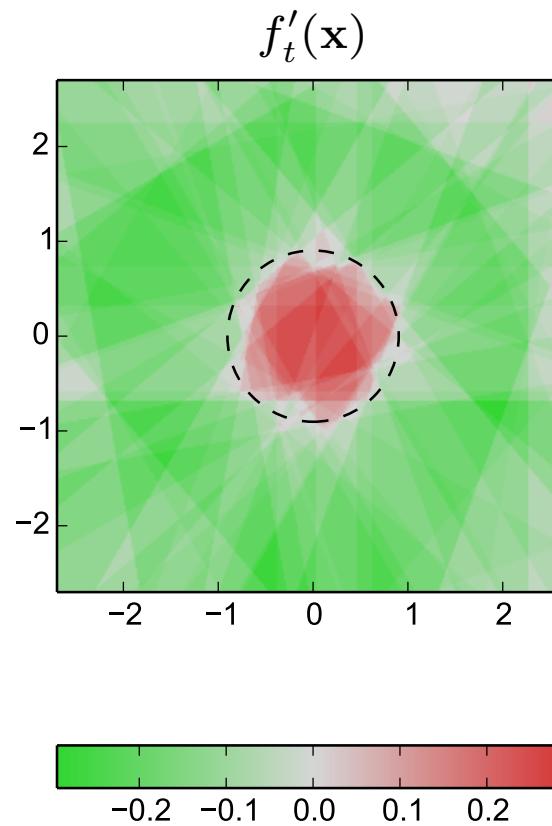
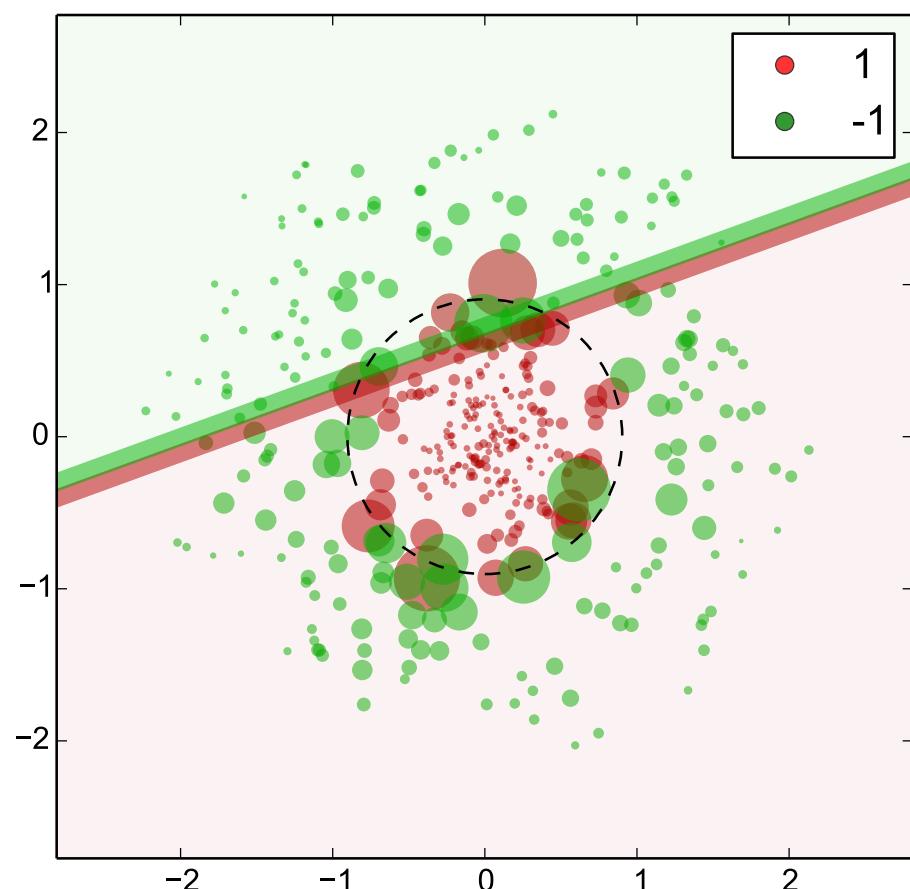
$$\epsilon_{H_t}^{\text{train}} = 0.00\%$$

$$\epsilon_{H_t}^{\text{test}} = 3.35\%$$

$$Z_t = 0.972$$

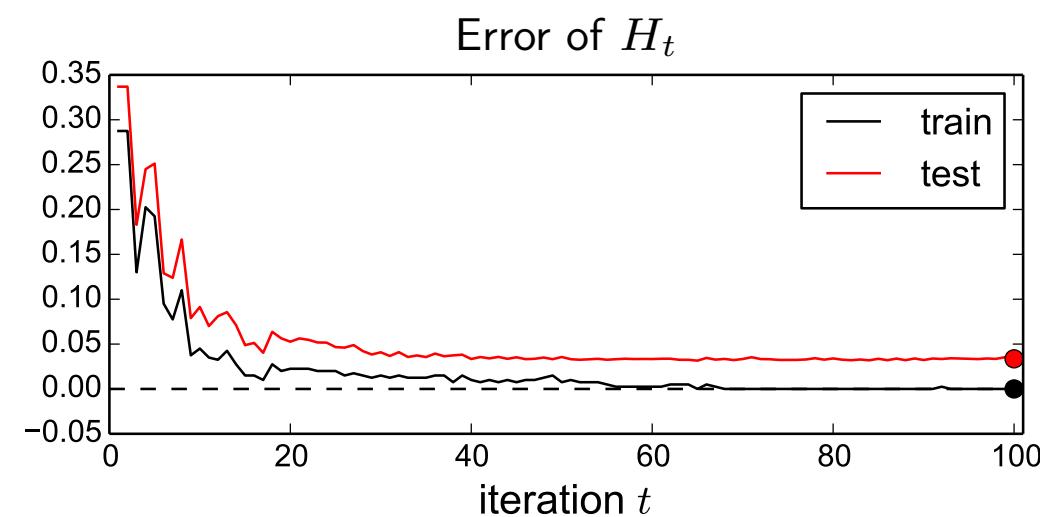
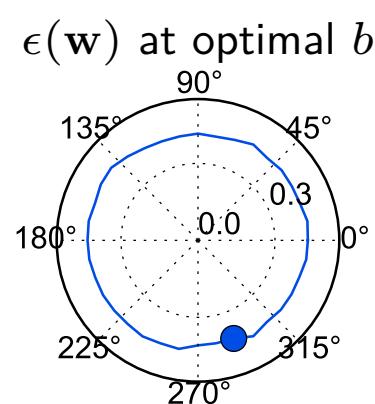


Example 1 – iteration 100



$$\epsilon_t = 40.7\%$$

$$\alpha_t = 0.189$$

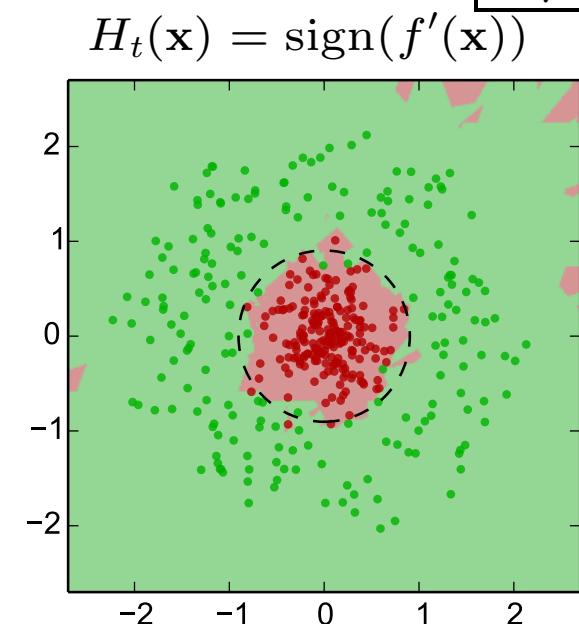
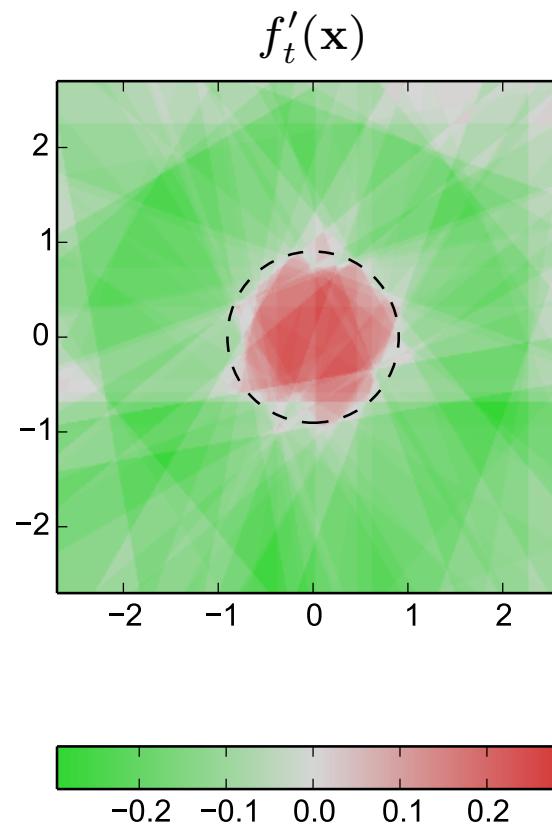
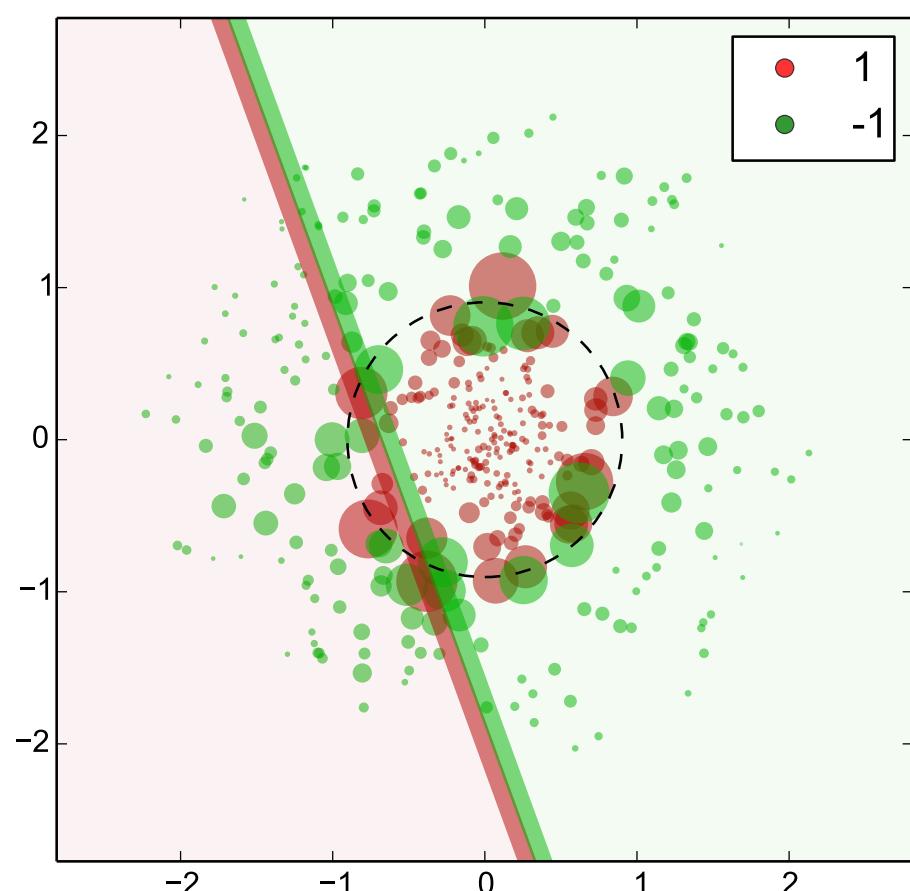


$$\epsilon_{H_t}^{\text{train}} = 0.00\%$$

$$\epsilon_{H_t}^{\text{test}} = 3.36\%$$

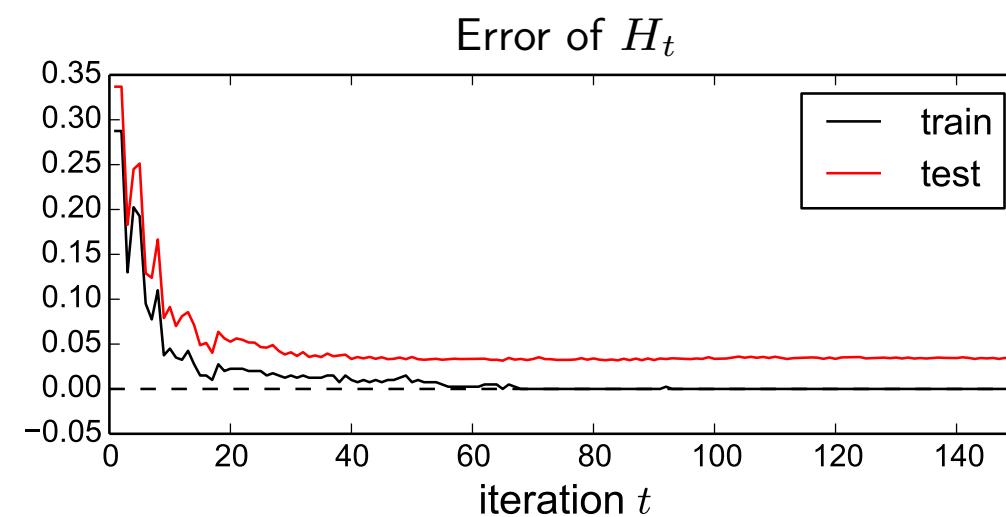
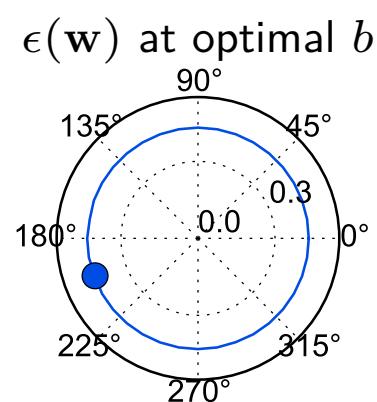
$$Z_t = 0.982$$

Example 1 – iteration 150



$$\epsilon_t = 42.6\%$$

$$\alpha_t = 0.149$$



$$\epsilon_{H_t}^{\text{train}} = 0.00\%$$

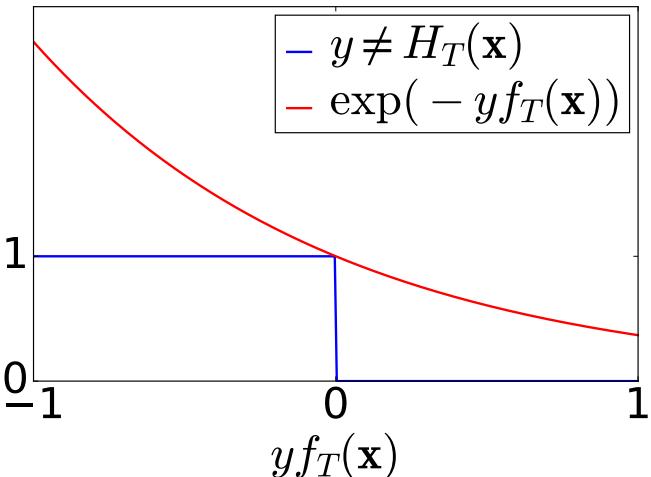
$$\epsilon_{H_t}^{\text{test}} = 3.40\%$$

$$Z_t = 0.989$$

Upper bound theorem (1/2)

Theorem: The following upper bound holds, in iteration T , for the training error ϵ of H_T :

$$\epsilon = \frac{1}{L} \sum_{i=1}^L \llbracket y_i \neq H_T(\mathbf{x}_i) \rrbracket \leq \prod_{t=1}^T Z_t.$$



Proof: There holds that:

$$\llbracket H_T(\mathbf{x}_i) \neq y_i \rrbracket \leq \exp(-y_i f_T(\mathbf{x}_i)), \quad (4)$$

which can be checked by a simple observation (the inequality follows from the first and last columns):

$\llbracket H_T(\mathbf{x}) \neq y \rrbracket$	classification	$y H_T(\mathbf{x})$	$y f_T(\mathbf{x})$	$\exp(-y f_T(\mathbf{x}))$
0	correct	1	> 0	≥ 0
1	incorrect	-1	< 0	≥ 1

Summing over the training dataset and dividing by L , we get

$$\epsilon = \frac{1}{L} \sum_i \llbracket H_T(\mathbf{x}_i) \neq y_i \rrbracket \leq \frac{1}{L} \sum_i \exp(-y_i f_T(\mathbf{x}_i))$$

Upper bound theorem (2/2)

Theorem: The following upper bound holds, in iteration T , for the training error ϵ of H_T :

$$\epsilon = \frac{1}{L} \sum_{i=1}^L \llbracket y_i \neq H_T(\mathbf{x}_i) \rrbracket \leq \prod_{t=1}^T Z_t.$$

Proof (contd.):

$$\epsilon = \frac{1}{L} \sum_i \llbracket H_T(\mathbf{x}_i) \neq y_i \rrbracket \leq \frac{1}{L} \sum_i \exp(-y_i f_T(\mathbf{x}_i))$$

But from the distribution update rule:

$$D_{T+1}(i) = \frac{\exp(-y_i f_T(\mathbf{x}_i))}{L \prod_{t=1}^T Z_t}$$

we have that

$$\frac{1}{L} \sum_i \exp(-y_i f_T(\mathbf{x}_i)) = \underbrace{\left(\prod_{t=1}^T Z_t \right)}_{=1} \left(\sum_i D_{T+1}(i) \right),$$

which completes the proof.

AdaBoost as a Minimiser of the Upper Bound on the Empirical Error

- ◆ The main objective is to minimize $\epsilon = \frac{1}{L} \sum_{i=1}^L \llbracket y_i \neq H_T(\mathbf{x}_i) \rrbracket$ (plus maximize the margin).
- ◆ ϵ has just been shown to be upperbounded: $\epsilon(H_T) \leq \prod_{t=1}^T Z_t$.
- ◆ Adaboost is minimizing this upper bound.
- ◆ It does so by greedily minimizing Z_t in each iteration.
- ◆ Recall that

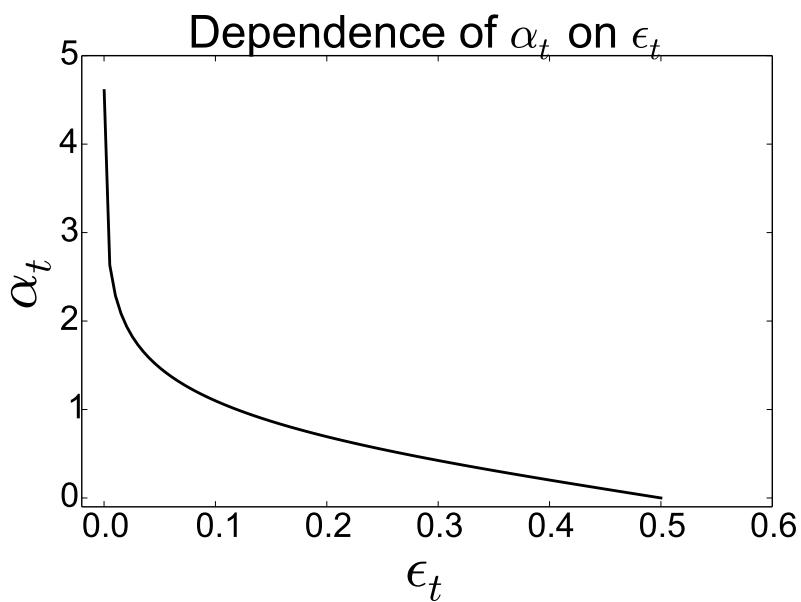
$$Z_t = \sum_{i=1}^L D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)};$$

given the dataset $\{(\mathbf{x}_i, y_i)\}$ and the distribution D_t in iteration t , the variables to minimize Z_t over are α_t and h_t .

Choosing α_t

Let us minimize $Z_t = \sum_i D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}$ with respect to α_t :

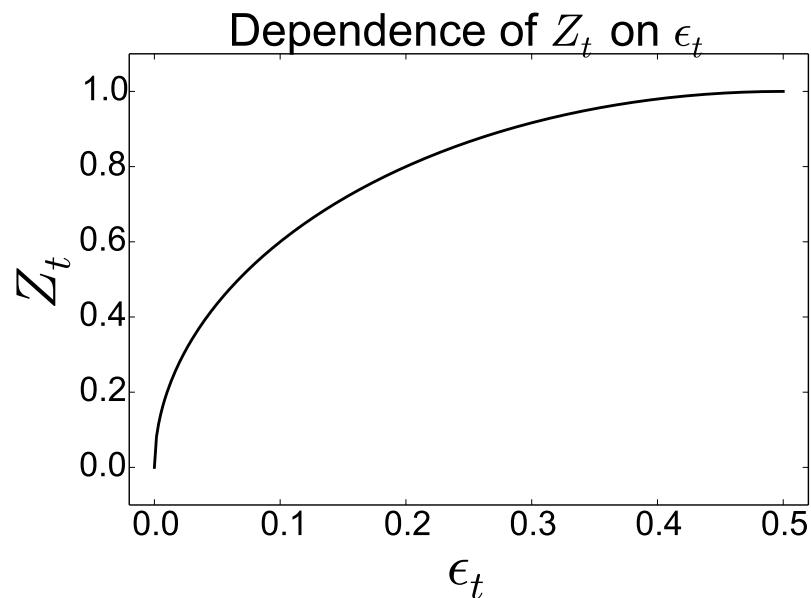
$$\begin{aligned} \frac{dZ}{d\alpha_t} &= - \sum_{i=1}^L D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} y_i h_t(\mathbf{x}_i) = 0 \\ &- \underbrace{\sum_{i:y_i=h_t(\mathbf{x}_i)} D_t(i) e^{-\alpha_t}}_{1 - \epsilon_t} + \underbrace{\sum_{i:y_i \neq h_t(\mathbf{x}_i)} D_t(i) e^{\alpha_t}}_{\epsilon_t} = 0 \\ -e^{-\alpha_t}(1 - \epsilon_t) + e^{\alpha_t}\epsilon_t &= 0 \\ \alpha_t + \log \epsilon_t &= -\alpha_t + \log(1 - \epsilon_t) \\ \alpha_t &= \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t} \end{aligned}$$



Choosing h_t

Let us substitute $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$ into Z_t :

$$\begin{aligned}
 Z_t &= \sum_{i=1}^L D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} \\
 &= \sum_{i:y_i=h_t(\mathbf{x}_i)} D_t(i) e^{-\alpha_t} + \sum_{i:y_i \neq h_t(\mathbf{x}_i)} D_t(i) e^{\alpha_t} \\
 &= (1 - \epsilon_t) e^{-\alpha_t} + \epsilon_t e^{\alpha_t} \\
 &= 2\sqrt{\epsilon_t(1 - \epsilon_t)}
 \end{aligned}$$



⇒ Z_t is minimised by selecting h_t with minimal weighted error ϵ_t .

Weak classifier examples

- ◆ Decision tree, Perceptron – \mathcal{B} infinite
- ◆ Selecting the best one from a given *finite* set \mathcal{B}

Minimization of an Upper Bound on the Empirical Error - Recapitulation

Choosing α_t and h_t

- ◆ For any weak classifier h_t with error ϵ_t , $Z_t(\alpha)$ is a convex differentiable function with a single minimum at α_t :

$$\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$$

- ◆ $Z_t = 2\sqrt{\epsilon_t(1 - \epsilon_t)} \leq 1$ for optimal $\alpha_t \Rightarrow Z_t$ is minimized by h_t with minimal ϵ_t .

Comments

- ◆ The process of selecting α_t and $h_t(x)$ can be interpreted as a single optimisation step minimising the upper bound on the empirical error. Improvement of the bound is guaranteed, provided that $\epsilon < 1/2$.
- ◆ The process can be interpreted as a component-wise local optimisation (Gauss-Southwell iteration) in the (possibly infinite dimensional!) space of $\vec{\alpha} = (\alpha_1, \alpha_2, \dots)$ starting from $\vec{\alpha}_0 = (0, 0, \dots)$.

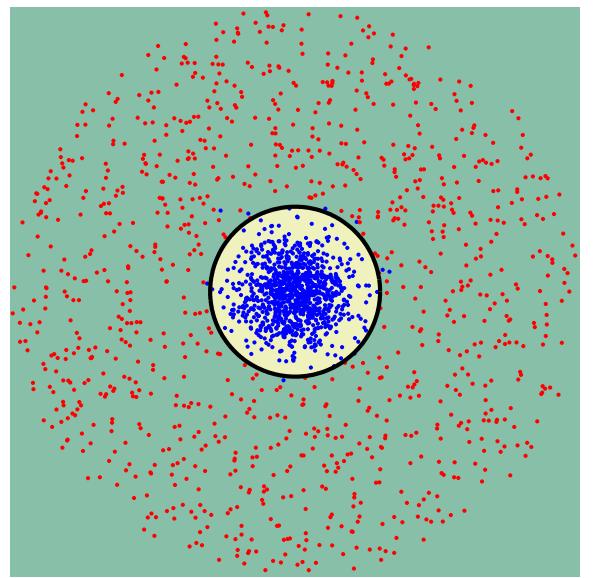
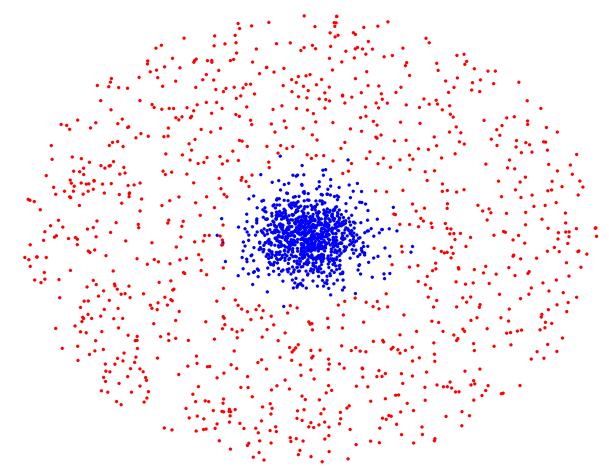
Summary of the Algorithm

Initialization ...

Summary of the Algorithm

Initialization ...

For $t = 1, \dots, T$:

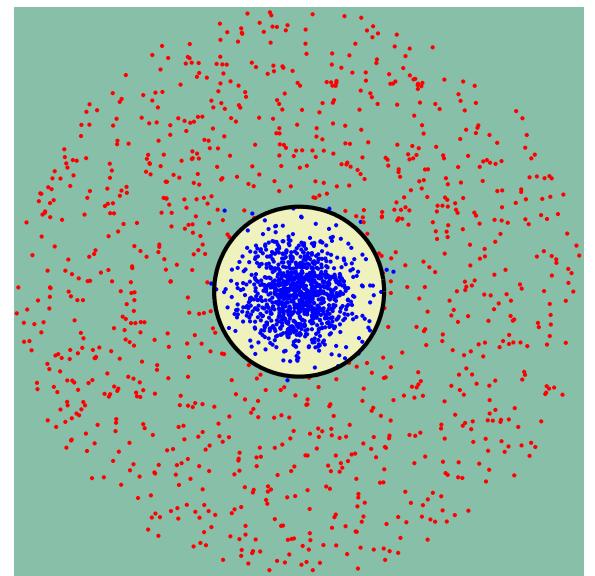
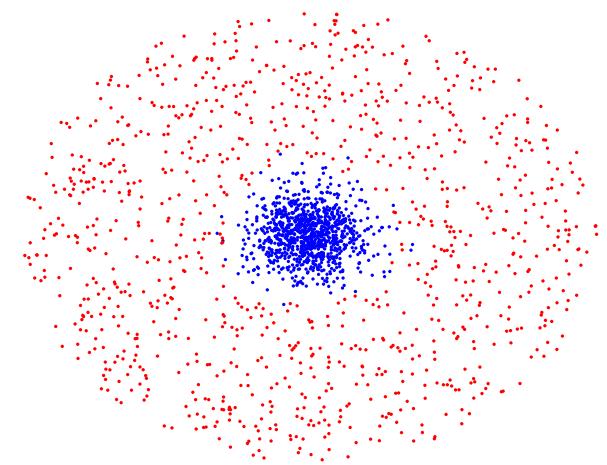


Summary of the Algorithm

Initialization ...

For $t = 1, \dots, T$:

- ◆ Find $h_t = \arg \min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^L D_t(i) \llbracket y_i \neq h(\mathbf{x}_i) \rrbracket$



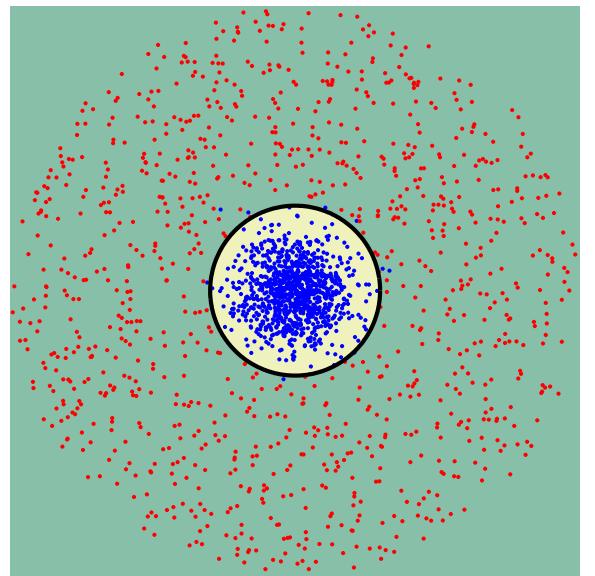
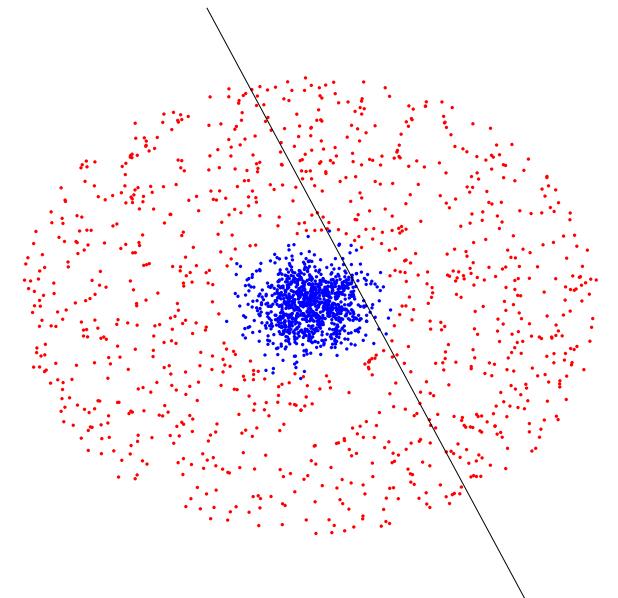
Summary of the Algorithm

Initialization ...

$t = 1$

For $t = 1, \dots, T$:

- ◆ Find $h_t = \arg \min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^L D_t(i) \llbracket y_i \neq h(\mathbf{x}_i) \rrbracket$
- ◆ If $\epsilon_t \geq 1/2$ then stop



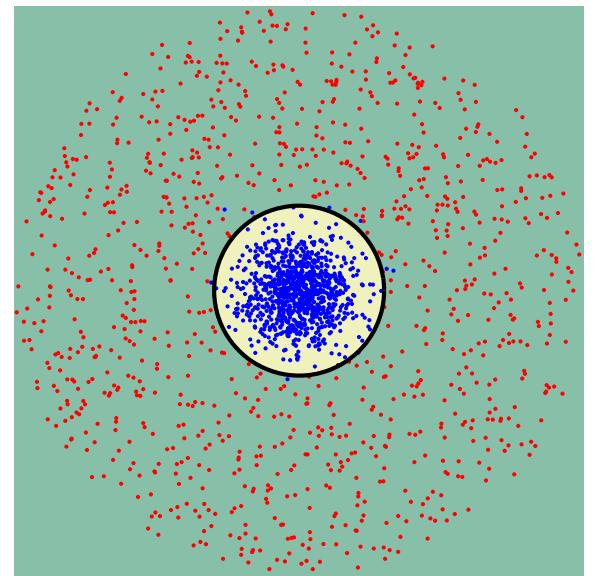
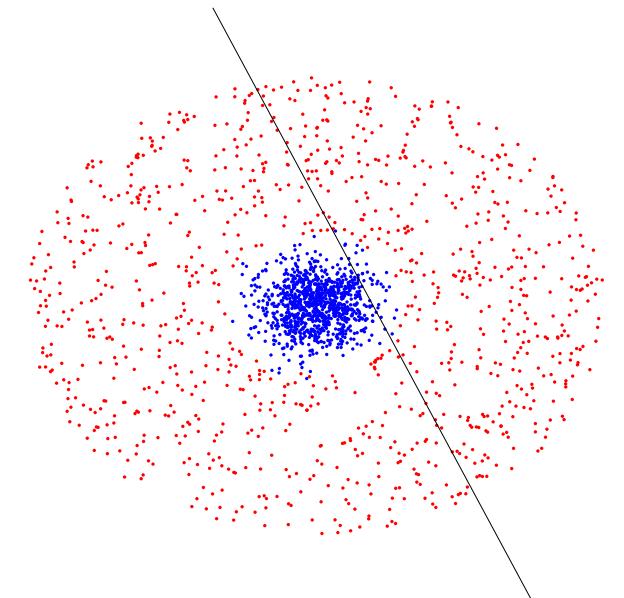
Summary of the Algorithm

Initialization ...

$t = 1$

For $t = 1, \dots, T$:

- ◆ Find $h_t = \arg \min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^L D_t(i) \llbracket y_i \neq h(\mathbf{x}_i) \rrbracket$
- ◆ If $\epsilon_t \geq 1/2$ then stop
- ◆ Set $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$



Summary of the Algorithm

Initialization ...

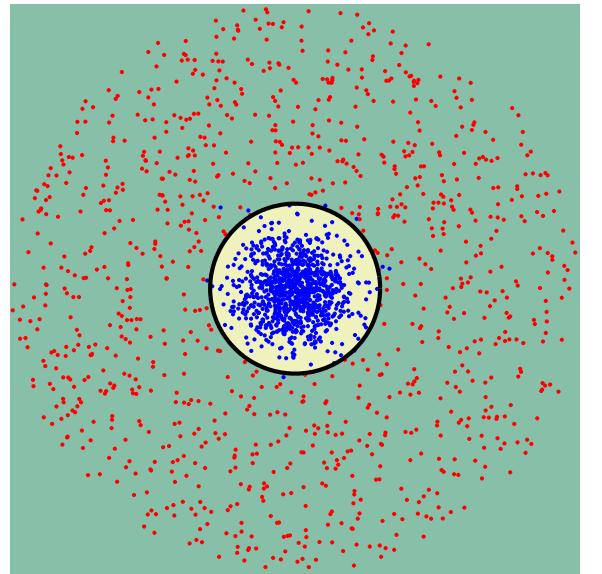
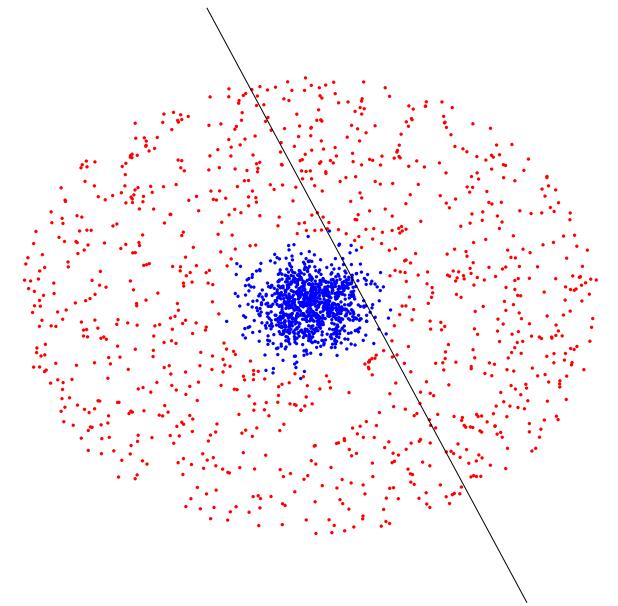
$t = 1$

For $t = 1, \dots, T$:

- ◆ Find $h_t = \arg \min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^L D_t(i) \llbracket y_i \neq h(\mathbf{x}_i) \rrbracket$
- ◆ If $\epsilon_t \geq 1/2$ then stop
- ◆ Set $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
- ◆ Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

$$Z_t = \sum_{i=1}^L D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$



Summary of the Algorithm

Initialization ...

$t = 1$

For $t = 1, \dots, T$:

- ◆ Find $h_t = \arg \min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^L D_t(i) \llbracket y_i \neq h(\mathbf{x}_i) \rrbracket$

- ◆ If $\epsilon_t \geq 1/2$ then stop

- ◆ Set $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

- ◆ Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

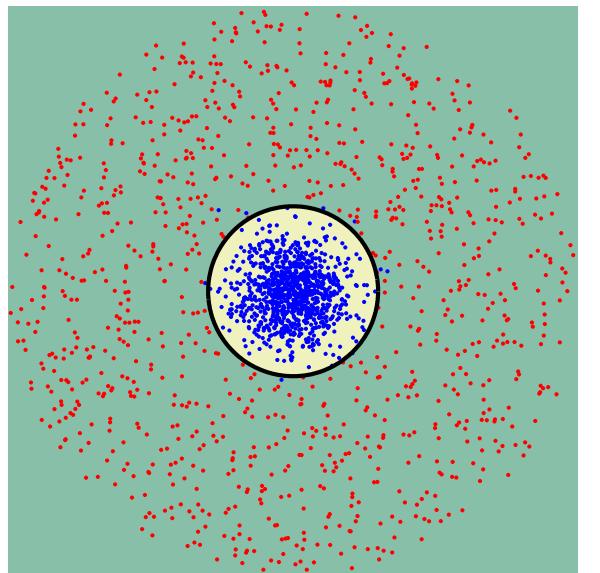
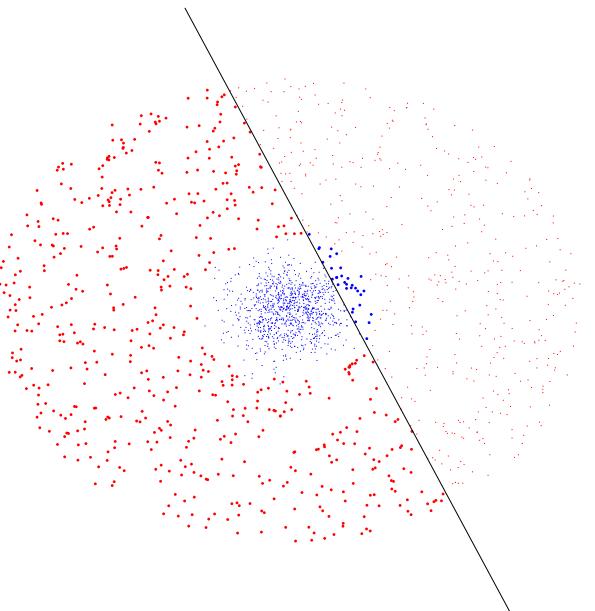
$$Z_t = \sum_{i=1}^L D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$$

Comments

- ◆ The computational complexity of selecting h_t is independent of t
- ◆ All information about previously selected “features” is captured in D_t



Summary of the Algorithm

Initialization ...

$t = 1$

For $t = 1, \dots, T$:

- ◆ Find $h_t = \arg \min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^L D_t(i) \llbracket y_i \neq h(\mathbf{x}_i) \rrbracket$

- ◆ If $\epsilon_t \geq 1/2$ then stop

- ◆ Set $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

- ◆ Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

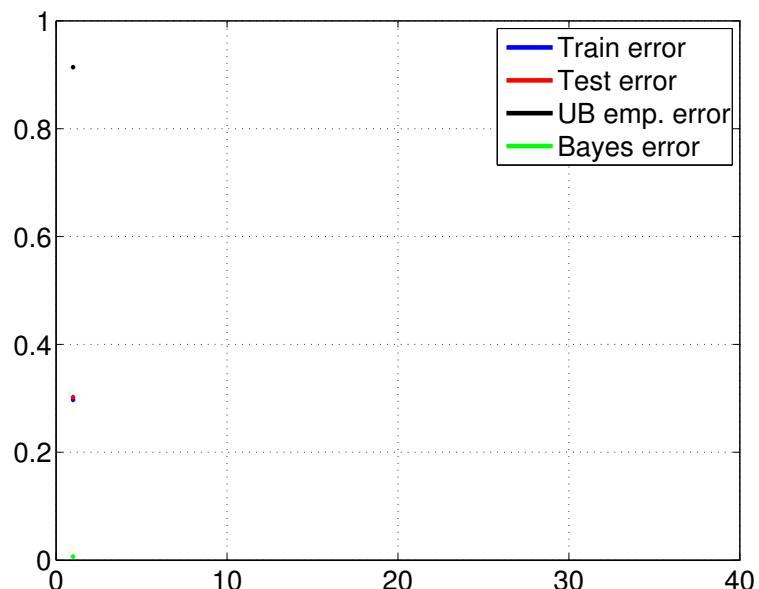
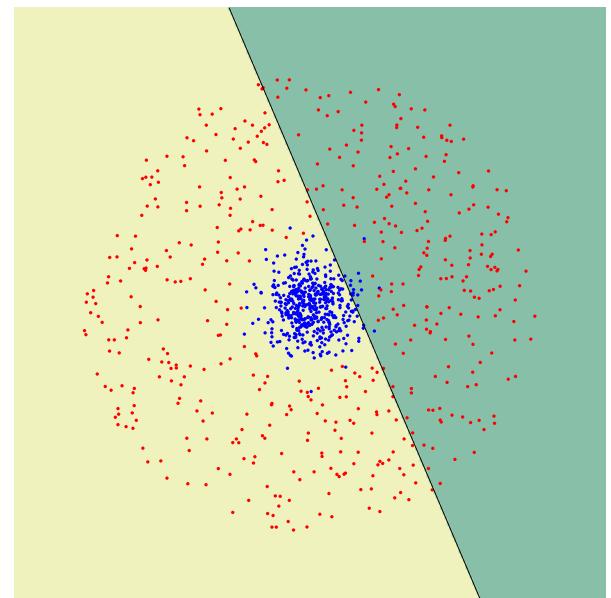
$$Z_t = \sum_{i=1}^L D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$$

Comments

- ◆ The computational complexity of selecting h_t is independent of t
- ◆ All information about previously selected “features” is captured in D_t



Summary of the Algorithm

Initialization ...

$t = 2$

For $t = 1, \dots, T$:

- ◆ Find $h_t = \arg \min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^L D_t(i) \llbracket y_i \neq h(\mathbf{x}_i) \rrbracket$

- ◆ If $\epsilon_t \geq 1/2$ then stop

- ◆ Set $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

- ◆ Update
$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

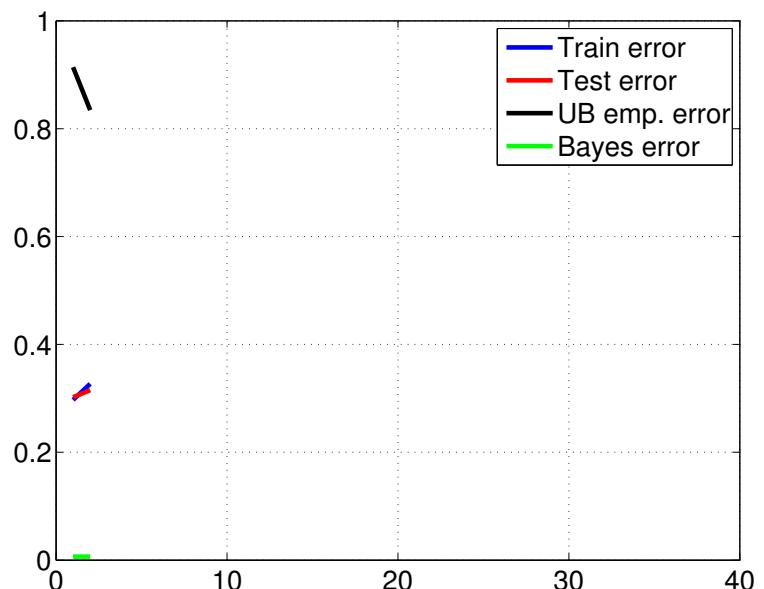
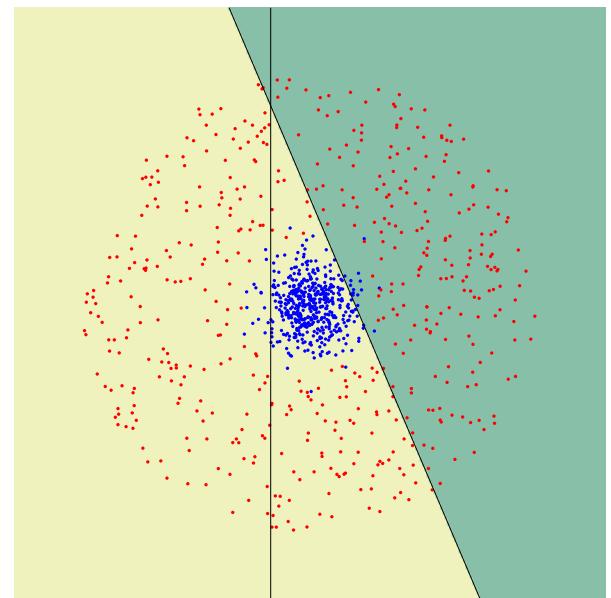
$$Z_t = \sum_{i=1}^L D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$$

Comments

- ◆ The computational complexity of selecting h_t is independent of t
- ◆ All information about previously selected “features” is captured in D_t



Summary of the Algorithm

Initialization ...

$t = 3$

For $t = 1, \dots, T$:

- ◆ Find $h_t = \arg \min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^L D_t(i) \llbracket y_i \neq h(\mathbf{x}_i) \rrbracket$

- ◆ If $\epsilon_t \geq 1/2$ then stop

- ◆ Set $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

- ◆ Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

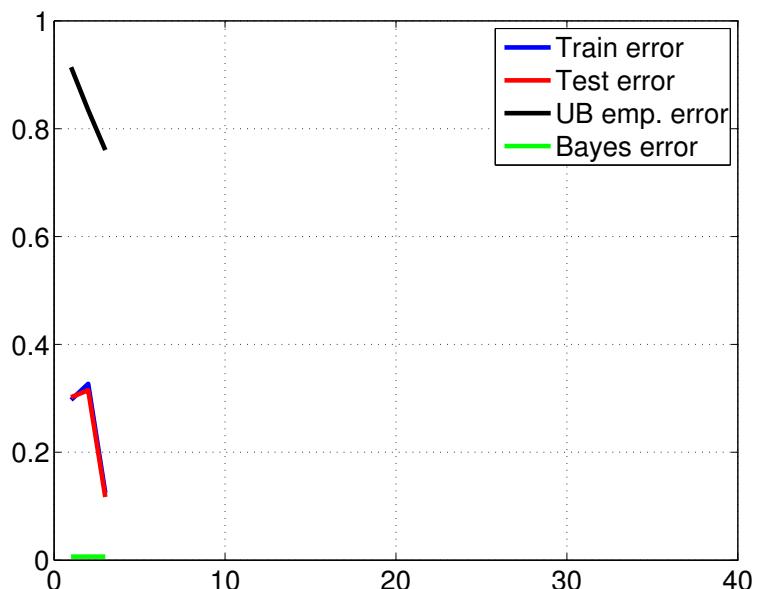
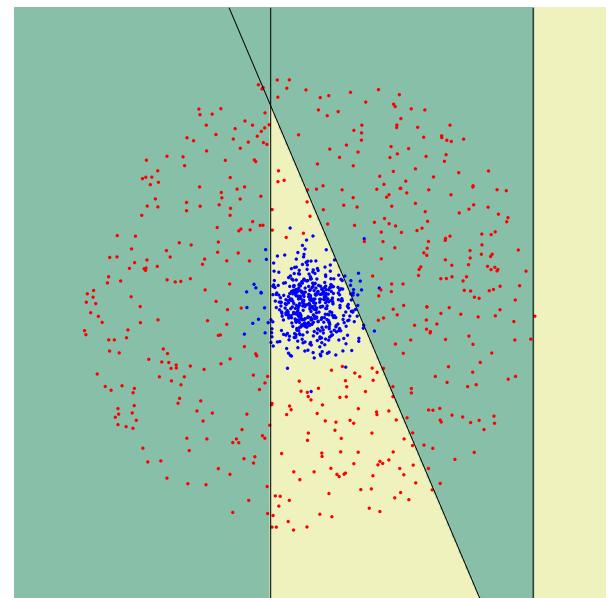
$$Z_t = \sum_{i=1}^L D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$$

Comments

- ◆ The computational complexity of selecting h_t is independent of t
- ◆ All information about previously selected “features” is captured in D_t



Summary of the Algorithm

Initialization ...

$t = 4$

For $t = 1, \dots, T$:

- ◆ Find $h_t = \arg \min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^L D_t(i) \llbracket y_i \neq h(\mathbf{x}_i) \rrbracket$

- ◆ If $\epsilon_t \geq 1/2$ then stop

- ◆ Set $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

- ◆ Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

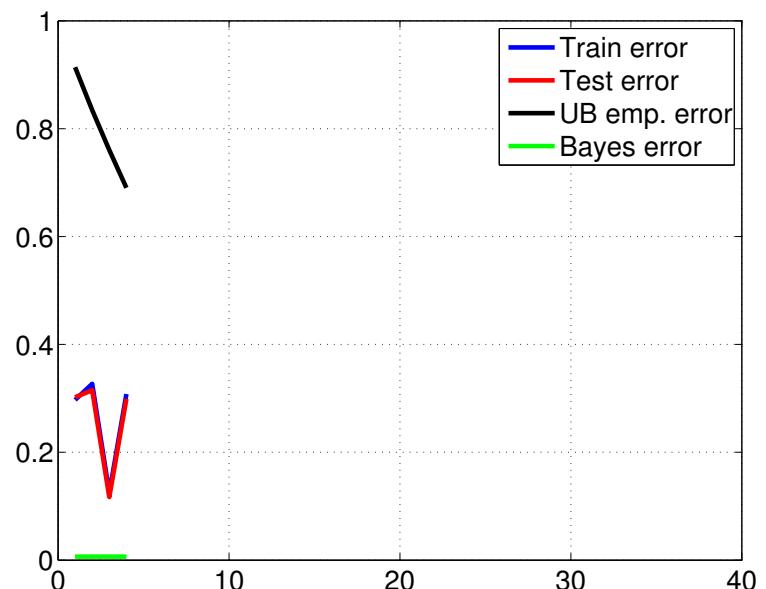
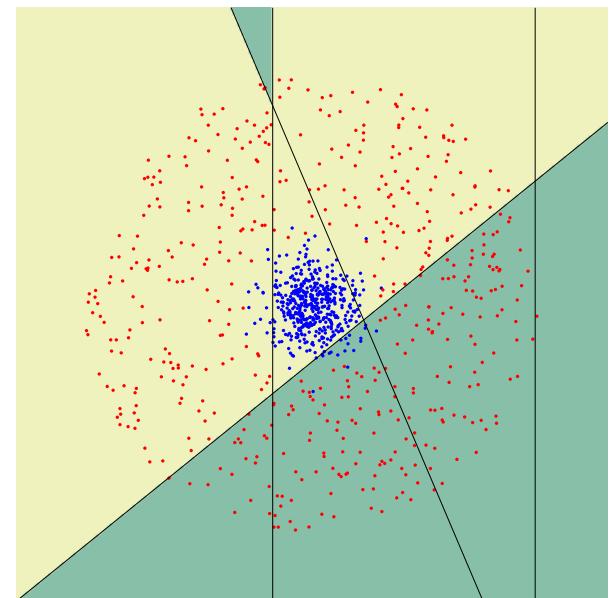
$$Z_t = \sum_{i=1}^L D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$$

Comments

- ◆ The computational complexity of selecting h_t is independent of t
- ◆ All information about previously selected “features” is captured in D_t



Summary of the Algorithm

Initialization ...

$t = 5$

For $t = 1, \dots, T$:

- ◆ Find $h_t = \arg \min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^L D_t(i) \llbracket y_i \neq h(\mathbf{x}_i) \rrbracket$

- ◆ If $\epsilon_t \geq 1/2$ then stop

- ◆ Set $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

- ◆ Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

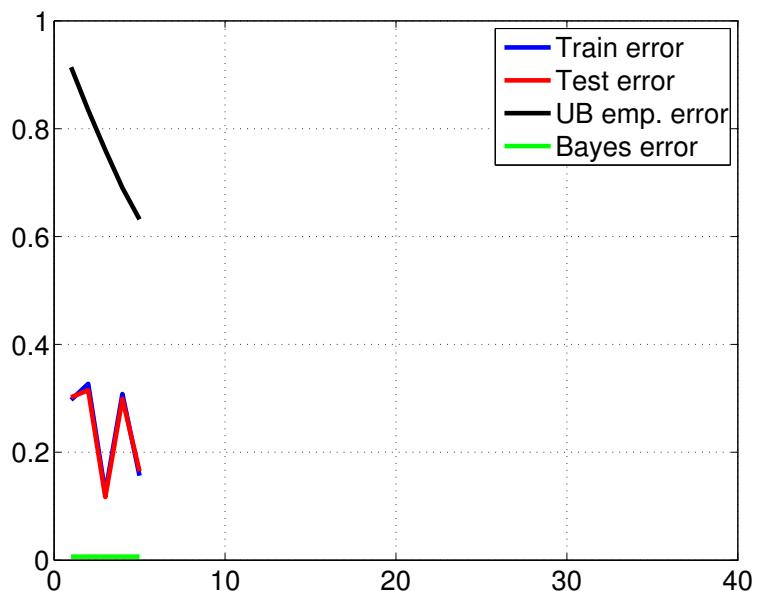
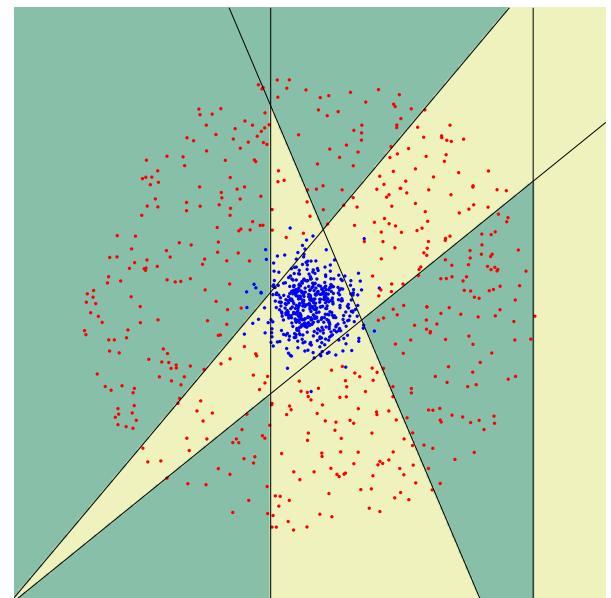
$$Z_t = \sum_{i=1}^L D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$$

Comments

- ◆ The computational complexity of selecting h_t is independent of t
- ◆ All information about previously selected “features” is captured in D_t



Summary of the Algorithm

Initialization ...

$t = 6$

For $t = 1, \dots, T$:

- ◆ Find $h_t = \arg \min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^L D_t(i) \llbracket y_i \neq h(\mathbf{x}_i) \rrbracket$

- ◆ If $\epsilon_t \geq 1/2$ then stop

- ◆ Set $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

- ◆ Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

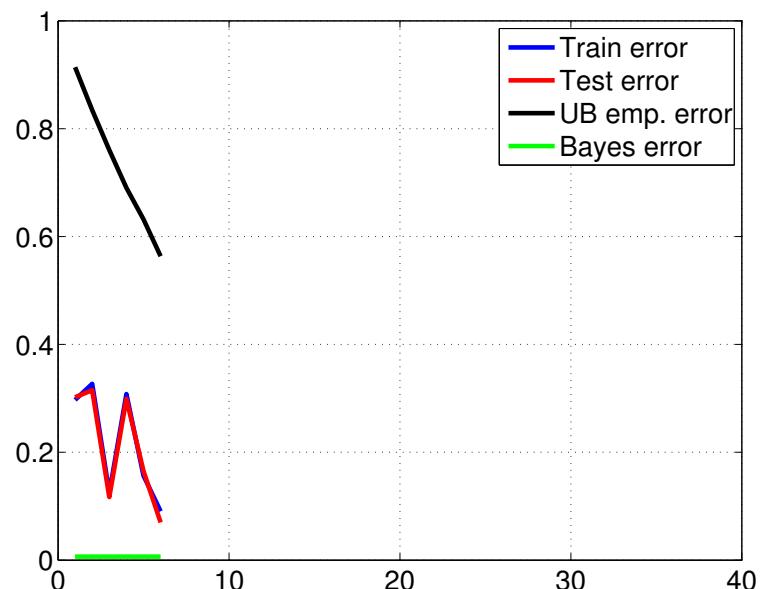
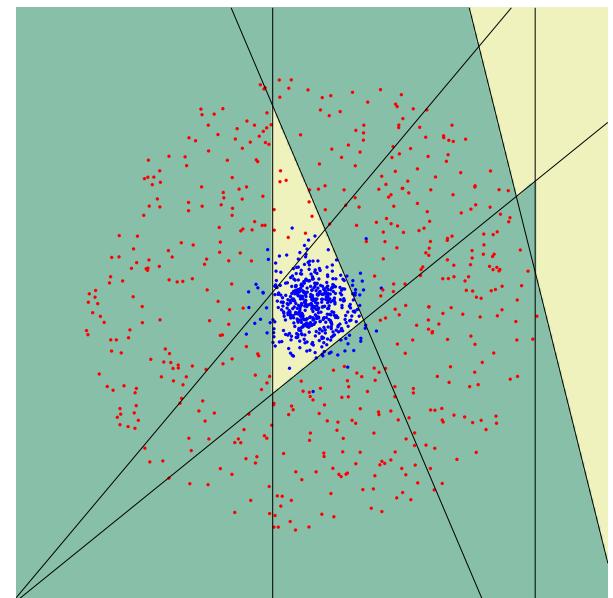
$$Z_t = \sum_{i=1}^L D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$$

Comments

- ◆ The computational complexity of selecting h_t is independent of t
- ◆ All information about previously selected “features” is captured in D_t



Summary of the Algorithm

Initialization ...

$t = 7$

For $t = 1, \dots, T$:

- ◆ Find $h_t = \arg \min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^L D_t(i) \llbracket y_i \neq h(\mathbf{x}_i) \rrbracket$

- ◆ If $\epsilon_t \geq 1/2$ then stop

- ◆ Set $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

- ◆ Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

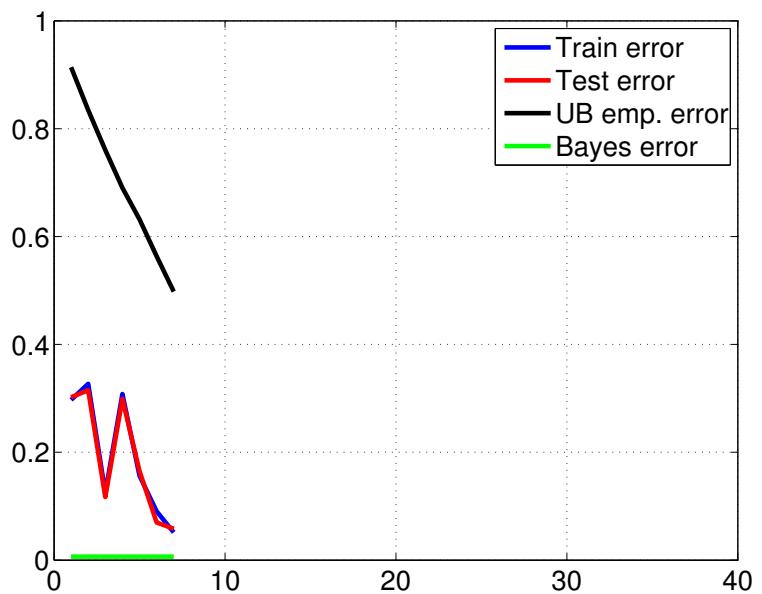
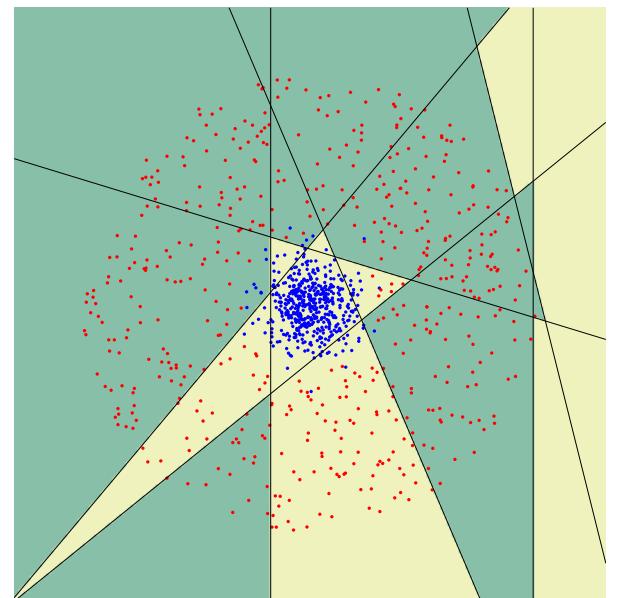
$$Z_t = \sum_{i=1}^L D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$$

Comments

- ◆ The computational complexity of selecting h_t is independent of t
- ◆ All information about previously selected “features” is captured in D_t



Summary of the Algorithm

Initialization ...

$t = 40$

For $t = 1, \dots, T$:

- ◆ Find $h_t = \arg \min_{h \in \mathcal{B}} \epsilon_t$; $\epsilon_j = \sum_{i=1}^L D_t(i) \llbracket y_i \neq h(\mathbf{x}_i) \rrbracket$

- ◆ If $\epsilon_t \geq 1/2$ then stop

- ◆ Set $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

- ◆ Update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$$

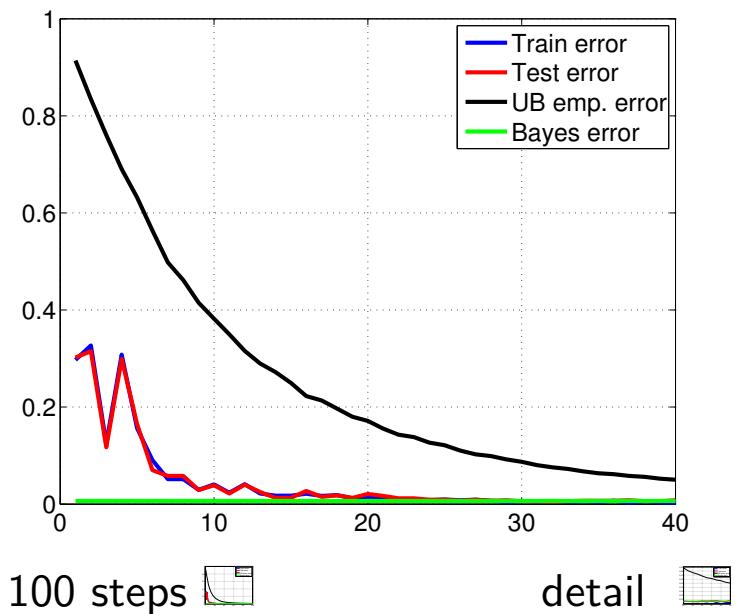
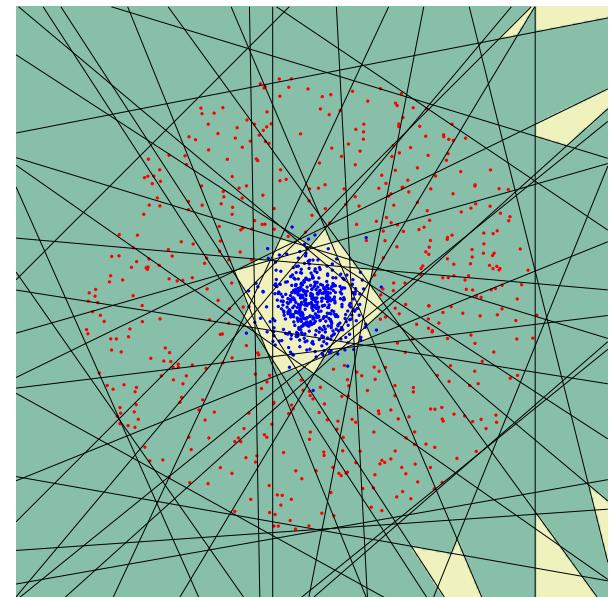
$$Z_t = \sum_{i=1}^L D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Output the final classifier:

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}) \right)$$

Comments

- ◆ The computational complexity of selecting h_t is independent of t
- ◆ All information about previously selected “features” is captured in D_t



Does AdaBoost generalise?

Margins in SVM

$$\max \min_{(\mathbf{x}, y) \in \mathcal{T}} \frac{y(\vec{\alpha} \cdot \vec{h}(\mathbf{x}))}{\|\vec{\alpha}\|_2}$$

Margins in AdaBoost

$$\max \min_{(\mathbf{x}, y) \in \mathcal{T}} \frac{y(\vec{\alpha} \cdot \vec{h}(\mathbf{x}))}{\|\vec{\alpha}\|_1}$$

Maximising margins in AdaBoost

$$P_S[yf(\mathbf{x}) \leq \theta] \leq 2^T \prod_{t=1}^T \sqrt{\epsilon_t^{1-\theta} (1 - \epsilon_t)^{1+\theta}} \quad \text{where } f(\mathbf{x}) = \frac{\vec{\alpha} \cdot \vec{h}(\mathbf{x})}{\|\vec{\alpha}\|_1}$$

Upper bounds based on margin

$$P_{\mathcal{D}}[yf(\mathbf{x}) \leq 0] \leq P_S[yf(\mathbf{x}) \leq \theta] + \mathcal{O} \left(\frac{1}{\sqrt{L}} \left(\frac{d \log^2(L/d)}{\theta^2} + \log(1/\delta) \right)^{1/2} \right)$$

Pros and cons of AdaBoost

Advantages

- ◆ Very simple to implement
- ◆ Feature selection on very large sets of features
- ◆ Fairly good generalisation
- ◆ Linear classifier with all its desirable properties.
- ◆ Output converges to the logarithm of likelihood ratio.
- ◆ Feature selector with a principled strategy (minimisation of upper bound on empirical error)
- ◆ Close to sequential decision making (it produces a sequence of gradually more complex classifiers).

Disadvantages

- ◆ Suboptimal solution for $\vec{\alpha}$
- ◆ Can overfit in the presence of noise

AdaBoost variants

Freund & Schapire 1995

- ◆ Discrete ($h : \mathcal{X} \rightarrow \{0, 1\}$)
- ◆ Multiclass AdaBoost.M1 ($h : \mathcal{X} \rightarrow \{0, 1, \dots, k\}$)
- ◆ Multiclass AdaBoost.M2 ($h : \mathcal{X} \rightarrow [0, 1]^k$)
- ◆ Real valued AdaBoost.R ($Y = [0, 1], h : \mathcal{X} \rightarrow [0, 1]$)

Schapire & Singer 1997

- ◆ Confidence rated prediction ($h : \mathcal{X} \rightarrow R$, two-class)
- ◆ Multilabel AdaBoost.MR, AdaBoost.MH (different formulation of minimised loss)

... Many other modifications since then (WaldBoost, cascaded AB, online AB, ...)

Neural Networks

lecturer: Jiří Matas, matas@cmp.felk.cvut.cz

authors: O. Drbohlav, J. Matas, D. Mishkin

Czech Technical University, Faculty of Electrical Engineering
Department of Cybernetics, Center for Machine Perception
121 35 Praha 2, Karlovo nám. 13, Czech Republic

<http://cmp.felk.cvut.cz>

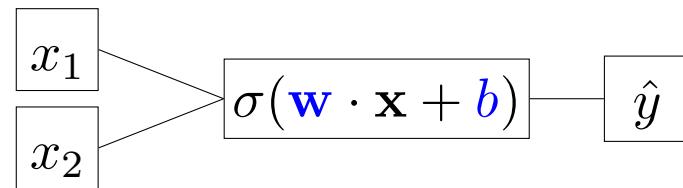
Last update: Nov 2020

Neural Networks - Motivation (1)

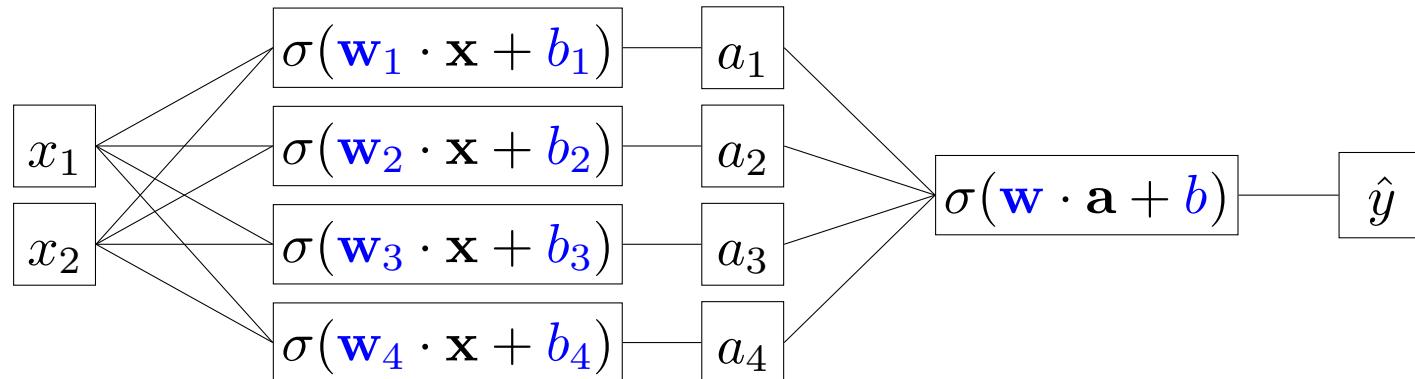
Recall the Perceptron: given the perceptron parameters $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$, the classification \hat{y} to two classes $\{-1, 1\}$ for a vector $\mathbf{x} \in \mathbb{R}^n$ is performed as

$$\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b), \quad (1)$$

which is an affine (often called linear) function $l(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, followed by a non-linear function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$, $\sigma(z) = \text{sign}(z)$. The perceptron is also often depicted as follows:



to make the linear combination of elements of the input vector explicit. Perceptron is a linear classifier; these are well understood, have low VC dimension, etc. and thus it is a natural next step to combine them to a more complex classifiers: By letting more of them sharing the input, as well as using their outputs a_i 's as inputs to other perceptrons:



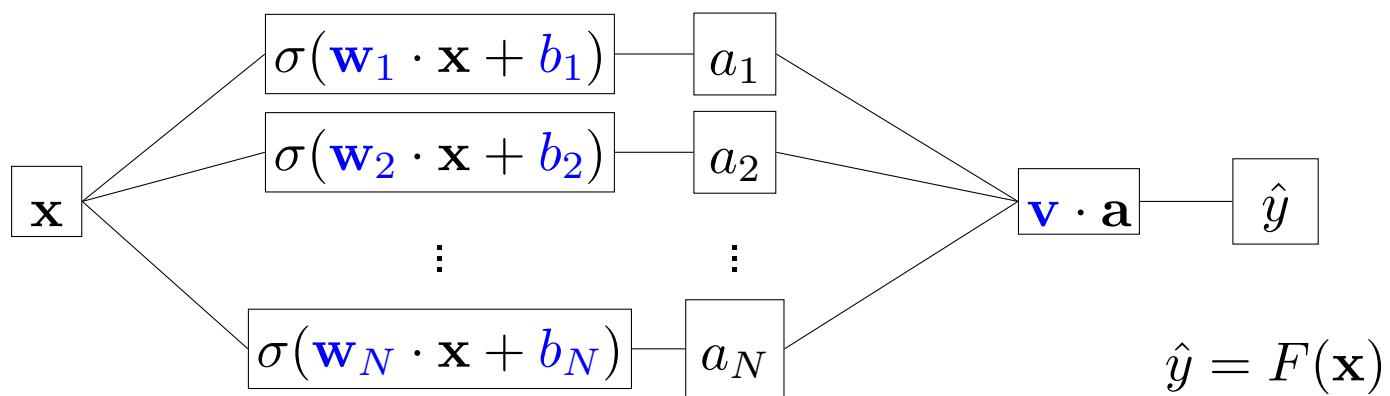
Neural Networks - Motivation (2)

Universal Approximation Theorem. Another strong motivation for forming such combinations of simple classifiers is the theorem which states that if $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is a nonconstant, bounded and continuous function and f is a continuous function on unit hypercube $[0, 1]^n$ then for any $\epsilon > 0$ there exists $N \in \mathbb{N}$, $v_i, b_i \in \mathbb{R}$ and $\mathbf{w}_i \in \mathbb{R}^n$ such that

$$F(\mathbf{x}) = \sum_{i=1}^N v_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i), \text{ and} \quad (2)$$

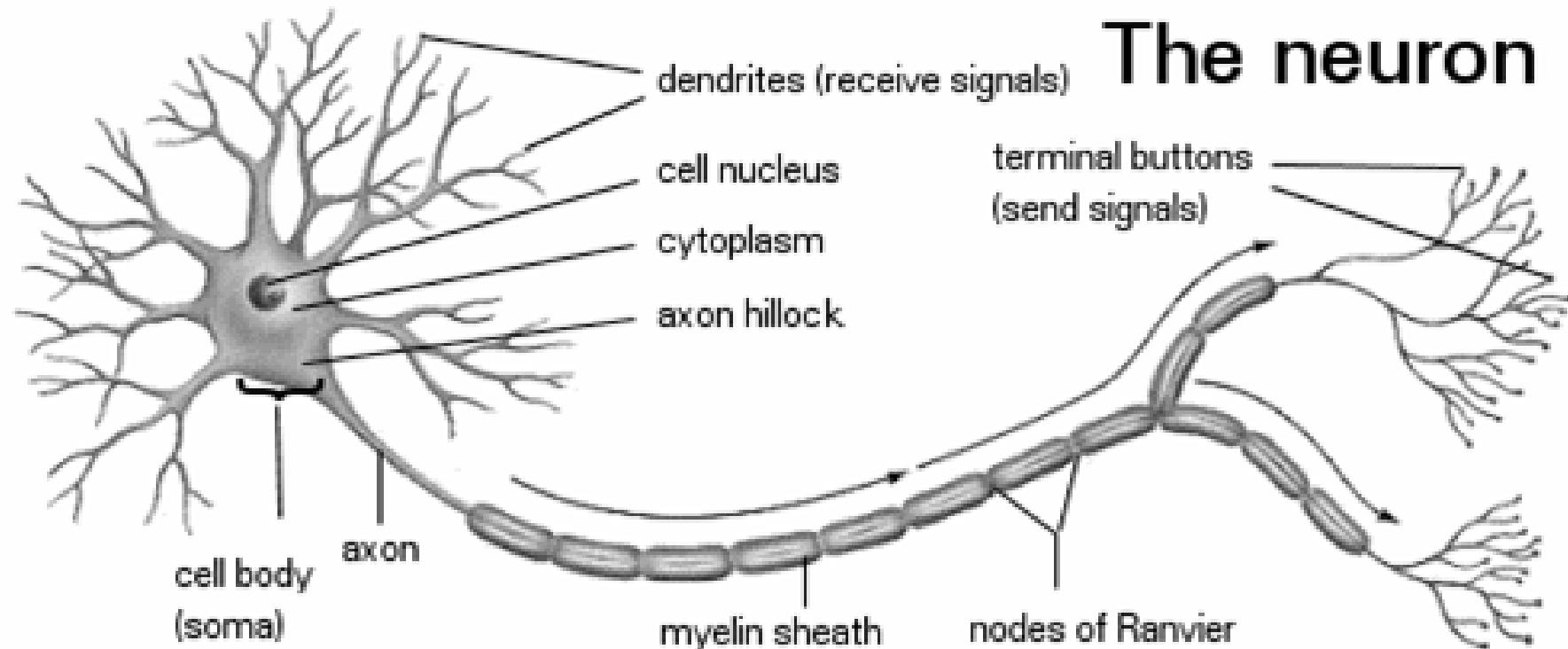
$$|F(\mathbf{x}) - f(\mathbf{x})| < \epsilon, \quad \forall \mathbf{x} \in [0, 1]^m \quad (3)$$

By comparison, we see that the approximation is exactly captured by the following network with single hidden layer and linear output:



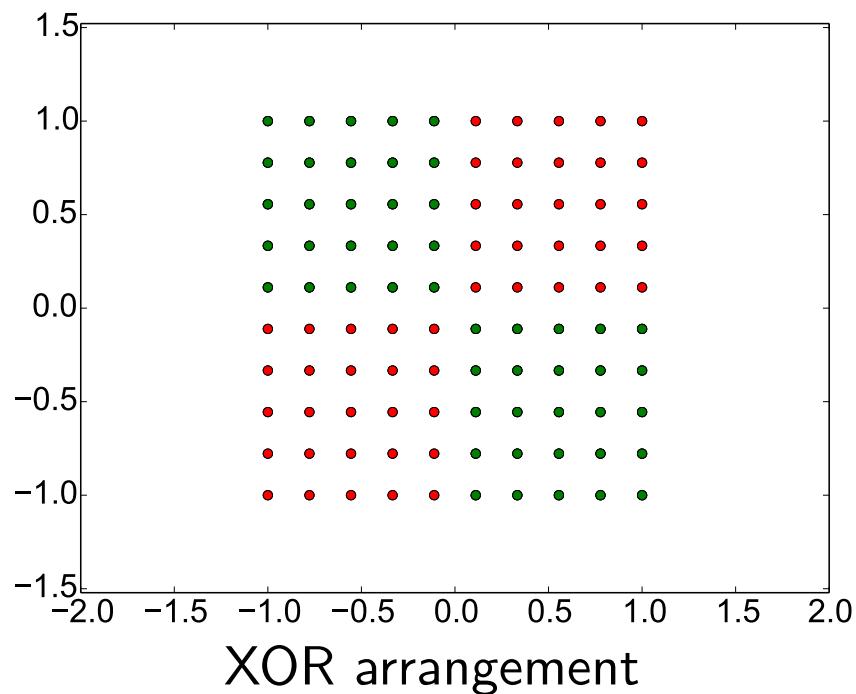
Neural Networks - Motivation (3)

'Biological' motivation - a real neuron is known to combine inputs to an output which is then passed to other neurons.



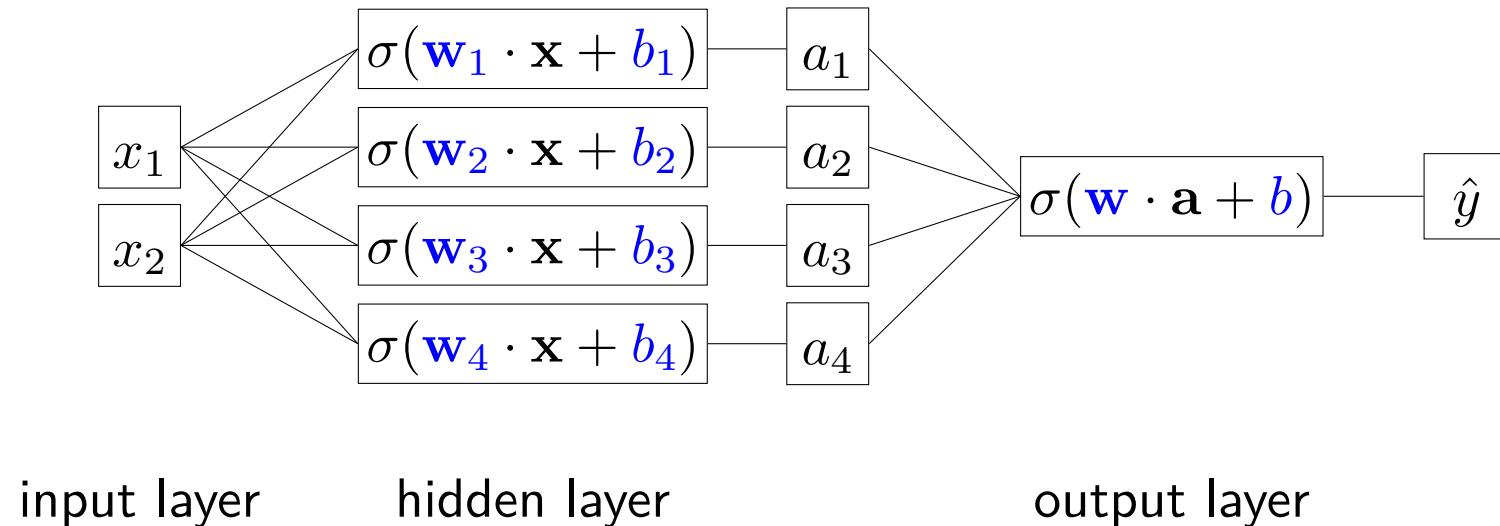
Historical Note

- ◆ Perceptron (Rosenblatt, 1956) with its simple learning algorithm generated a lot of excitement
- ◆ Minsky and Papert (1969) showed that even a simple XOR cannot be learnt by a perceptron
- ◆ But chaining perceptrons to a network (Multi-Layer Perceptron, MLP) *can* learn XOR



Layers, Concise Equivalent Representation

The network



can be rewritten in a more concise form as follows:

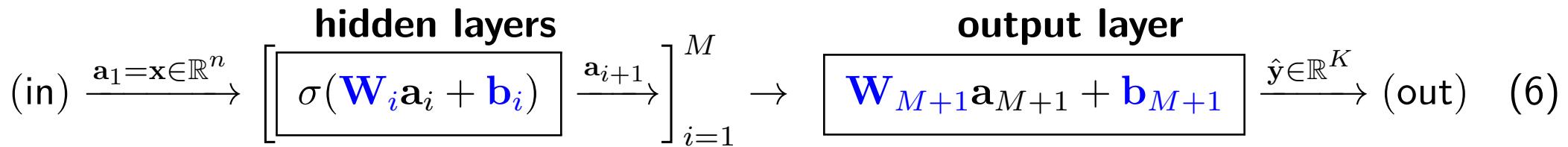
$$(in) \xrightarrow{x \in \mathbb{R}^2} \boxed{\sigma(\mathbf{W}x + \mathbf{b})} \xrightarrow{a \in \mathbb{R}^4} \boxed{\sigma(\mathbf{w} \cdot \mathbf{a} + b)} \xrightarrow{\hat{y} \in \mathbb{R}} (out) \quad (4)$$

where we introduced a convention that for $\sigma: \mathbb{R} \rightarrow \mathbb{R}$ and $\mathbf{z} \in \mathbb{R}^k$, $\sigma(\mathbf{z}) = [\sigma(z_1), \sigma(z_2), \dots, \sigma(z_k)]^\top$, i.e. the function is applied per component of the vector \mathbf{z} .

A network with M hidden layers can be written as (here shown with affine output layer)

$$(in) \xrightarrow{a_1=x \in \mathbb{R}^n} \left[\boxed{\sigma(\mathbf{W}_i \mathbf{a}_i + \mathbf{b}_i)} \xrightarrow{a_{i+1}} \right]_{i=1}^M \rightarrow \boxed{\mathbf{W}_{M+1} \mathbf{a}_{M+1} + \mathbf{b}_{M+1}} \xrightarrow{\hat{y} \in \mathbb{R}^K} (out) \quad (5)$$

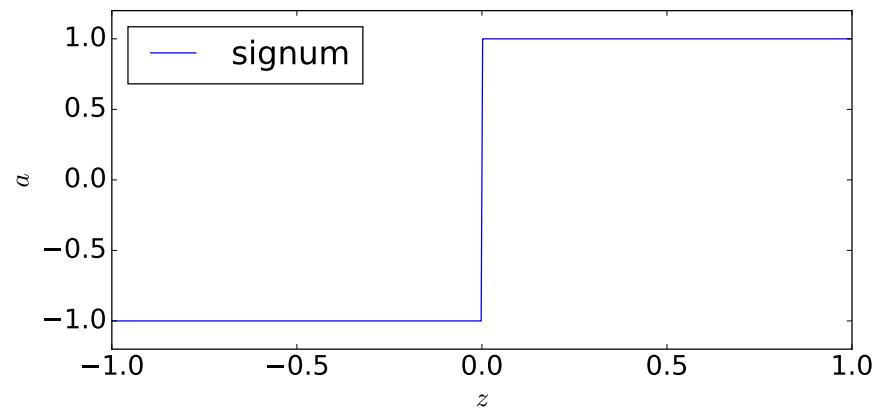
Layer Anatomy



Each hidden layer of an NN is composed of an affine function, followed by a non-linearity.

Non-linear functions

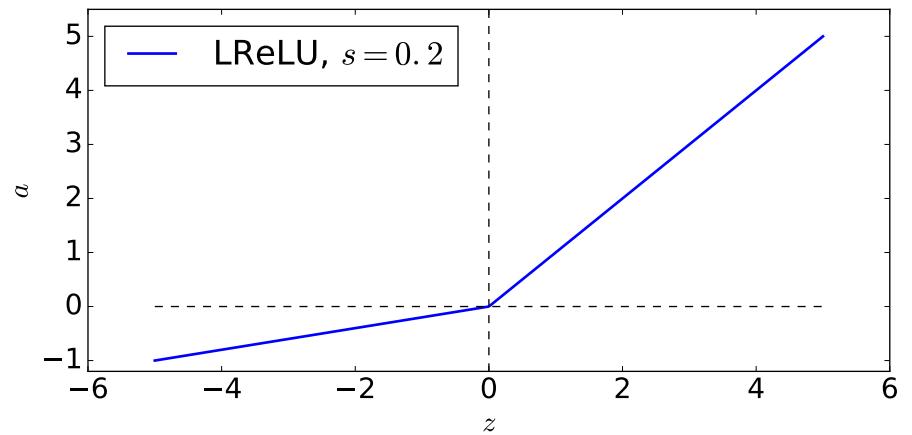
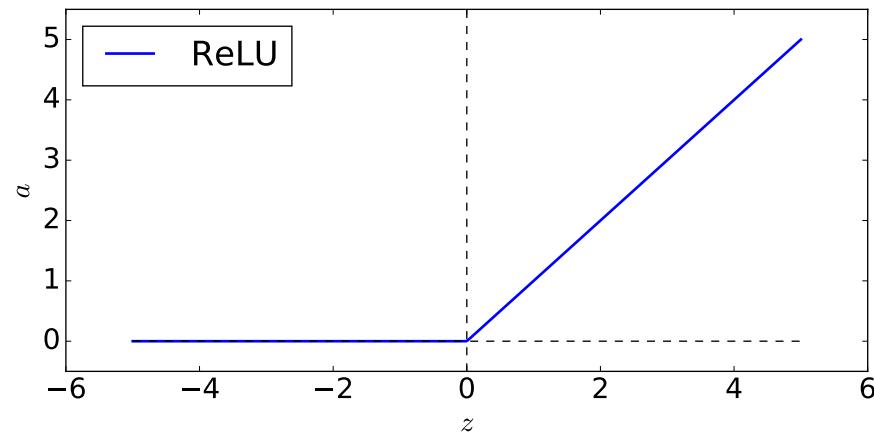
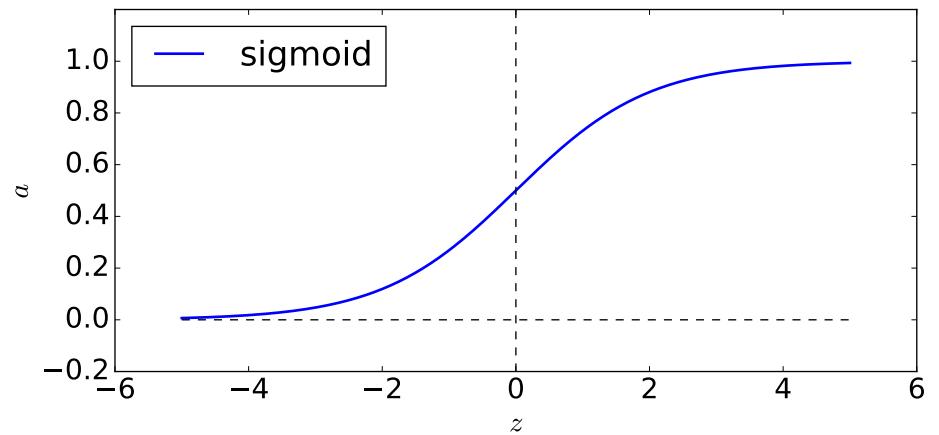
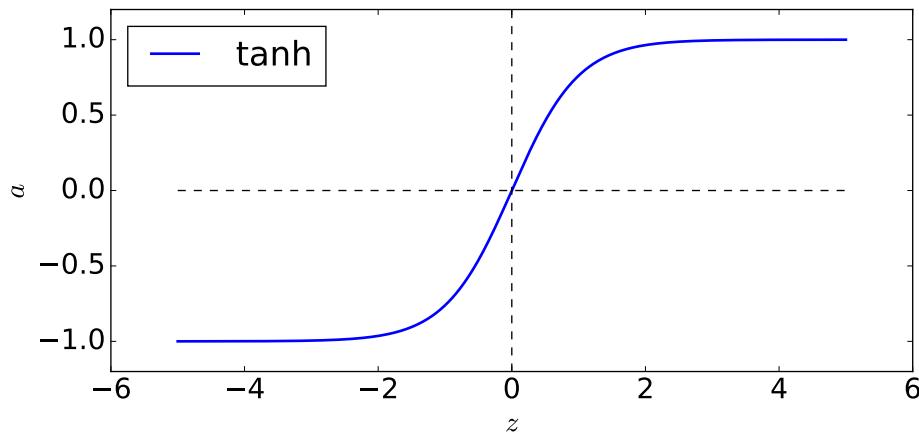
- ◆ $\sigma(z) = \text{sign}(z)$: used for the original perceptron. Unusable for an NN because this non-linear function is discontinuous and not differentiable at 0, and everywhere else has zero gradient. So, once we would like to optimize the parameters of the combination of perceptrons, gradient descent and other methods based on first and second order approximations could not proceed. This is why the original sign function has been replaced by functions with ‘nicer’ properties.



Layer Anatomy, Non-Linear Functions

Non-linear functions

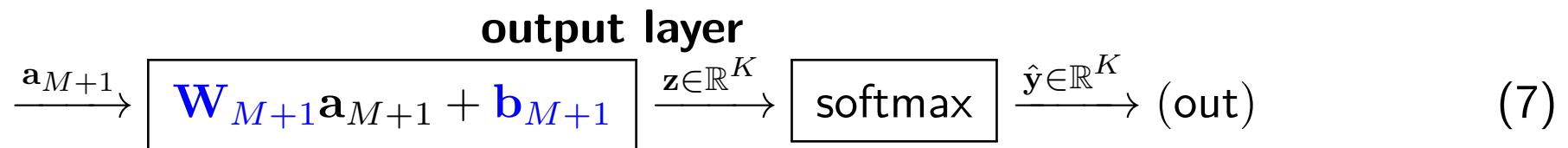
- ◆ $\sigma(z) = 1/(1 + e^{-z})$: logistic sigmoid, $\sigma : \mathbb{R} \rightarrow [0, 1]$
- ◆ $\sigma(z) = \tanh(z) = (e^z - e^{-z})/(e^z + e^{-z})$: tanh sigmoid, $\sigma : \mathbb{R} \rightarrow [-1, 1]$
- ◆ $\sigma(z) = \max(0, z)$: ReLU (rectified linear unit)
- ◆ $\sigma(z) = \max(0, z) + \min(0, sz)$ ($0 < s < 1$): Leaky ReLU
- ◆ Many other



Output Layer and Loss Functions. K -class Classification (1)

So far, we have made a general assumption that the output of NN is a K -element vector $\hat{\mathbf{y}} \in \mathbb{R}^K$. Now we will discuss what form the output should take and how we will measure how good the output is.

K -class Classification. We may formally number the classes as $1, 2, \dots, K$. But if classes are not ordinal, these numbers are really just labels; it doesn't mean that objects from classes 2 and 3 are in some sense closer than objects from classes 1 and K , say. Therefore, it would make no sense to have a one-dimensional output \hat{y} trying to predict the class label. Instead, the output layer will consist of an affine function producing a K -dimensional vector, followed by the **softmax** which will convert it to class probabilities:



with

$$[\text{softmax}(\mathbf{z})]_k = \frac{\exp z_k}{\sum_{l=1}^K \exp z_l}. \quad (8)$$

For representing the class y of the training data point (\mathbf{x}, y) , **one-hot** representation is used which simply makes a K -dimensional vector which is everywhere zero except for the y -th component which is 1:

$$\text{onehot}(y) = [\delta_{1y}, \delta_{2y}, \dots, \delta_{Ky}]^\top = [0, 0, \dots, \underset{\substack{\uparrow \\ \text{y-th place}}}{1}, \dots, 0]^\top \in \mathbb{R}^K \quad (9)$$

Output Layer and Loss Functions. K -class Classification (2)

For a training data point (\mathbf{x}, y) , how to measure how far the prediction $\hat{\mathbf{y}} \in [0, 1]^K$ for \mathbf{x} is from the target vector $\mathbf{y} = \text{onehot}(y) \in \{0, 1\}^K$?

- ◆ Squared difference:

$$J(\hat{\mathbf{y}}, \mathbf{y}) = \|\hat{\mathbf{y}} - \mathbf{y}\|^2. \quad (10)$$

This loss = 0 when the prediction matches the target and > 0 otherwise.

- ◆ Negative log-likelihood. To avoid confusion, let us denote the index of the target class l for a training data point (\mathbf{x}, l) ; then

$$J(\hat{\mathbf{y}}, \mathbf{y}) = -\mathbf{y}^\top \log \hat{\mathbf{y}} = -\sum_{i=1}^K y_i \log \hat{y}_i = -\log \hat{y}_l. \quad (11)$$

Example: K -class logistic regression. The class conditionals

$\hat{\mathbf{y}}(\mathbf{x}) = [p(1|\mathbf{x}), p(2|\mathbf{x}), \dots, p(K|\mathbf{x})] \in \mathbb{R}^K$ can be written as ($\mathbf{W} \in \mathbb{R}^{n \times K}$, $\mathbf{b} \in \mathbb{R}^K$):



Output Layer and Loss Functions. 2-class Classification

For two classes, the output \hat{y} of the NN can be 1-dimensional, modeling the class posterior $p(1|\mathbf{x}) \in [0, 1]$. The posterior $p(2|\mathbf{x})$ is computed simply as $1 - \hat{y}$.

Example: 2-class logistic regression. The posteriors $p(1|\mathbf{x})$, $p(2|\mathbf{x})$ are modeled as $(\mathbf{x}, \mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R})$:

$$p(1|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{x} + b)}}, \quad p(2|\mathbf{x}) = 1 - p(1|\mathbf{x}) \quad (13)$$

This can be written as a single-neuron NN with output $\hat{y} = p(1|\mathbf{x})$:

$$\text{(in)} \xrightarrow{\mathbf{x}} \boxed{\sigma(\mathbf{w} \cdot \mathbf{x} + b)} \xrightarrow{\hat{y}} \text{(out)} \quad (14)$$

where $\sigma(z) = 1/(1 + e^{-z})$ is the logistic sigmoid non-linearity.

Recall that negative log-likelihood loss has been used for the 2-class logistic regression:

$$J(\hat{y}, y) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})] \quad (15)$$

where class labels y have conveniently been changed as $(1, 2) \mapsto (1, 0)$. The procedure for finding the optimal parameters $\boldsymbol{\theta} = \{\mathbf{w}, b\}$ was the gradient descent.

Output Layer and Loss Functions. Regression

The NN can of course also be used for regression, that is, finding a function which predicts a target $\mathbf{y} \in \mathbb{R}^K$ which is not constrained to represent probability distribution.

Squared difference is an obvious choice for such a problem.

Training the NN

Let the training data be $\mathcal{T} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}$, where without loss of generality we assume that class labels y_i 's have been converted to their one-hot representations \mathbf{y}_i 's if needed. Let $\boldsymbol{\theta}$ denote all parameters of the NN. We want to minimize

$$J(\mathcal{T}; \boldsymbol{\theta}) = \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{T}} J(\hat{\mathbf{y}}(\mathbf{x}), \mathbf{y}). \quad (16)$$

Gradient-based methods are used most of the time for minimizing the loss function $J(\mathcal{T}; \boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$. We need to evaluate the gradient of loss w.r.t. the NN parameters, $\frac{\partial J(\hat{\mathbf{y}}(\mathbf{x}), \mathbf{y})}{\partial \boldsymbol{\theta}}$, in order to use it for updates of the gradient-descent type:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \mu \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{T}' \subseteq \mathcal{T}} \frac{\partial J(\hat{\mathbf{y}}(\mathbf{x}), \mathbf{y})}{\partial \boldsymbol{\theta}} \quad (17)$$

where μ is the learning rate and the summation is not necessarily over the entire dataset.

Computing gradient (1)

When computing the gradient, we will make use of the chain rule. Let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ and $g: \mathbb{R}^m \rightarrow \mathbb{R}^n$. For $\mathbf{x} \in \mathbb{R}^m$, let $\mathbf{y} = g(\mathbf{x})$. Let us consider the composition of these functions, $f(g(\mathbf{x})) = f(\mathbf{y})$ ($f \circ g: \mathbb{R}^m \rightarrow \mathbb{R}$). There holds

$$\frac{\partial f(g(\mathbf{x}))}{\partial x_k} = \sum_{i=1}^n \frac{\partial f}{\partial y_i} \frac{\partial y_i}{\partial x_k}. \quad (18)$$

This can be written in a matrix form,

$$\frac{\partial f(g(\mathbf{x}))}{\partial \mathbf{x}} = \frac{\partial f}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = f' \mathbf{y}', \quad (19)$$

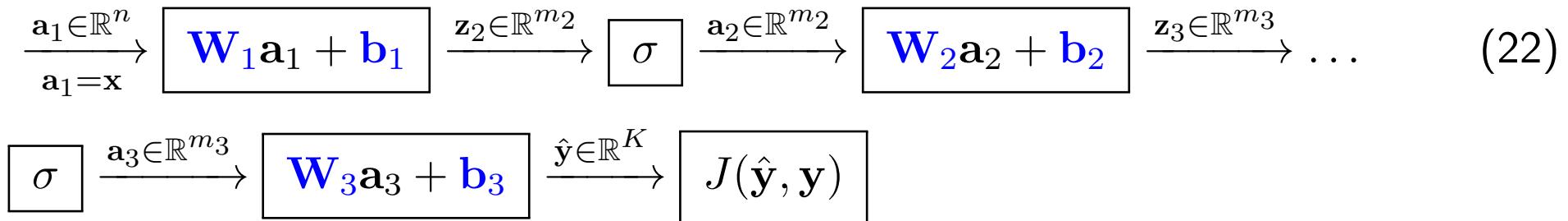
where $f' = \partial f / \partial \mathbf{y}$ and $\mathbf{y}' = \partial \mathbf{y} / \partial \mathbf{x}$ are Jacobian matrices:

$$f' = \left[\frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial y_2}, \dots, \frac{\partial f}{\partial y_n} \right], \quad (20)$$

$$\mathbf{y}' = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \dots & \frac{\partial y_1}{\partial x_m} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \dots & \frac{\partial y_2}{\partial x_m} \\ \vdots & & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \frac{\partial y_n}{\partial x_2} & \dots & \frac{\partial y_n}{\partial x_m} \end{bmatrix} \quad (21)$$

Example: Gradient by Back Propagation (1)

Let us have the following network:



where the non-linearities have been explicitly put to the chain. The set of NN parameters is $\theta = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$. For the loss, let us consider $J(\hat{\mathbf{y}}, \mathbf{y}) = \|\hat{\mathbf{y}} - \mathbf{y}\|^2$.

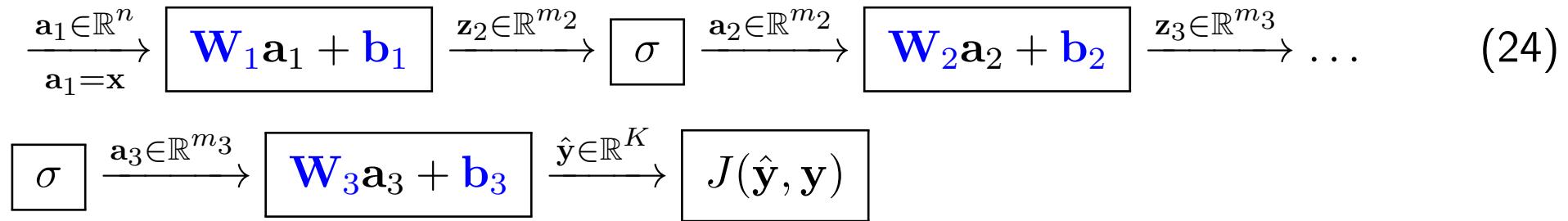
1. Compute $\mathbf{v}_3 = \frac{\partial J}{\partial \hat{\mathbf{y}}} \Big|_{\hat{\mathbf{y}}}$. This is a row vector, $\mathbf{v}_3 \in \mathbb{R}^K$, $\mathbf{v}_3 = 2(\hat{\mathbf{y}} - \mathbf{y})^\top$.
2. $\frac{\partial J}{\partial \mathbf{b}_3} = \mathbf{v}_3 \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{b}_3} \Big|_{\mathbf{a}_3} = \mathbf{v}_3 \mathbb{1} = \mathbf{v}_3$
3. $\frac{\partial J}{\partial (\mathbf{W}_3)_{kl}} = \mathbf{v}_3 \frac{\partial \hat{\mathbf{y}}}{\partial (\mathbf{W}_3)_{kl}} = \mathbf{v}_3 [0, 0, \dots, (\mathbf{a}_3)_l, \dots, 0, 0]^\top = (\mathbf{v}_3)_k (\mathbf{a}_3)_l$
 \uparrow
 at k -th element

So, arranging partial derivatives of J w.r.t. elements of $\mathbf{W}_3 \in \mathbb{R}^{K \times m_3}$ to a matrix of the same dimensions then we can write

$$\frac{\partial J}{\partial \mathbf{W}_3} = \left[\begin{array}{cccc} \frac{\partial J}{\partial (\mathbf{W}_3)_{11}} & \frac{\partial J}{\partial (\mathbf{W}_3)_{12}} & \cdots & \frac{\partial J}{\partial (\mathbf{W}_3)_{1m_1}} \\ \frac{\partial J}{\partial (\mathbf{W}_3)_{21}} & \frac{\partial J}{\partial (\mathbf{W}_3)_{22}} & \cdots & \frac{\partial J}{\partial (\mathbf{W}_3)_{2m_1}} \\ \vdots & \ddots & & \\ \frac{\partial J}{\partial (\mathbf{W}_3)_{K1}} & \frac{\partial J}{\partial (\mathbf{W}_3)_{K2}} & \cdots & \frac{\partial J}{\partial (\mathbf{W}_3)_{Km_1}} \end{array} \right] = \mathbf{v}_3^\top \mathbf{a}_3^\top \quad (23)$$

Example: Gradient by Back Propagation (2)

Let us have the following network:



4. Compute $\mathbf{v}_2 = \mathbf{v}_3 \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{a}_3} \Big|_{\mathbf{a}_3} \frac{\partial \mathbf{a}_3}{\partial \mathbf{z}_3} \Big|_{\mathbf{z}_3} = \mathbf{v}_3 \mathbf{W}_3 \text{diag}[\sigma'(\mathbf{z}_3)] = (\mathbf{v}_3 \mathbf{W}_3) \odot \sigma'(\mathbf{z}_3)$;

Here $\sigma'(\mathbf{x}) = [\sigma'(x_1), \sigma'(x_2), \dots, \sigma'(x_n)]$ with $\mathbf{x} \in \mathbb{R}^n$ and σ' the derivative of σ , $\text{diag}(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^n$ forms an n -by- n diagonal matrix with elements of \mathbf{x} on the diagonal, and \odot is the Hadamard (element-wise) product.

5. $\frac{\partial J}{\partial \mathbf{b}_2} = \mathbf{v}_2$

6. $\frac{\partial J}{\partial \mathbf{W}_2} = \mathbf{v}_2^\top \mathbf{a}_2^\top$

7. Compute $\mathbf{v}_1 = \mathbf{v}_2 \mathbf{W}_2 \text{diag}[\sigma'(\mathbf{z}_2)] = (\mathbf{v}_2 \mathbf{W}_2) \odot \sigma'(\mathbf{z}_2)$

8. $\frac{\partial J}{\partial \mathbf{b}_1} = \mathbf{v}_1$

9. $\frac{\partial J}{\partial \mathbf{W}_1} = \mathbf{v}_1^\top \mathbf{a}_1^\top$

Gradient-based Optimization, Back Propagation

- ◆ As it has been just shown, gradient for all parameters can be efficiently computed (without repeating already performed computations) by computation flow from the last layer to the first – hence the name Back Propagation (aka backprop)
- ◆ Recall the update rule:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \mu \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{T}' \subseteq \mathcal{T}} \frac{\partial J(\hat{\mathbf{y}}(\mathbf{x}), \mathbf{y})}{\partial \boldsymbol{\theta}} \quad (25)$$

- How to choose the learning rate μ ? The rate is often changed adaptively during learning, based on monitoring of the learning process
- The summation is often done over a subset \mathcal{T}' of the training data. This provides an estimate of the actual gradient and the technique is called Stochastic Gradient Descent (SGD). Many alternatives (important one: SGD with momentum)
- How to initialize the parameters? Rule of thumb (core idea, many variants): Initialize them such that variance of outputs is equal to 1 for all layers.

Deep Learning

Deep Learning

- ◆ Extremely successful branch of Machine Learning methods
- ◆ Lot of progress in recent (>10) years
- ◆ Includes:
 - Convolutional Neural Nets (CNNs): suitable for inputs which are translation-invariant and 'warpable'. Typical examples are visual signals (images)
 - Recurrent neural networks (RNNs): allow previous outputs to be used as inputs; recognition of time-series signals, speech recognition
 - Autoencoders: The goal is to output the inputs; The hidden layers have a bottleneck (fewer neurons than input layer) and thus the network is forced to learn 'compressed' representation of the input
 - . . .

Example: Convolutional Neural Networks

Consider an image (an input layer) which is 64×64 pixels large. Let the next layer be a layer of the same size. If this is modeled as a fully connected network, the number of connections is $(64^2)^2 \approx 16M$.

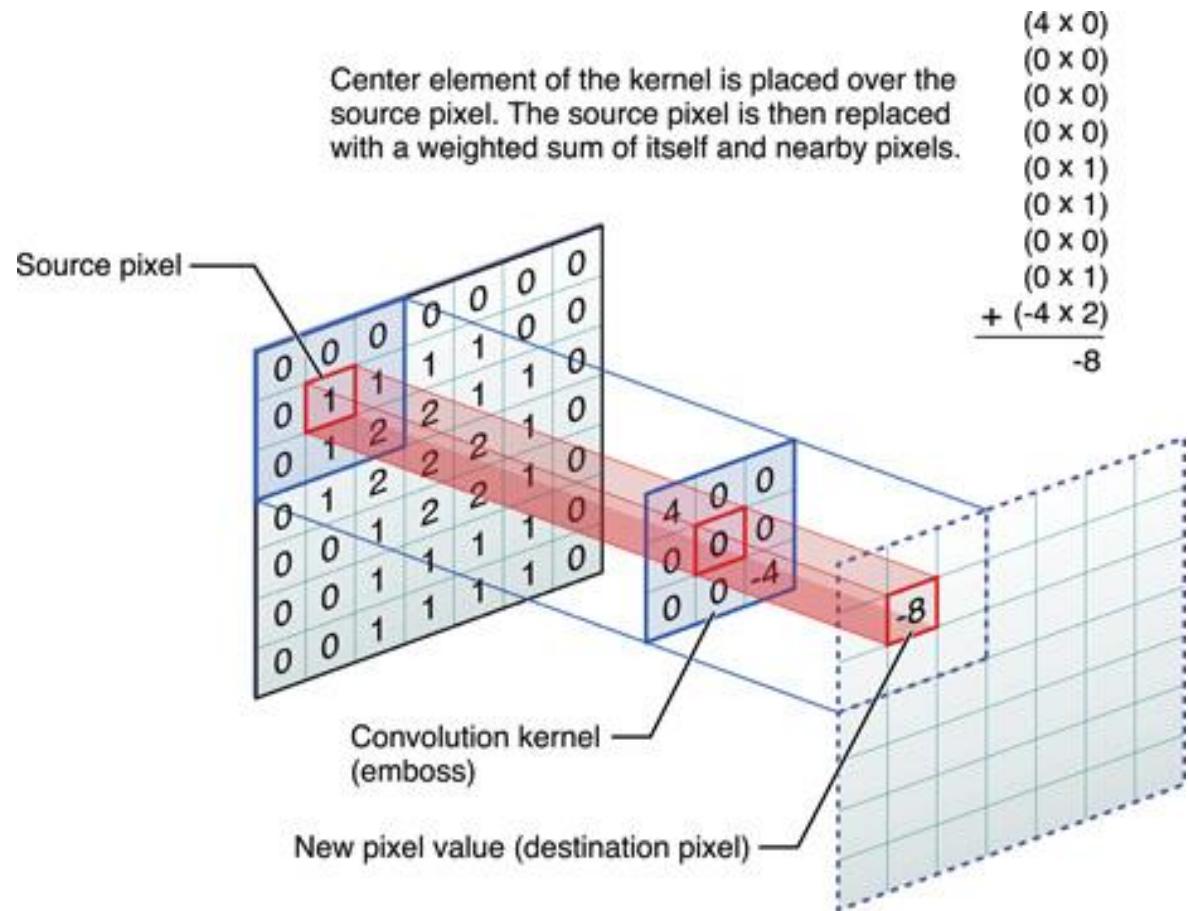
In contrast to that,

1. Make the connections only in a 5×5 neighbourhood of each neuron in the second layer; This would lower the number of parameters to $64^2 \cdot 25 \approx 102k$
2. Make the parameters of all 5×5 connections *shared*; This lowers the number of parameters to only 25.

Doing this corresponds to learning a *convolutional filter* of size 5×5 . In practice, N (e.g. $N = 32$) filters are learnt in the first layer, forming N -channel output. The next layer then operates on all N channels, thus when e.g. $N = 32$ and the receptive field is 3×3 , each of the next convolutional filters has $32 \cdot 3^2$ parameters.

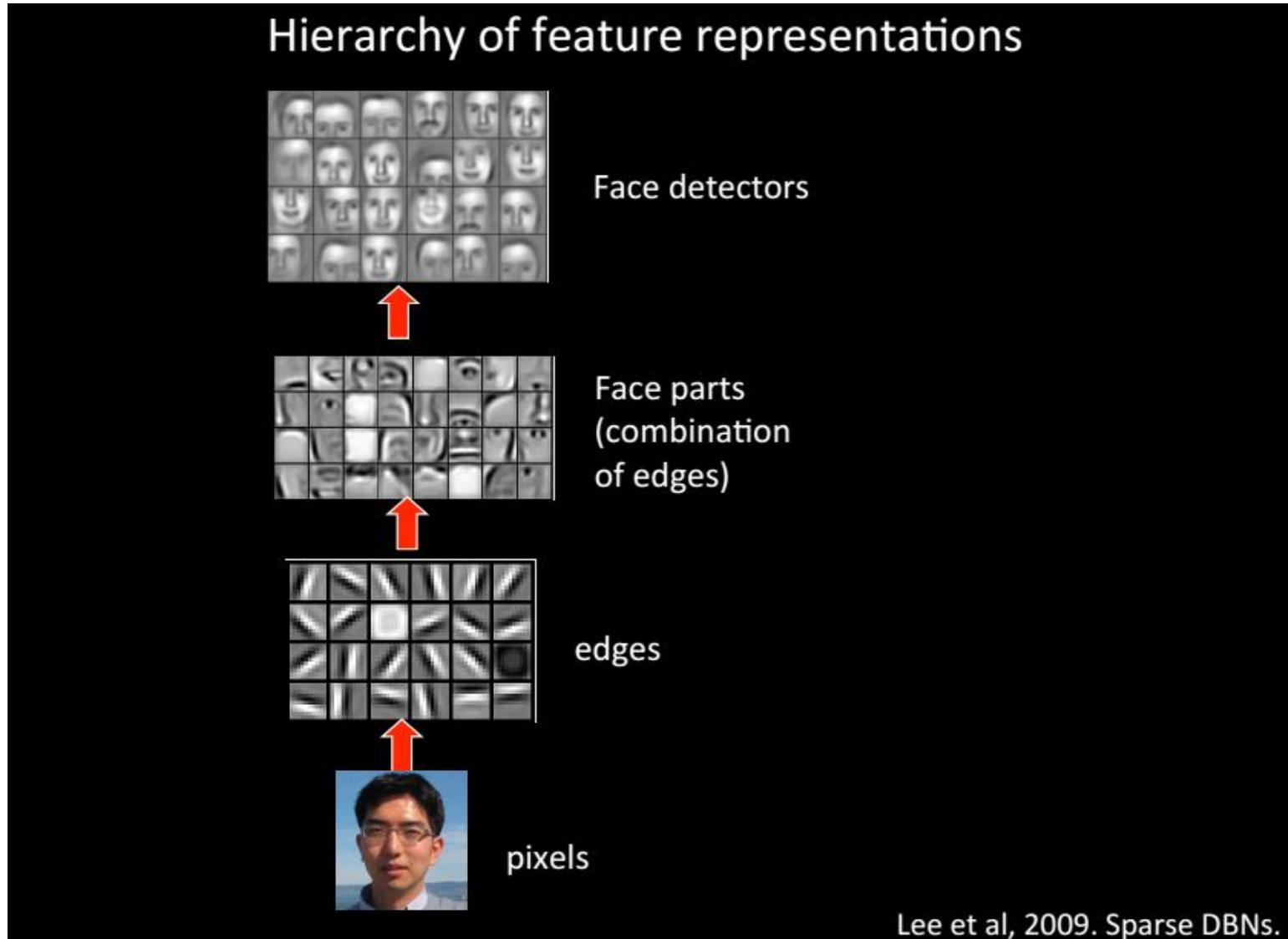
Convolutional structure reduces the number of parameters by orders of magnitude.

WHAT IS CONVOLUTION



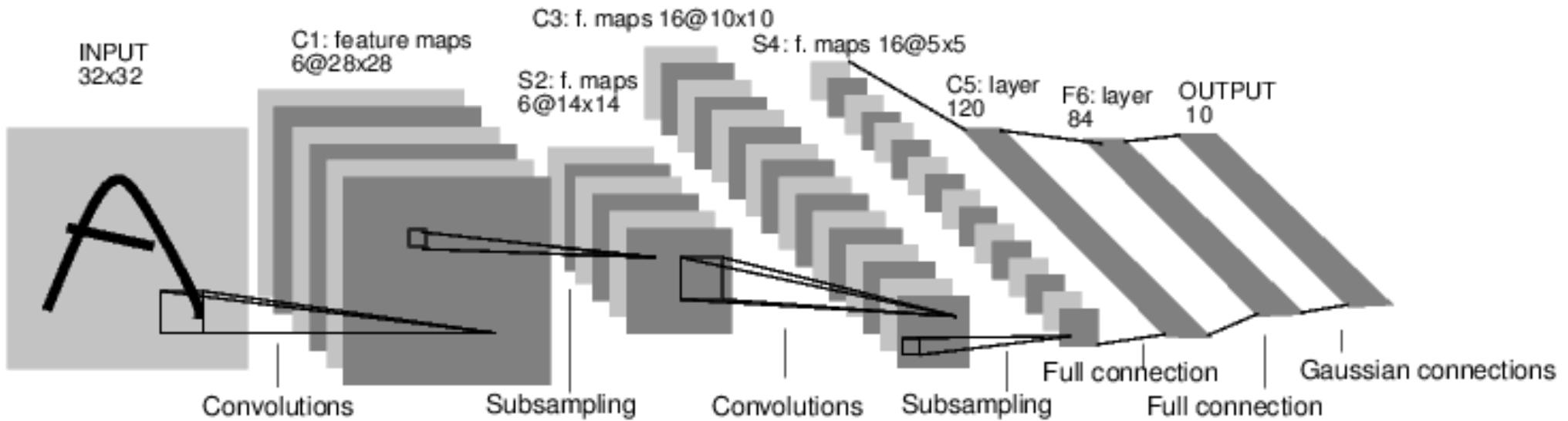
Classical NN
for image
is convolution
with image
size kernel

DEEP LEARNING IS HIERARCHICAL REPRESENTATION LEARNING



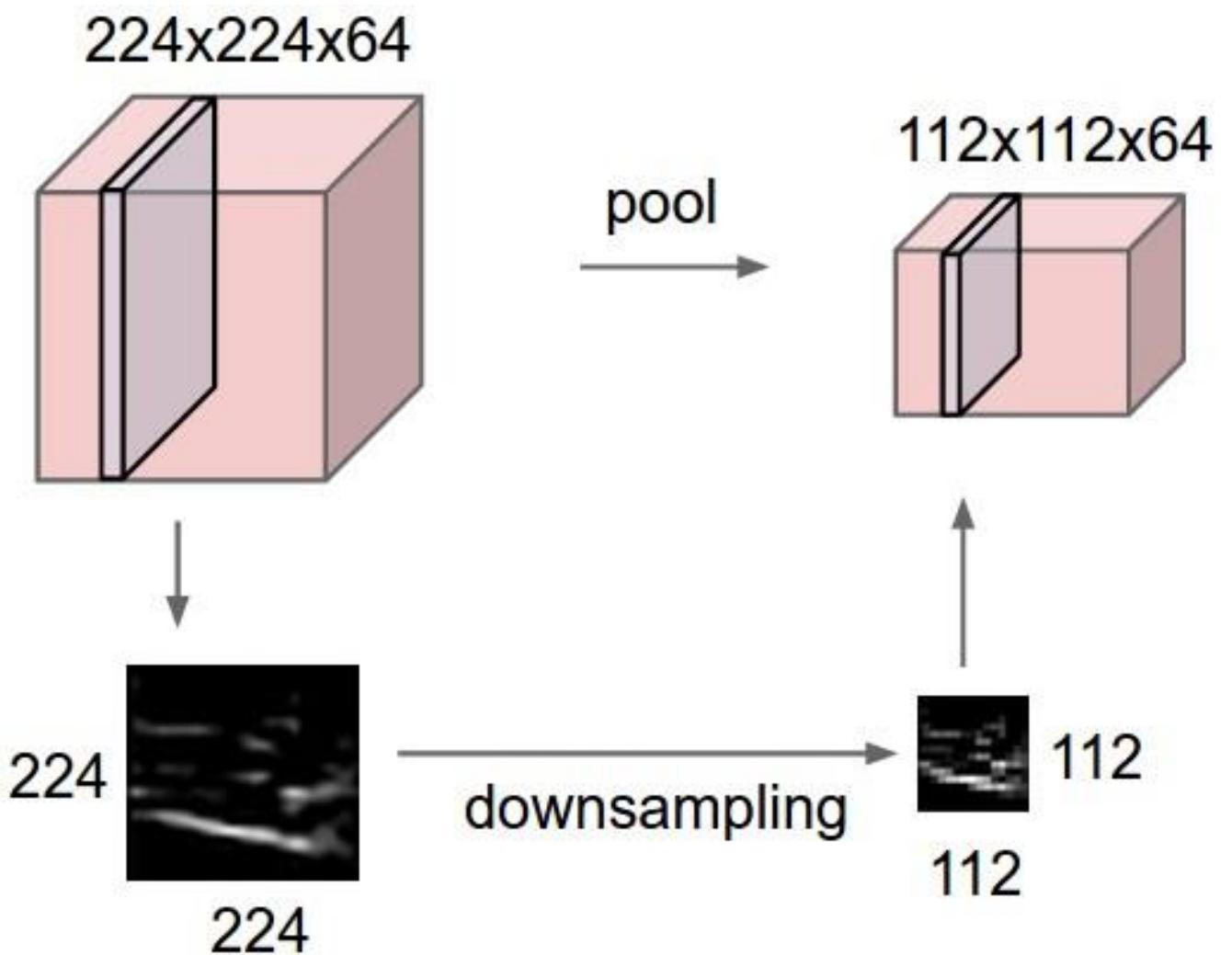
Quoc.V.Le et.al.,2011. Building high-level features using large scale unsupervised learning

TYPICAL CNN STRUCTURE (LENET-5)



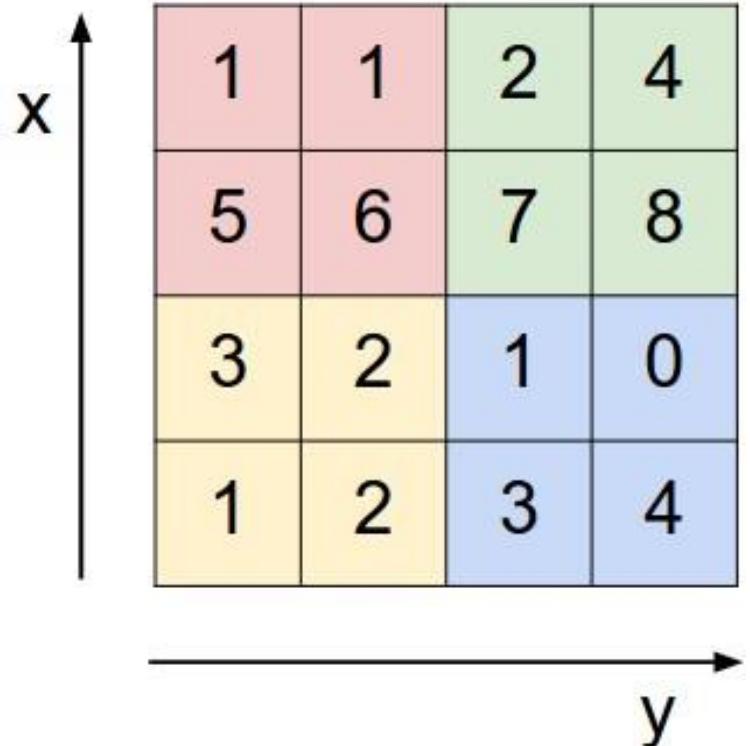
- (Conv-ReLU-Pool)xN Softmax. Simple
- (Conv-ReLU)xN-Pool- (Conv-Relu)x2N-Pool....Softmax. Popular.
- Some Inception arch. Have fun :)

POOLING



MAX POOLING

Single depth slice



max pool with 2x2 filters
and stride 2

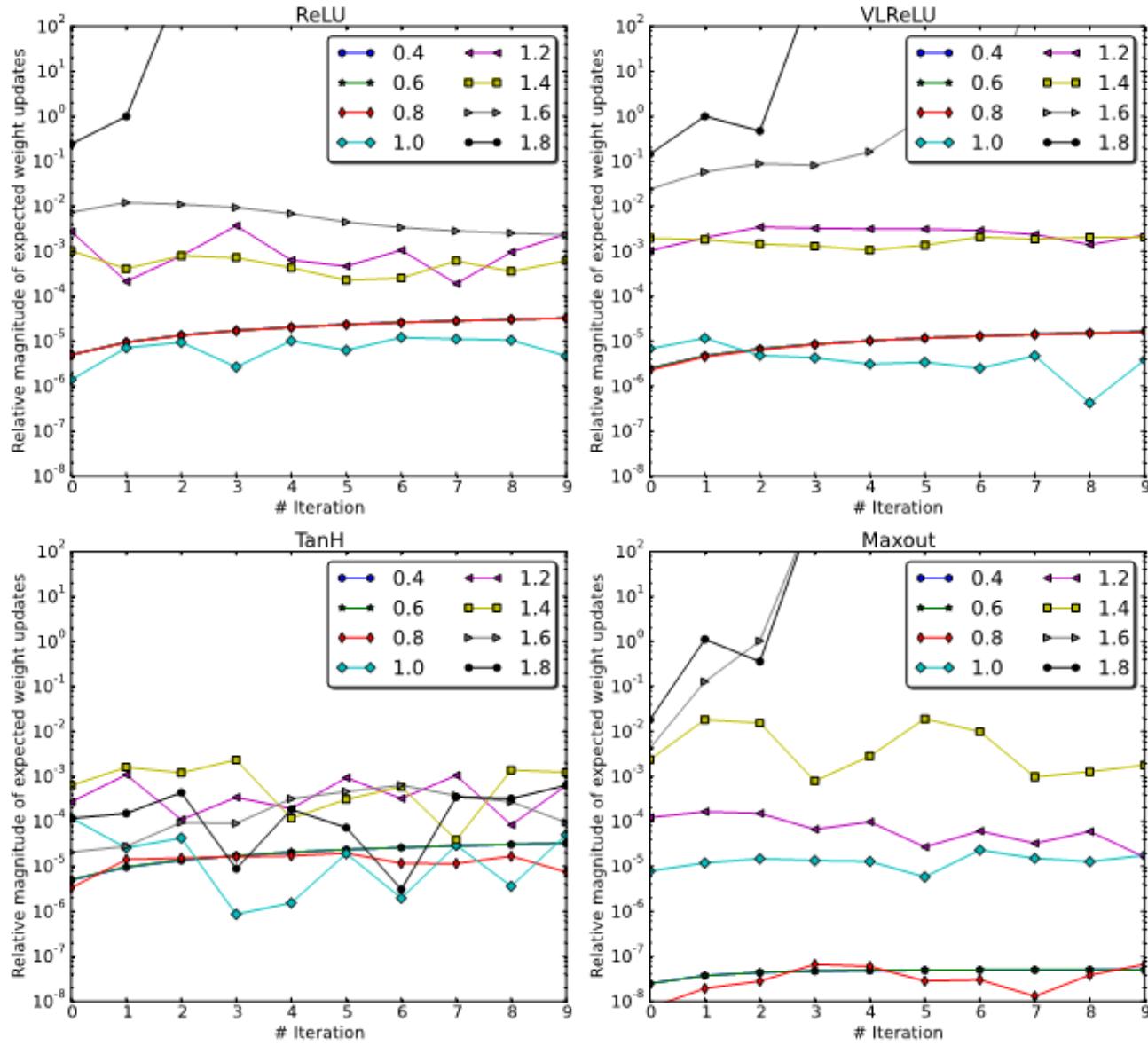
A 2x2 output tensor resulting from max pooling with 2x2 filters and stride 2. The values are:

6	8
3	4

CNNs: Important details/concepts

- ◆ weights init
- ◆ dropout
- ◆ batch normalization
- ◆ data augmentation
- ◆ padding

WEIGHTS INITIALIZATION



- Preserve $\text{var}=1$ of all layers output.
How?
There are lots of papers with variants

WEIGHTS INITIALIZATION

- Gaussian noise with some coefficient:
 - Xavier: $n_l \text{Var}[w_l] = 1, \quad \forall l.$
 - He ($0.5 * \text{Xavier}$ for ReLU)
- Orthonormal (Saxe et.al. 2013)
- Data-dependent: LSUV

Algorithm 1 Layer-sequential unit-variance orthogonal initialization. L – convolution or full-connected layer, W_L - its weights, B_L - its output blob., Tol_{var} - variance tolerance, T_i – current trial, T_{max} – max number of trials.

```

Pre-initialize network with orthonormal matrices as in Saxe et al. (2014)
for each layer  $L$  do
    while  $|\text{Var}(B_L) - 1.0| \geq Tol_{var}$  and  $(T_i < T_{max})$  do
        do Forward pass with a mini-batch
        calculate  $\text{Var}(B_L)$ 
         $W_L = W_L / \sqrt{\text{Var}(B_L)}$ 
    end while
end for

```

BATCH NORMALIZATION

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{mini-batch variance}$$

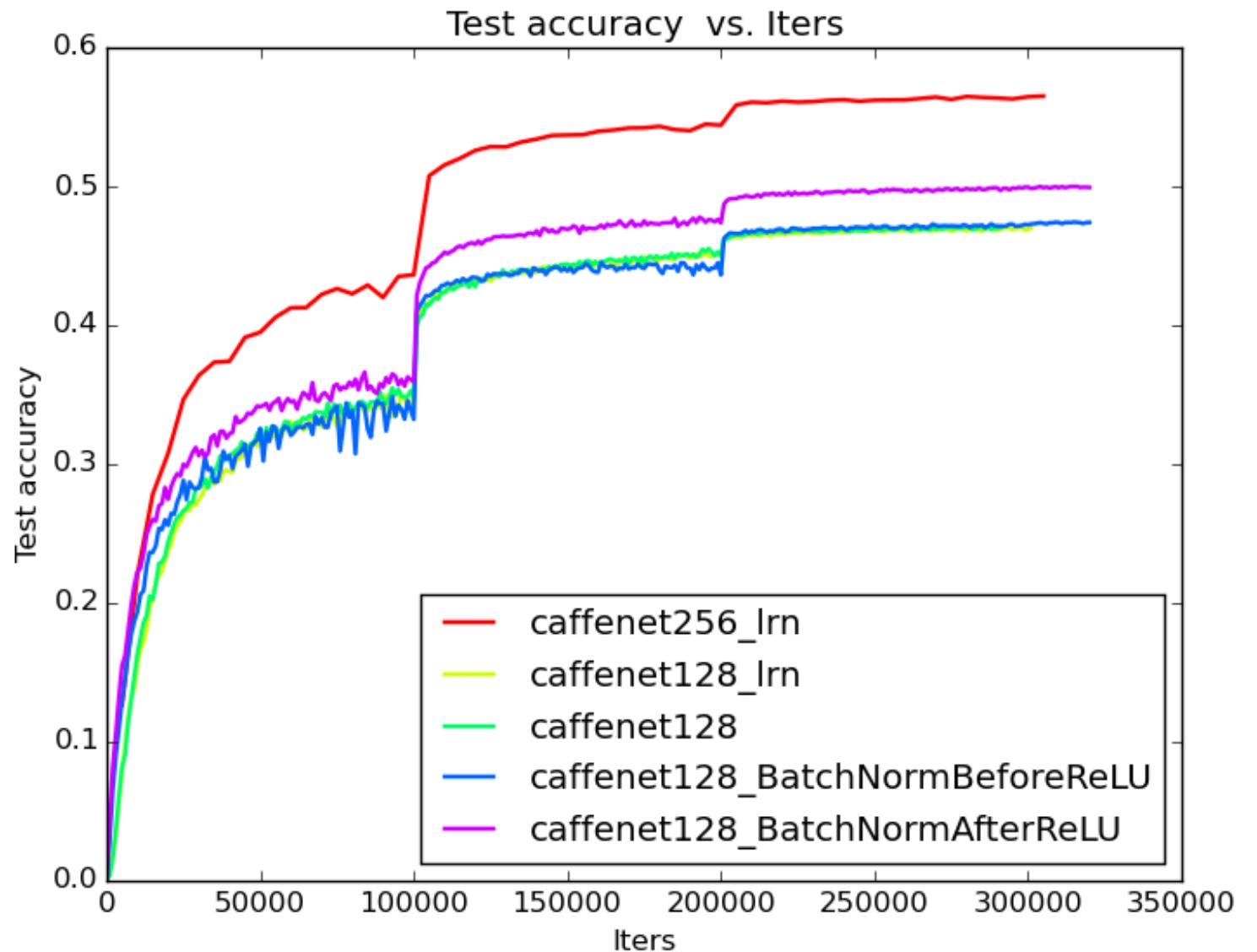
$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

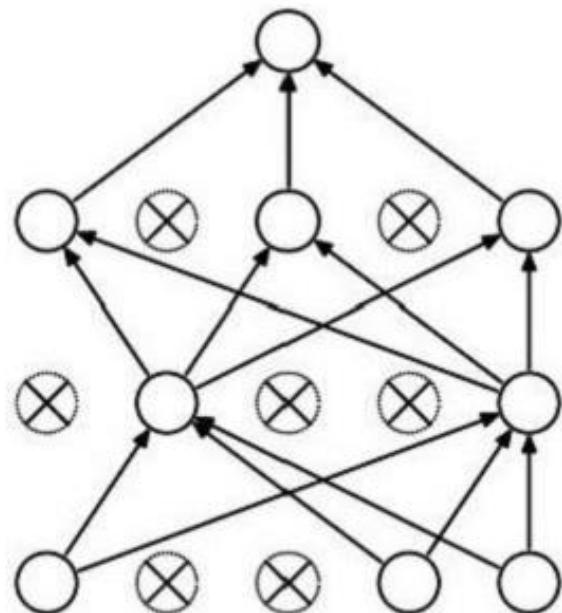
Ioffe et.al 2015

BATCH NORMALIZATION



DROPOUT

Dropout

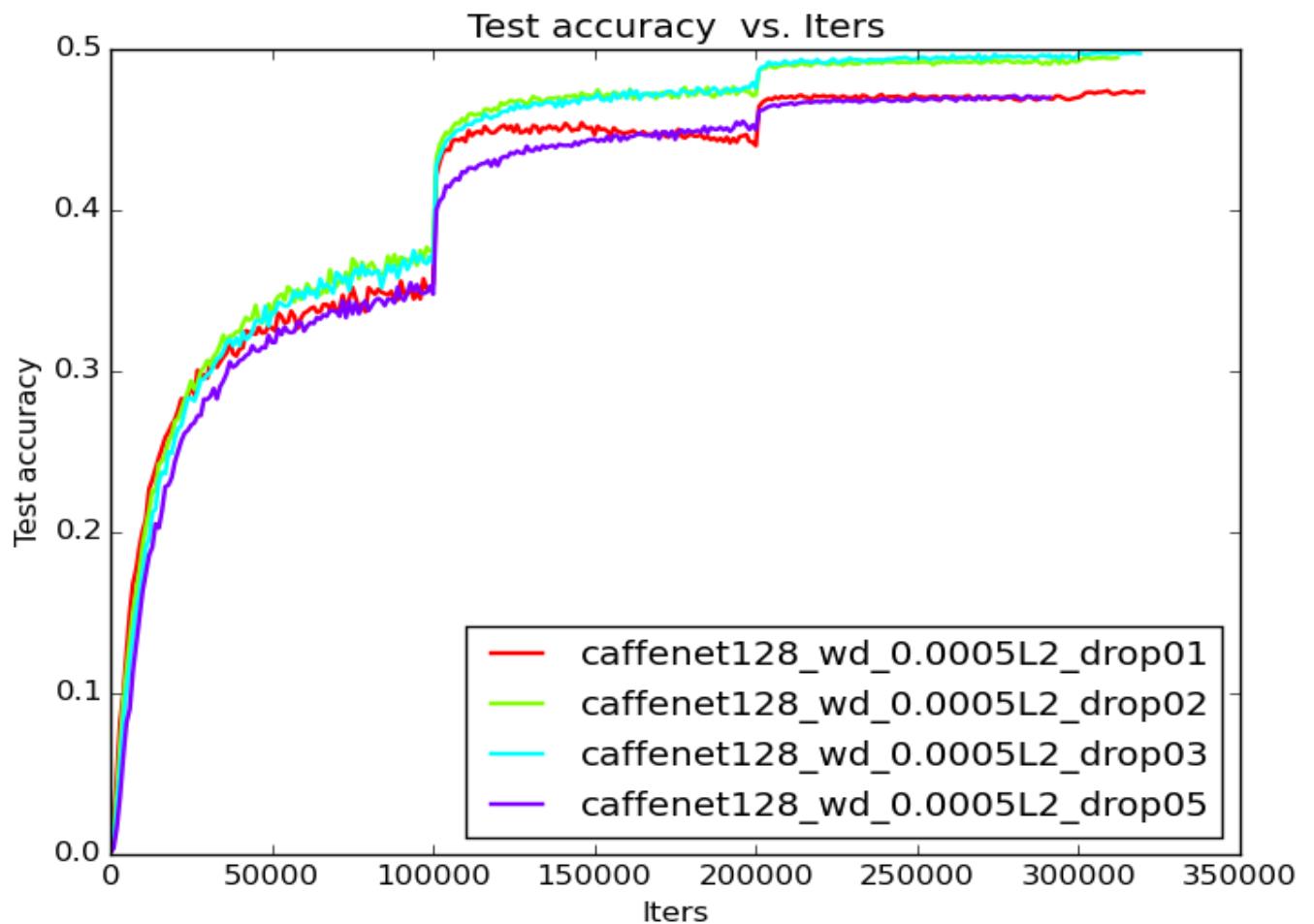


Forces the network to have a redundant representation.



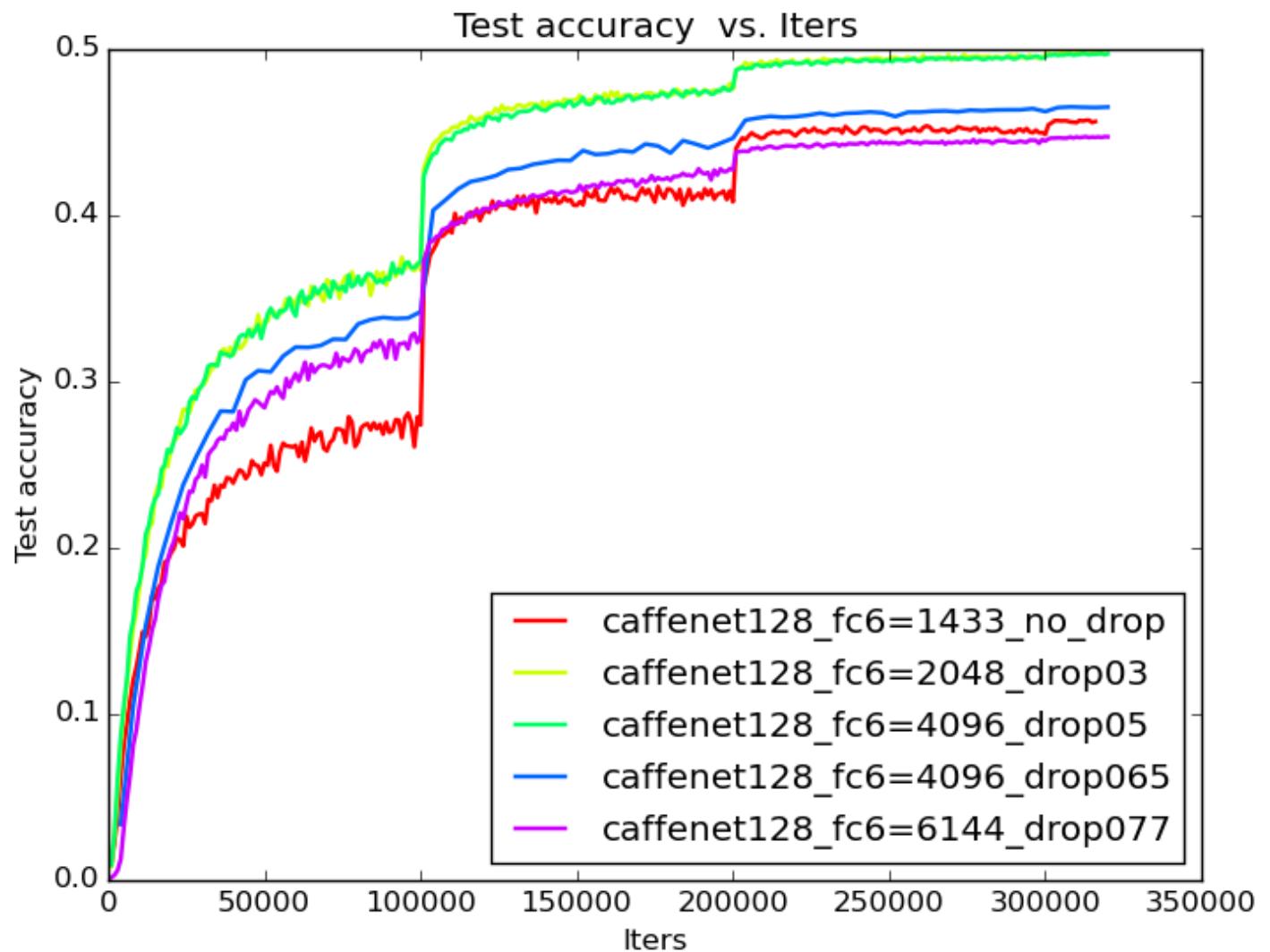
DROPOUT

- Play with rates. 0.5 is rarely optimal choice (but often good)



DROPOUT

- Dropout_rate * width = constant – doesn't work!



DATA AUGMENTATION

- Common (helps 99% cases):

- Random crop: e.g., 227x227 from 256x256 px (AlexNet)
- Horizontal mirror

- Dataset dependent:

- Random rotation
- Affine transform
- Random scale
- Color augmentation
- Noise input
- Thin plate deformation
- Unleash your imagination

PADDING. VALID AND SAME CONVOLUTION

$$\begin{bmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{bmatrix} * \begin{bmatrix} 1 & 3 & 1 \\ 0 & 5 & 0 \\ 2 & 1 & 2 \end{bmatrix}$$

full same valid

17	75	90	35	40	53	15
23	159	165	45	105	137	16
38	198	120	165	205	197	52
56	95	160	200	245	184	35
19	117	190	255	235	106	53
20	89	160	210	75	90	6
22	47	90	65	70	13	18

Same = padding with zeros by $\frac{1}{2}$ kernel size.

The most common choice

PADDING

- Padding:

- Preserving spatial size, not “washing out” information
- Dropout-like augmentation by zeros

Caffenet128

with conv padding: **47%** top-1 acc

w/o conv padding: **41%** top-1 acc.

It is huge difference



Notable approaches & Applications

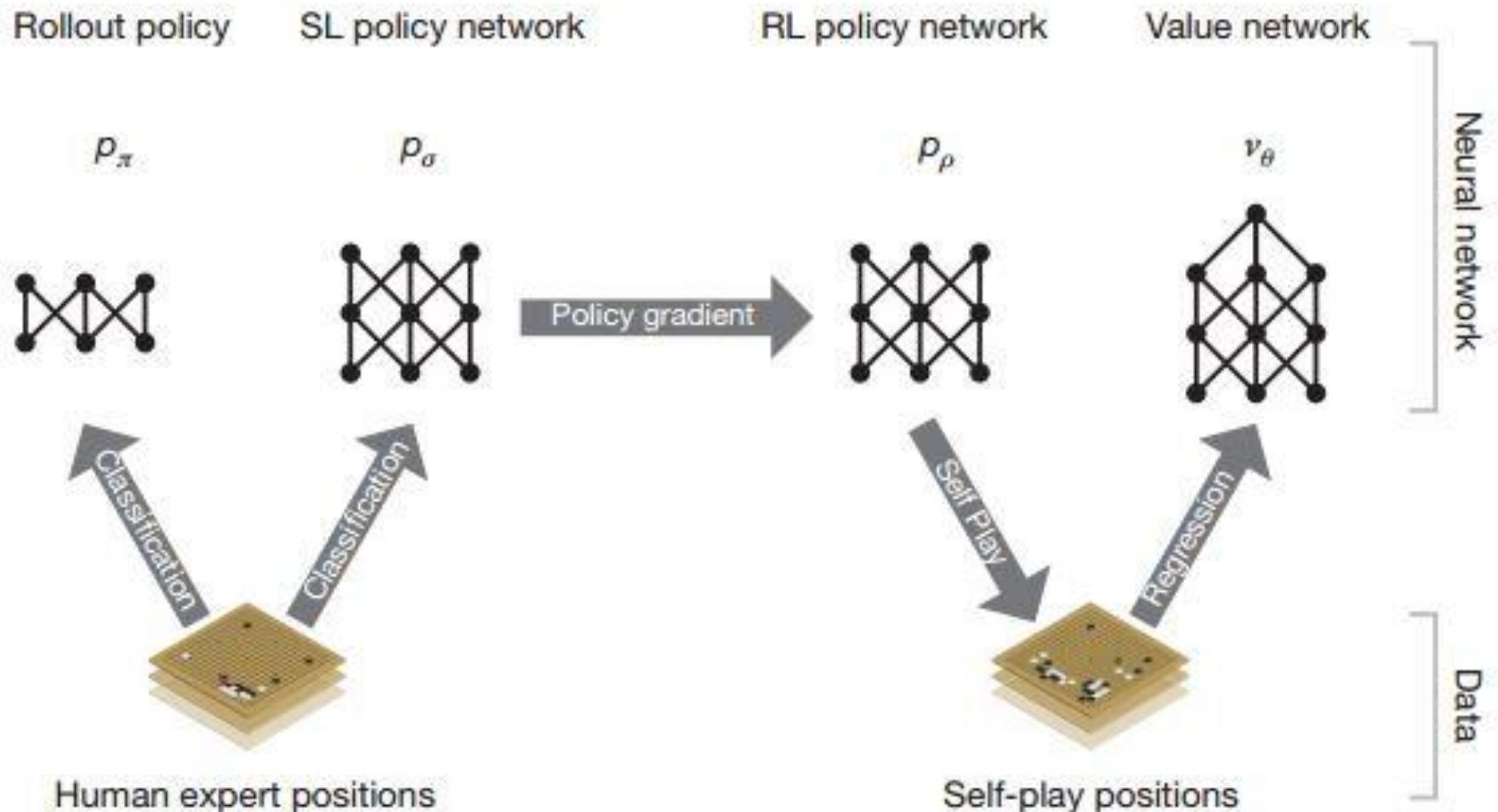
HOW TO DO – LET`S GO TO WHITEBOARD

- **Image retrieval** [Babenko et. al \(2014\)](#)
- **Person identification** [Chopra et. al 2006](#)
- **Ranking** [Wang et.al 2014](#)
- Playing games. [Atari \(2013\)](#) [Go \(2016\)](#)
- Text generation [https://github.com/karpathy/char-rnn](#)
- Image generation [Radford et.al 2016](#)
- Action recognition [Simonyan et.al 2014](#)
- **Anomaly detection**
[https://www.youtube.com/watch?v=ds73ULGjnpc&feature=youtu.be](#)
- Translation [Cho et al 2014](#)
- Fraud detection at PayPal
[http://university.h2o.ai/cds-lp/cds02.html](#)

DEEP LEARNING APPLICATIONS

- Alpha Go :)
- Image recognition
- Speech Recognition. Cortana, Siri
- Translation
- Anomaly detection
- Fraud detection
- Video recognition
- Robotics
- Recommendation systems
- DNA, biology, and more..

ALPHAGO

a

Mastering the game of Go with deep neural networks and tree search
Silver et.al 2016

IMAGE CLASSIFICATION

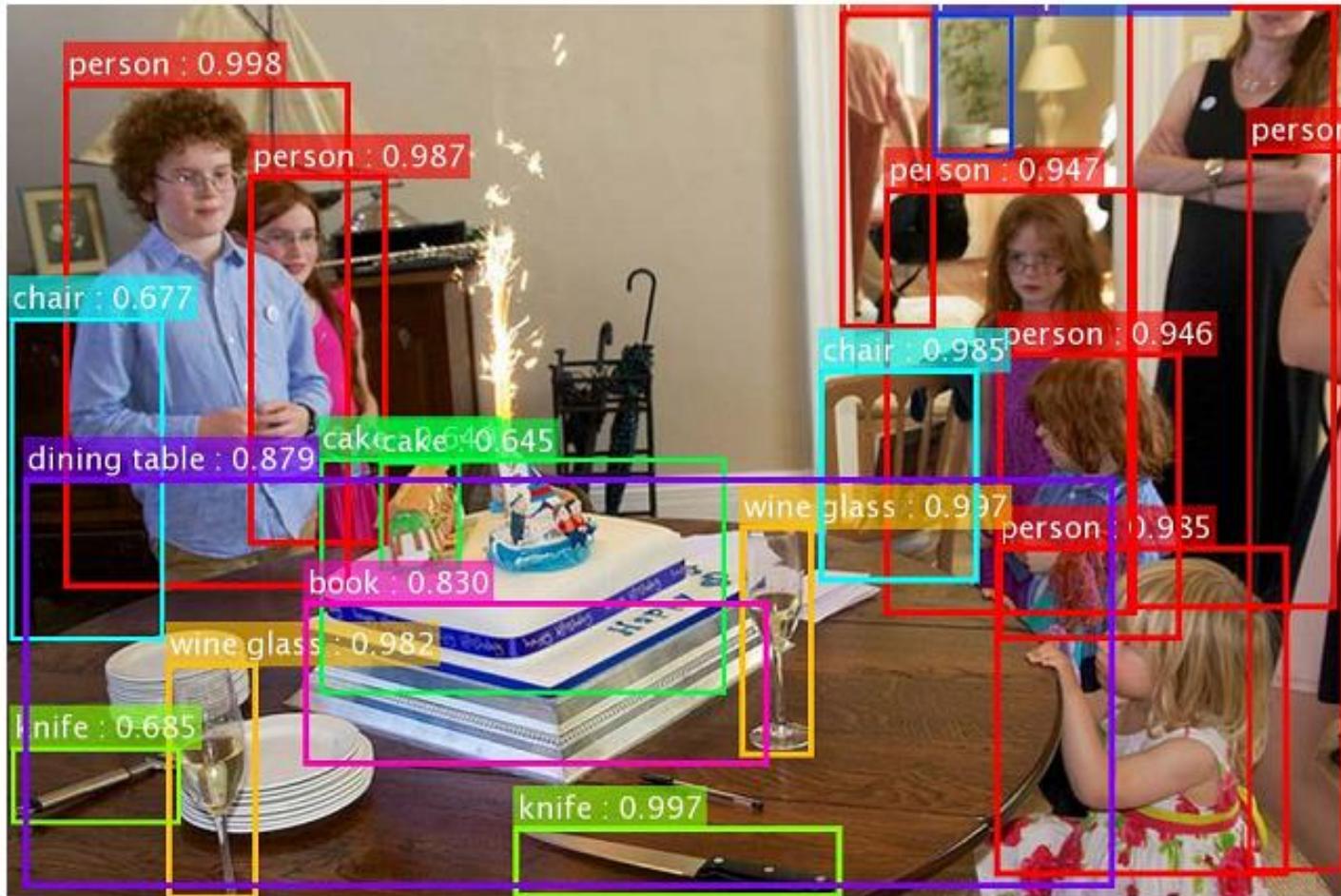
Select all dogs. Our assignment...almost :)



State-of-art since 2012. Krizhevsky et.al 2012
Superhuman level an ImageNet classification since 2015.
He et.al 2015, Szegedy et.al 2015

OBJECT DETECTION

Microsoft
Research



Our results on COCO – too many objects, let's check carefully!

*the original image is from the COCO dataset

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". NIPS 2015.

SPEECH RECOGNITION

- Cortana
- Siri
- OK, Google

Figure 1. CNN architecture for speech recognition

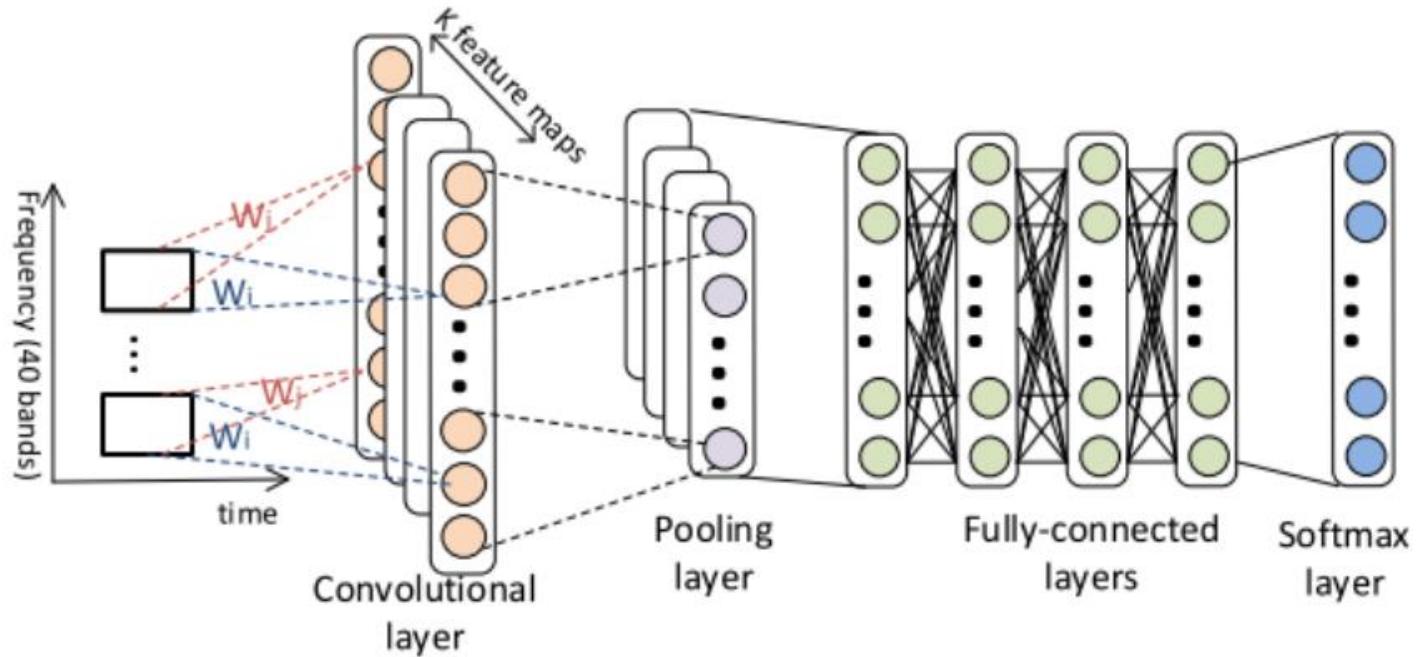


Figure from Huang et.al. 2015.

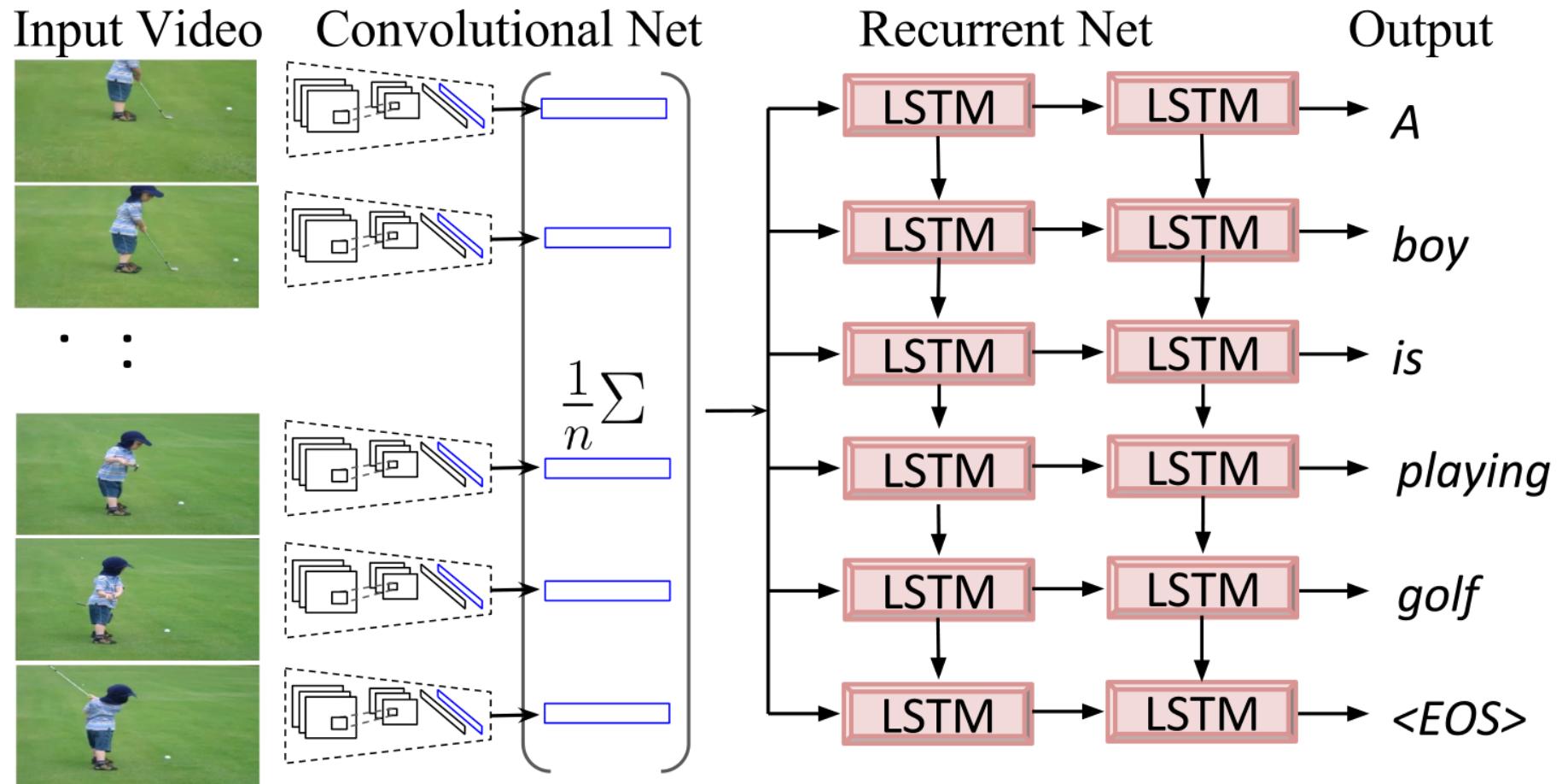
ANOMALY DETECTION

Virtual doctor: Image Recognition for Healthcare

Igor Kostiuk



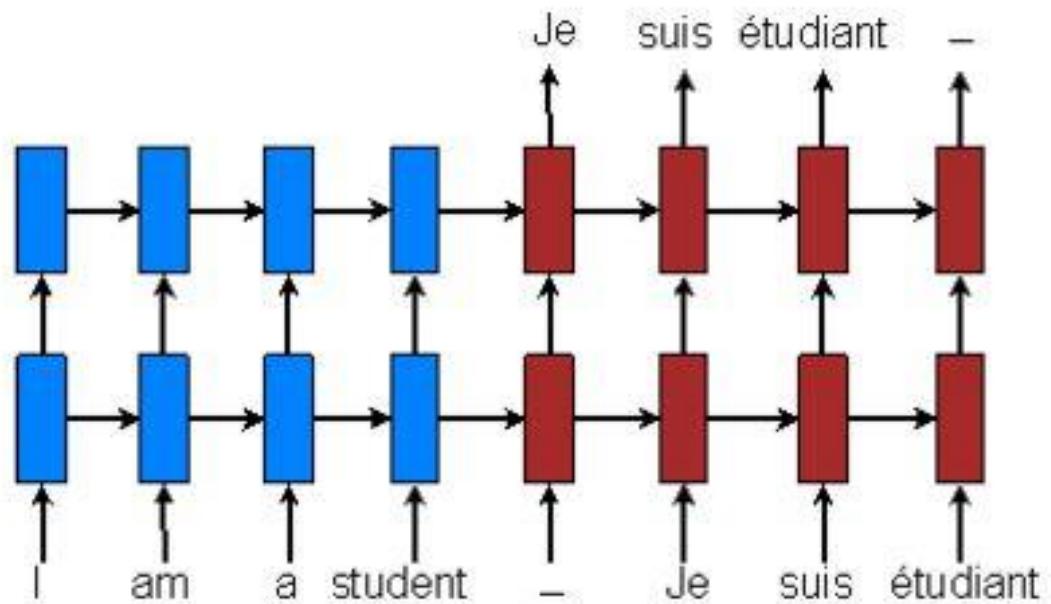
VIDEO CAPTIONING



Translating Videos to Natural Language Using Deep Recurrent Neural Networks. Venugopalan et.al. 2015

TEXT TRANSLATION

Neural Machine Translation (NMT)



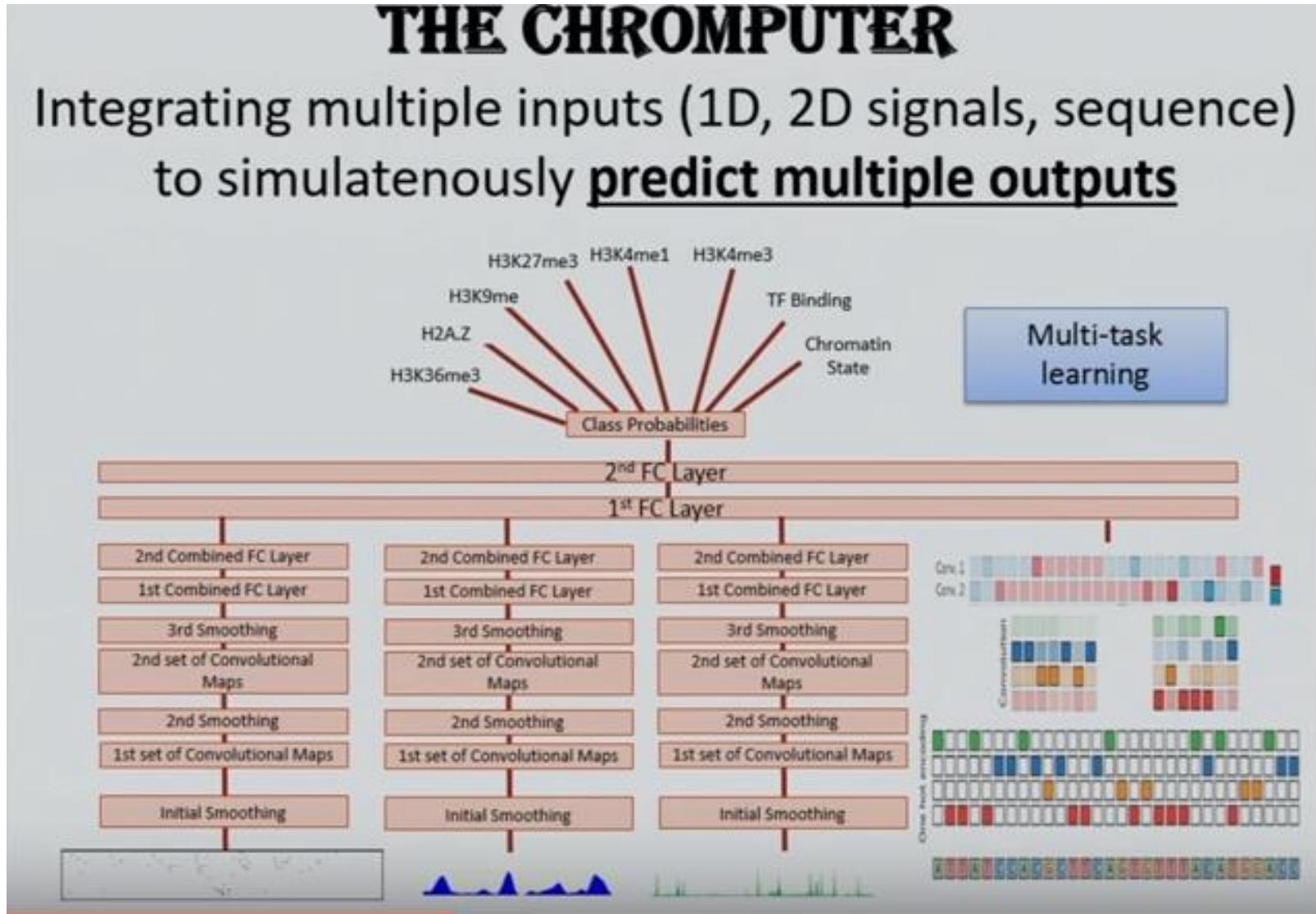
- RNNs trained end-to-end (Sutskever et al., 2014).

From [Bahadanau et al., 2015] slides at ICLR 2015.

DEEP LEARNING FRAMEWORKS FOR REGULATORY GENOMICS AND EPIGENOMICS

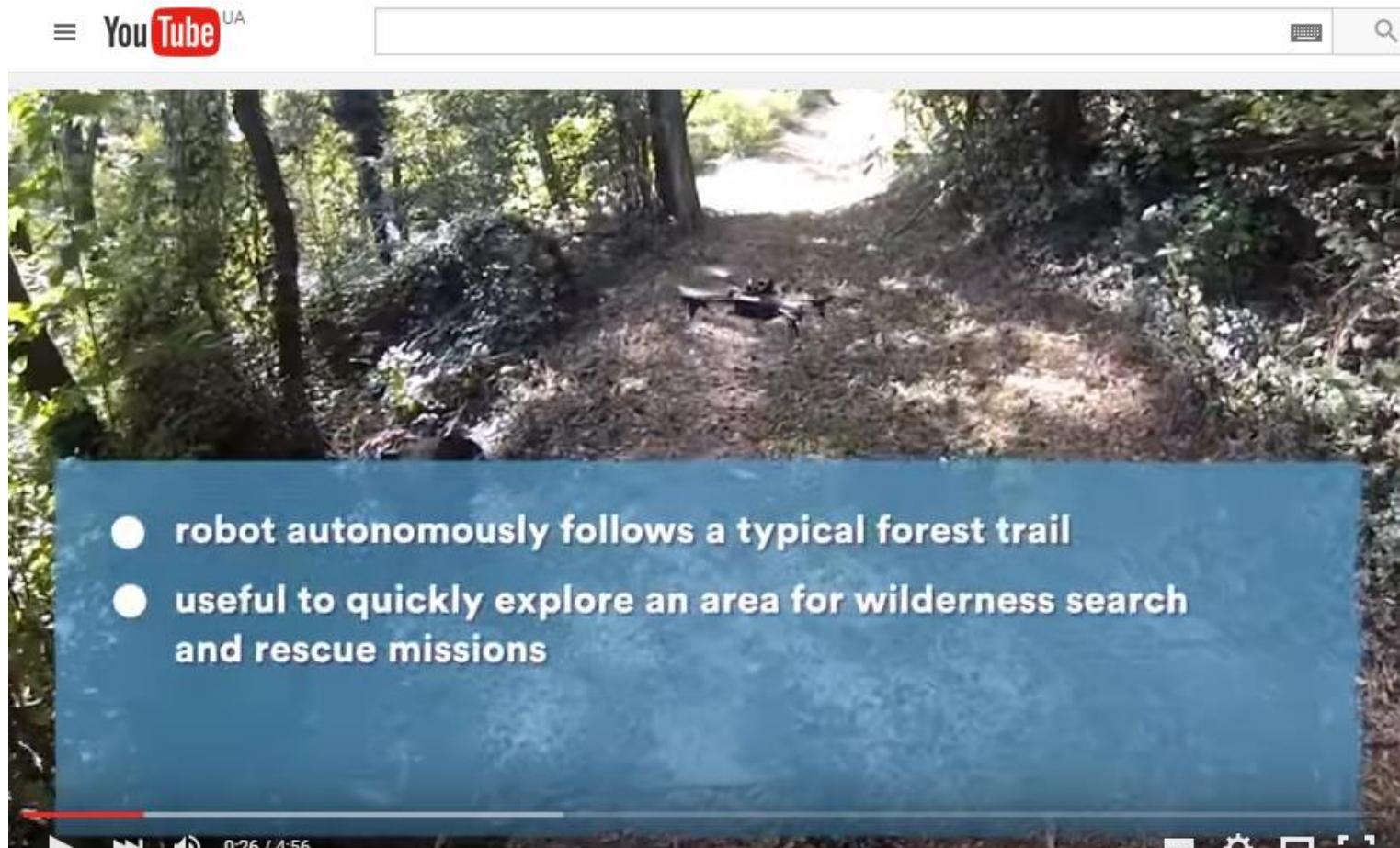
THE CHROMPUTER

Integrating multiple inputs (1D, 2D signals, sequence) to simultaneously **predict multiple outputs**



<https://www.youtube.com/watch?v=2vpKB3j-OY0>

ROBOTICS: NAVIGATION



Quadcopter Navigation in the Forest using Deep Neural Networks



AAAI Video Competition

<https://www.youtube.com/watch?v=umRdt3zGgpU>

FRAUD DETECTION

Experimental Design

10 million rows/1500 features (60% training; 20% validation; 20% test)

Parameter	Range
# of hidden layers	2, 4, 6, 8
# of neurons	200, 300, 400, 500, 600, 700
activation function	Rectifier; Tanh; Maxout; RectifierWithDropout
feature subset	All, subset1 – subset7
test data set	All, week4 – week8
L1/L2 regularization	0 - 1
epoch	500

Results

How much depth is required?

# of hidden layers (Rectifier, 2 layer, 200 neurons, 500 epoch, L1/L2 = 0)	Area Under ROC Curve (AUC)
2	0.762
4	0.821
6	0.839
8	0.839

Best performance with 6 layers

As simple classification

<http://www.slideshare.net/0xdata/>

paypal-fraud-detection-with-deep-learning-in-h2o-presentationh2oworld2014

Neural Networks Summary

- + Flexible technique; can be employed both for classification and regression
- + Multiclass classification
- + Outputs probabilities of all classes, giving access to confidence of prediction
- Interpretability
- Objective function has multiple extrema and gradient-based learning is not guaranteed to find the global minimum

K-Means

lecturer: Jiří Matas, matas@cmp.felk.cvut.cz

authors: J. Matas, O. Drbohlav

Czech Technical University, Faculty of Electrical Engineering
Department of Cybernetics, Center for Machine Perception
121 35 Praha 2, Karlovo nám. 13, Czech Republic

<http://cmp.felk.cvut.cz>

4/Dec/2015

Last update: 7/Dec/2015, 11am

LECTURE PLAN

- ◆ Least squares clustering problem statement
- ◆ K-means, algorithm, properties
- ◆ Initialization by K-means++
- ◆ Related methods

Formulation of the Least-Squares Clustering Problem

Given:

$\mathcal{T} = \{\mathbf{x}_l\}_{l=1}^L$ the set of observations

K the desired number of cluster prototypes

Output:

$\{\mathbf{c}_k\}_{k=1}^K$ the set of cluster prototypes (etalons)

$\{\mathcal{T}_k\}_{k=1}^K$ the clustering (partitioning) of the data

$\bigcup_{k=1}^K \mathcal{T}_k = \mathcal{T}, \mathcal{T}_i \cap \mathcal{T}_j = \emptyset \text{ for } i \neq j$

The result is obtained by solving the following optimization problem:

$$(\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K; \mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_K) = \underset{\text{all } \mathbf{c}'_k, \mathcal{T}'_k}{\operatorname{argmin}} J(\mathbf{c}'_1, \mathbf{c}'_2, \dots, \mathbf{c}'_K; \mathcal{T}'_1, \mathcal{T}'_2, \dots, \mathcal{T}'_K) \quad (1)$$

where

$$J(\mathbf{c}'_1, \mathbf{c}'_2, \dots, \mathbf{c}'_K; \mathcal{T}'_1, \mathcal{T}'_2, \dots, \mathcal{T}'_K) = \sum_{k=1}^K \sum_{\mathbf{x} \in \mathcal{T}'_k} \|\mathbf{x} - \mathbf{c}'_k\|^2. \quad (2)$$

Formulation of the Least-Squares Clustering Problem

Given:

$\mathcal{T} = \{\mathbf{x}_l\}_{l=1}^L$ the set of observations
 K the desired number of cluster prototypes

Output:

$\{\mathbf{c}_k\}_{k=1}^K$ the set of cluster prototypes (etalons)
 $\{\mathcal{T}_k\}_{k=1}^K$ the clustering (partitioning) of the data
 $\cup_{k=1}^K \mathcal{T}_k = \mathcal{T}, \mathcal{T}_i \cap \mathcal{T}_j = \emptyset \text{ for } i \neq j$

Note that this problem can be equivalently rewritten as an optimization in cluster centres only:

$$(\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K) = \operatorname{argmin}_{\{\mathbf{c}'_k\}_{k=1}^K} J(\mathbf{c}'_1, \mathbf{c}'_2, \dots, \mathbf{c}'_K), \quad (3)$$

$$J(\mathbf{c}'_1, \mathbf{c}'_2, \dots, \mathbf{c}'_K) = \sum_{l=1}^L \min_{k \in \{1, 2, \dots, K\}} \|\mathbf{x}_l - \mathbf{c}'_k\|^2, \quad (4)$$

with \mathcal{T}_k 's then computed as

$$\mathcal{T}_k = \{\mathbf{x} \in \mathcal{T} : \forall j, \|\mathbf{x} - \mathbf{c}_k\|^2 \leq \|\mathbf{x} - \mathbf{c}_j\|^2\} \quad (\forall k = 1, 2, \dots, K). \quad (5)$$

K-Means: An Algorithm for the LS Clustering Problem

Given:

$\mathcal{T} = \{\mathbf{x}_l\}_{l=1}^L$ the set of observations
 K the desired number of cluster prototypes

Output:

$\{\mathbf{c}_k\}_{k=1}^K$ the set of cluster prototypes (etalons)
 $\{\mathcal{T}_k\}_{k=1}^K$ the clustering (partitioning) of the data
 $\bigcup_{k=1}^K \mathcal{T}_k = \mathcal{T}, \mathcal{T}_i \cap \mathcal{T}_j = \emptyset \text{ for } i \neq j$

K-Means Algorithm:

1. Initialize the cluster centres $\{\mathbf{c}_k\}_{k=1}^K$ (e.g. by random selection from the data points \mathcal{T} , without replacement)
2. Assignment optimization (assign to closest etalon):

$$\mathcal{T}_k = \{\mathbf{x} \in \mathcal{T} : \forall j, \|\mathbf{x} - \mathbf{c}_k\|^2 \leq \|\mathbf{x} - \mathbf{c}_j\|^2\} \quad (\forall k = 1, 2, \dots, K) \quad (6)$$

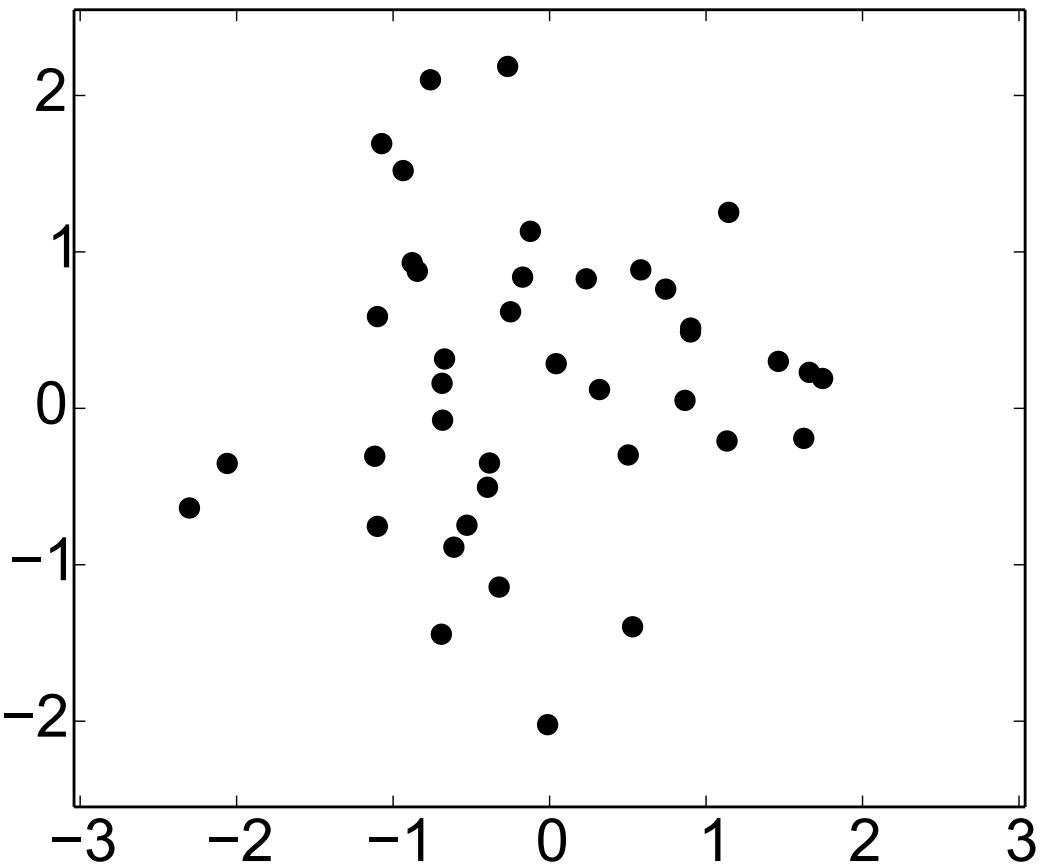
3. Prototype optimization (updated etalon is the mean of data assigned to it):

$$\mathbf{c}_k = \begin{cases} \frac{1}{|\mathcal{T}_k|} \sum_{\mathbf{x} \in \mathcal{T}_k} \mathbf{x} & \text{if } |\mathcal{T}_k| > 0 \\ \text{re-initialize} & \text{if } \mathcal{T}_k = \emptyset \end{cases} \quad (\forall k = 1, 2, \dots, K) \quad (7)$$

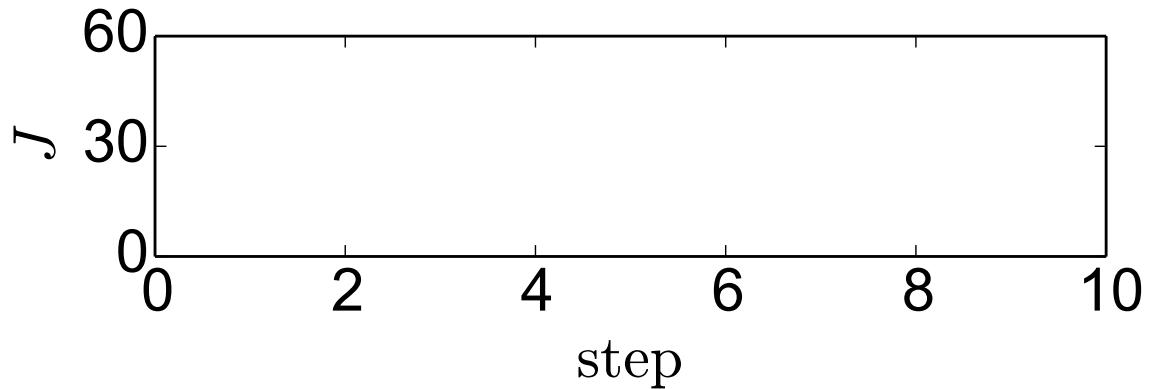
4. Terminate if $\forall k : \mathcal{T}_k^{t+1} = \mathcal{T}_k^t$, otherwise goto 2

K-Means: Example

data points

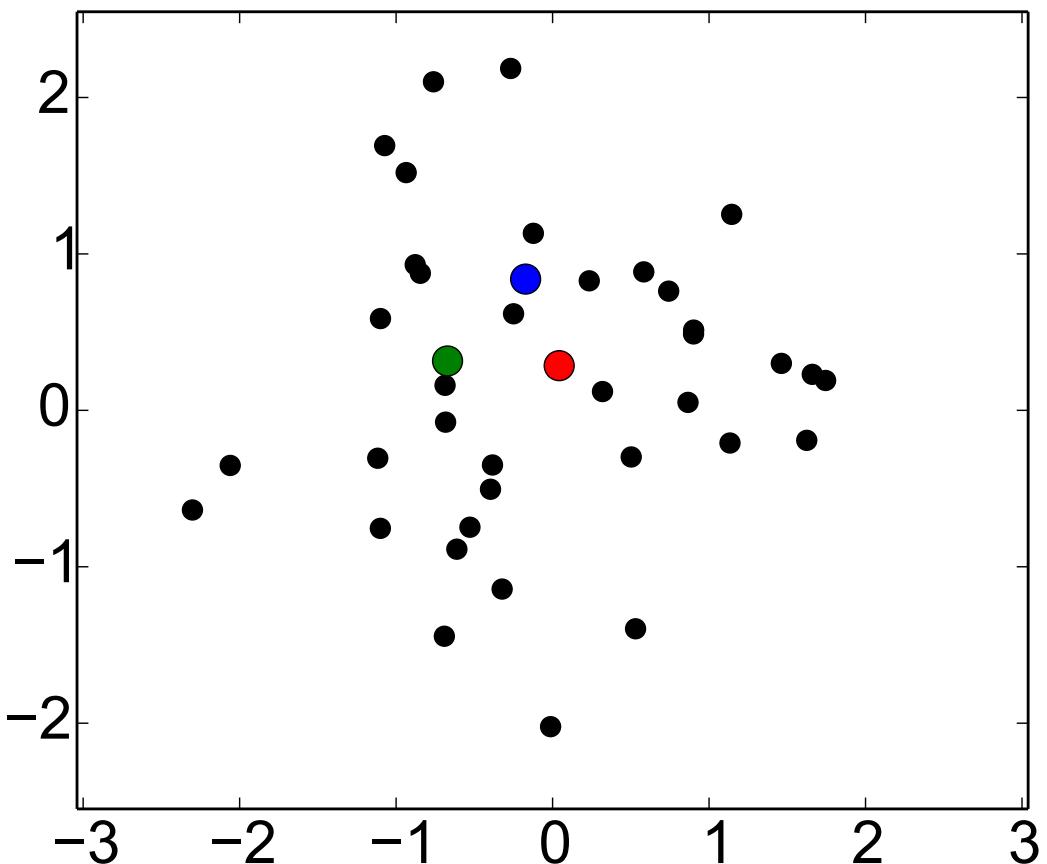


Cluster the data points to $K = 3$ clusters



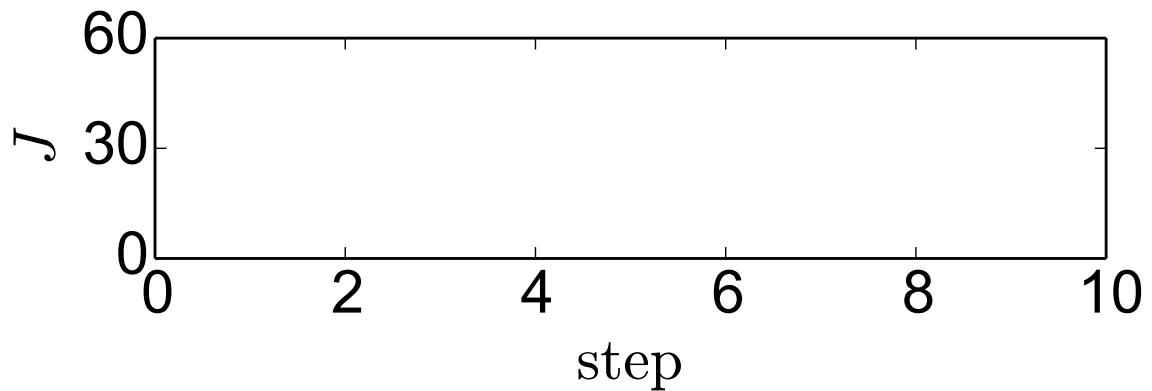
K-Means: Example

● ● ● initial cluster centers



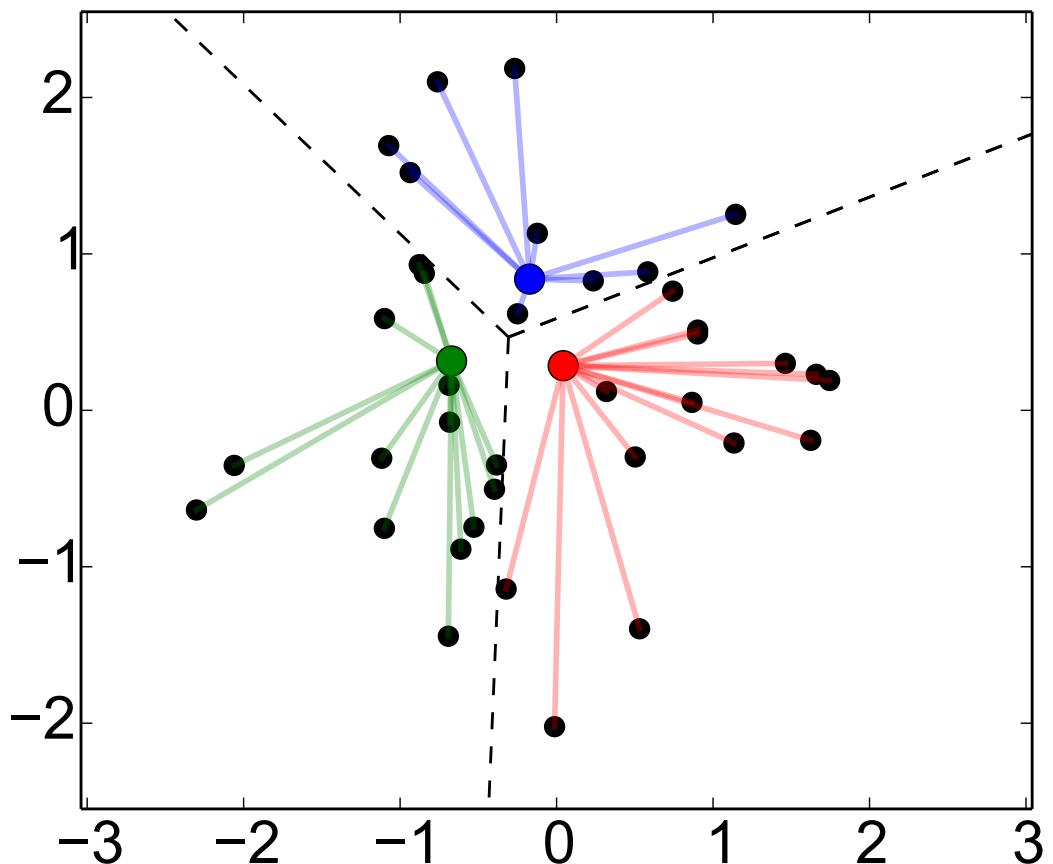
Cluster the data points to $K = 3$ clusters

Initial cluster centers (here selected randomly from data points)



K-Means: Example

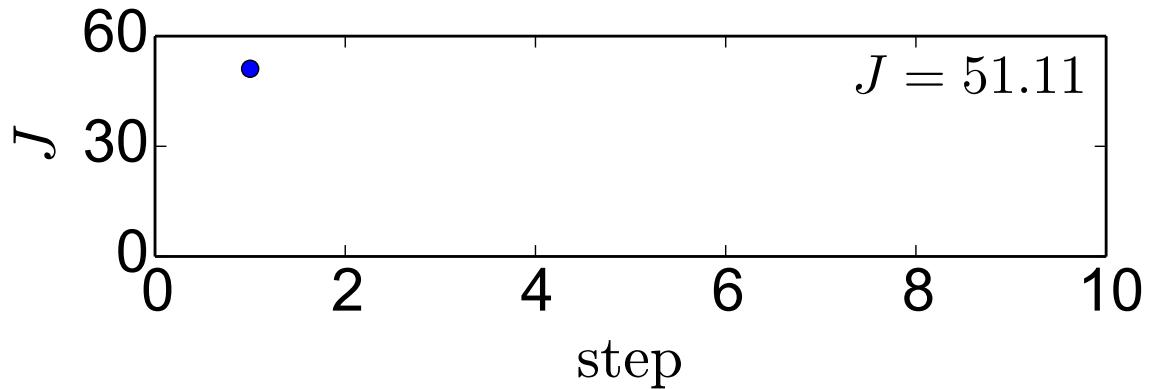
(-- Voronoi boundaries)



Cluster the data points to $K = 3$ clusters

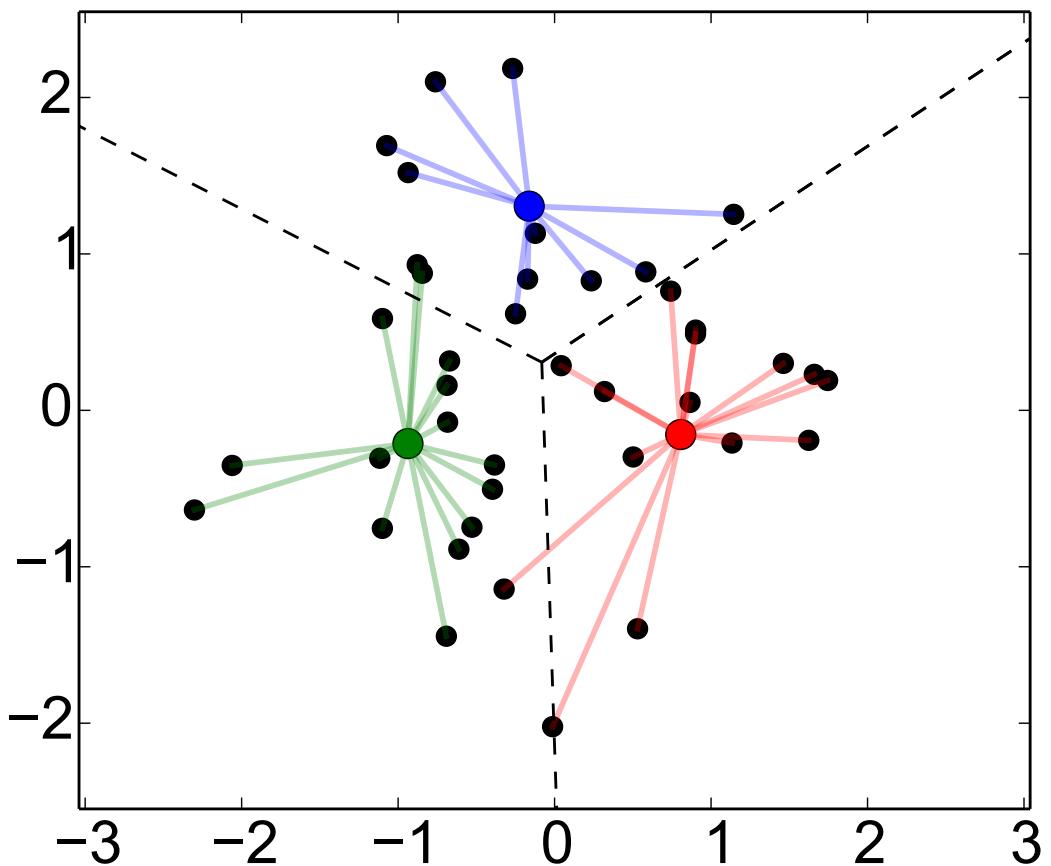
Initial cluster centers (here selected randomly from data points)

step 1, compute assignments



K-Means: Example

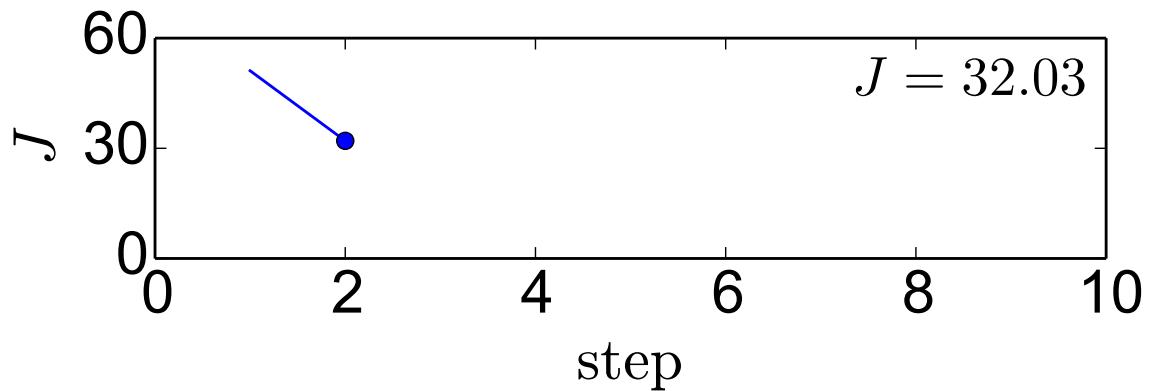
(-- Voronoi boundaries)



Cluster the data points to $K = 3$ clusters

Initial cluster centers (here selected randomly from data points)

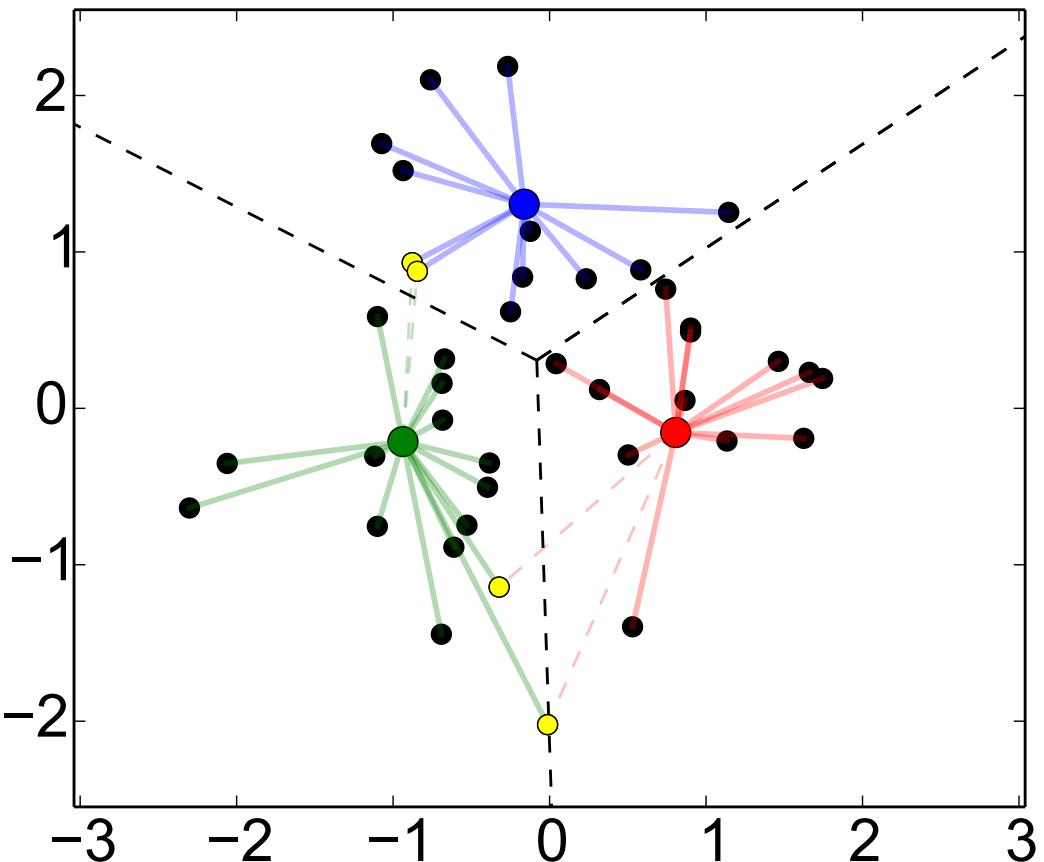
step 1, compute assignments
step 2, recompute centers



K-Means: Example

(-- Voronoi boundaries)

● points with changed assignment



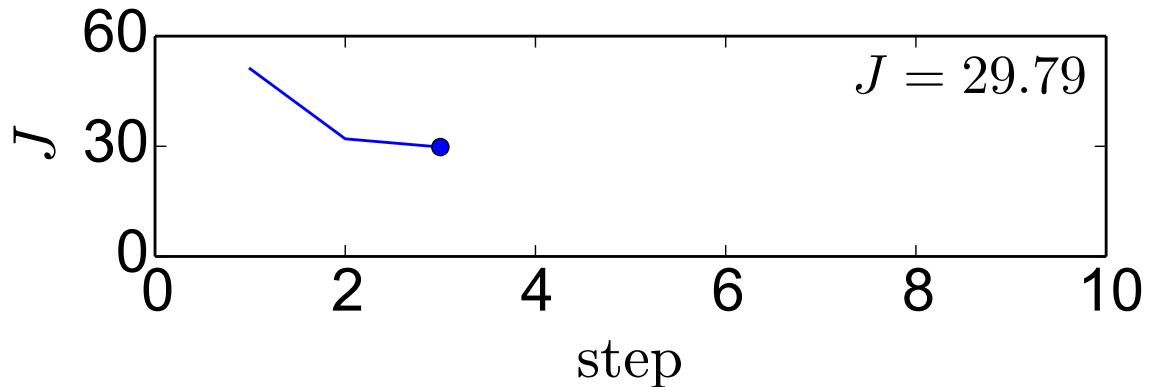
Cluster the data points to $K = 3$ clusters

Initial cluster centers (here selected randomly from data points)

step 1, compute assignments

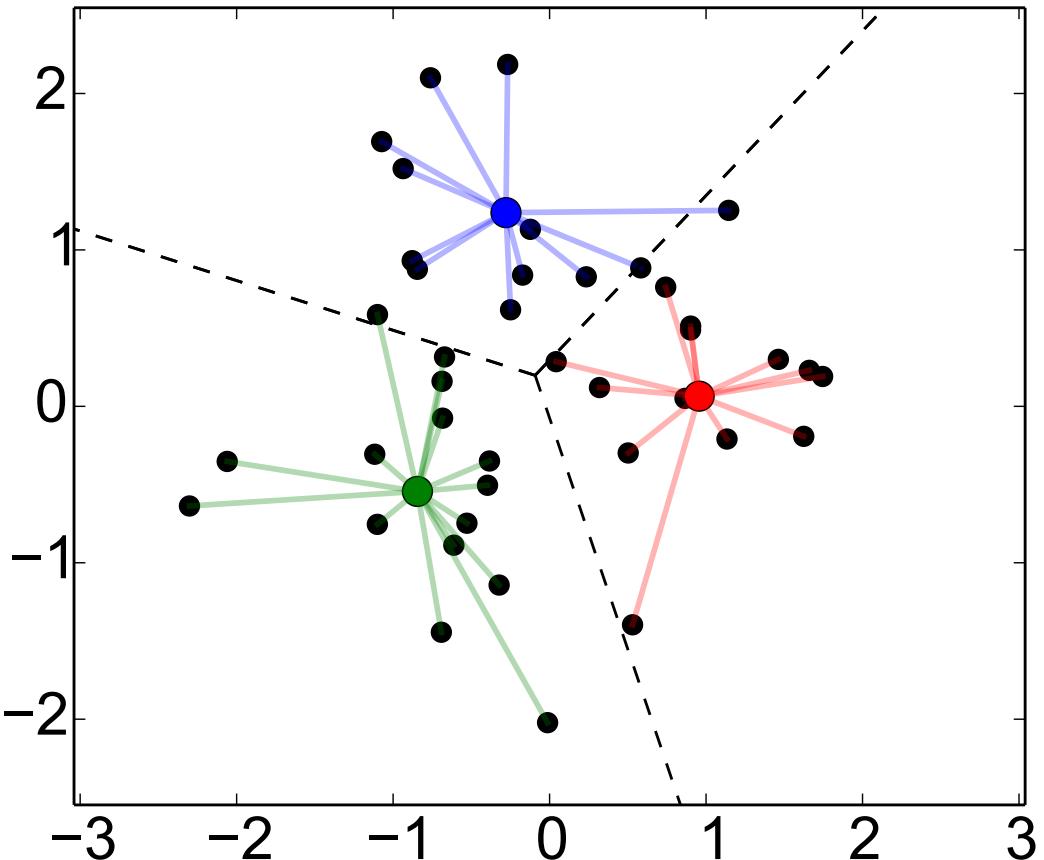
step 2, recompute centers

step 3, recompute assignments



K-Means: Example

(-- Voronoi boundaries)
● points with changed assignment



Cluster the data points to $K = 3$ clusters

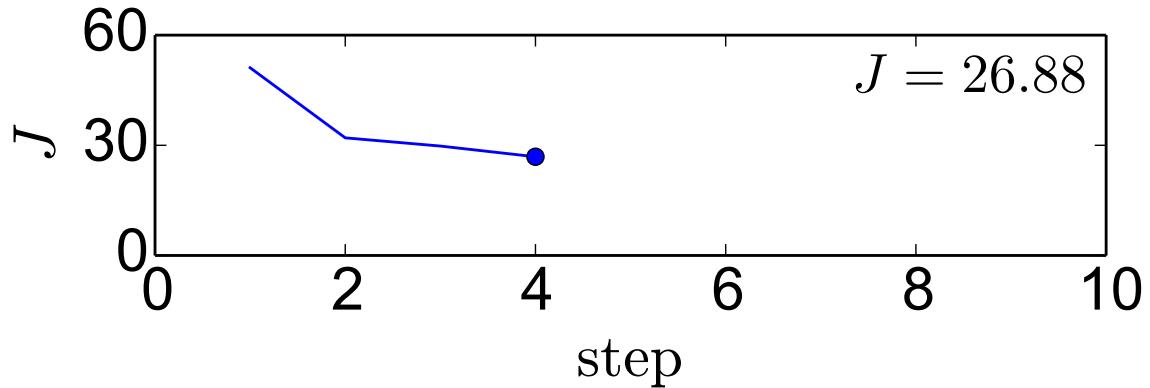
Initial cluster centers (here selected randomly from data points)

step 1, compute assignments

step 2, recompute centers

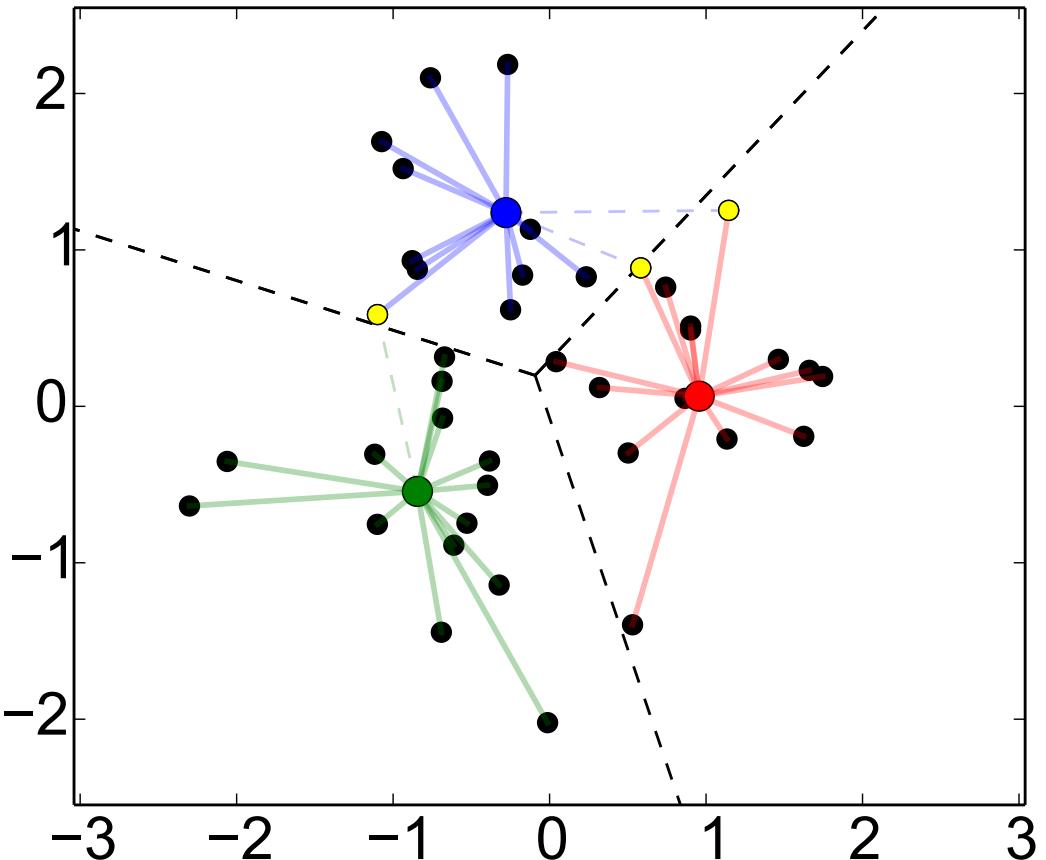
step 3, recompute assignments

step 4, recompute centers



K-Means: Example

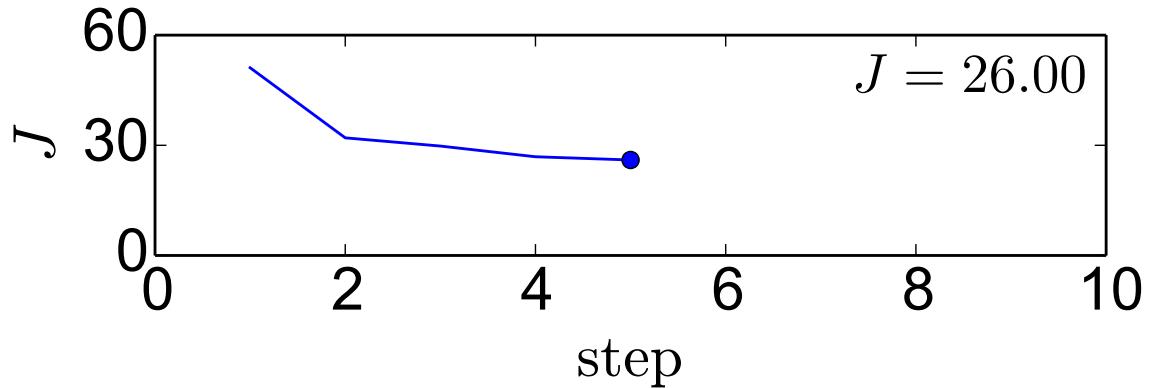
(-- Voronoi boundaries)
 ● points with changed assignment



Cluster the data points to $K = 3$ clusters

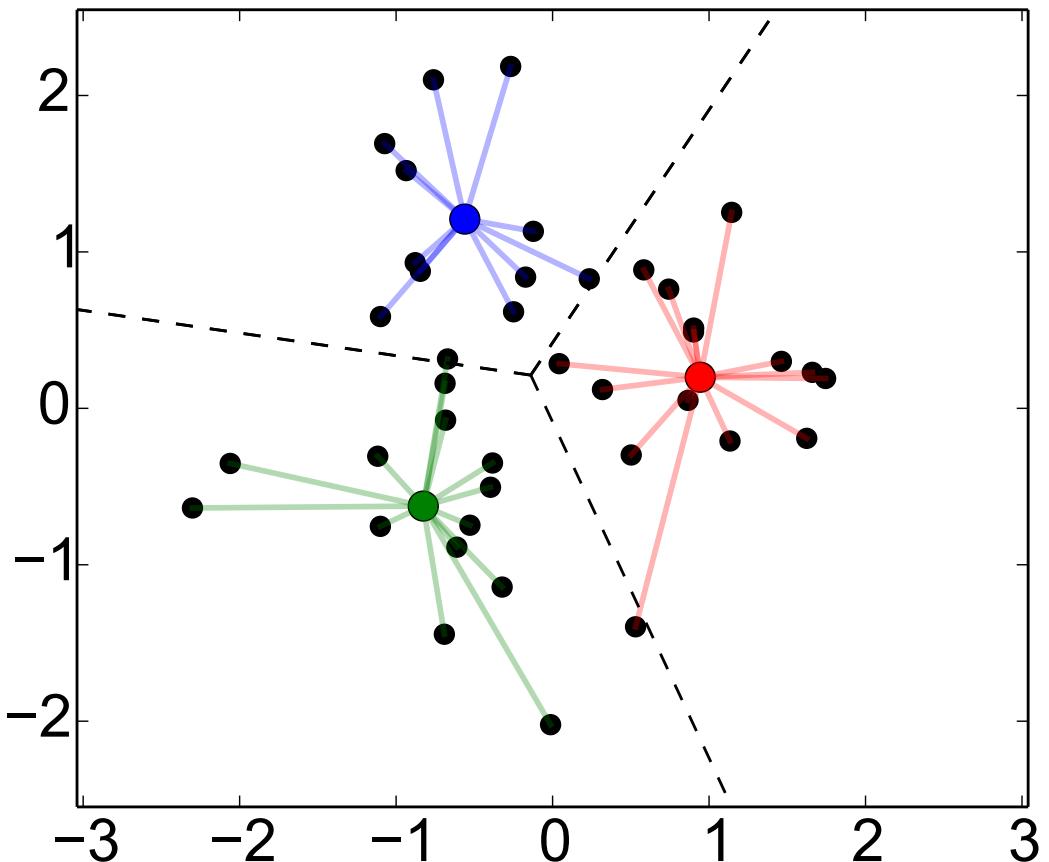
Initial cluster centers (here selected randomly from data points)

- step 1, compute assignments
- step 2, recompute centers
- step 3, recompute assignments
- step 4, recompute centers
- step 5, recompute assignments



K-Means: Example

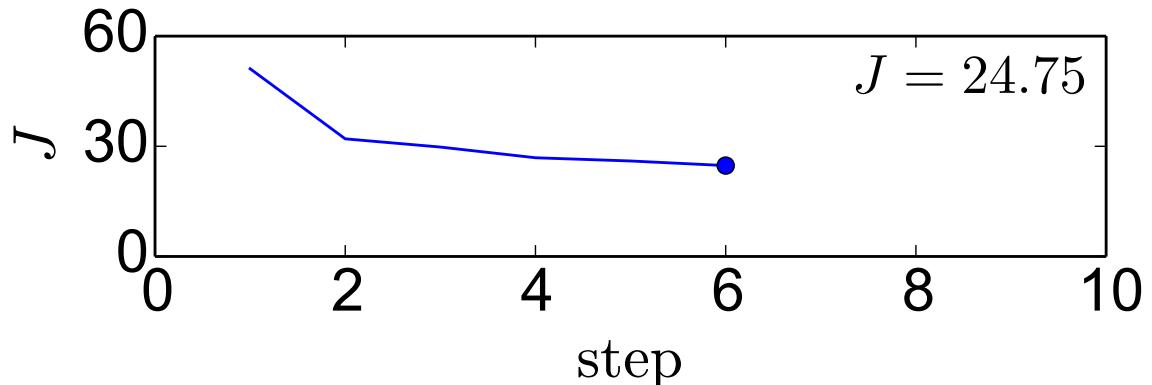
(-- Voronoi boundaries)
● points with changed assignment



Cluster the data points to $K = 3$ clusters

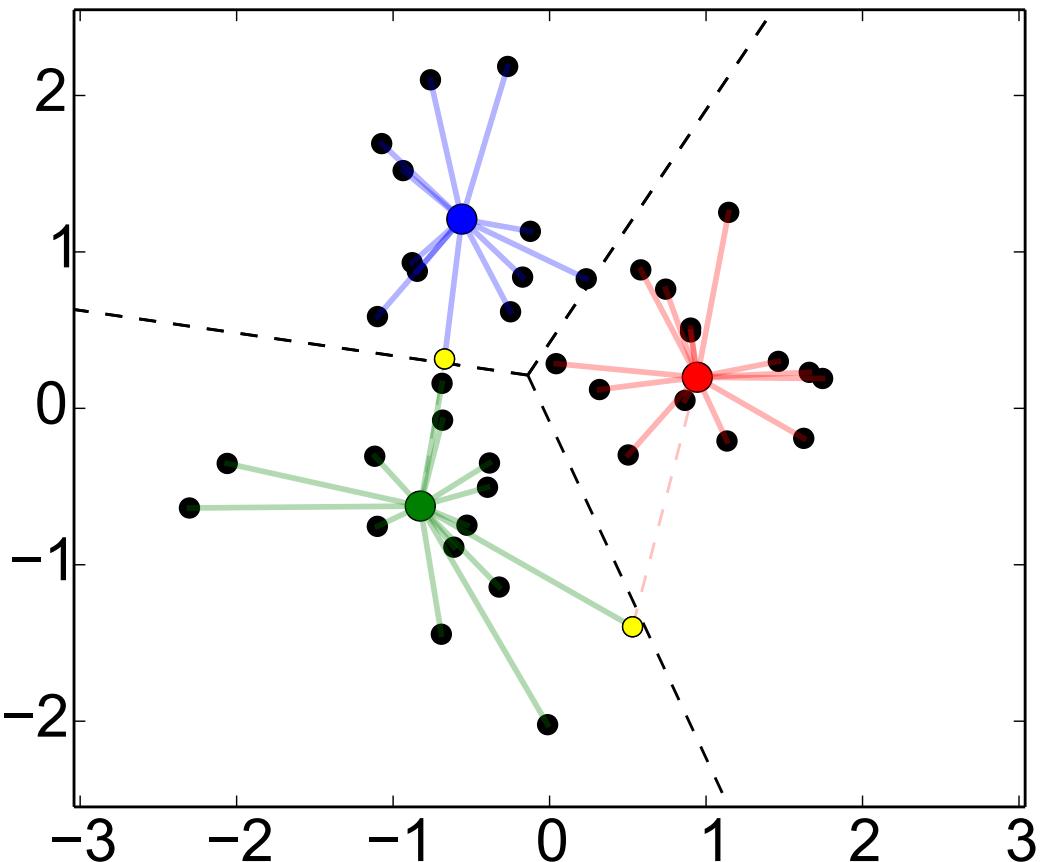
Initial cluster centers (here selected randomly from data points)

- step 1, compute assignments
- step 2, recompute centers
- step 3, recompute assignments
- step 4, recompute centers
- step 5, recompute assignments
- step 6, recompute centers



K-Means: Example

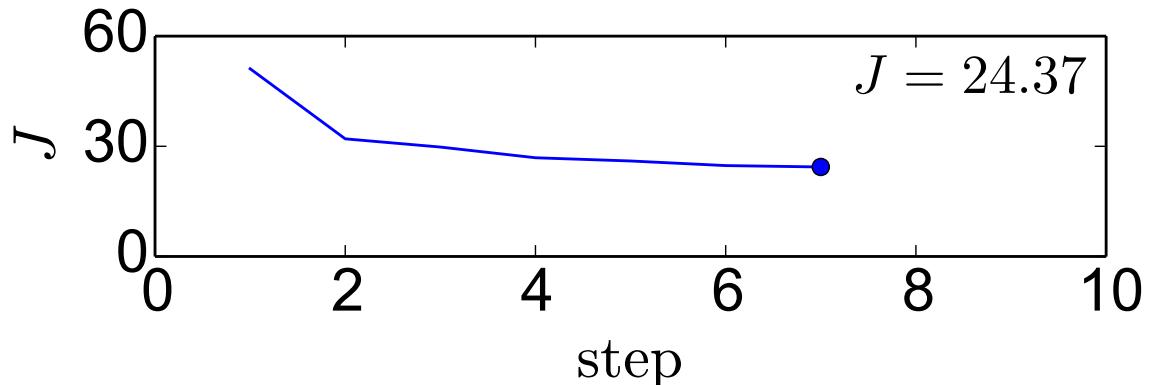
(-- Voronoi boundaries)
● points with changed assignment



Cluster the data points to $K = 3$ clusters

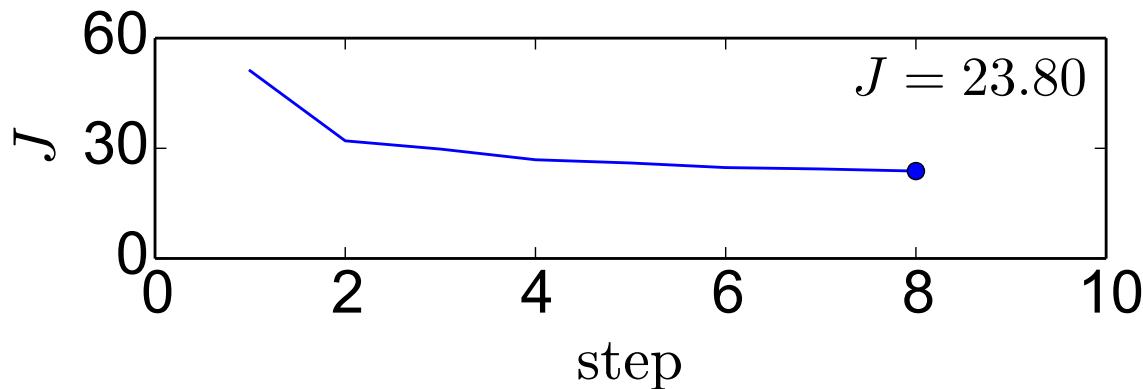
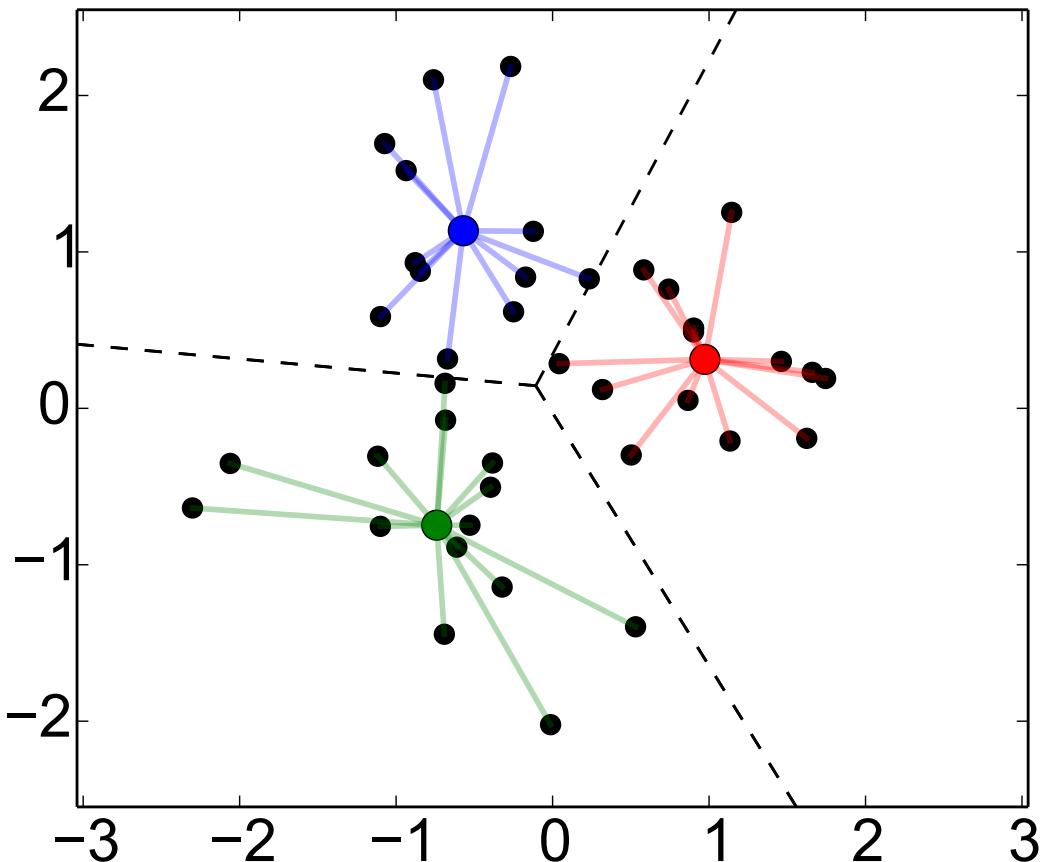
Initial cluster centers (here selected randomly from data points)

- step 1, compute assignments
- step 2, recompute centers
- step 3, recompute assignments
- step 4, recompute centers
- step 5, recompute assignments
- step 6, recompute centers
- step 7, recompute assignments



K-Means: Example

(-- Voronoi boundaries)
 ● points with changed assignment



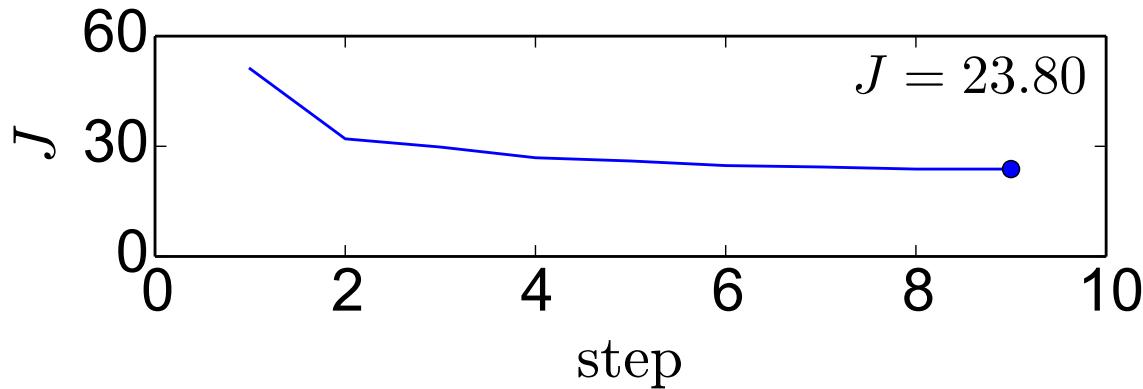
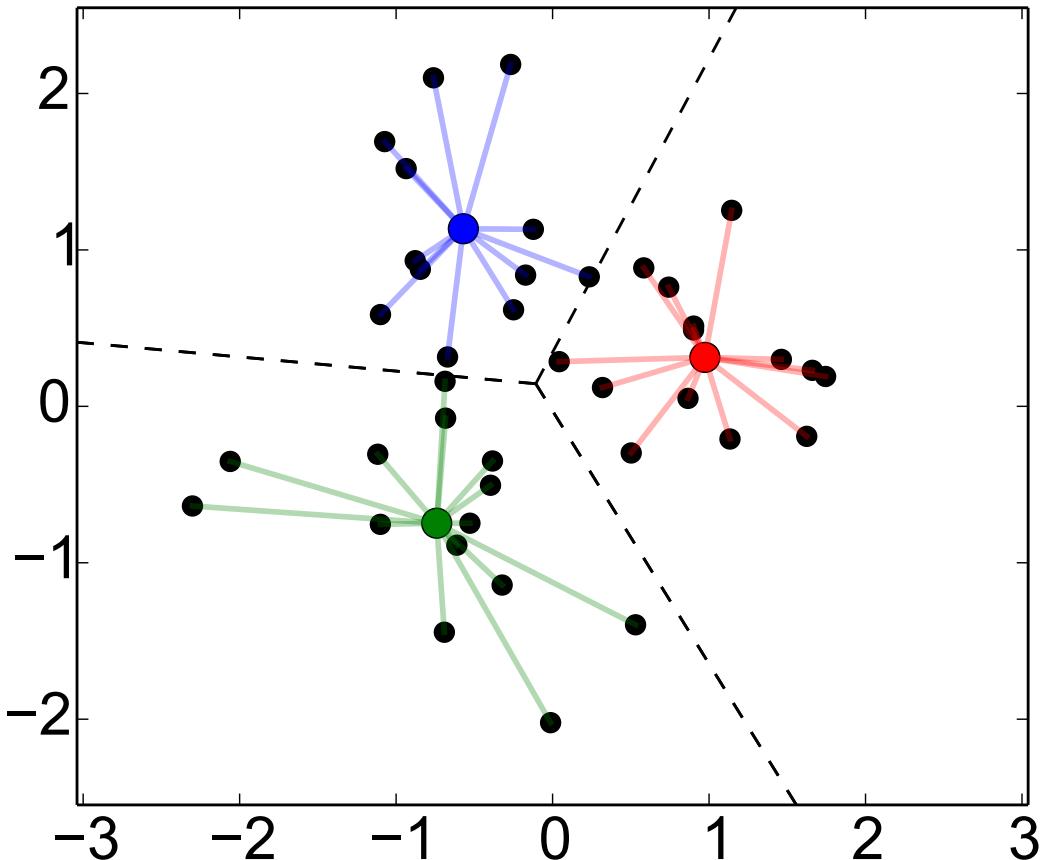
Cluster the data points to $K = 3$ clusters

Initial cluster centers (here selected randomly from data points)

- step 1, compute assignments
- step 2, recompute centers
- step 3, recompute assignments
- step 4, recompute centers
- step 5, recompute assignments
- step 6, recompute centers
- step 7, recompute assignments
- step 8, recompute centers

K-Means: Example

(-- Voronoi boundaries)
● points with changed assignment



Cluster the data points to $K = 3$ clusters

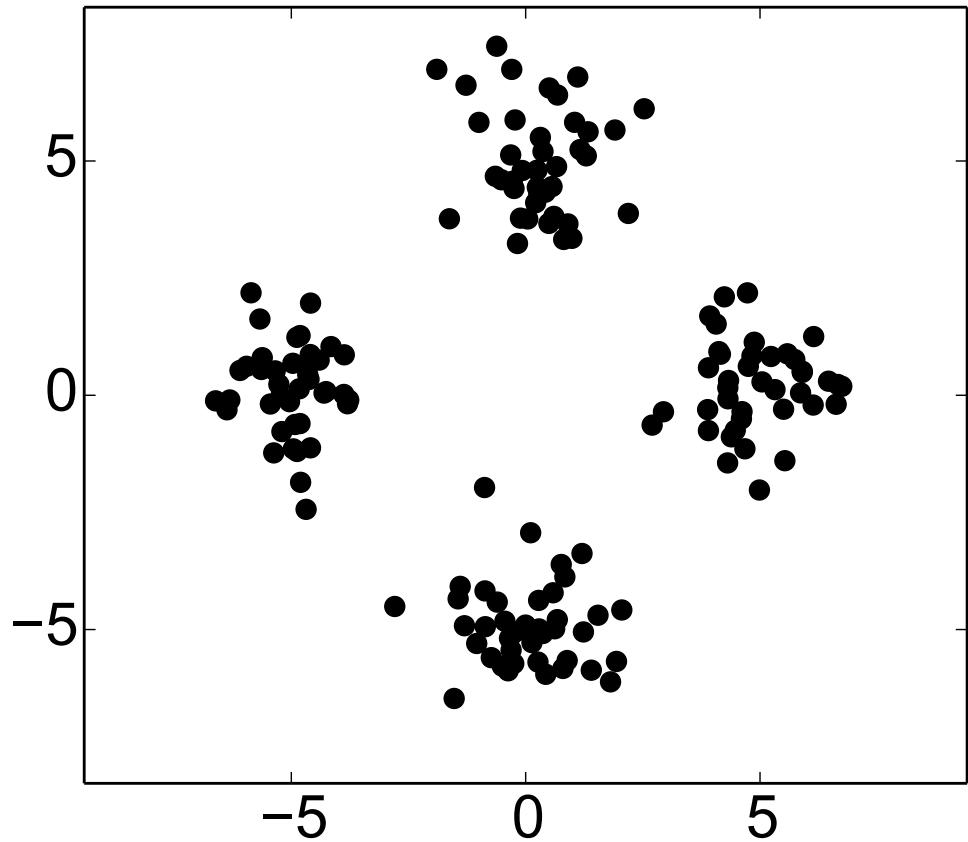
Initial cluster centers (here selected randomly from data points)

- step 1, compute assignments
- step 2, recompute centers
- step 3, recompute assignments
- step 4, recompute centers
- step 5, recompute assignments
- step 6, recompute centers
- step 7, recompute assignments
- step 8, recompute centers
- step 9, recompute assignments

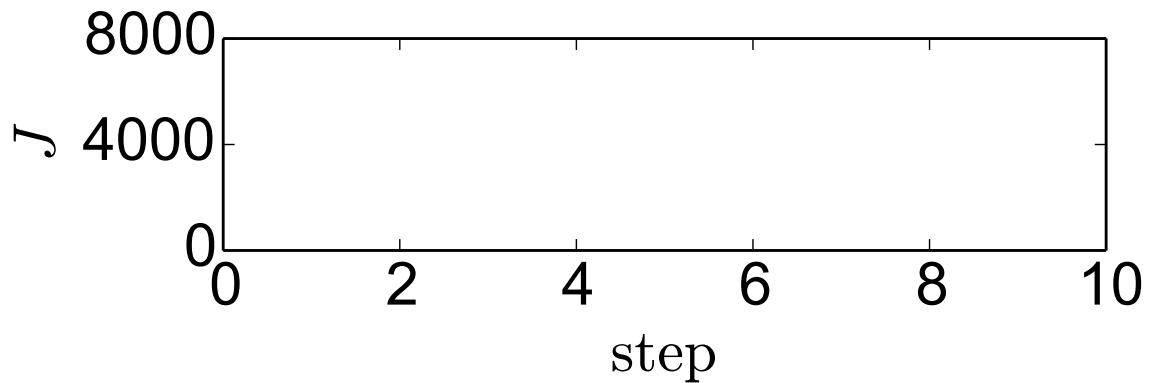
The assignments have not changed.
Done.

K-Means: Example with Reinitialization

data points

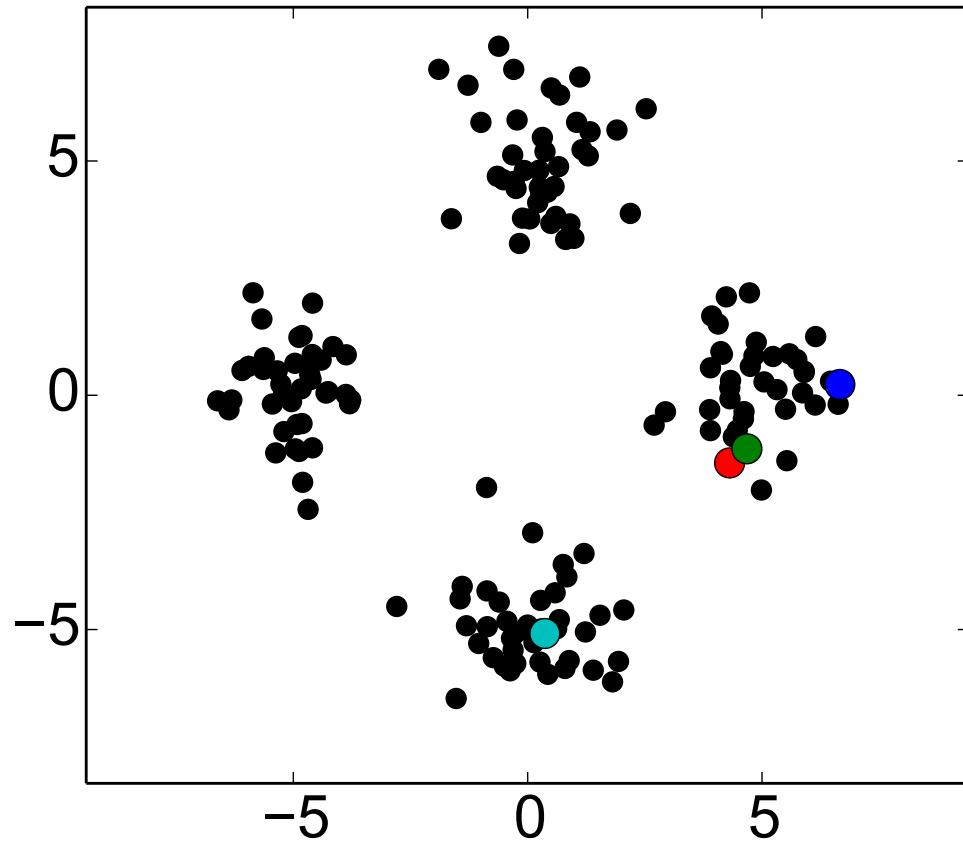


Cluster the data points to $K = 4$ clusters



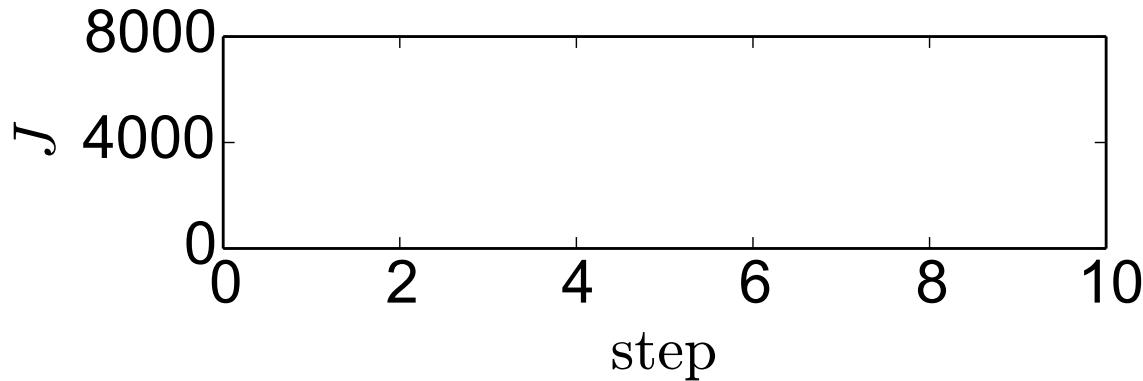
K-Means: Example with Reinitialization

● ● ● ● initial cluster centers



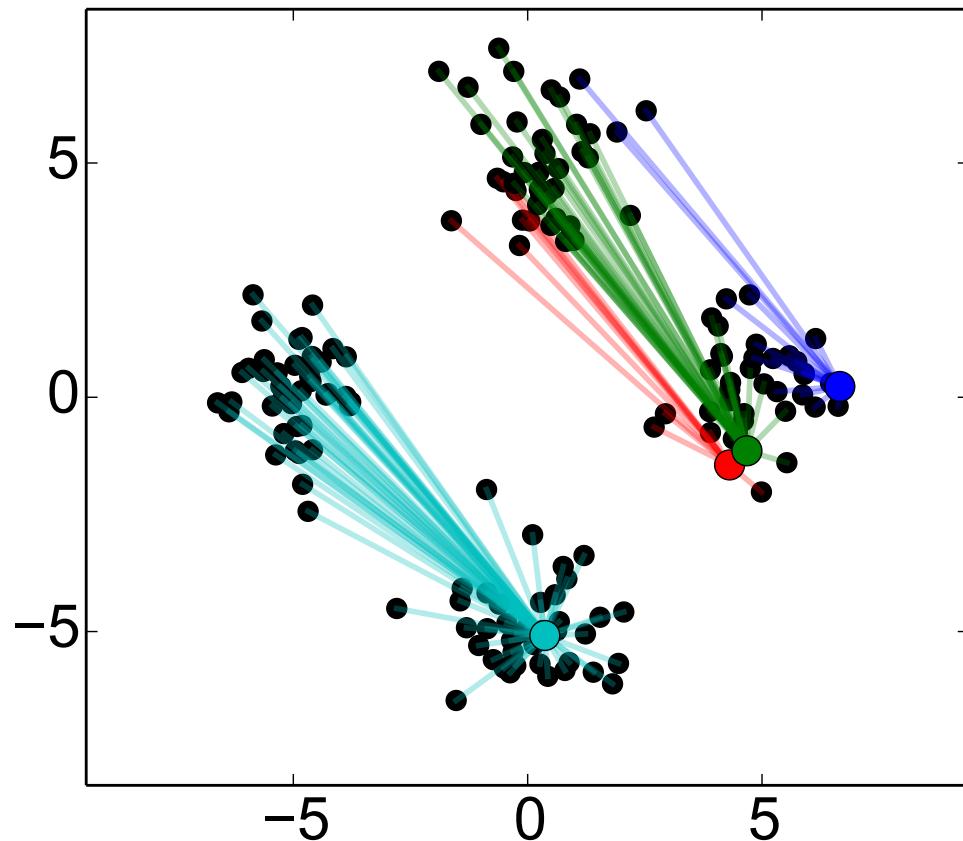
Cluster the data points to $K = 4$ clusters

Initial cluster centers (here selected randomly from data points)



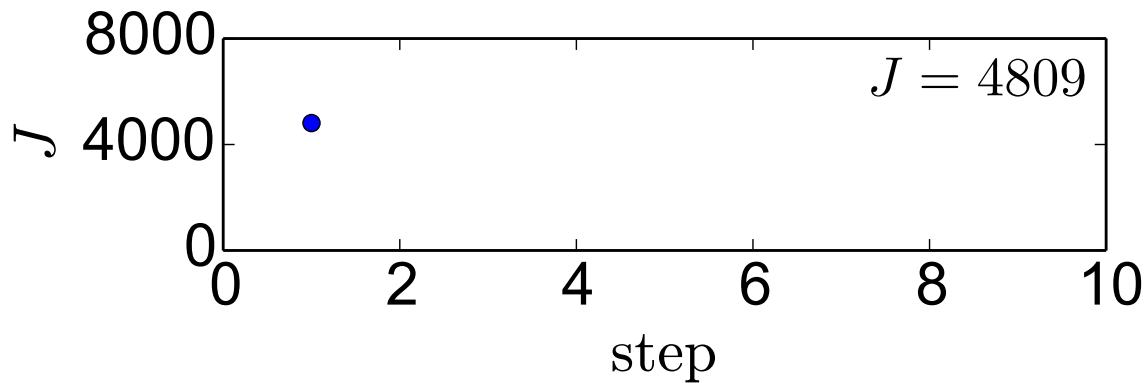
K-Means: Example with Reinitialization

(-- Voronoi boundaries)



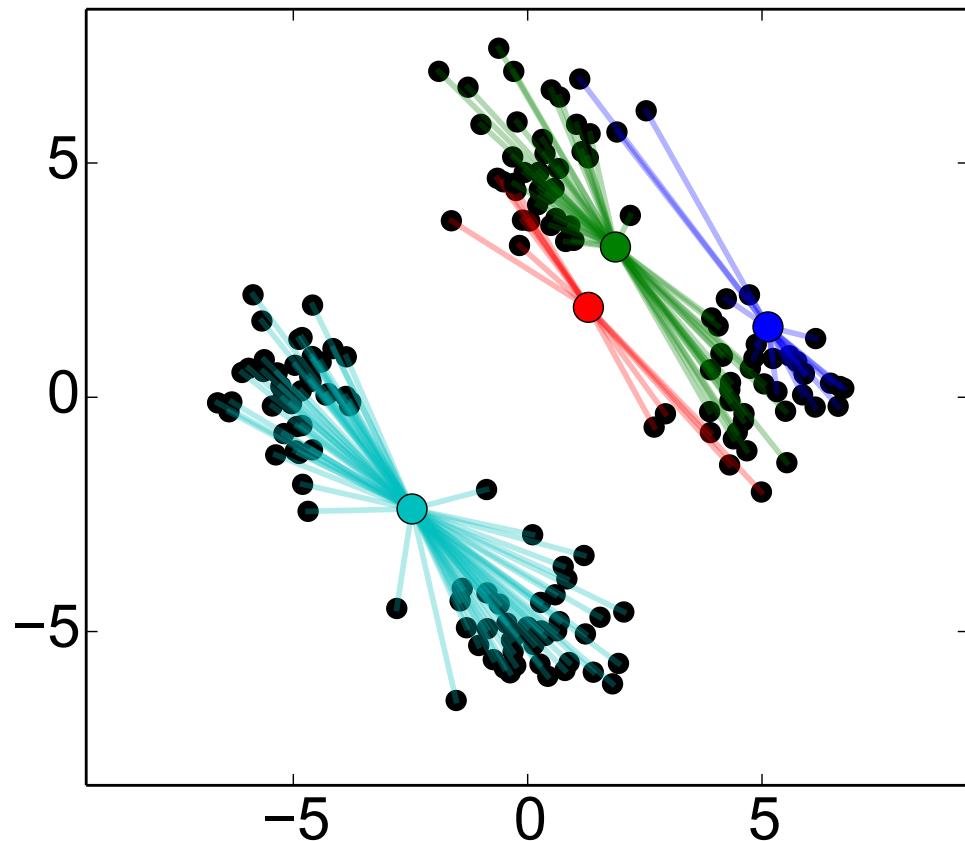
Cluster the data points to $K = 4$ clusters

Step 1, compute assignments



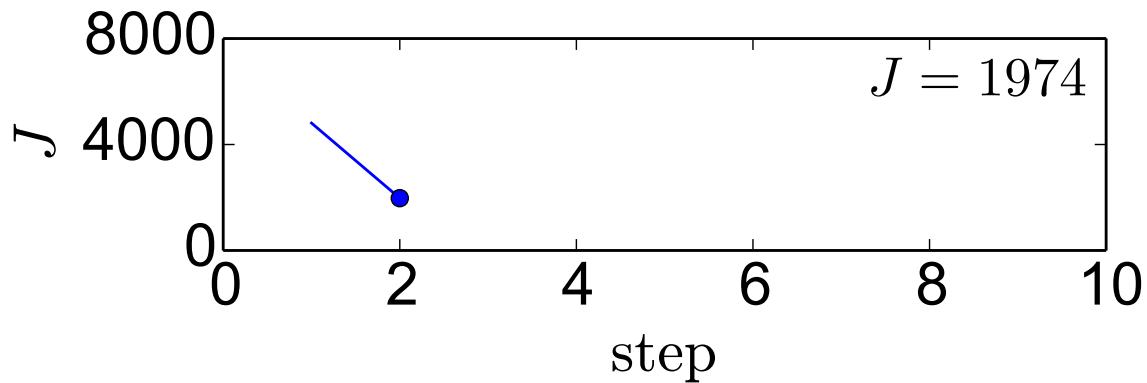
K-Means: Example with Reinitialization

(-- Voronoi boundaries)



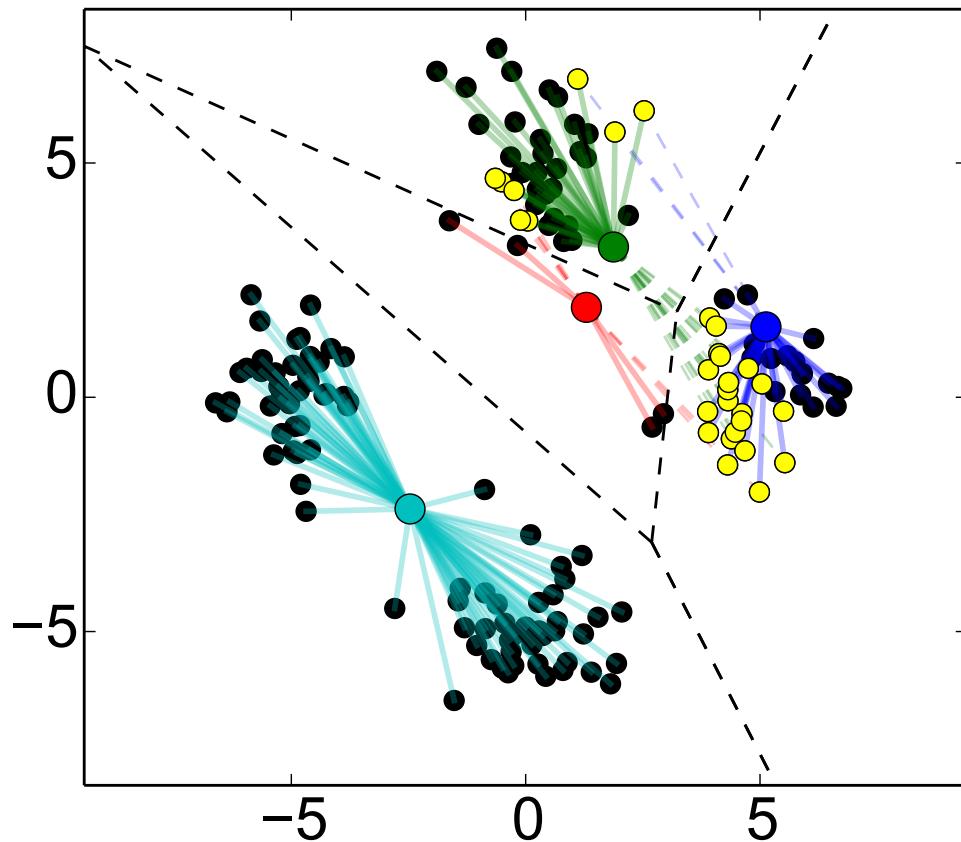
Cluster the data points to $K = 4$ clusters

Step 2, recompute centres



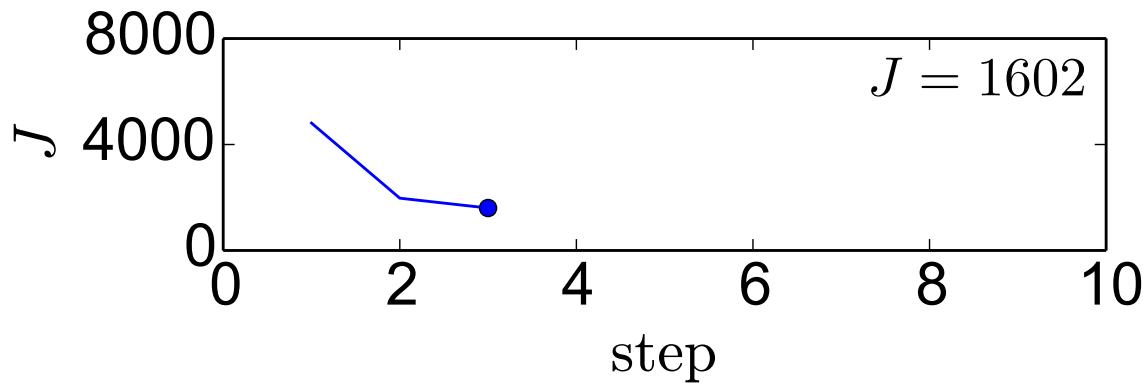
K-Means: Example with Reinitialization

(-- Voronoi boundaries)



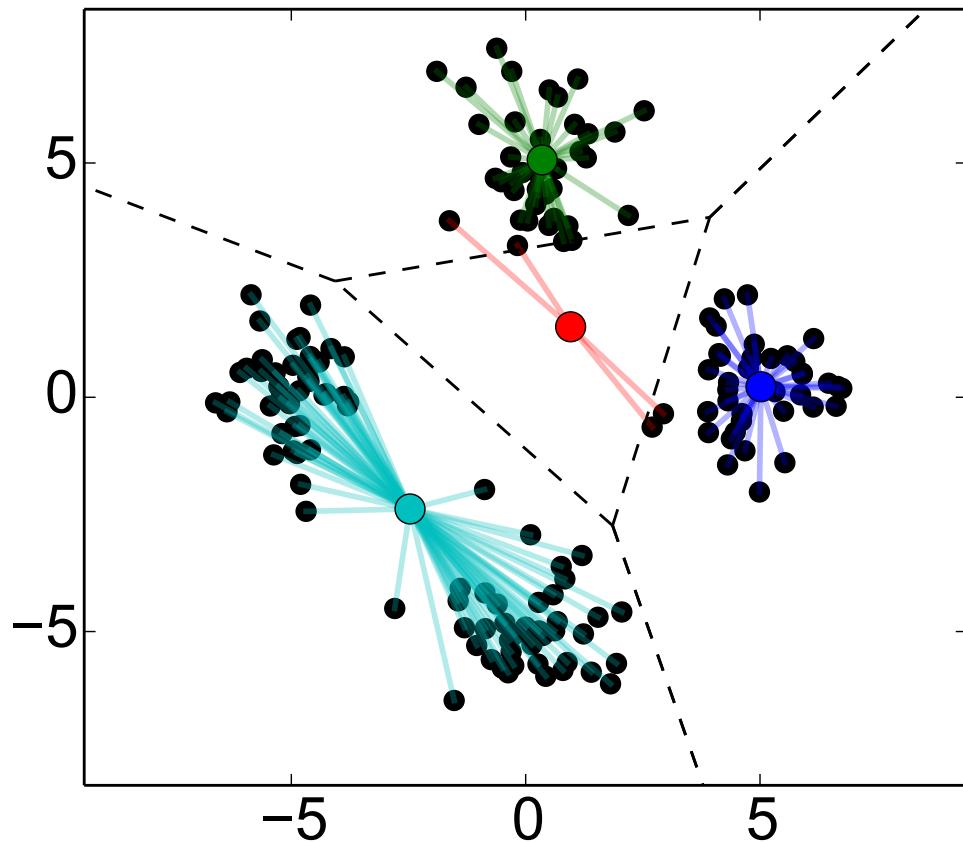
Cluster the data points to $K = 4$ clusters

Step 3, recompute assignments



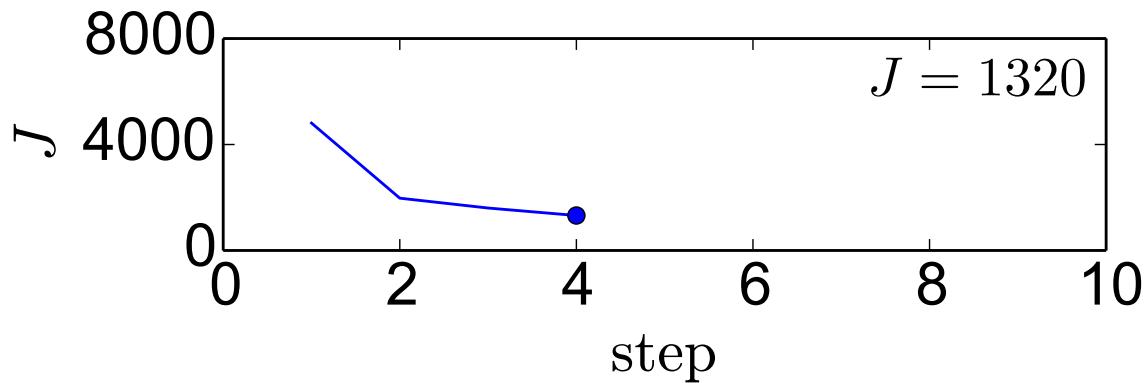
K-Means: Example with Reinitialization

(-- Voronoi boundaries)



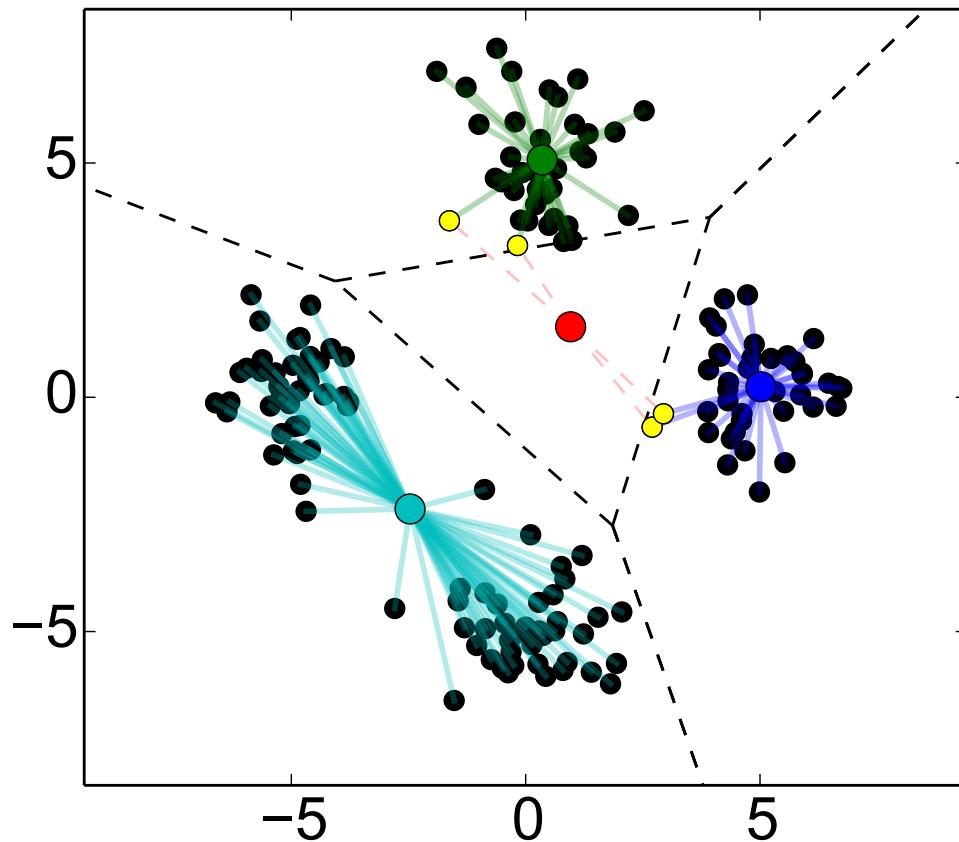
Cluster the data points to $K = 4$ clusters

Step 4, recompute centres



K-Means: Example with Reinitialization

(-- Voronoi boundaries)

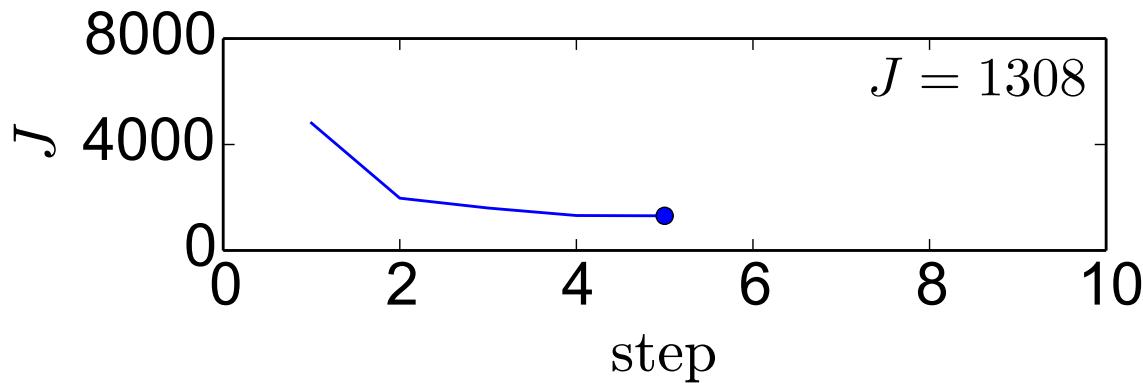


Cluster the data points to $K = 4$ clusters

Step 5, recompute assignments

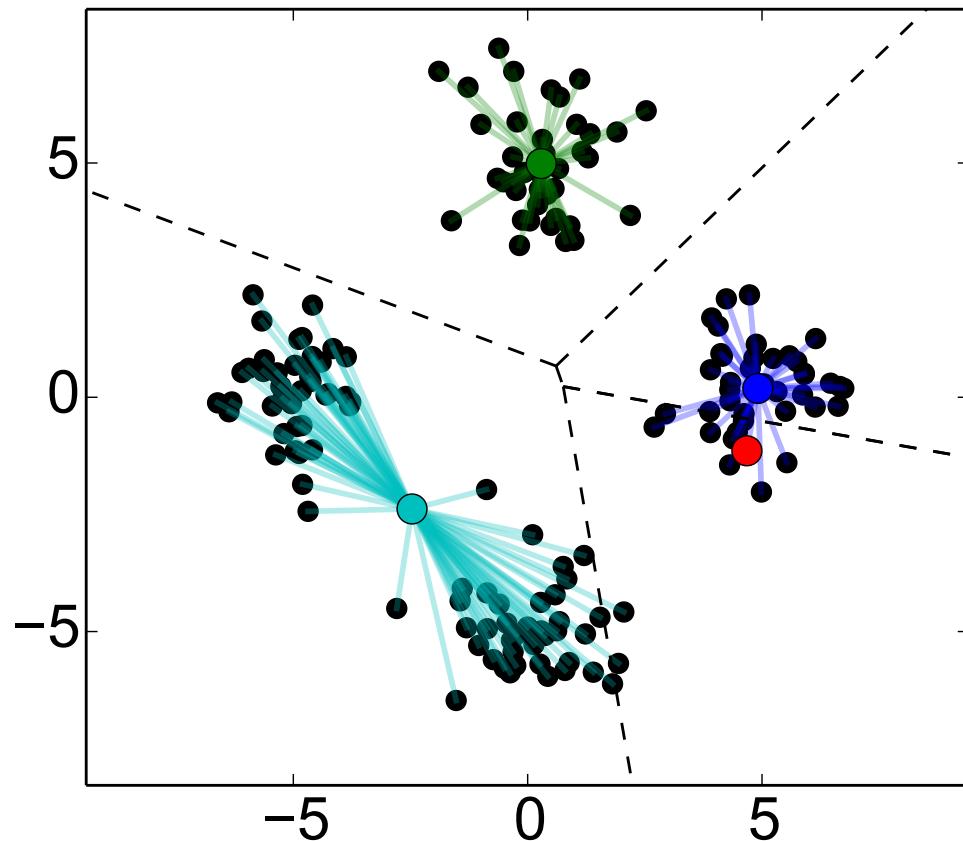
No points assigned to cluster ●

⇒ it will be reinitialized.



K-Means: Example with Reinitialization

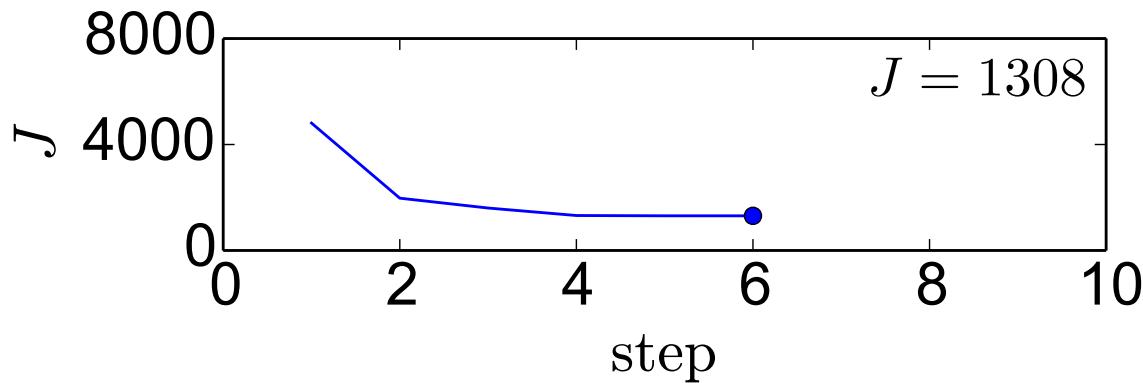
(-- Voronoi boundaries)



Cluster the data points to $K = 4$ clusters

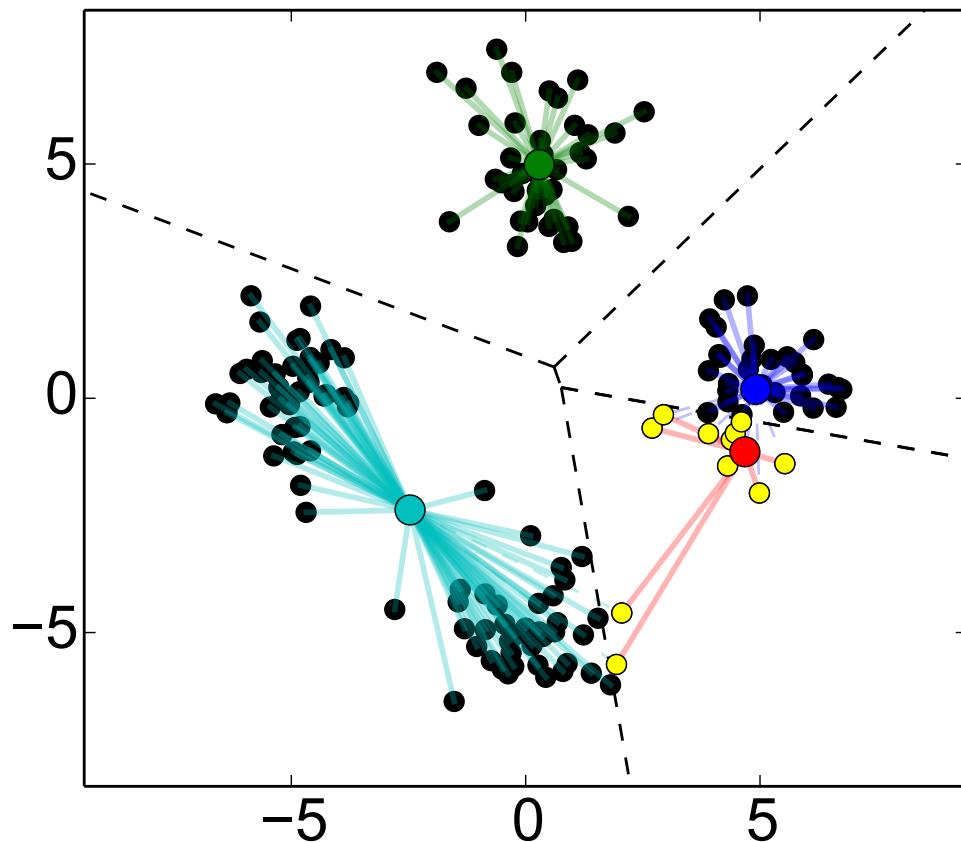
Step 6

- recompute ● ● ●
- reinitialize ●



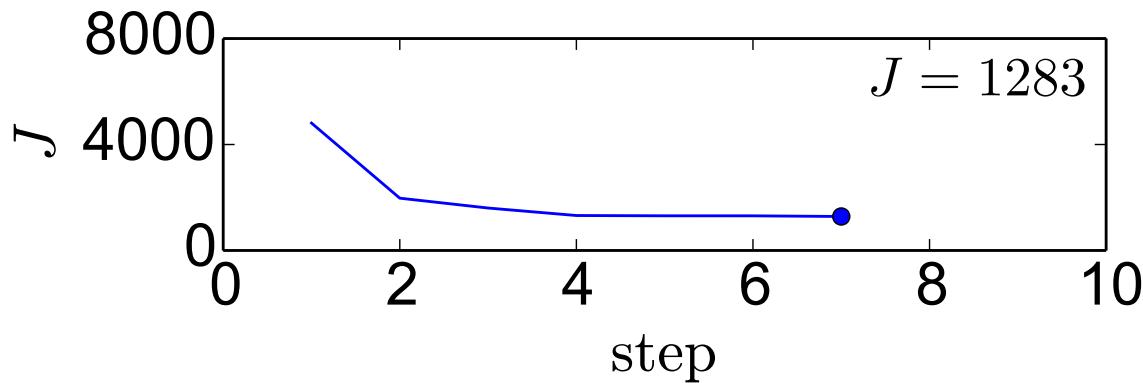
K-Means: Example with Reinitialization

(-- Voronoi boundaries)



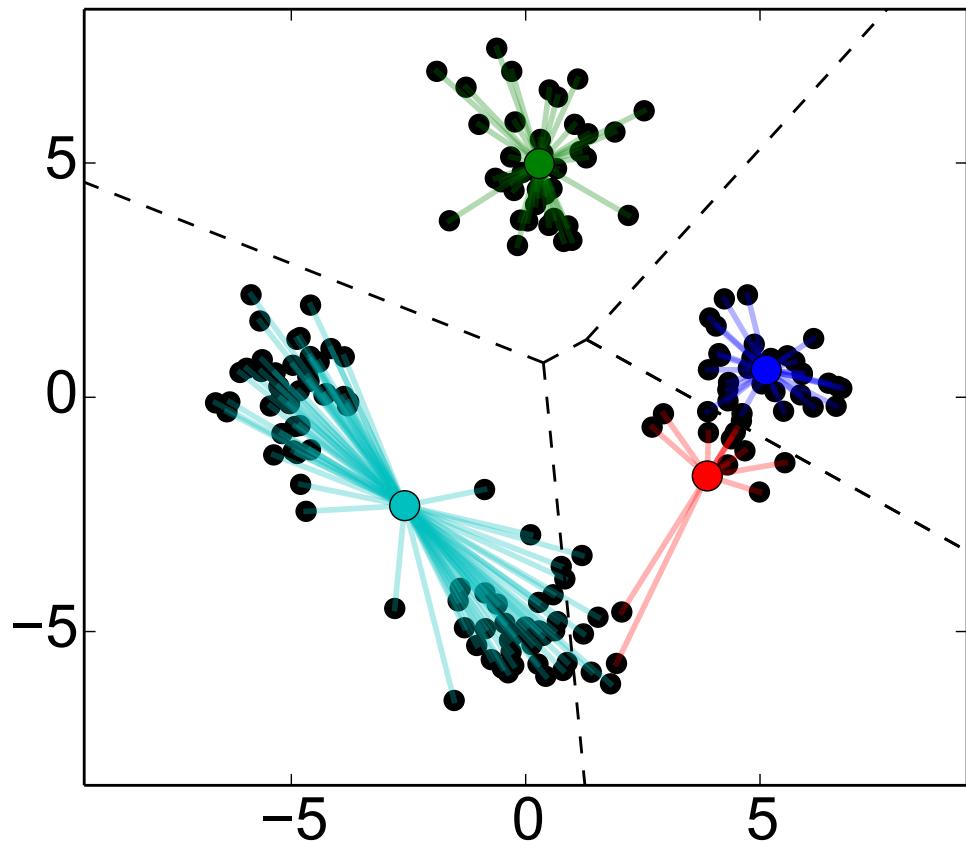
Cluster the data points to $K = 4$ clusters

Step 7, recompute assignments



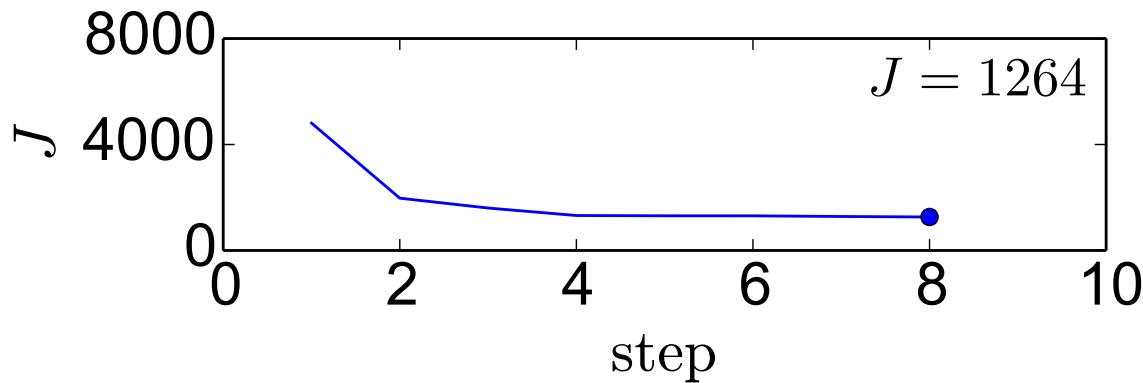
K-Means: Example with Reinitialization

(-- Voronoi boundaries)



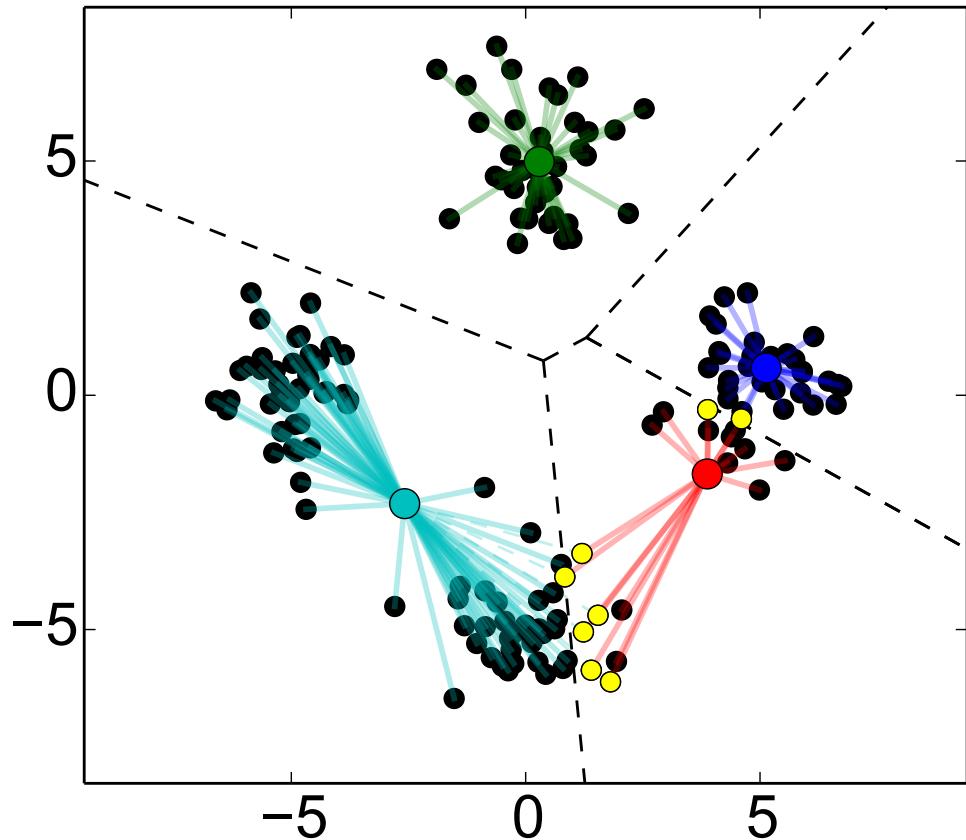
Cluster the data points to $K = 4$ clusters

Step 8, recompute centres



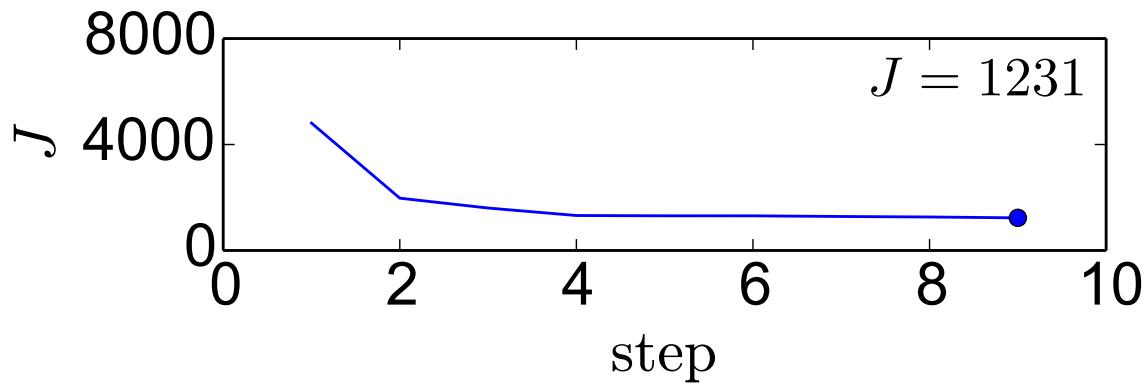
K-Means: Example with Reinitialization

(-- Voronoi boundaries)



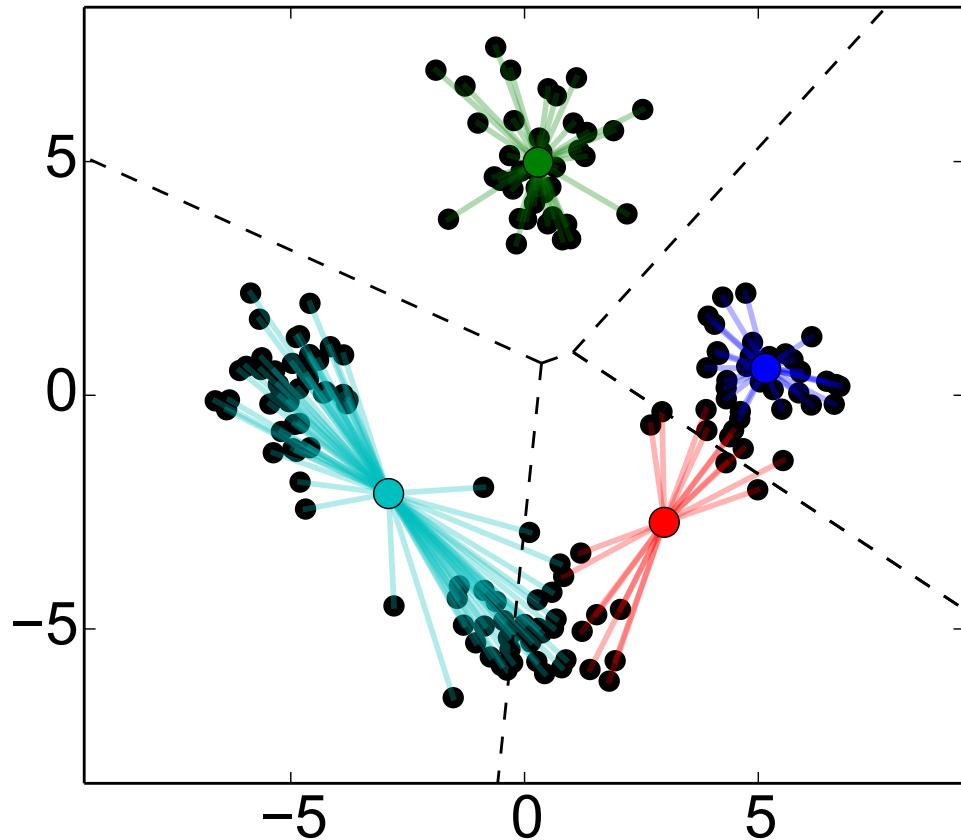
Cluster the data points to $K = 4$ clusters

Step 9, recompute assignments



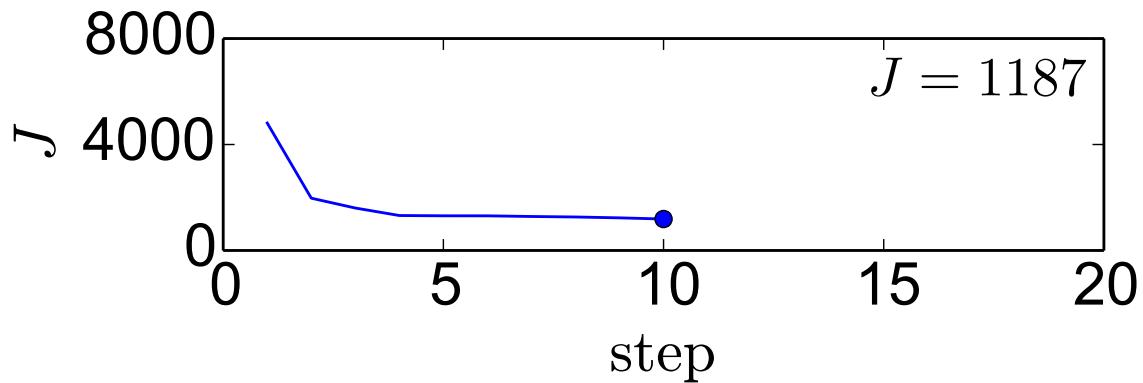
K-Means: Example with Reinitialization

(-- Voronoi boundaries)



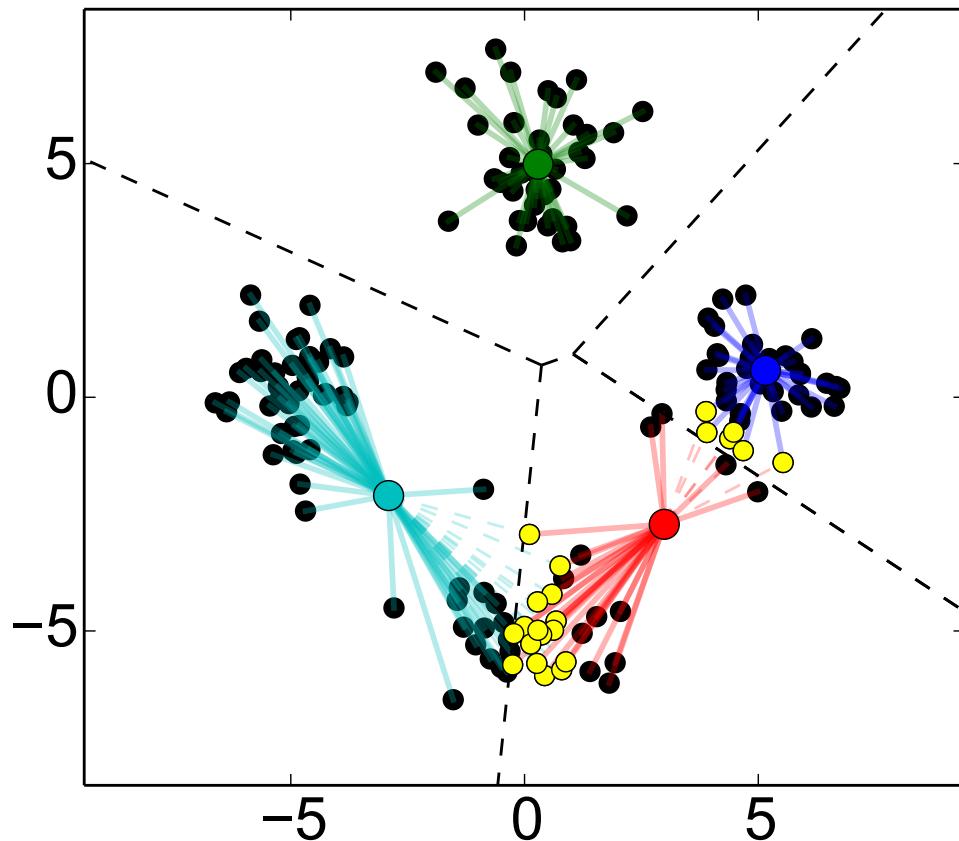
Cluster the data points to $K = 4$ clusters

Step 10, recompute centres



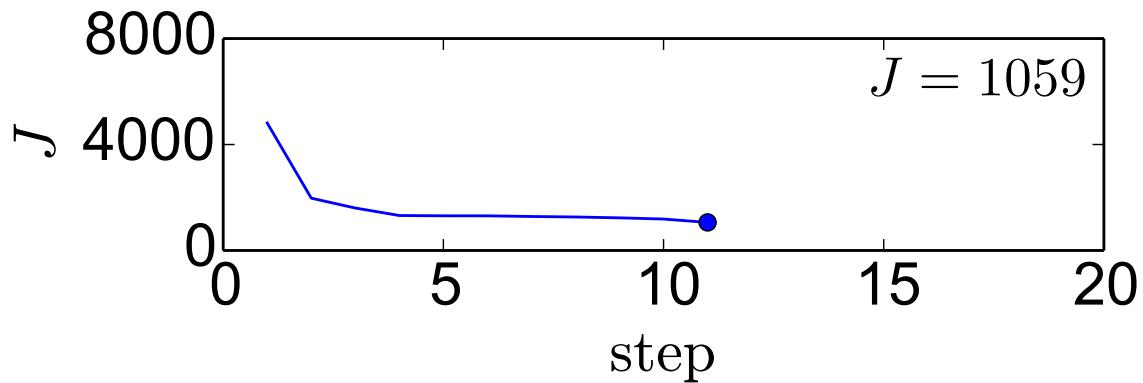
K-Means: Example with Reinitialization

(-- Voronoi boundaries)



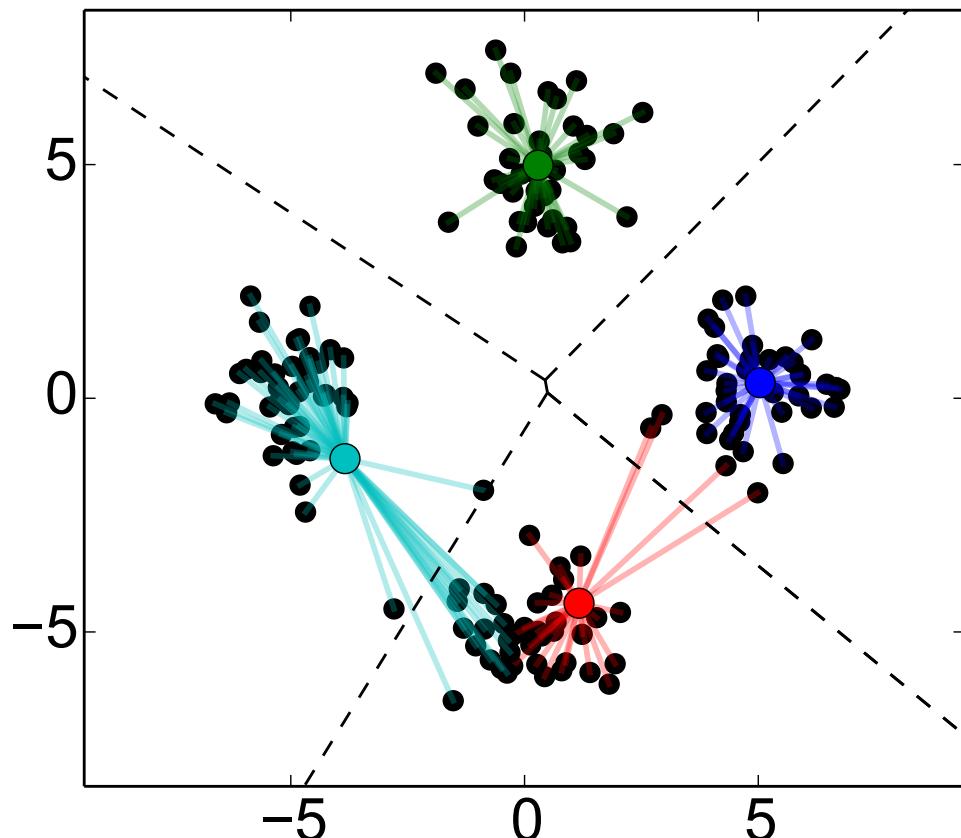
Cluster the data points to $K = 4$ clusters

Step 11, recompute assignments



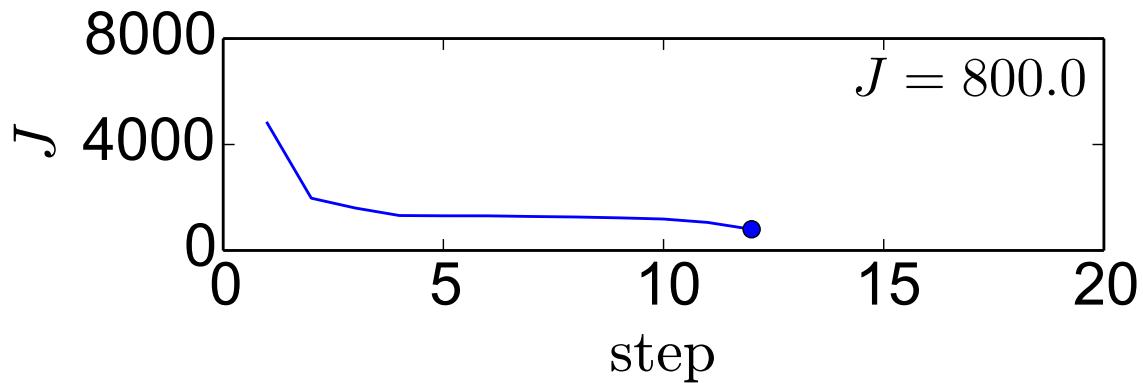
K-Means: Example with Reinitialization

(-- Voronoi boundaries)



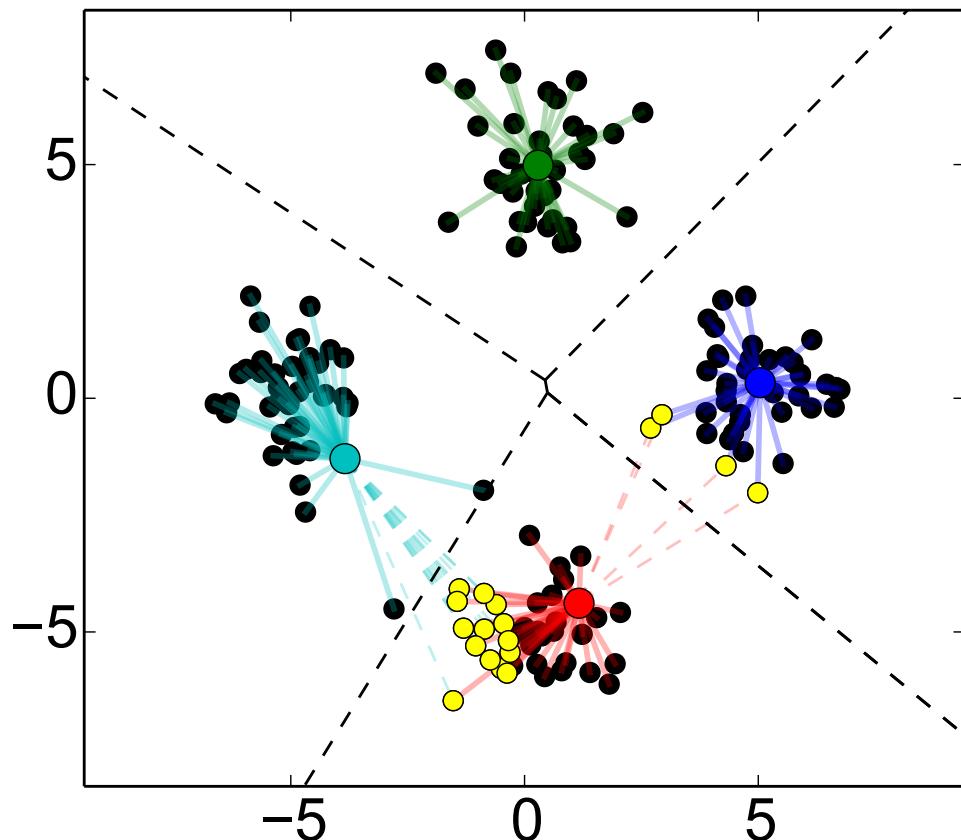
Cluster the data points to $K = 4$ clusters

Step 12, recompute centres



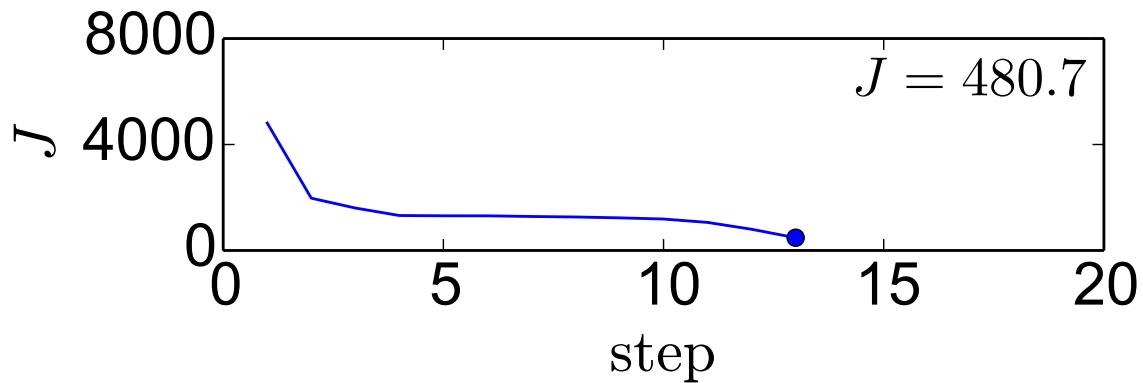
K-Means: Example with Reinitialization

(-- Voronoi boundaries)



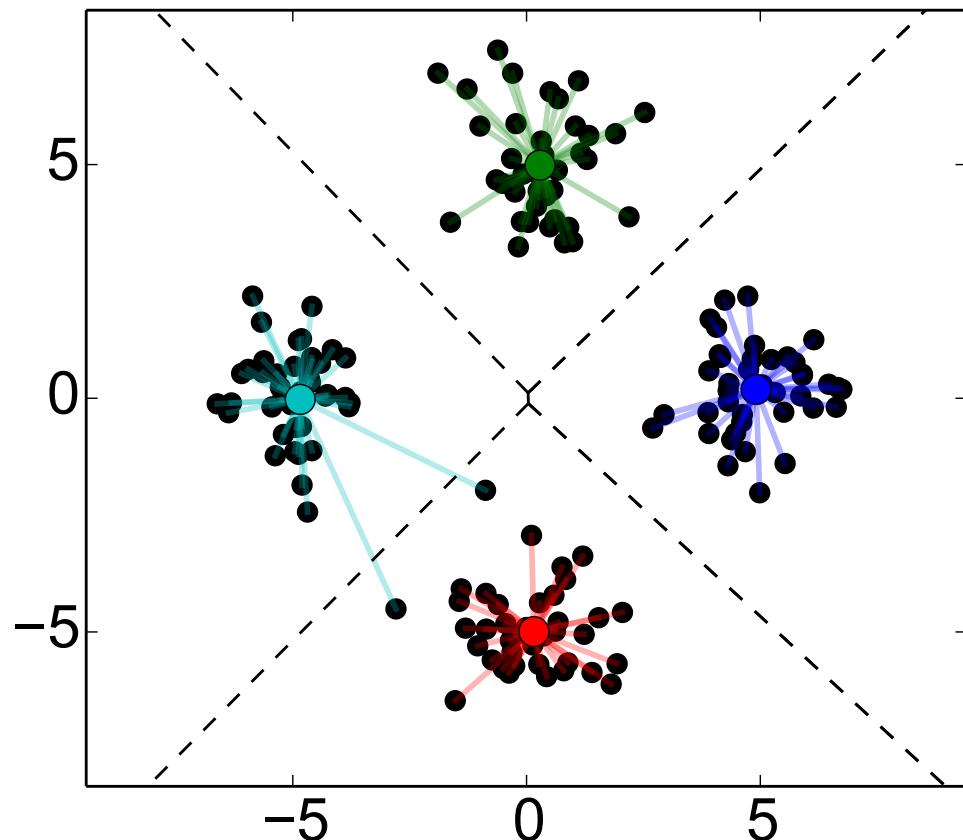
Cluster the data points to $K = 4$ clusters

Step 13, recompute assignments



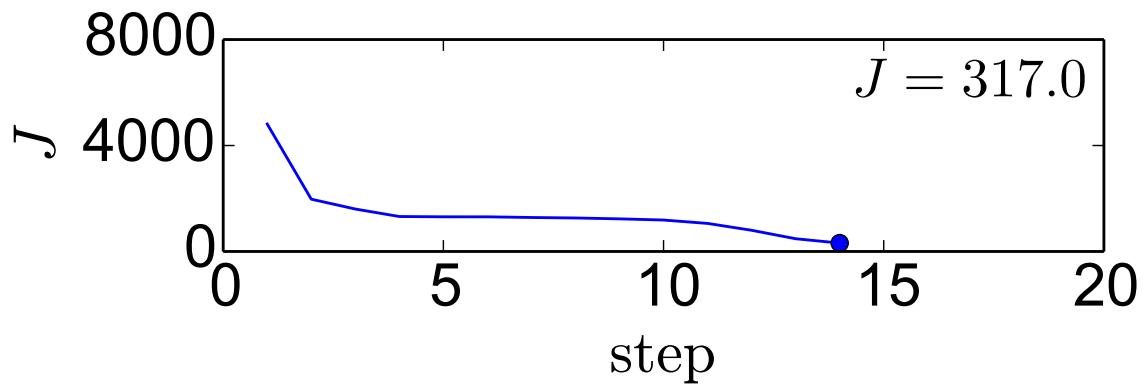
K-Means: Example with Reinitialization

(-- Voronoi boundaries)



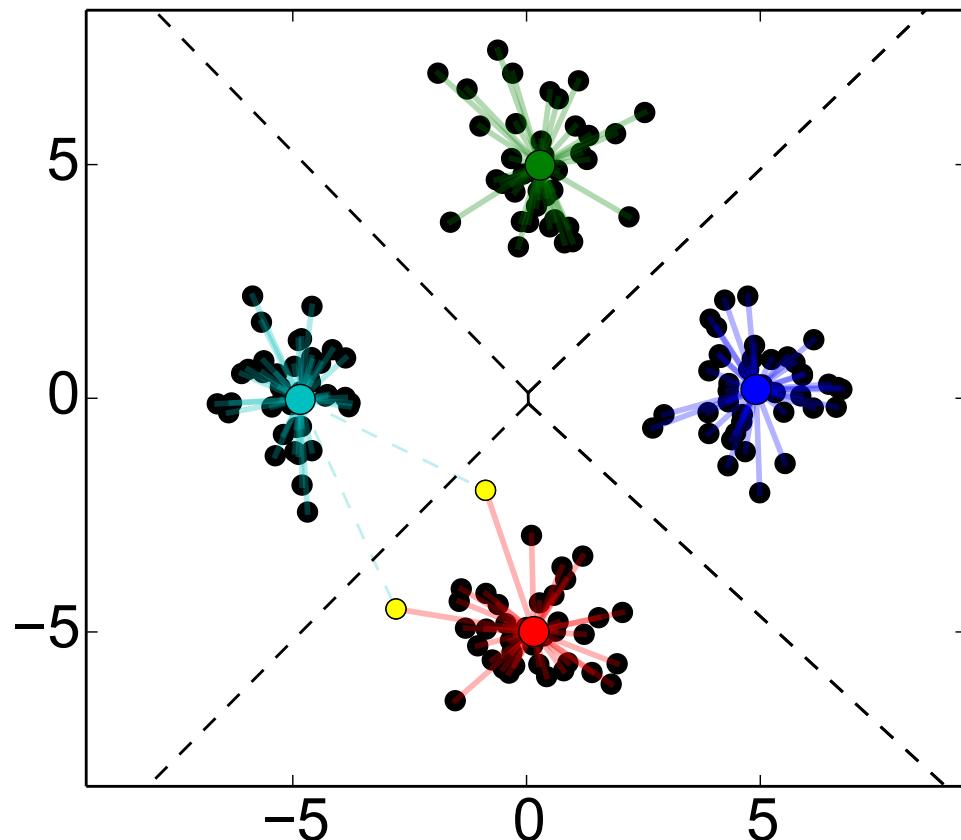
Cluster the data points to $K = 4$ clusters

Step 14, recompute centres



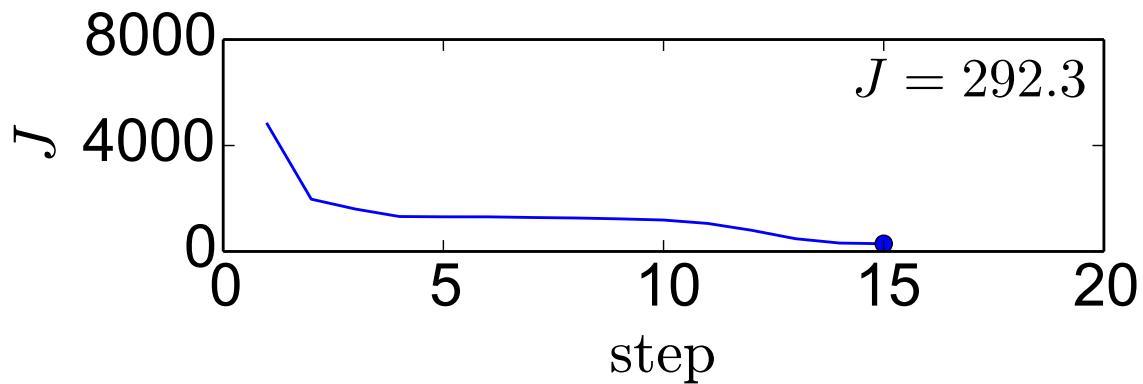
K-Means: Example with Reinitialization

(-- Voronoi boundaries)



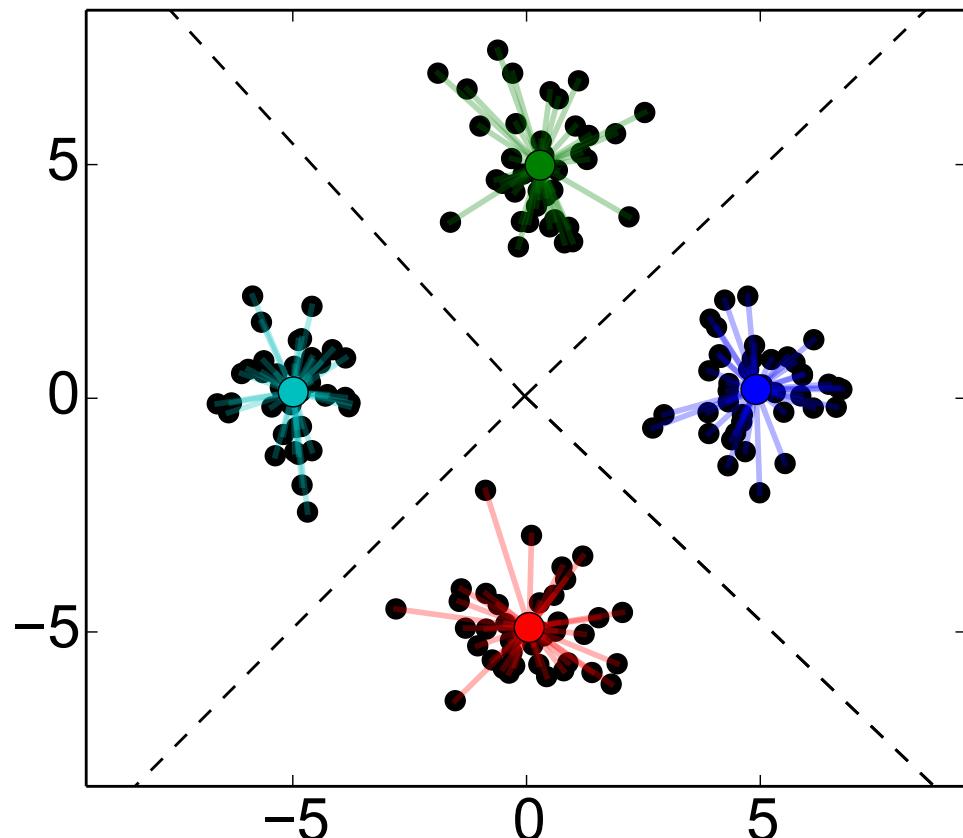
Cluster the data points to $K = 4$ clusters

Step 15, recompute assignments



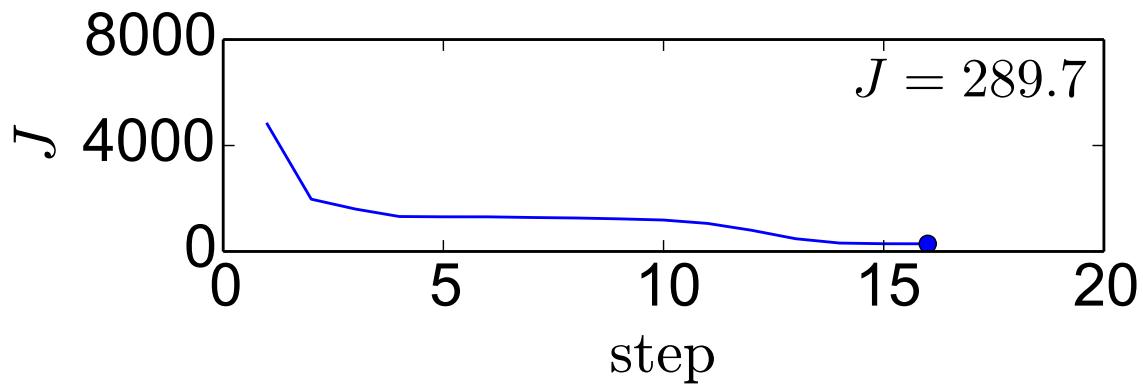
K-Means: Example with Reinitialization

(-- Voronoi boundaries)



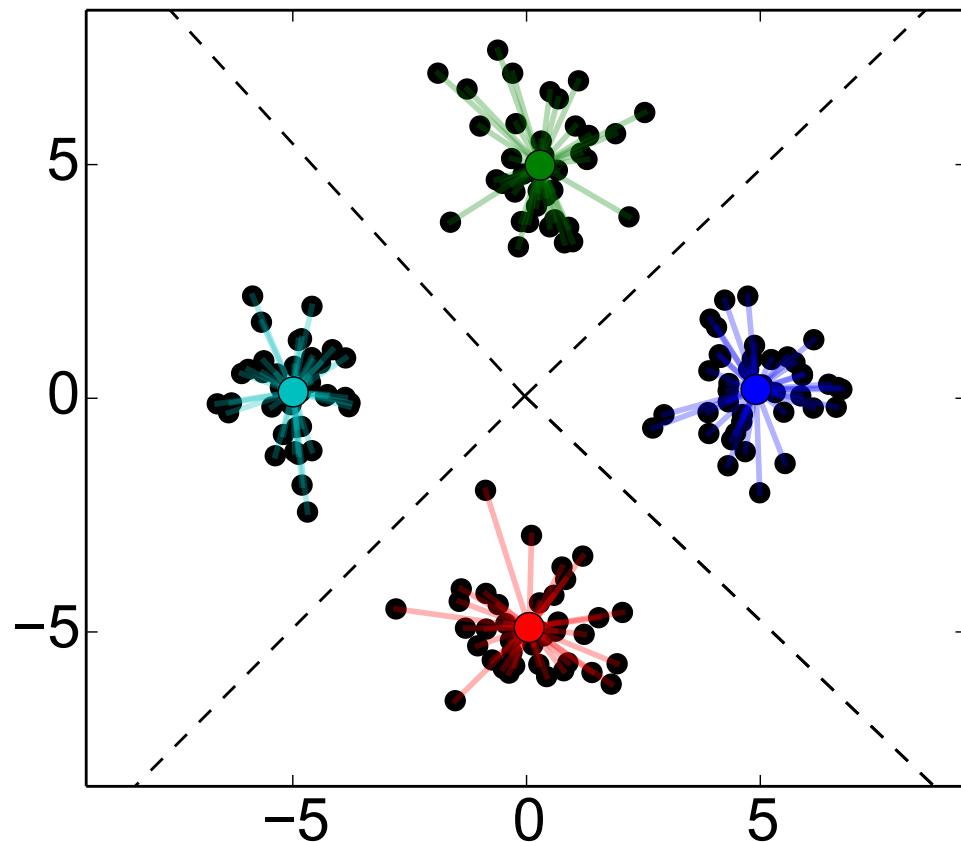
Cluster the data points to $K = 4$ clusters

Step 16, recompute centres



K-Means: Example with Reinitialization

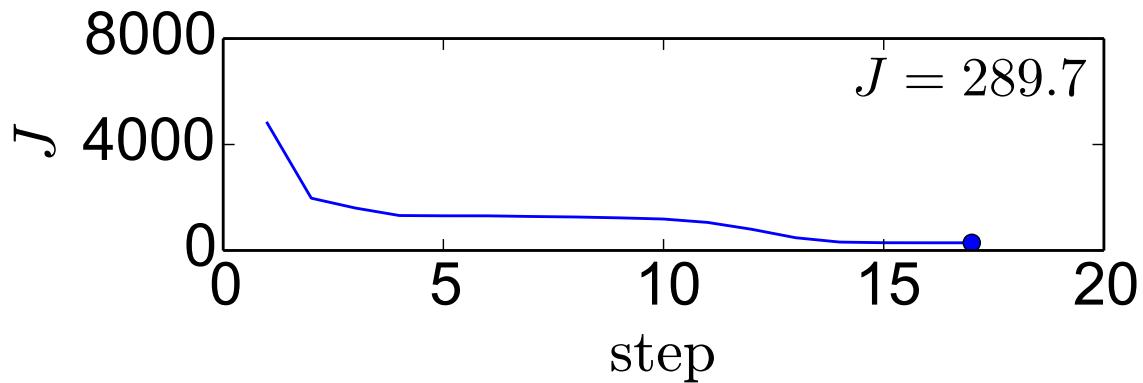
(-- Voronoi boundaries)



Cluster the data points to $K = 4$ clusters

Step 17, recompute assignments

Assignments haven't changed. **Done.**



K-Means: Properties

Clustering criterion: $J(\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K; \mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_K) = \sum_{k=1}^K \sum_{\mathbf{x} \in \mathcal{T}_k} \|\mathbf{x} - \mathbf{c}_k\|^2.$

K-means algorithm skeleton:

1. Initialization
2. Assignment optimization (assign to closest etalon)
3. Cluster centres optimization (\mathbf{c}_k set to average of data in \mathcal{T}_k)
4. Goto 2 if the assignments have changed

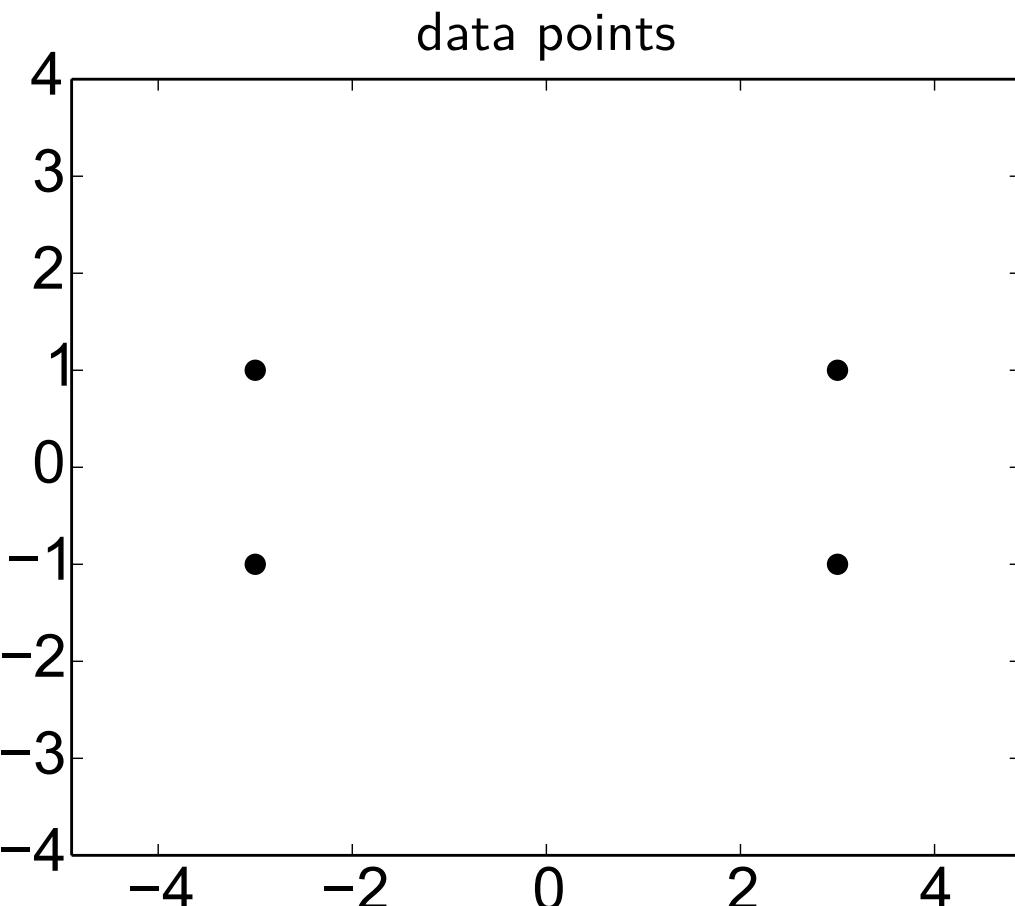
Convergence:

- ◆ During the run of the algorithm, J monotonically decreases because:
 - Step 2: The contribution of each \mathbf{x}_l to J either stays the same, or gets lower,
 - Step 3: For a fixed assignment \mathcal{T}_k , the mean of the data points in \mathcal{T}_k is the optimal solution under the least squares criterion J . If \mathcal{T}_k is empty and re-initialization is done for \mathbf{c}_k then this has no effect on J at this point, but it can only cause additional decrease in J in the subsequent Step 2.
- ◆ Since there is a **finite number of assignments** (how many?) and no assignment is visited twice (why?), the K-means algorithm **reaches a local minimum** after a **finite** number of steps.

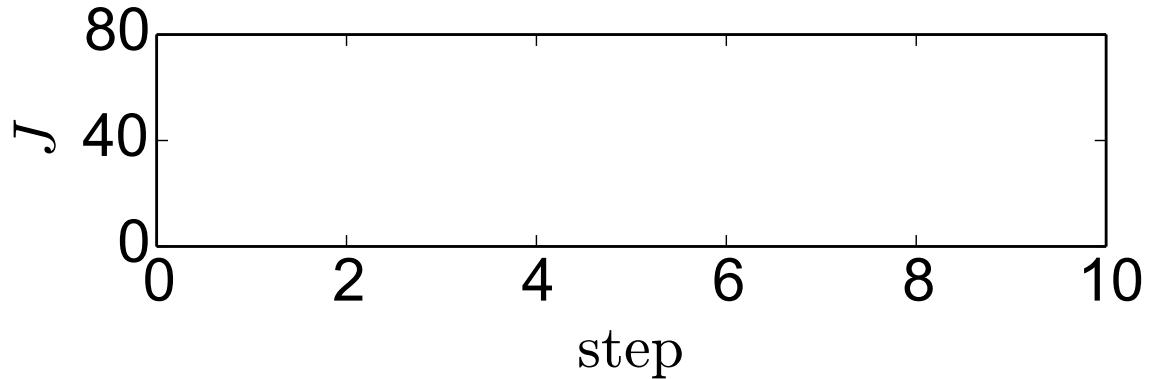
K-Means: Notes

- ◆ K-means is clearly not a guaranteed global minimum minimizer.
- ◆ In theory, there may be a problem of infinite looping through a set of assignments with equal J , but this is not the case when breaking the ties in Step 2 is done consistently (e.g. assigning to cluster with the lowest index if a point is equidistant to multiple cluster centres.)
- ◆ As for the computational time, the complexity of assignment computation dominates, as for every observation the nearest prototype is sought. Trivially implemented, this requires $O(LK)$ distance computations per iteration. Any idea for a speed-up?
- ◆ The algorithm is sometimes modified in a way that initialization is done by setting \mathcal{T}_k 's and swapping steps 2 and 3 in the iteration loop.

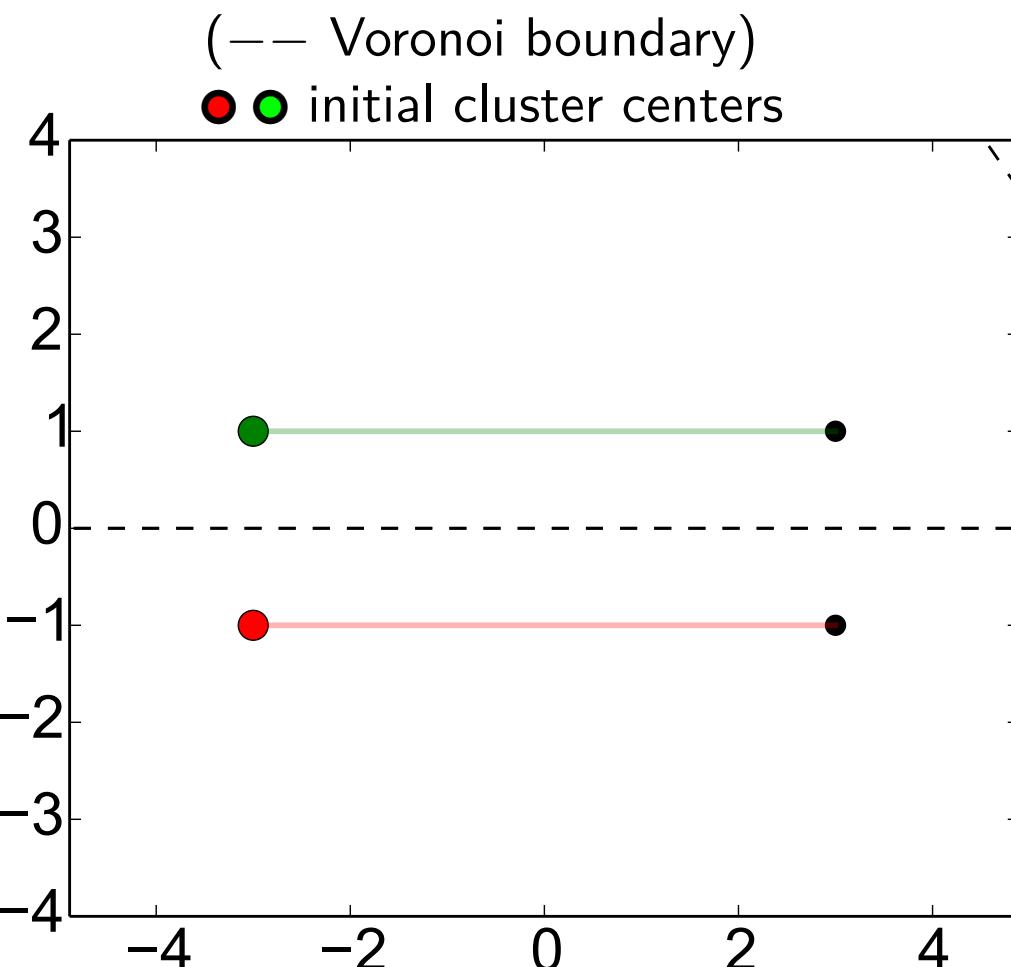
Example of Local Minima



Cluster the data points to $K = 2$ clusters

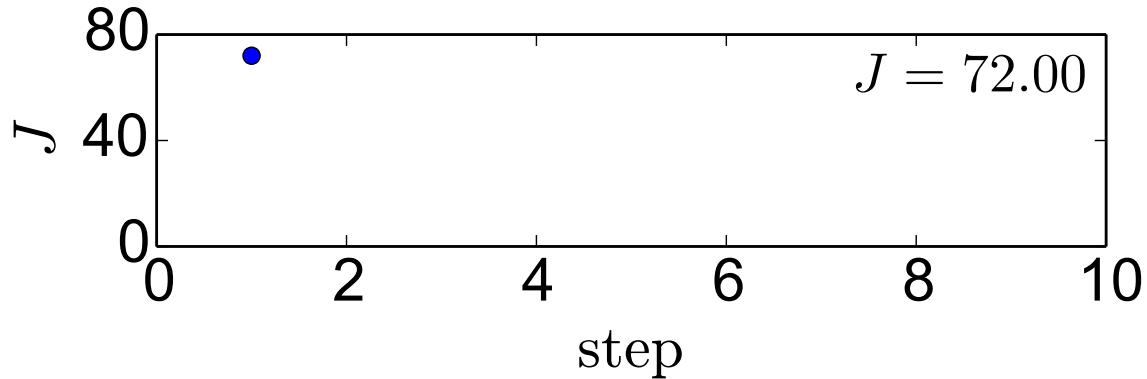


Example of Local Minima



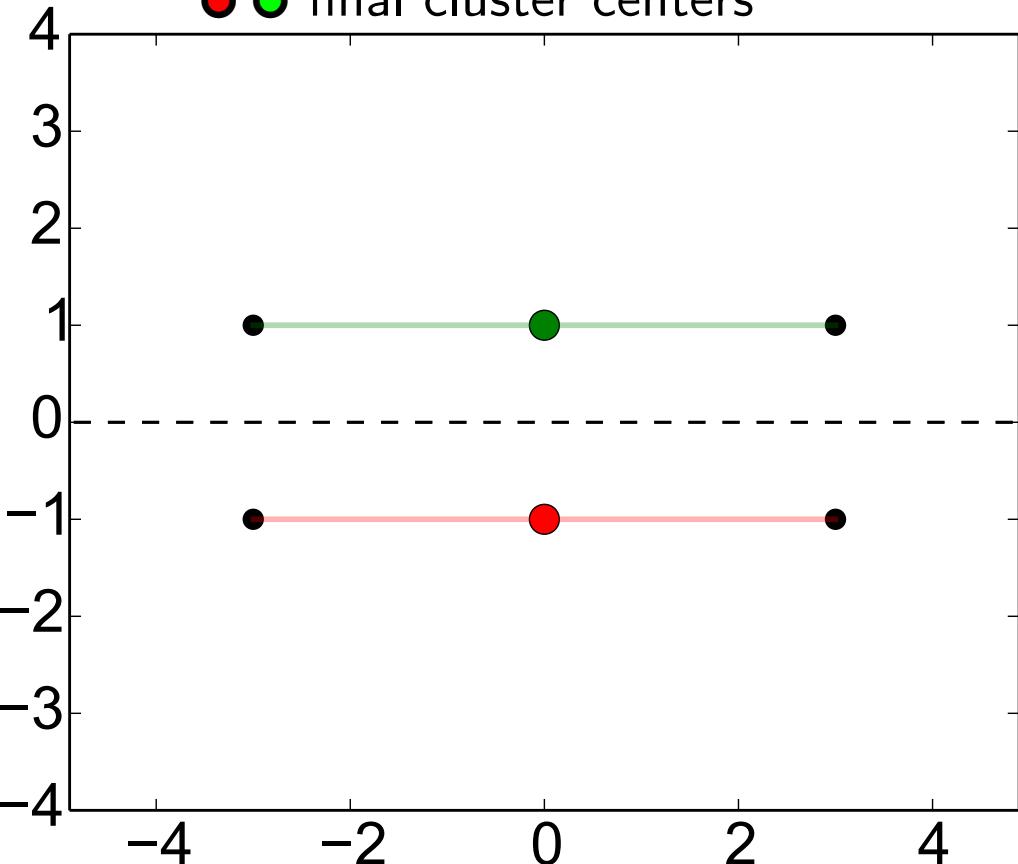
Cluster the data points to $K = 2$ clusters

Initial cluster centers (selected randomly from data points)



Local Minimum 1, $J = 36$

(-- Voronoi boundary)
● ● final cluster centers



Cluster the data points to $K = 2$ clusters

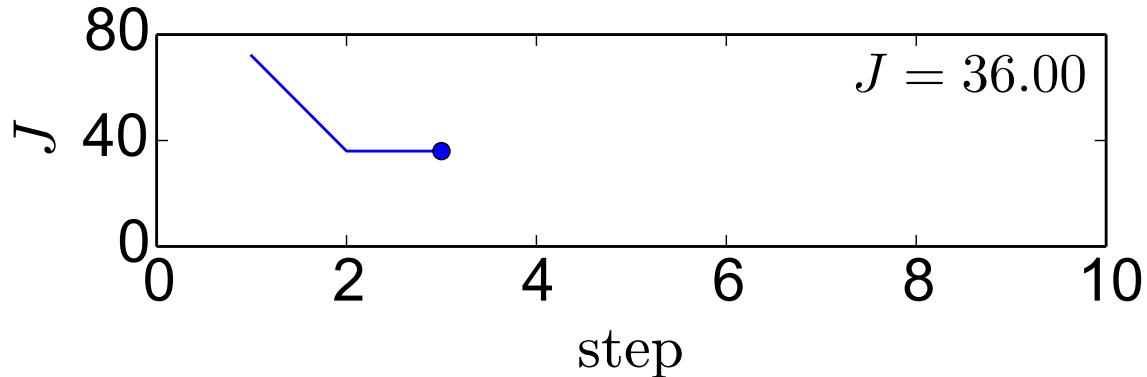
Initial cluster centers (selected randomly from data points)

step 1, compute assignments

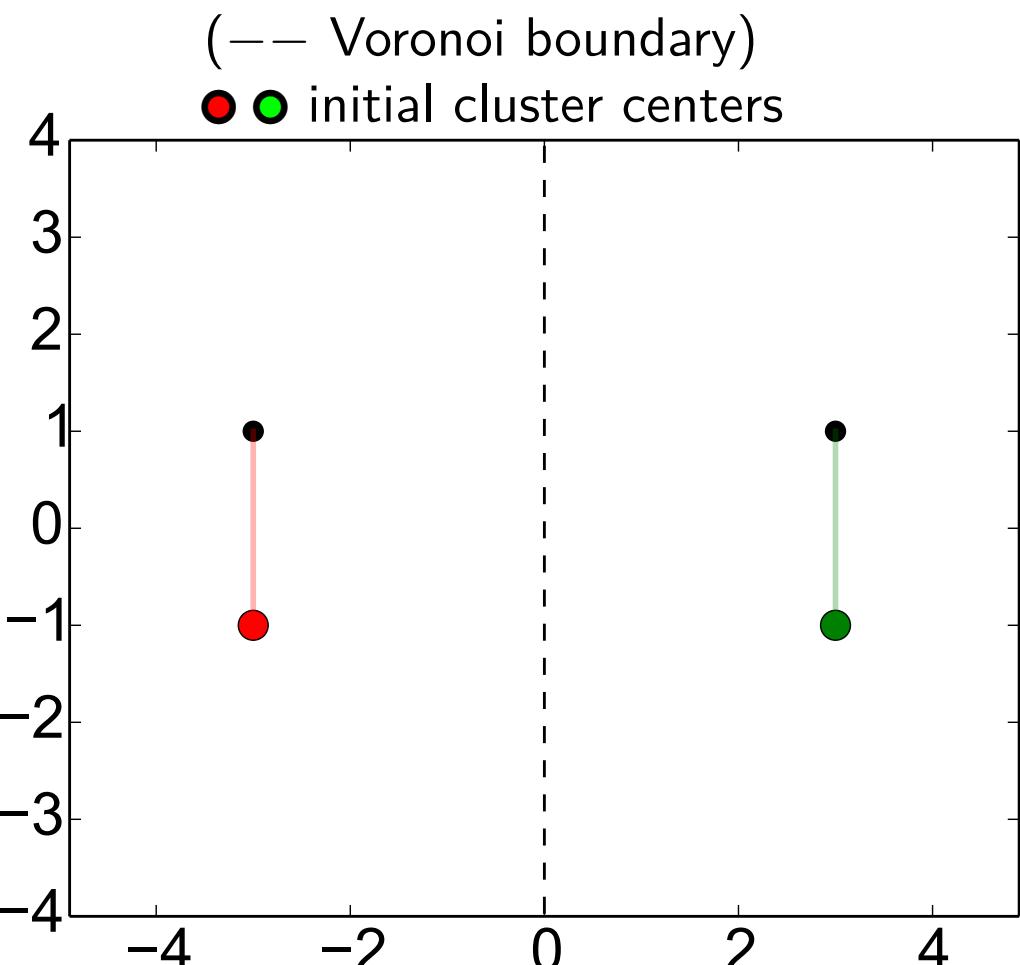
step 2, recompute centers

step 3, recompute assignments

Done.

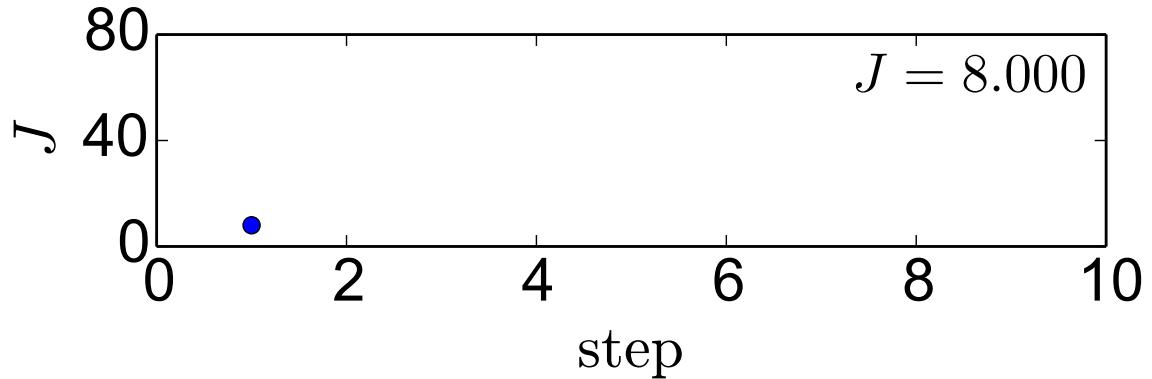


Different Initialization

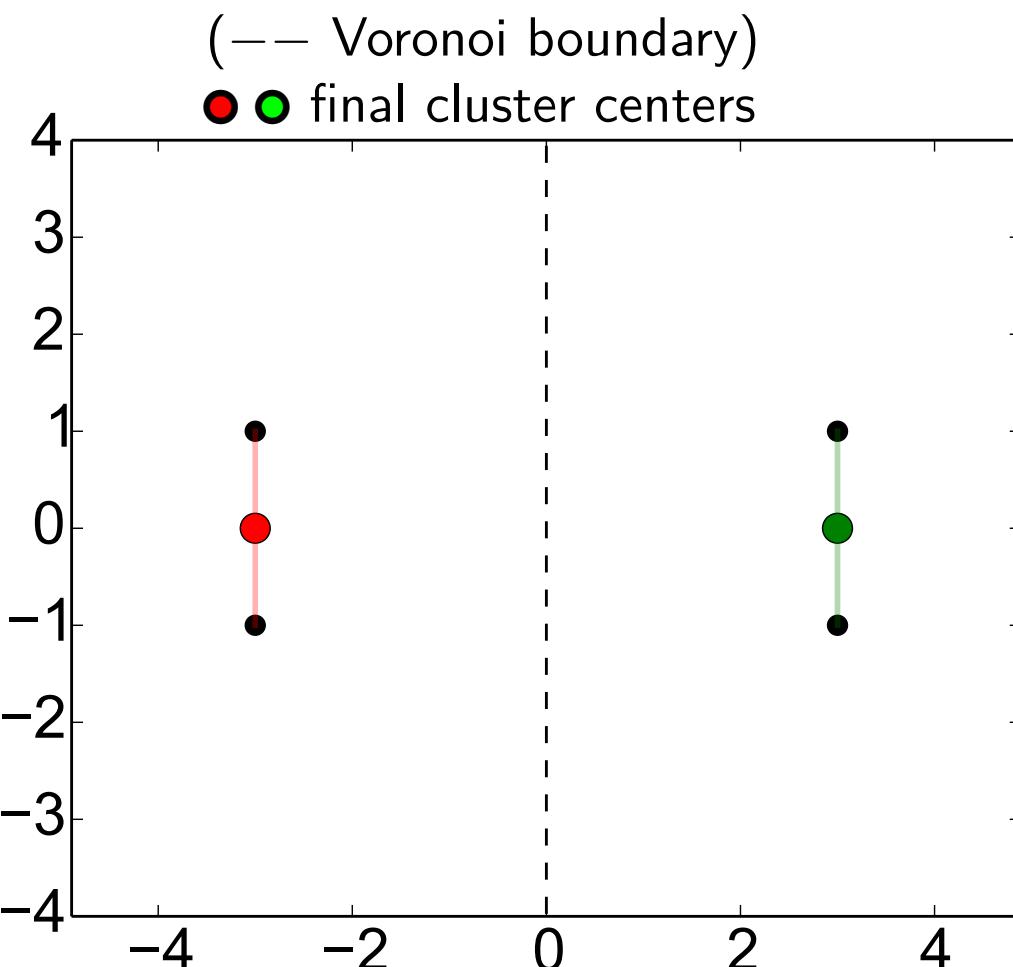


Cluster the data points to $K = 2$ clusters

Initial cluster centers (selected randomly from data points)



Local Minimum 2, $J = 4$



Cluster the data points to $K = 2$ clusters

Initial cluster centers (selected randomly from data points)

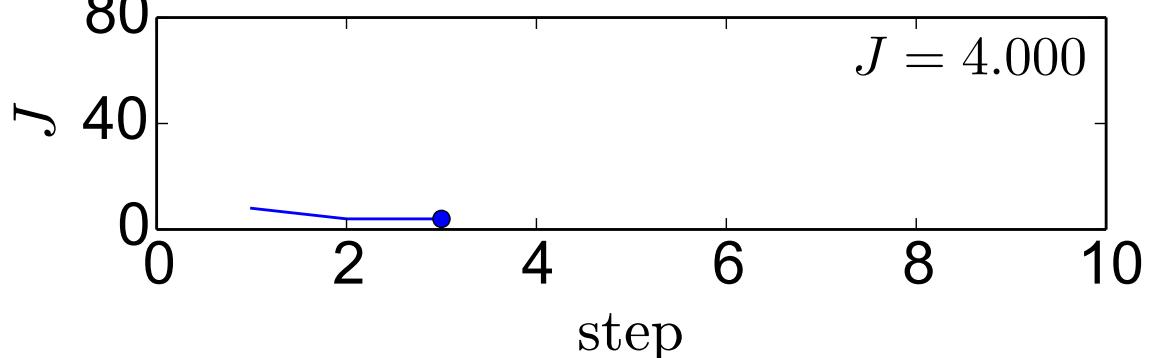
step 1, compute assignments

step 2, recompute centers

step 3, recompute assignments

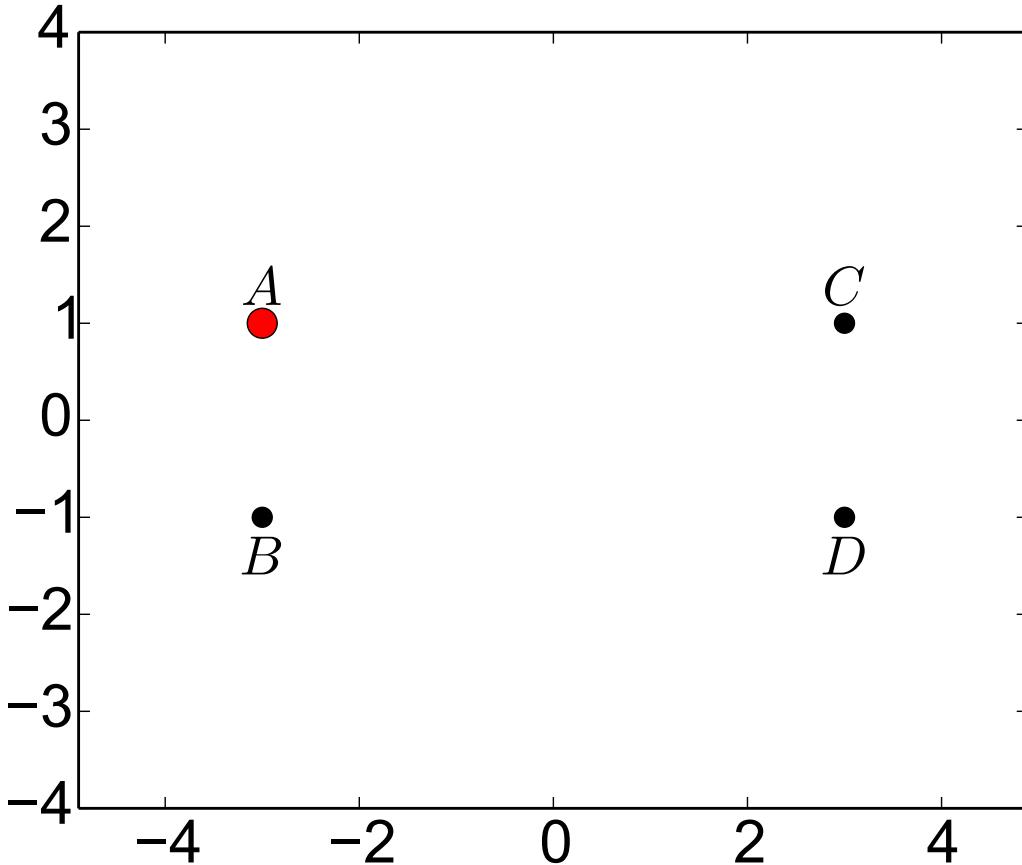
Done.

This minimum is the global one, reaching the optimum $J_{\text{opt}} = 4$.



There is no other local minimum besides these two.

Which Local Minimum Will Be Reached?



This depends on initialization. Let us assume that A has been randomly selected from data as the first cluster centre (●).

If B is selected as the second cluster centre, the output will be Minimum 1 ($J = 36$).

If C or D is selected as the second cluster centre, the output will be Minimum 2 ($J = J_{\text{opt}} = 4$).

All of the points B , C , D have equal chance of being randomly selected as the second cluster centre. Thus, the algorithm outcome can be summarized as follows:

K-means output	value of J	odds
Minimum 1	36	1/3
Minimum 2	4	2/3

K-Means++

K-Means++ is the K-means with clever **initialization** of cluster centers. The motivation is to make initializations which make K-means more likely to end up in better local minima (minima with lower J).

K-Means++ uses the following randomized sampling strategy for constructing the initial cluster centers set \mathcal{C} :

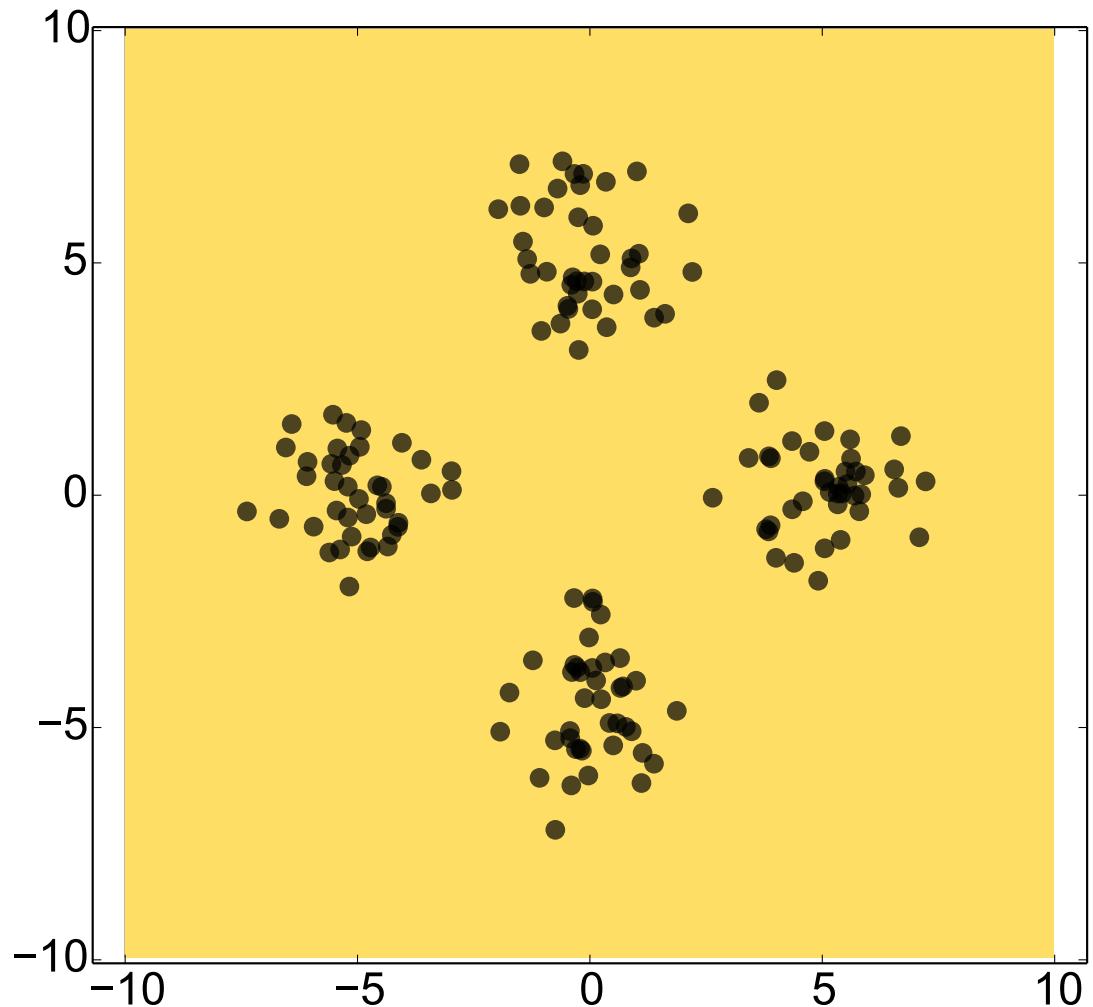
1. Choose the first cluster centre \mathbf{c}_1 uniformly at random from \mathcal{T} . Set $\mathcal{C} = \{\mathbf{c}_1\}$.
2. For each data point \mathbf{x}_l , compute the distance d_l to its nearest cluster in \mathcal{C} :

$$d_l = \min_{\mathbf{c} \in \mathcal{C}} \|\mathbf{x}_l - \mathbf{c}\| \quad (\forall l = 1, 2, \dots, L) \quad (8)$$

3. Select a point \mathbf{x}_l from \mathcal{T} with probability proportional to d_l^2 . This involves constructing a distribution $p(l)$ from d_l as $p(l) = \frac{d_l^2}{\sum_{l=1}^L d_l^2}$ and sampling from it to get the index l .
4. $\mathcal{C} \leftarrow \mathcal{C} \cup \mathbf{x}_l$.
5. Stop if $|\mathcal{C}| = K$, otherwise goto 2.

After this initialization, standard K-means algorithm is employed.

K-means++, Example



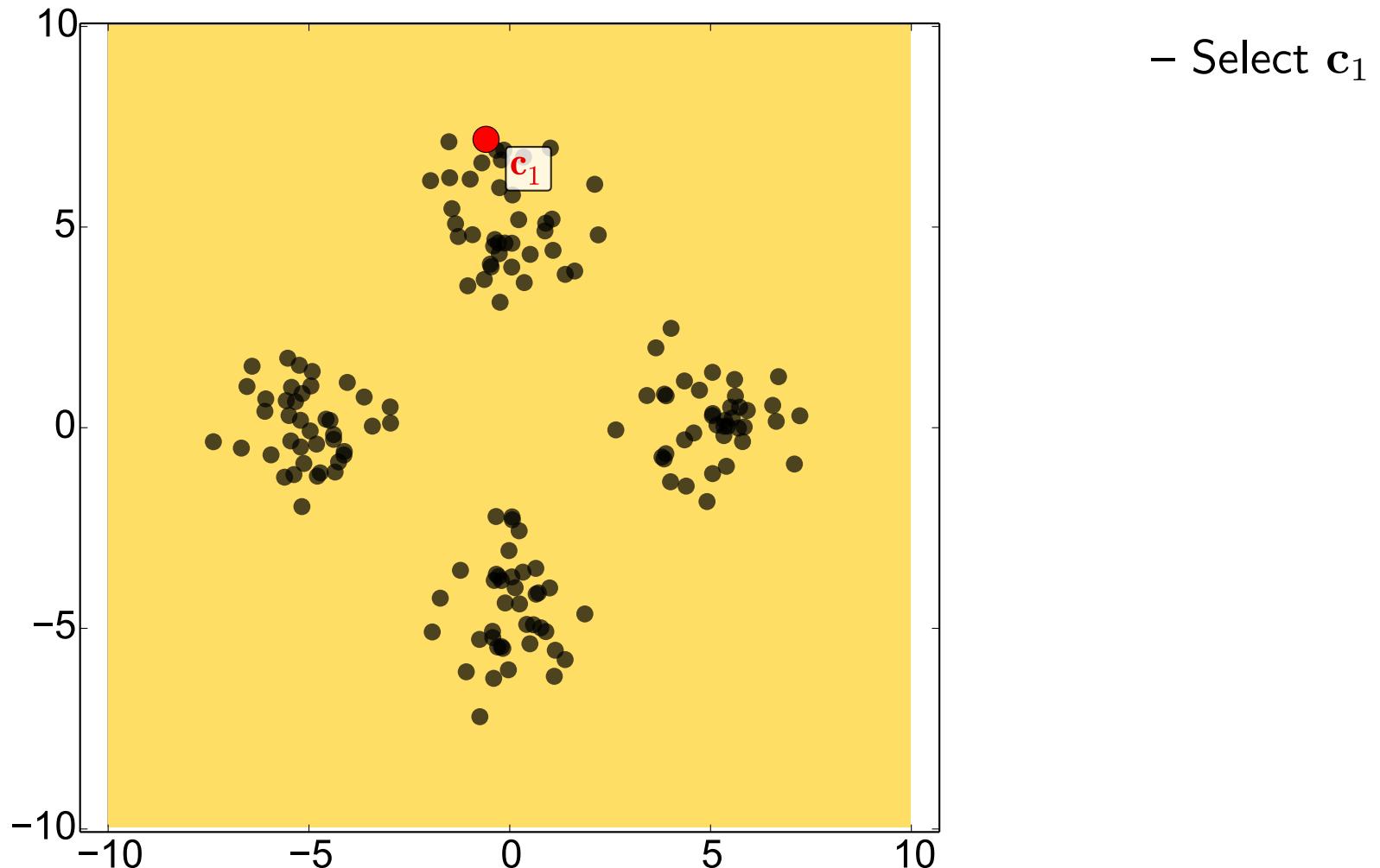
Sampling distribution $p(l)$ is shown as the scatter plot. Area of circle shown at \mathbf{x}_l is proportional to $p(l)$.

Data: Points sampled from normal distribution with unit variance, at each of the following four positions (40 samples each):

$$[-5, 0], [5, 0], [0, -5], [0, 5].$$

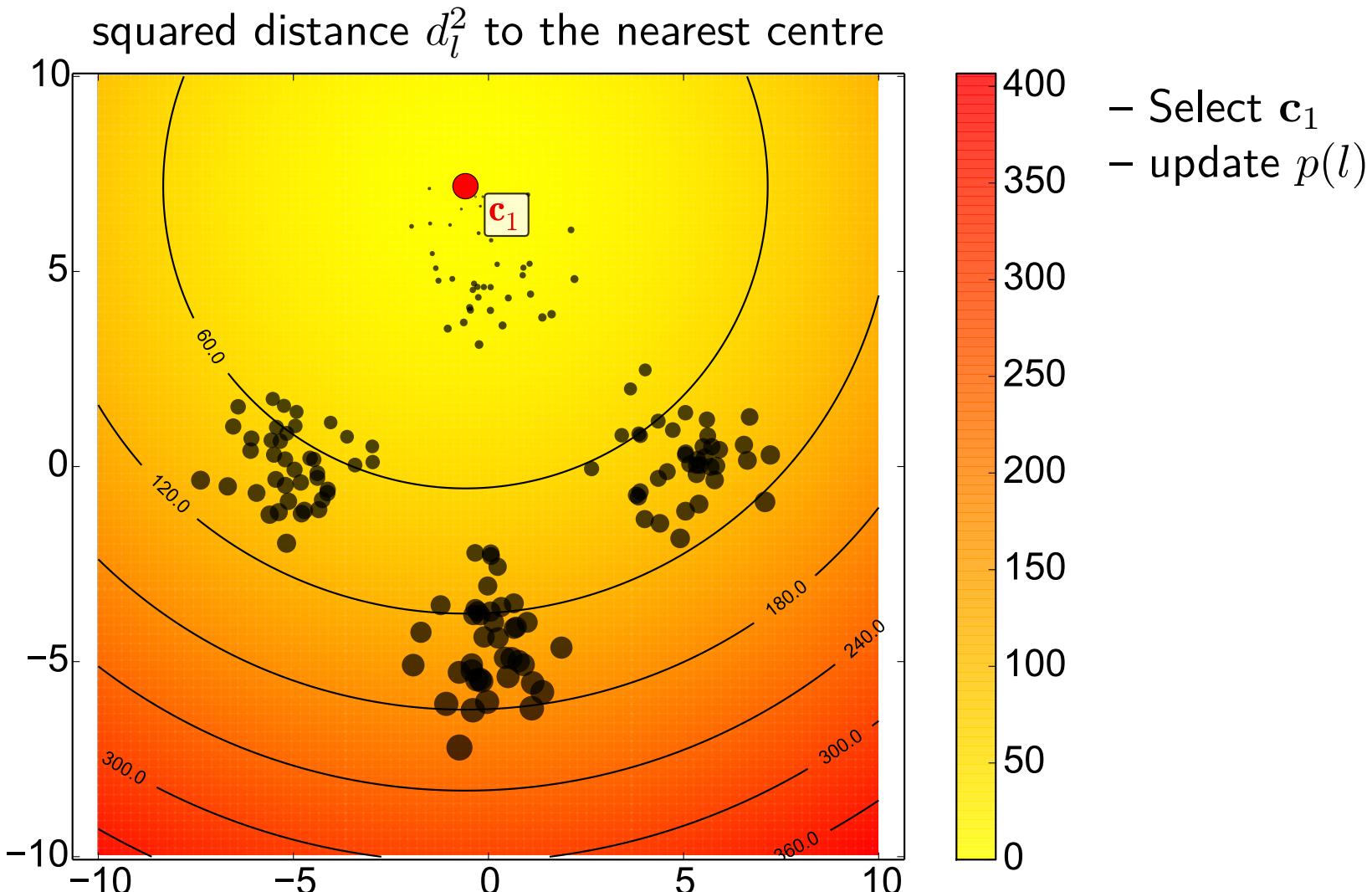
Problem: Initialize $K = 4$ cluster centres using K-means++.

K-means++, Example



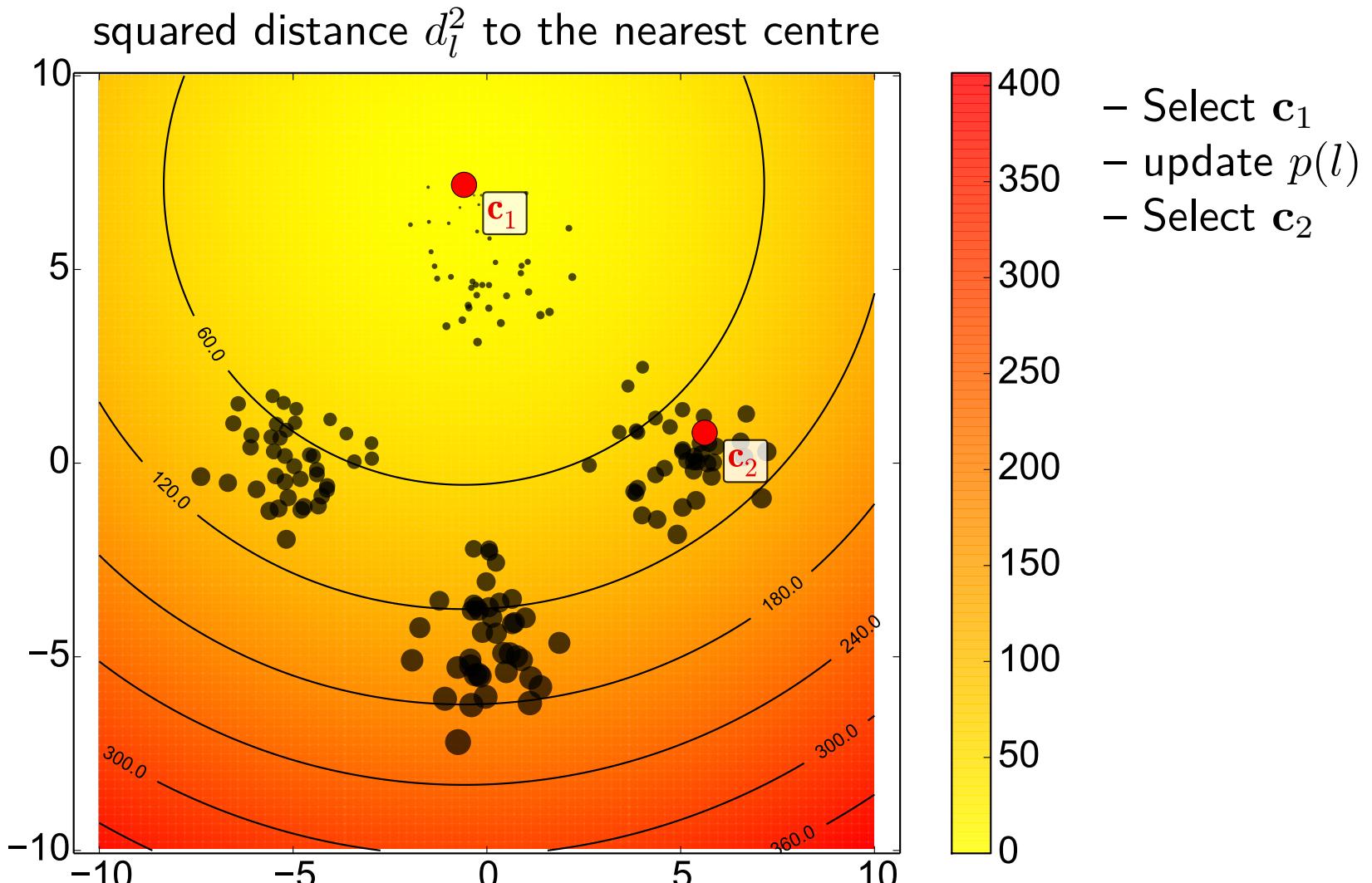
Sampling distribution $p(l)$ is shown as the scatter plot. Area of circle shown at x_l is proportional to $p(l)$.

K-means++, Example



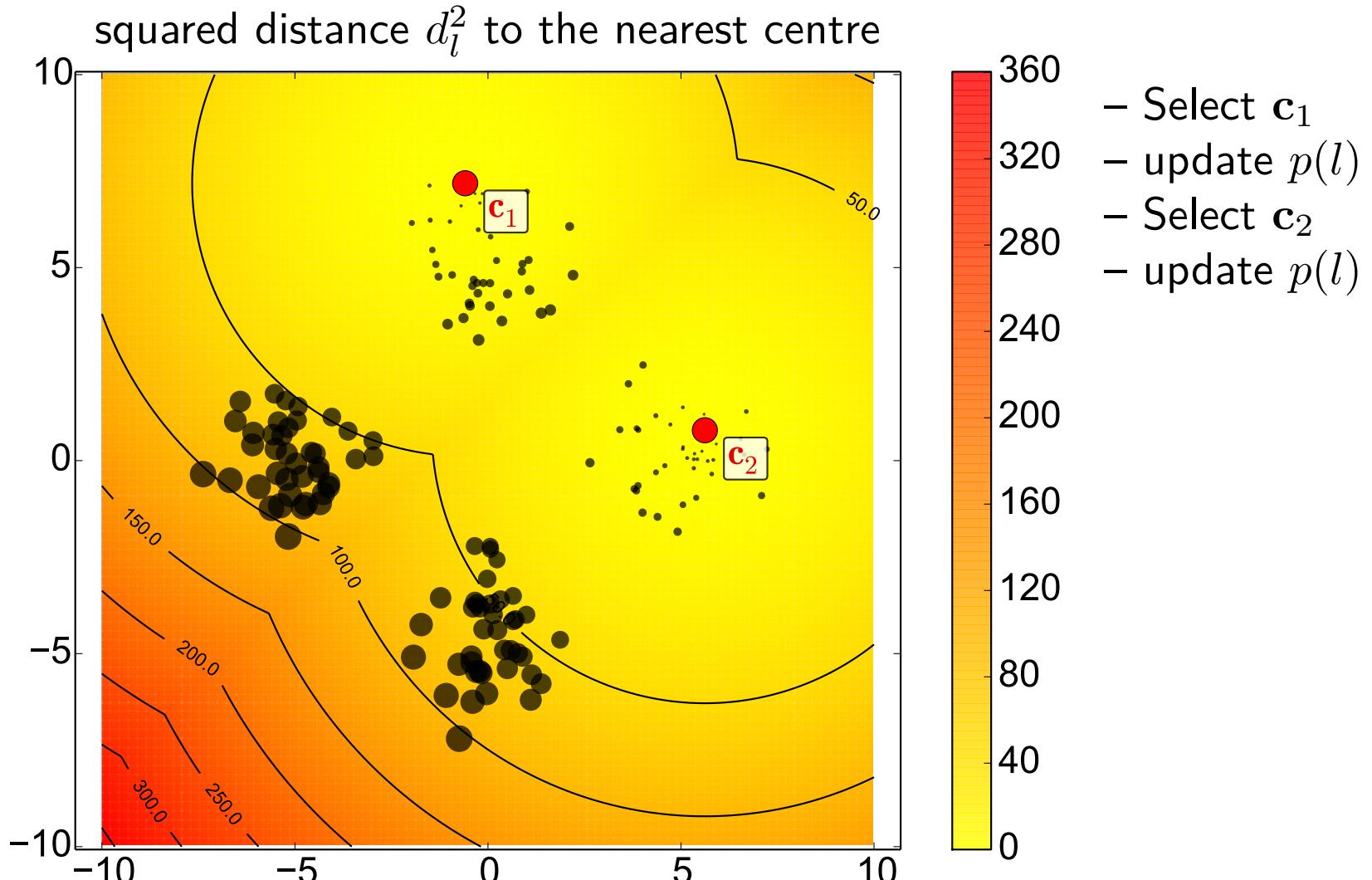
Sampling distribution $p(l)$ is shown as the scatter plot. Area of circle shown at x_l is proportional to $p(l)$.

K-means++, Example



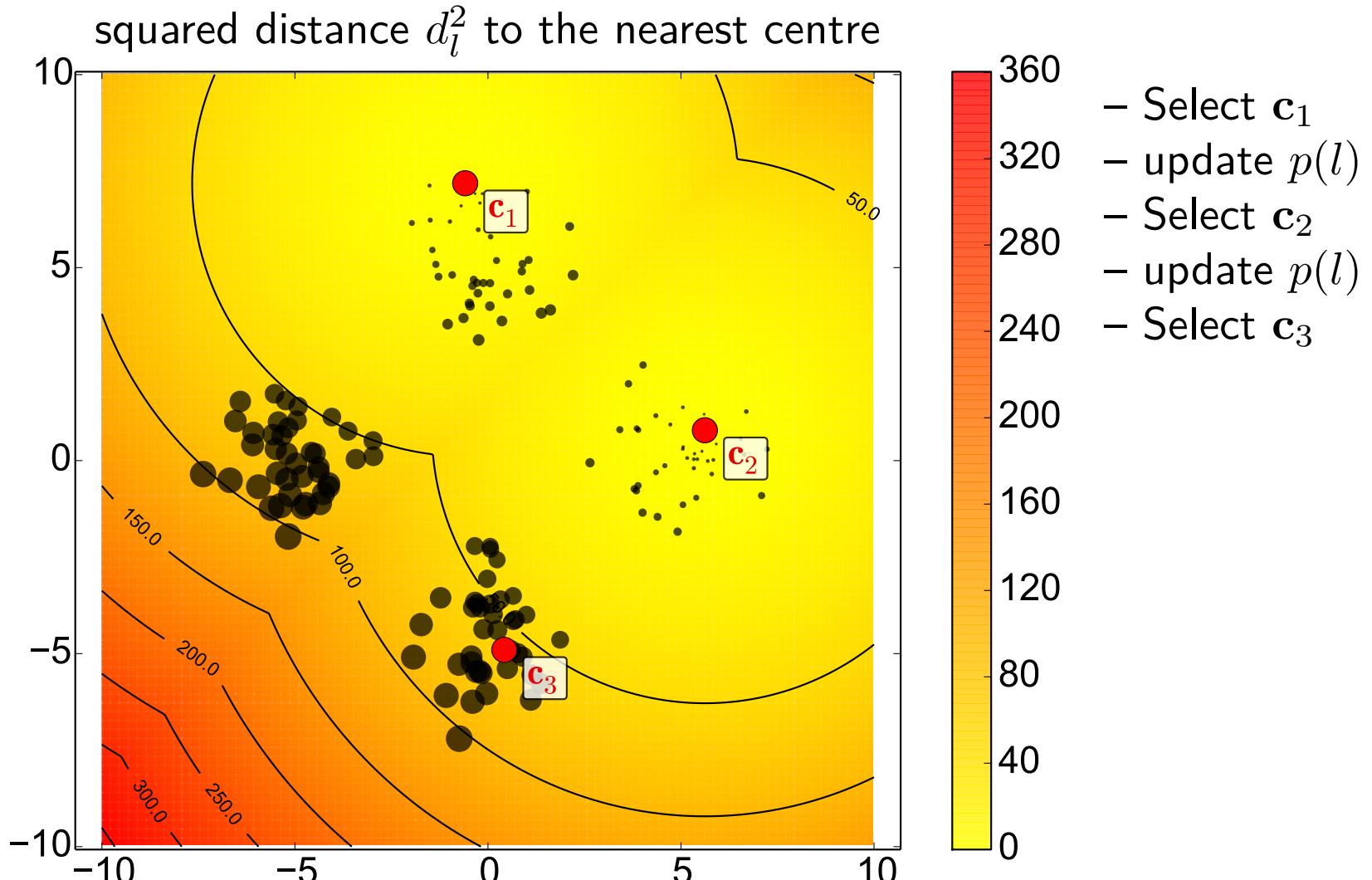
Sampling distribution $p(l)$ is shown as the scatter plot. Area of circle shown at x_l is proportional to $p(l)$.

K-means++, Example



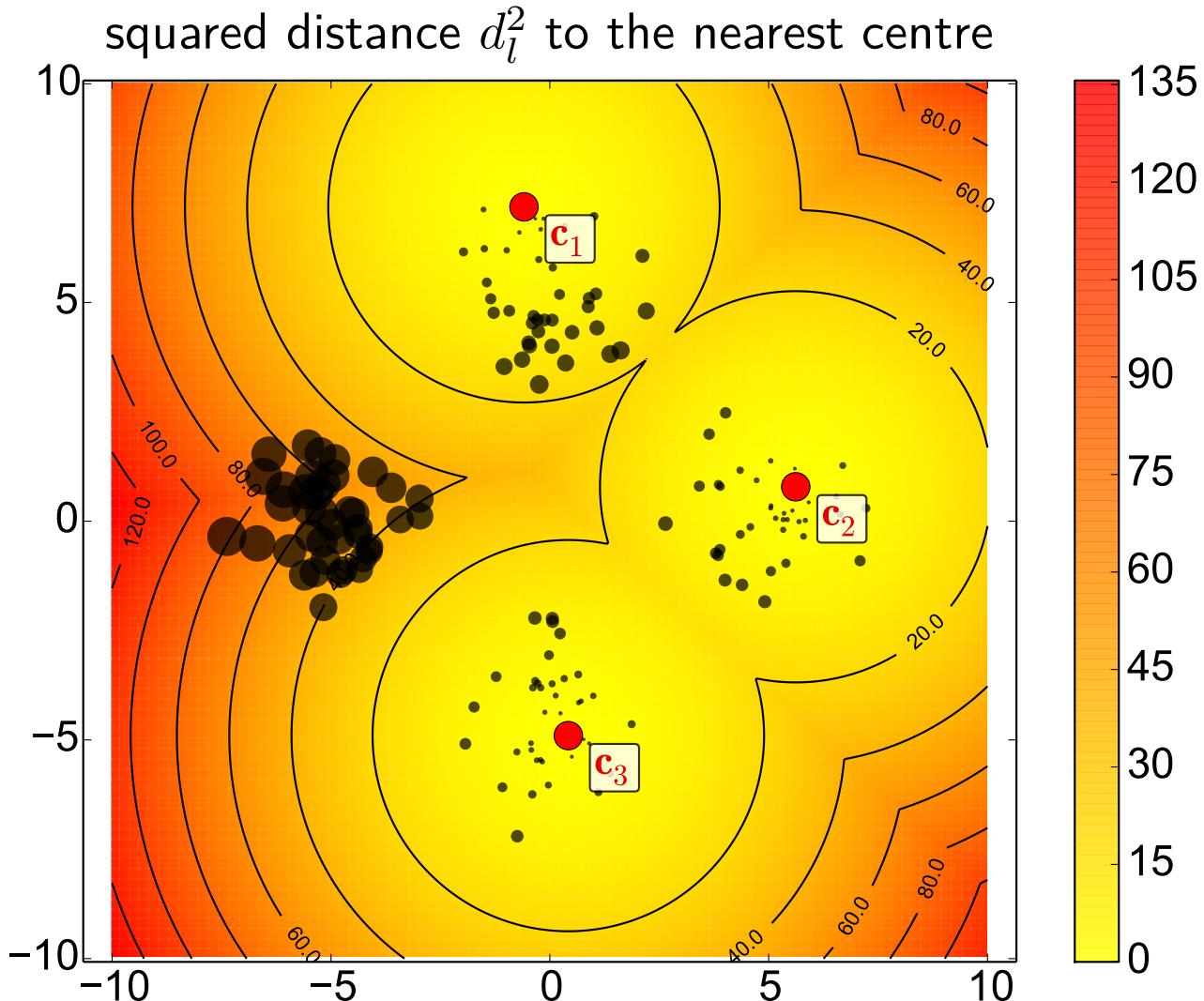
Sampling distribution $p(l)$ is shown as the scatter plot. Area of circle shown at x_l is proportional to $p(l)$.

K-means++, Example



Sampling distribution $p(l)$ is shown as the scatter plot. Area of circle shown at x_l is proportional to $p(l)$.

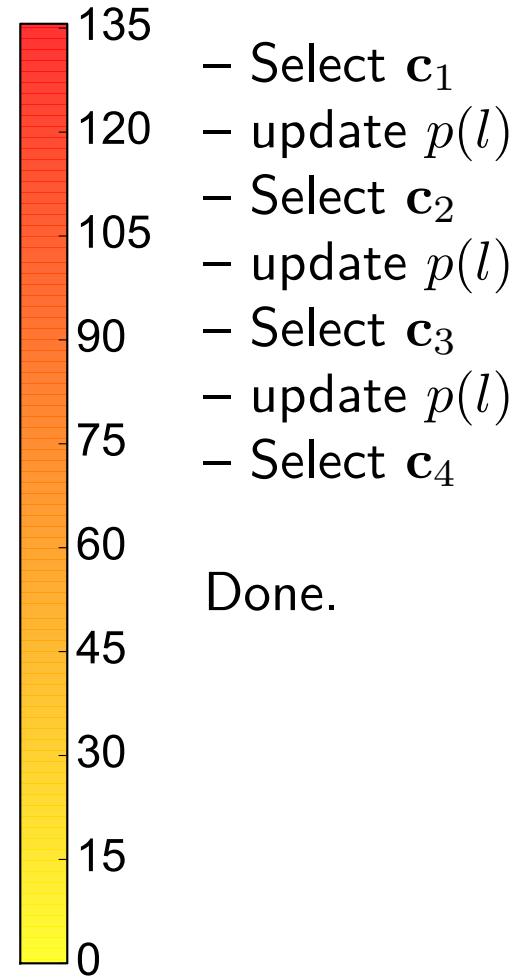
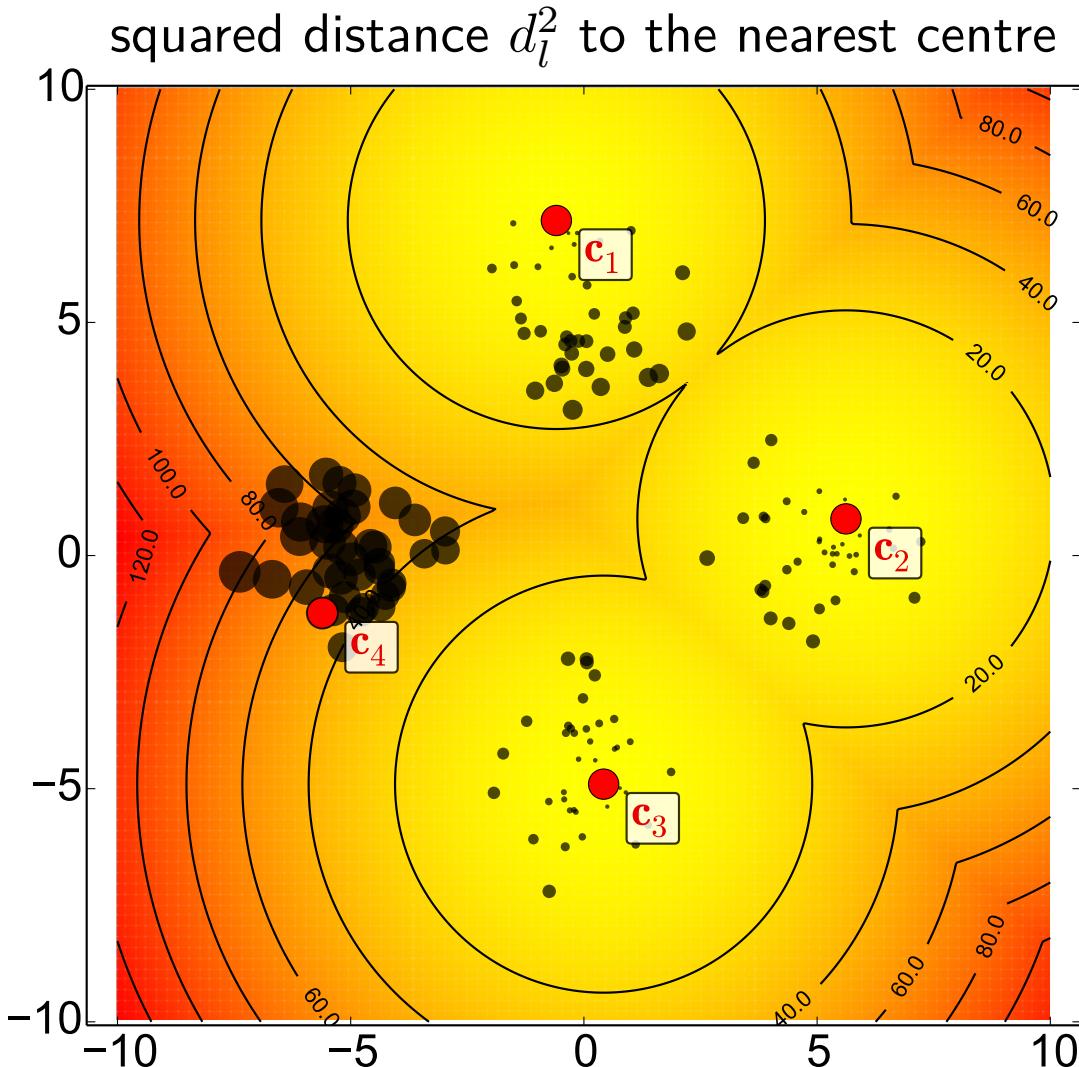
K-means++, Example



- Select c_1
- update $p(l)$
- Select c_2
- update $p(l)$
- Select c_3
- update $p(l)$

Sampling distribution $p(l)$ is shown as the scatter plot. Area of circle shown at \mathbf{x}_l is proportional to $p(l)$.

K-means++, Example



Sampling distribution $p(l)$ is shown as the scatter plot. Area of circle shown at x_l is proportional to $p(l)$.

K-means++, Bound on $E(J)$

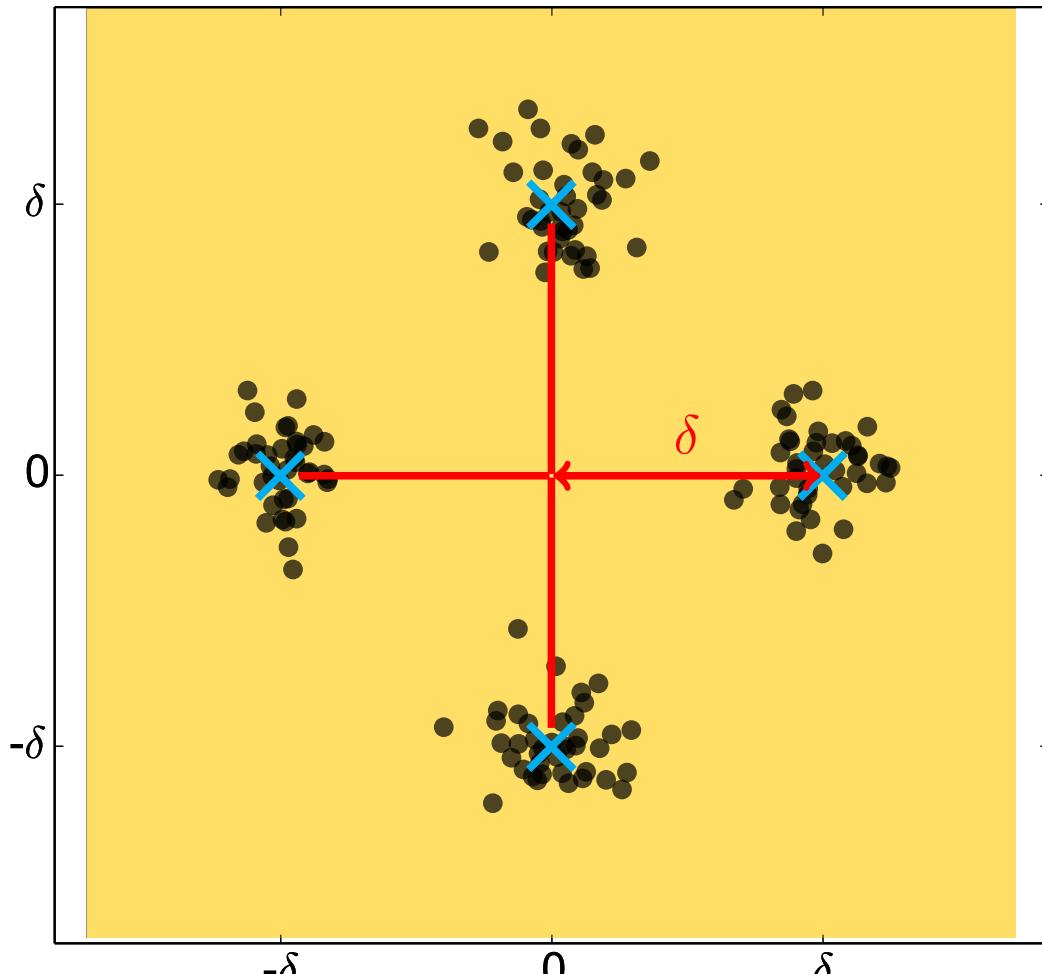
- ◆ The following bound on expectation $E(J)$ of the criterion value J exists when K-means++ is used for initialization:

$$E(J) \leq 8(\ln K + 2)J_{\text{opt}} \quad (9)$$

In the classical initialization (selecting all centres from data uniformly at random), no such bound exists.

- ◆ Arthur, D. and Vassilvitskii, S. (2007). "*k-means++: the advantages of careful seeding*". Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics Philadelphia, PA, USA. pp. 1027–1035.

K-means++, Effect on K-means outcome, Example 1



(shown for $\delta = 7$)

$$J_{\text{opt}} = 289.7$$

Data: points sampled from normal distribution with unit variance, at four different positions (40 samples each):

$$\begin{aligned}\mu_1 &= [-\delta, 0] \\ \mu_2 &= [\delta, 0] \\ \mu_3 &= [0, \delta] \\ \mu_4 &= [0, -\delta]\end{aligned}$$

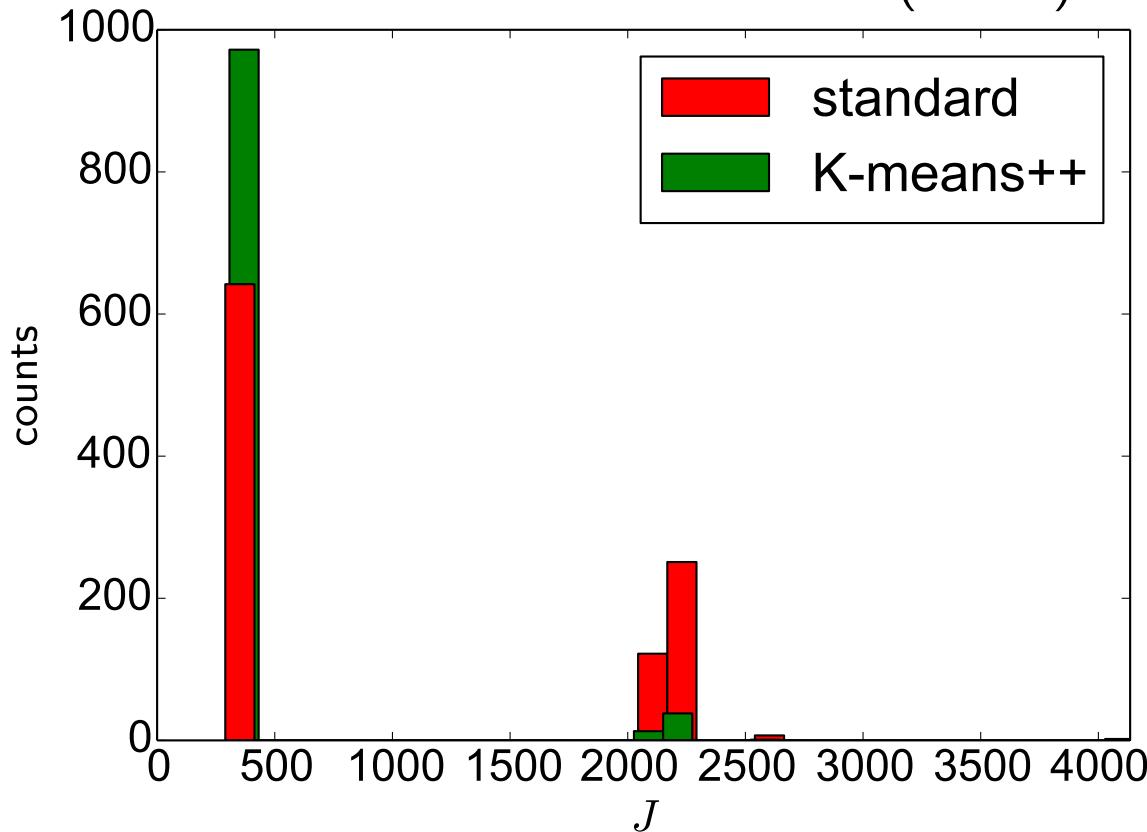
Experiment: Cluster the data repeatedly to $K = 4$ clusters, using (i) standard and (ii) K-means++ initializations. Store the values of J obtained in individual runs of K-means. Compare distributions of J for the two initializations.

K-means++, Effect on K-means outcome, Example 1

Results (for $\delta = 7$):

	J_{mean}	J_{min}	J_{max}
standard init.	1002	289.7	4135
K-means++	386.5	289.7	2637

histogram of values of J obtained across 1024 runs of K-means ($\delta = 7$)

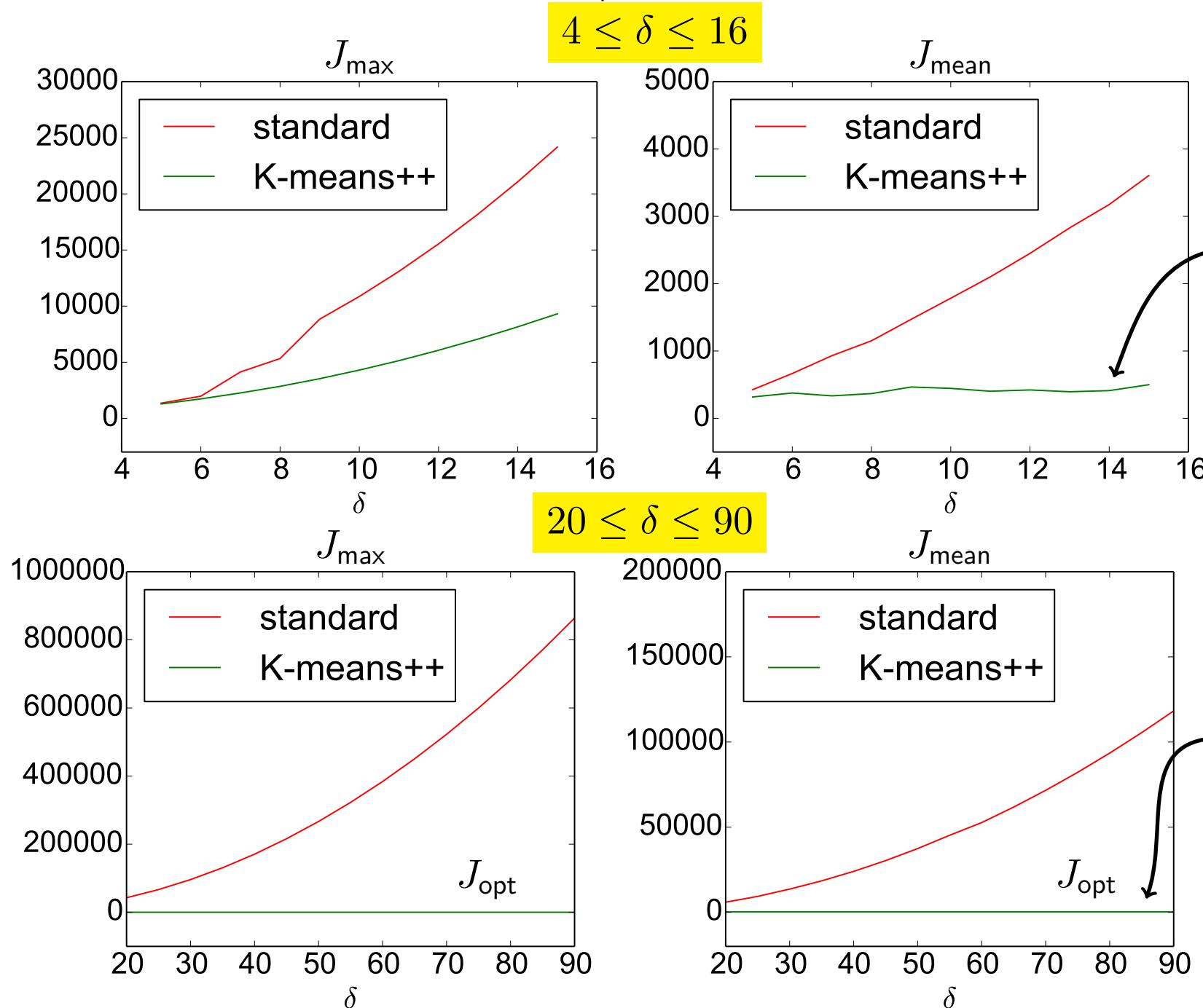


Things to note:

- ◆ both initialization methods found the optimal clustering and reach $J_{\text{opt}} = 289.7$
- ◆ K-means++ achieved better clustering on average (lower J_{mean})
- ◆ K-means++ also achieved better worst case (lower J_{max})

K-means++, Effect on K-means outcome, Example 1

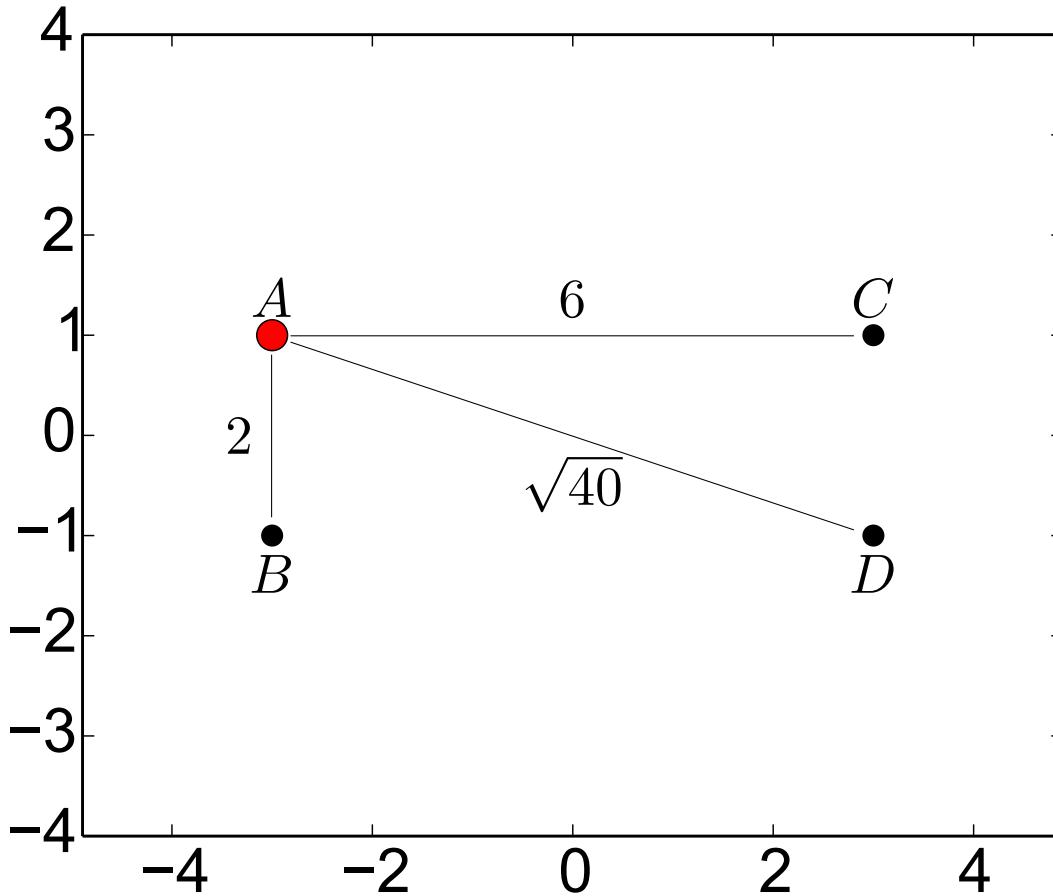
Dependence on δ . Results obtained by running K-means 128× for each δ (Note: $J_{\min} = J_{\text{opt}}$ for all δ 's and both methods.)



Note J_{mean} stays low for K-means++

Probability of generating initialization resulting in non-optimal outcome is so low that **no** non-optimal outcome is encountered across the 128 runs.

K-means++, Effect on K-means outcome, Example 2



Suppose A has been selected as the first cluster centre (●).

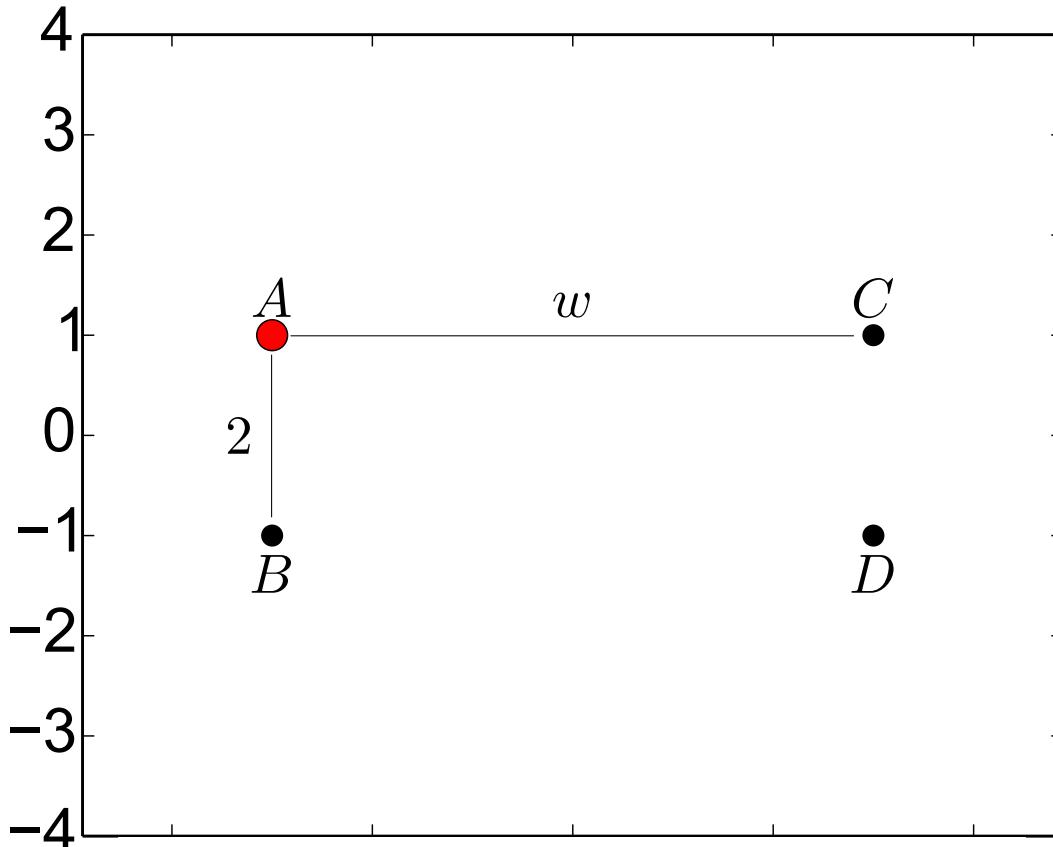
The points B , C , D will be selected to be the second cluster centre with odds $B : C : D = 4 : 36 : 40$. Hence the probabilities of being selected are:

$$\begin{aligned} p(B) &= 1/20, \\ p(C) &= 9/20, \\ p(D) &= 1/2. \end{aligned}$$

The algorithm outcome can be summarized as follows:

K-means output	value of J	odds (K-means++)	odds (standard init.)
Minimum 1	36	1/20	1/3
Minimum 2	4	19/20	2/3

K-means++, Effect on K-means outcome, Example 2



Suppose we let the points C and D go further away from A and B , with $|AC| = |BD| = w$ ($w \geq 2$).

Using the same arguments as before,

$$\begin{aligned} p(B) &= 4/Z, \\ p(C) &= w^2/Z, \\ p(D) &= (w^2 + 4)/Z, \end{aligned}$$

(Z is the normalization constant), and we arrive at the result summarized in this table:

K-means output	value of J	odds (K-means++)	odds (standard init.)
Minimum 1	w^2	$\frac{2}{w^2 + 4}$	$1/3$
Minimum 2	4	$1 - \frac{2}{w^2 + 4}$	$2/3$

K-means++, Effect on K-means outcome, Example 2

The expectation $E(J)$ of J is ($w \geq 2$):

$$E(J) = w^2 \frac{2}{w^2 + 4} + 4 \left(1 - \frac{2}{w^2 + 4}\right) = 6 - \frac{16}{w^2 + 4} \quad (\text{K-means++}) \quad (10)$$

$$E(J) = \frac{w^2 + 8}{3} \quad (\text{standard initialization}) \quad (11)$$

There is a striking difference between the two as w increases:

	K-means++	standard init.
$E(J)$ for $w = 2$	4	4
$E(J)$ for $w = 6$	5.6	14.7
$E(J)$ for $w \rightarrow \infty$	6	∞

K-means output	value of J	odds (K-means++)	odds (standard init.)
Minimum 1	w^2	$\frac{2}{w^2 + 4}$	$1/3$
Minimum 2	4	$1 - \frac{2}{w^2 + 4}$	$2/3$

K-means Generalizations (K-medians, K-medoids, . . .)

K-means can be generalized for minimizing criterion other than squared Euclidean.

Given:

- $\mathcal{T} = \{\mathbf{x}_l\}_{l=1}^L$ the set of observations
- K the desired number of cluster prototypes
- $d(\cdot, \cdot)$ 'distance function' (not necessarily a metric)

Output:

- $\{\mathbf{c}_k\}_{k=1}^K$ the set of cluster prototypes (etalons)
- $\{\mathcal{T}_k\}_{k=1}^K$ the clustering (partitioning) of the data
- $\bigcup_{k=1}^K \mathcal{T}_k = \mathcal{T}, \mathcal{T}_i \cap \mathcal{T}_j = \emptyset \text{ for } i \neq j$

1. Initialize the cluster centres $\{\mathbf{c}_k\}_{k=1}^K$ (e.g. by random selection from the data points \mathcal{T} , without replacement)
2. Assignment optimization (assign to closest etalon):

$$\mathcal{T}_k = \{\mathbf{x} \in \mathcal{T} : \forall j, d(\mathbf{x}, \mathbf{c}_k) \leq d(\mathbf{x}, \mathbf{c}_j)\} \quad (\forall k = 1, 2, \dots, K) \quad (12)$$

3. Prototype optimization:

$$\mathbf{c}_k = \begin{cases} \underset{\mathbf{c}}{\operatorname{argmin}} \sum_{\mathbf{x} \in \mathcal{T}_k} d(\mathbf{x}, \mathbf{c}) & \text{if } |\mathcal{T}_k| > 0 \\ \text{re-initialize} & \text{if } \mathcal{T}_k = \emptyset \end{cases} \quad (\forall k = 1, 2, \dots, K) \quad (13)$$

4. Terminate if $\forall k : \mathcal{T}_k^{t+1} = \mathcal{T}_k^t$, otherwise goto 2

K-means Generalization: K-medians

Given:

$\mathcal{T} = \{\mathbf{x}_l\}_{l=1}^L$ the set of observations, $\mathbf{x} \in \mathbb{R}^D$

K the desired number of cluster prototypes

$d(\cdot, \cdot)$ $\|\mathbf{c} - \mathbf{x}\|_1 = \sum_{i=1}^D |c_i - x_i|$ (L_1 metric)

Output:

$\{\mathbf{c}_k\}_{k=1}^K$ the set of cluster prototypes (etalons)

$\{\mathcal{T}_k\}_{k=1}^K$ the clustering (partitioning) of the data

$\cup_{k=1}^K \mathcal{T}_k = \mathcal{T}, \mathcal{T}_i \cap \mathcal{T}_j = \emptyset$ for $i \neq j$

1. Initialize the cluster centres $\{\mathbf{c}_k\}_{k=1}^K$ (e.g. by random selection from the data points \mathcal{T} , without replacement)
2. Assignment optimization (assign to closest etalon):

$$\mathcal{T}_k = \{\mathbf{x} \in \mathcal{T} : \forall j, d(\mathbf{x}, \mathbf{c}_k) \leq d(\mathbf{x}, \mathbf{c}_j)\} \quad (\forall k = 1, 2, \dots, K) \quad (14)$$

3. Prototype optimization:

$$\mathbf{c}_k = \begin{cases} \text{median}\{\mathcal{T}_k\} & \\ \text{re-initialize} & \text{if } \mathcal{T}_k = \emptyset \end{cases} \quad (\forall k = 1, 2, \dots, K) \quad (15)$$

4. Terminate if $\forall k : \mathcal{T}_k^{t+1} = \mathcal{T}_k^t$, otherwise goto 2

K-means Generalization: Clustering Strings

Given:

$\mathcal{T} = \{\mathbf{x}_l\}_{l=1}^L$ observations are strings

K the desired number of cluster prototypes

$d(\mathbf{s}_1, \mathbf{s}_2)$ Levenshtein distance, number of edit operations to transform \mathbf{s}_1 to \mathbf{s}_2

Output:

$\{\mathbf{c}_k\}_{k=1}^K$ the set of cluster prototypes (etalons)

$\{\mathcal{T}_k\}_{k=1}^K$ the clustering (partitioning) of the data

$$\bigcup_{k=1}^K \mathcal{T}_k = \mathcal{T}, \mathcal{T}_i \cap \mathcal{T}_j = \emptyset \text{ for } i \neq j$$

1. Initialize the cluster centres $\{\mathbf{c}_k\}_{k=1}^K$ (e.g. by random selection from the data points \mathcal{T} , without replacement)
2. Assignment optimization (assign to closest etalon):

$$\mathcal{T}_k = \{\mathbf{x} \in \mathcal{T} : \forall j, d(\mathbf{x}, \mathbf{c}_k) \leq d(\mathbf{x}, \mathbf{c}_j)\} \quad (\forall k = 1, 2, \dots, K) \quad (16)$$

3. Prototype optimization:

$$\mathbf{c}_k = \begin{cases} \underset{\mathbf{c}}{\operatorname{argmin}} \sum_{\mathbf{x} \in \mathcal{T}_k} d(\mathbf{x}, \mathbf{c}) & \text{if } |\mathcal{T}_k| > 0 \\ \text{re-initialize} & \text{if } \mathcal{T}_k = \emptyset \end{cases} \quad (\forall k = 1, 2, \dots, K) \quad (17)$$

4. Terminate if $\forall k : \mathcal{T}_k^{t+1} = \mathcal{T}_k^t$, otherwise goto 2

K-means Generalization: Clustering Strings, Notes

- ◆ the calculation of $d(\cdot, \cdot)$ may be non trivial
- ◆ it may be hard to minimize $\sum_{x \in \mathcal{T}_k} d(x, c)$ over the space of all strings. The minimization may be restricted to $c \in \mathcal{T}$.
- ◆ is the algorithm guaranteed to terminate if step 2 (step 3) is only improving J , not finding the minimum (given \mathcal{T}_k or c_k), respectively?

K-means Generalization: Euclidean Clustering

Given:

$\mathcal{T} = \{\mathbf{x}_l\}_{l=1}^L$ the set of observations, $\mathbf{x} \in \mathbb{R}^D$

K the desired number of cluster prototypes

$d(\cdot, \cdot)$ $\|\mathbf{c} - \mathbf{x}\|$ (L_2 metric)

Output:

$\{\mathbf{c}_k\}_{k=1}^K$ the set of cluster prototypes (etalons)

$\{\mathcal{T}_k\}_{k=1}^K$ the clustering (partitioning) of the data

$\cup_{k=1}^K \mathcal{T}_k = \mathcal{T}, \mathcal{T}_i \cap \mathcal{T}_j = \emptyset$ for $i \neq j$

1. Initialize the cluster centres $\{\mathbf{c}_k\}_{k=1}^K$ (e.g. by random selection from the data points \mathcal{T} , without replacement)
2. Assignment optimization (assign to closest etalon):

$$\mathcal{T}_k = \{\mathbf{x} \in \mathcal{T} : \forall j, d(\mathbf{x}, \mathbf{c}_k) \leq d(\mathbf{x}, \mathbf{c}_j)\} \quad (\forall k = 1, 2, \dots, K) \quad (18)$$

3. Prototype optimization: no closed-form solution for *geometric median*. Use e.g. iterative Weiszfeld's algorithm.

$$\mathbf{c}_k = \begin{cases} \underset{\mathbf{c}}{\operatorname{argmin}} \sum_{\mathbf{x} \in \mathcal{T}_k} \|\mathbf{x} - \mathbf{c}\| & \text{if } |\mathcal{T}_k| > 0 \\ \text{re-initialize} & \text{if } \mathcal{T}_k = \emptyset \end{cases} \quad (\forall k = 1, 2, \dots, K) \quad (19)$$

4. Terminate if $\forall k : \mathcal{T}_k^{t+1} = \mathcal{T}_k^t$, otherwise goto 2

Weiszfeld Algorithm for Computing Geometric Median

- ◆ Uses iteratively re-weighted least squares
- ◆ Given $\mathbf{x}_i \in \mathbb{R}^D$ ($i = 1, 2, \dots, I$), the geometric median $\mathbf{m} \in \mathbb{R}^D$:

$$\mathbf{m} = \operatorname{argmin}_{\mathbf{m}'} \sum_{i=1}^I \|\mathbf{x}_i - \mathbf{m}'\|. \quad (20)$$

- ◆ Algorithm:
 1. $t = 0$. Initialize $\mathbf{m}^{(0)}$ (e. g. take the mean of \mathbf{x}_i 's)
 2. Compute weights w_i :

$$w_i = \frac{1}{\|\mathbf{x}_i - \mathbf{m}^{(t)}\|} \quad (i = 1, 2, \dots, I) \quad (21)$$

- 3. Obtain new estimate for \mathbf{m} as a weighted average of \mathbf{x}_i 's:

$$\mathbf{m}^{(t+1)} = \frac{\sum_{i=1}^I w_i \mathbf{x}_i}{\sum_{i=1}^I w_i} \quad (22)$$

- 4. Finish if the termination condition is met. Otherwise, $t \leftarrow t + 1$ and goto 2.

Expectation Maximization (EM) Algorithm

lecturer: J. Matas, matas@cmp.felk.cvut.cz

authors: O. Drbohlav, J. Matas

Czech Technical University, Faculty of Electrical Engineering
Department of Cybernetics, Center for Machine Perception
121 35 Praha 2, Karlovo nám. 13, Czech Republic

<http://cmp.felk.cvut.cz>

4/Dec/2020

LECTURE PLAN

- ◆ Motivation: Observations with missing values
- ◆ Sketch of the algorithm, relation to K-means
- ◆ EM algorithm derivation and properties

EM Algorithm

- ◆ Used to find maximum likelihood parameters of a statistical model when the equations cannot be directly solved.
- ◆ Two typical cases of use:
 - **Missing data:** Some observations are incomplete. E.g. features are vectors in 5-dimensional space $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5) \in \mathbb{R}^D$ but observations have a component missing, e.g.: $(2, 5, \bullet, 1, 2)$ or $(\bullet, \bullet, 1, 4, 2)$, where ' \bullet ' are the unobserved components.
 - **Latent variables:** Observations are complete but the model can be formulated and solved more simply if further variables are introduced to it. A typical example are *mixture models* where for each observed point it is advantageous to introduce a random variable which specifies which component of the mixture generated that point.

EM for Maximum Likelihood Estimation, Example (1)

Consider multivariate normal distribution in 2D. For simplicity, let us consider the isotropic case for which the covariance matrix Σ is diagonal and parametrized by a single parameter σ^2 , $\Sigma = \text{diag}(\sigma^2, \sigma^2)$. The normal distribution $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \sigma^2)$ for this case is then

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \sigma^2) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{\|\mathbf{x}-\boldsymbol{\mu}\|^2}{\sigma^2}}, \quad (1)$$

where $\mathbf{x} \in \mathbb{R}^2$ is the random variable and $\boldsymbol{\mu} \in \mathbb{R}^2$ is the mean.

Having the data $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, the MLE for the parameters $\boldsymbol{\mu}$ and σ^2 are computed as:

$$\hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \quad (2)$$

$$\hat{\sigma}^2 = \frac{1}{2N} \sum_{i=1}^N \|\mathbf{x}_i - \hat{\boldsymbol{\mu}}\|^2 \quad (3)$$

($2N$ in the denominator of Eq. (3) is not a mistake. It follows from the parametrization of Σ and the dimensionality of the considered space.)

EM for Maximum Likelihood Estimation, Example (2)

Now consider the case that the data are the result of random sampling from a mixture of two such distributions (denoted A and B):

$$p(\mathbf{x}|\pi_A, \pi_B, \boldsymbol{\mu}_A, \boldsymbol{\mu}_B, \sigma_A^2, \sigma_B^2) = \pi_A \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_A, \sigma_A^2) + \pi_B \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_B, \sigma_B^2), \quad (4)$$

where π_A and π_B imply the frequency with which a sample is realized from the respective distribution ($\pi_A + \pi_B = 1$) and other parameters have obvious meaning.

Analytical derivation of MLE in this case will involve logarithm of the **sum** of two exp terms. This is not as easily solvable.

This is where the EM algorithm comes in.

EM for Maximum Likelihood Estimation, Example (3)

1. Initialize $\hat{\pi}_A, \hat{\pi}_B, \hat{\mu}_A, \hat{\mu}_B, \hat{\sigma}_A^2, \hat{\sigma}_B^2$
2. For each of the data \mathbf{x}_k , compute

$$v_k^A = \hat{\pi}_A \mathcal{N}(\mathbf{x}_k | \hat{\mu}_A, \hat{\sigma}_A^2), \quad v_k^B = \hat{\pi}_B \mathcal{N}(\mathbf{x}_k | \hat{\mu}_B, \hat{\sigma}_B^2) \quad (5)$$

$$q_k^A = \frac{v_k^A}{v_k^A + v_k^B}, \quad q_k^B = \frac{v_k^B}{v_k^A + v_k^B} \quad (6)$$

3. Use q_k^A and q_k^B as weights. That is, if, say, $(q_k^A, q_k^B) = (0.2, 0.8)$, act as if 20% of point \mathbf{x}_k were from distribution A and 80% of that point were from distribution B. Update the estimates for the respective distributions as follows:

$$\hat{\mu}_A = \frac{1}{\sum_{i=1}^N q_k^A} \sum_{i=1}^N q_k^A \mathbf{x}_k \quad (7)$$

$$\hat{\sigma}_A^2 = \frac{1}{2 \sum_{i=1}^N q_k^A} \sum_{i=1}^N q_k^A \|\mathbf{x}_k - \hat{\mu}_A\|^2 \quad (8)$$

$$\hat{\pi}_A = \frac{1}{N} \sum_{i=1}^N q_k^A \quad (9)$$

4. (and analogously for B). Iterate.

Example: Mixture of Gaussians (general non-isotropic case)

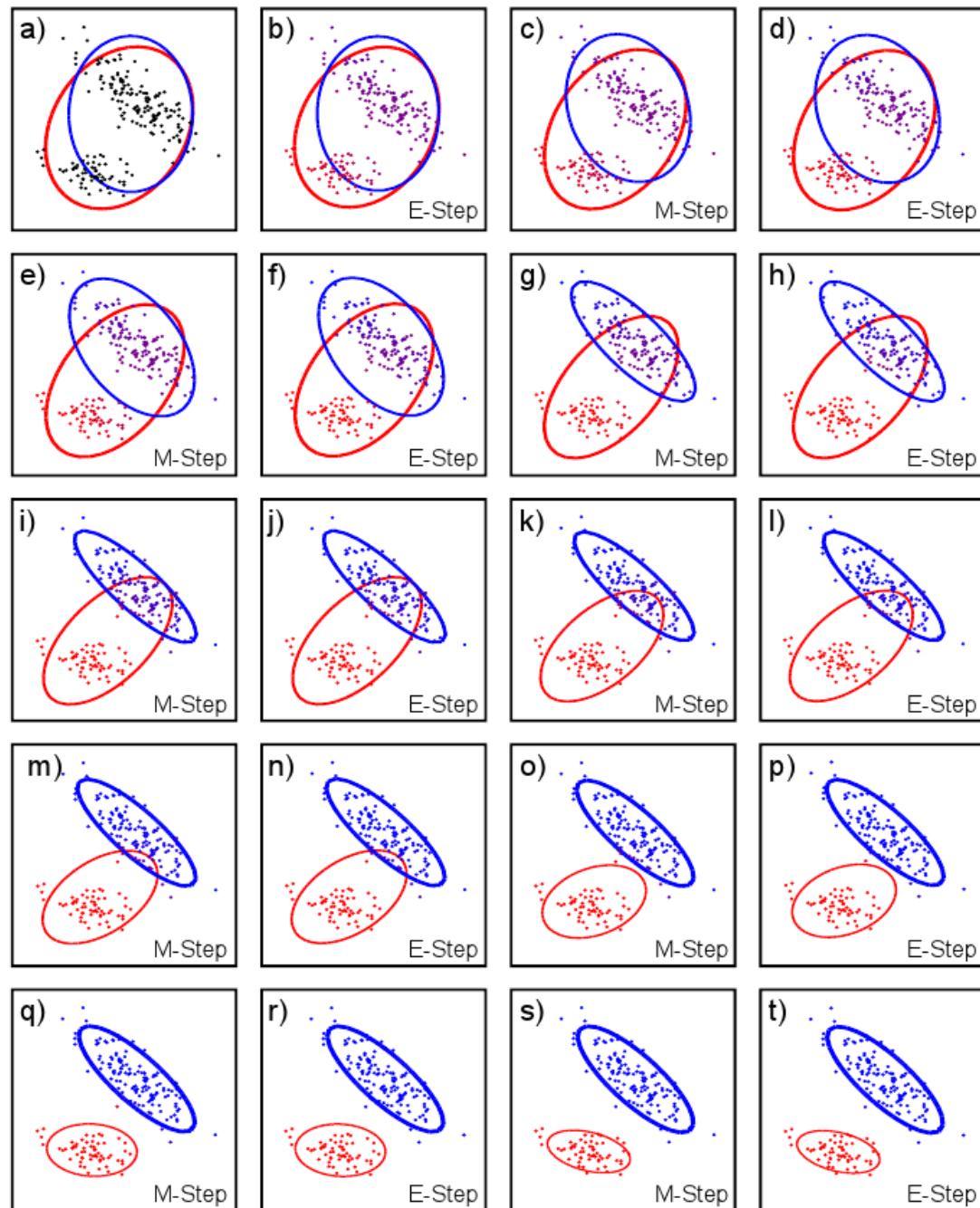


Figure 7.10 a) Initial model. b) E-step. For each data point the posterior probability that it was generated from each Gaussian is calculated (indicated by color of point). c) M-step. The mean, variance and weight of each Gaussian is updated based on these posterior probabilities. Ellipse shows Mahalanobis distance of two. Weight (thickness) of ellipse indicates weight of Gaussian. d-t) Further E-step and M-step iterations.

Image courtesy of Simon Prince. Computer Vision: Models, Learning and Inference, 2012

Toy Example 1: Estimating Means of Two Normal Distributions

We measure lengths of vehicles. The observation space has two dimensions, with $x \in \{\text{car, truck}\}$ capturing vehicle type and $y \in \mathbb{R}$ capturing length.

$$p(x, y) : \text{distribution , } \quad x \in \{\text{car, truck}\} , \quad y \in \mathbb{R} \quad (10)$$



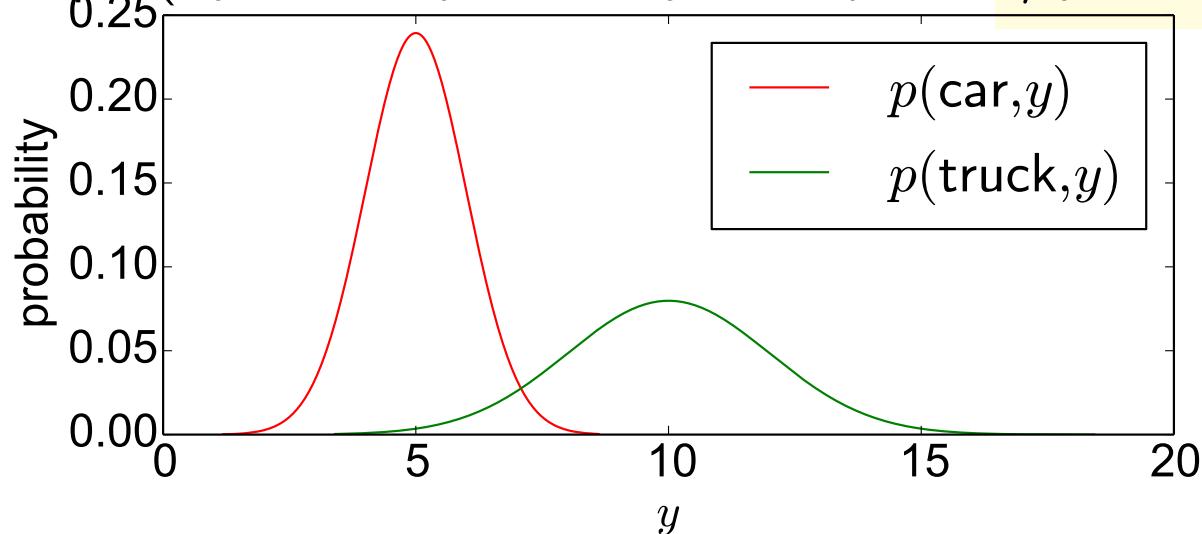
$$p(\text{car}, y) = \pi_c \mathcal{N}(y | \mu_c, \sigma_c^2 = 1) = \kappa_c \exp \left\{ -\frac{1}{2} (y - \mu_c)^2 \right\} , \quad (\kappa_c = \frac{\pi_c}{\sqrt{2\pi}}) \quad (11)$$



$$p(\text{truck}, y) = \pi_t \mathcal{N}(y | \mu_t, \sigma_t^2 = 4) = \kappa_t \exp \left\{ -\frac{1}{8} (y - \mu_t)^2 \right\} , \quad (\kappa_t = \frac{\pi_t}{\sqrt{8\pi}}) \quad (12)$$

Suppose $\kappa_c, \kappa_t, \sigma_c, \sigma_t$ are known. The **only unknowns** are μ_c and μ_t . We want to recover μ_c and μ_t using Maximum Likelihood.

Example ($\pi_c = 0.6, \pi_t = 0.4, \sigma_c = 1, \sigma_t = 2, \mu_c = 5, \mu_t = 10$)



Toy Example 1, Complete Data → Easy

The observations are:

$$\mathcal{T} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\} \quad (13)$$

$$= \underbrace{\{(\text{car}, y_1^{(c)}), (\text{car}, y_2^{(c)}), \dots, (\text{car}, y_C^{(c)})\}}_{C \text{ car observations}}, \underbrace{\{(\text{truck}, y_1^{(t)}), (\text{truck}, y_2^{(t)}), \dots, (\text{truck}, y_T^{(t)})\}}_{T \text{ truck observations}} \quad (14)$$

Log-likelihood $\ell(\mathcal{T}) = \ln p(\mathcal{T} | \mu_c, \mu_t)$:

$$\ell(\mathcal{T}) = \sum_{i=1}^N \ln p(x_i, y_i | \mu_c, \mu_t) = C \ln \kappa_c - \frac{1}{2} \sum_{i=1}^C (y_i^{(c)} - \mu_c)^2 + T \ln \kappa_t - \frac{1}{8} \sum_{i=1}^T (y_i^{(t)} - \mu_t)^2 \quad (15)$$

Estimation of μ_1, μ_2 using ML is easy:

$$\frac{\partial \ell(\mathcal{T})}{\partial \mu_c} = \sum_{i=1}^C (y_i^{(c)} - \mu_c) = 0 \quad \Rightarrow \quad \mu_c = \frac{1}{C} \sum_{i=1}^C y_i^{(c)} \quad (16)$$

$$\frac{\partial \ell(\mathcal{T})}{\partial \mu_t} = \frac{1}{4} \sum_{i=1}^T (y_i^{(t)} - \mu_t) = 0 \quad \Rightarrow \quad \mu_t = \frac{1}{T} \sum_{i=1}^T y_i^{(t)} \quad (17)$$

Toy Example 1, Incomplete Data → Difficult (1)

Consider some observations to have the first coordinate **missing** (\bullet):

$$\mathcal{T} = \{(car, y_1^{(c)}), \dots, (car, y_C^{(c)}), (truck, y_1^{(t)}), \dots, (truck, y_T^{(t)}), \underbrace{(\bullet, y_1^\bullet), \dots, (\bullet, y_M^\bullet)}_{\text{data with unknown vehicle type}}\} \quad (18)$$

Probability $p(y^\bullet)$ of observing y^\bullet :

$$p(y^\bullet) = p(car, y^\bullet) + p(truck, y^\bullet)$$

Log-likelihood:

$$\ell(\mathcal{T}) = \sum_{i=1}^N \ln p(x_i, y_i | \mu_c, \mu_t) = \overbrace{C \ln \kappa_c - \frac{1}{2} \sum_{i=1}^C (y_i^{(c)} - \mu_c)^2 + T \ln \kappa_t - \frac{1}{8} \sum_{i=1}^T (y_i^{(t)} - \mu_t)^2}^{\text{same term as before}} \quad (19)$$

$$+ \sum_{i=1}^M \ln \left(\kappa_c \exp \left\{ -\frac{1}{2} (y_i^\bullet - \mu_c)^2 \right\} + \kappa_t \exp \left\{ -\frac{1}{8} (y_i^\bullet - \mu_t)^2 \right\} \right) \quad (20)$$

Toy Example 1, Incomplete Data → Difficult (2)

Log-likelihood:

$$\ell(\mathcal{T}) = C \ln \kappa_c - \frac{1}{2} \sum_{i=1}^C (y_i^{(c)} - \mu_c)^2 + T \ln \kappa_t - \frac{1}{8} \sum_{i=1}^T (y_i^{(t)} - \mu_t)^2 \quad (21)$$

$$+ \sum_{i=1}^M \ln \left(\kappa_c \exp \left\{ -\frac{1}{2} (y_i^\bullet - \mu_c)^2 \right\} + \kappa_t \exp \left\{ -\frac{1}{8} (y_i^\bullet - \mu_t)^2 \right\} \right) \quad (22)$$

Optimality condition (shown for μ_c only):

$$0 = \frac{\partial \ell(\mathcal{T})}{\partial \mu_c} = \sum_{i=1}^C (y_i^{(c)} - \mu_c) + \quad (23)$$

$$+ \sum_{i=1}^M \frac{\kappa_c \exp \left\{ -\frac{1}{2} (y_i^\bullet - \mu_c)^2 \right\}}{\kappa_c \exp \left\{ -\frac{1}{2} (y_i^\bullet - \mu_c)^2 \right\} + \kappa_t \exp \left\{ -\frac{1}{8} (y_i^\bullet - \mu_t)^2 \right\}} (y_i^\bullet - \mu_c) \quad (24)$$

Missing Values, Optimality Condition

Log-likelihood:

$$\ell(\mathcal{T}) = C \ln \kappa_c - \frac{1}{2} \sum_{i=1}^C (y_i^{(c)} - \mu_c)^2 + T \ln \kappa_t - \frac{1}{8} \sum_{i=1}^T (y_i^{(t)} - \mu_t)^2 \quad (25)$$

$$+ \sum_{i=1}^M \ln \left(\kappa_c \exp \left\{ -\frac{1}{2} (y_i^\bullet - \mu_c)^2 \right\} + \kappa_t \exp \left\{ -\frac{1}{8} (y_i^\bullet - \mu_t)^2 \right\} \right) \quad (26)$$

Optimality condition (shown for μ_c only):

$$0 = \frac{\partial \ell(\mathcal{T})}{\partial \mu_c} = \sum_{i=1}^C (y_i^{(c)} - \mu_c) + \underbrace{p(\text{car}, y_i^\bullet | \mu_c, \mu_t)}_{\kappa_c \exp \left\{ -\frac{1}{2} (y_i^\bullet - \mu_c)^2 \right\}} \quad (27)$$

$$+ \sum_{i=1}^M \frac{\underbrace{\kappa_c \exp \left\{ -\frac{1}{2} (y_i^\bullet - \mu_c)^2 \right\}}_{p(\text{car}, y_i^\bullet | \mu_c, \mu_t)} + \underbrace{\kappa_t \exp \left\{ -\frac{1}{8} (y_i^\bullet - \mu_t)^2 \right\}}_{p(\text{truck}, y_i^\bullet | \mu_c, \mu_t)}}{\underbrace{\kappa_c \exp \left\{ -\frac{1}{2} (y_i^\bullet - \mu_c)^2 \right\} + \kappa_t \exp \left\{ -\frac{1}{8} (y_i^\bullet - \mu_t)^2 \right\}}}_{(y_i^\bullet - \mu_c)} \quad (28)$$

Missing Values, Optimality Condition

Log-likelihood:

$$\ell(\mathcal{T}) = C \ln \kappa_c - \frac{1}{2} \sum_{i=1}^C (y_i^{(c)} - \mu_c)^2 + T \ln \kappa_t - \frac{1}{8} \sum_{i=1}^T (y_i^{(t)} - \mu_t)^2 \quad (29)$$

$$+ \sum_{i=1}^M \ln \left(\kappa_c \exp \left\{ -\frac{1}{2} (y_i^\bullet - \mu_c)^2 \right\} + \kappa_t \exp \left\{ -\frac{1}{8} (y_i^\bullet - \mu_t)^2 \right\} \right) \quad (30)$$

Optimality condition (shown for μ_c only):

$$0 = \frac{\partial \ell(\mathcal{T})}{\partial \mu_c} = \sum_{i=1}^C (y_i^{(c)} - \mu_c) + \underbrace{p(\text{car}|y_i^\bullet, \mu_c, \mu_t)}_{\kappa_c \exp \left\{ -\frac{1}{2} (y_i^\bullet - \mu_c)^2 \right\}} \quad (31)$$

$$+ \sum_{i=1}^M \frac{\kappa_c \exp \left\{ -\frac{1}{2} (y_i^\bullet - \mu_c)^2 \right\}}{\kappa_c \exp \left\{ -\frac{1}{2} (y_i^\bullet - \mu_c)^2 \right\} + \kappa_t \exp \left\{ -\frac{1}{8} (y_i^\bullet - \mu_t)^2 \right\}} (y_i^\bullet - \mu_c) \quad (32)$$

Missing Values, Optimality Conditions

Optimality conditions (shown for both μ_c and μ_t):

$$0 = \frac{\partial \ell(\mathcal{T})}{\partial \mu_c} = \sum_{i=1}^C (y_i^{(c)} - \mu_c) + \underbrace{p(\text{car}|y_i^\bullet, \mu_c, \mu_t)}_{\text{(33)}}$$

$$+ \sum_{i=1}^M \left[\frac{\kappa_c \exp \left\{ -\frac{1}{2} (y_i^\bullet - \mu_c)^2 \right\}}{\kappa_c \exp \left\{ -\frac{1}{2} (y_i^\bullet - \mu_c)^2 \right\} + \kappa_t \exp \left\{ -\frac{1}{8} (y_i^\bullet - \mu_t)^2 \right\}} \right] (y_i^\bullet - \mu_c) \quad \text{(34)}$$

$$0 = 4 \frac{\partial \ell(\mathcal{T})}{\partial \mu_t} = \sum_{i=1}^T (y_i^{(t)} - \mu_t) + \sum_{i=1}^M p(\text{truck}|y_i^\bullet, \mu_c, \mu_t) (y_i^\bullet - \mu_t) \quad \text{(35)}$$

Note:

- ◆ Complicated equations for the unknowns μ_c, μ_t
- ◆ Both equations contain μ_c and μ_t (cf. case with no missing variables)

Missing Values, EM Approach

Optimality conditions (shown for both μ_c and μ_t):

$$\sum_{i=1}^C (y_i^{(c)} - \mu_c) + \sum_{i=1}^M p(\text{car}|y_i^\bullet, \mu_c, \mu_t) (y_i^\bullet - \mu_c) = 0 \quad (36)$$

$$\sum_{i=1}^T (y_i^{(t)} - \mu_t) + \sum_{i=1}^M p(\text{truck}|y_i^\bullet, \mu_c, \mu_t) (y_i^\bullet - \mu_t) = 0 \quad (37)$$

If $p(\text{car}|y_i^\bullet, \mu_c, \mu_t)$ and $p(\text{truck}|y_i^\bullet, \mu_c, \mu_t)$ were known, the estimation would've been easy:

- ◆ Let z_i ($i = 1, 2, \dots, M$), $z_i \in \{\text{car}, \text{truck}\}$ denote the missing values. Define $q(z_i) = p(z_i|y_i^\bullet, \mu_c, \mu_t)$
- ◆ The equations lead to

$$\sum_{i=1}^C (y_i^{(c)} - \mu_c) + \sum_{i=1}^M q(z_i = \text{car}) (y_i^\bullet - \mu_c) = 0 \quad (38)$$

$$\Rightarrow \mu_c = \frac{\sum_{i=1}^C y_i^{(c)} + \sum_{i=1}^M q(z_i = \text{car}) y_i^\bullet}{C + \sum_{i=1}^M q(z_i = \text{car})} \quad (39)$$

and similarly,

$$\mu_t = \frac{\sum_{i=1}^T y_i^{(t)} + \sum_{i=1}^M q(z_i = \text{truck}) y_i^\bullet}{T + \sum_{i=1}^M q(z_i = \text{truck})} \quad (40)$$

Missing Values, EM Approach

$$\mu_c = \frac{\sum_{i=1}^C y_i^{(c)} + \sum_{i=1}^M q(z_i = \text{car}) y_i^\bullet}{C + \sum_{i=1}^M q(z_i = \text{car})} \quad (41)$$

$$\mu_t = \frac{\sum_{i=1}^T y_i^{(t)} + \sum_{i=1}^M q(z_i = \text{truck}) y_i^\bullet}{T + \sum_{i=1}^M q(z_i = \text{truck})} \quad (42)$$

- ◆ These expressions are weighted averages of the observed y 's. Data with non-missing x have weight 1, the data with missing x have weight $q(z_i)$. How about trying the following procedure for finding the ML estimate of μ_c and μ_t :
 1. Initialize μ_c, μ_t
 2. Compute $q(z_i) = p(z_i | y_i^\bullet, \mu_c, \mu_t)$ for all $i = 1, 2, \dots, M$
 3. Recompute μ_c, μ_t according to Eqs.(41, 42)
 4. If termination condition is met, finish. Otherwise goto 2.
- ◆ This is the essence of the **EM algorithm**, with Step 2 called the **Expectation (E)** step and Step 3 called the **Maximization (M)** step.

Clustering, Soft Assignment, Relation to K-means (1)

An extreme of the previous example is that **no** data have the x -coordinate value (car/truck vehicle type). Everything works just as well:

$$\mu_c = \frac{\sum_{i=1}^M q(z_i = \text{car}) y_i^\bullet}{\sum_{i=1}^M q(z_i = \text{car})} \quad (43)$$

$$\mu_t = \frac{\sum_{i=1}^M q(z_i = \text{truck}) y_i^\bullet}{\sum_{i=1}^M q(z_i = \text{truck})} \quad (44)$$

1. Initialize μ_c, μ_t
2. Compute $q(z_i) = p(z_i | y_i^\bullet, \mu_c, \mu_t)$ for all $i = 1, 2, \dots, M$
3. Recompute μ_c, μ_t according to Eqs.(45, 46)
4. If termination condition is met, finish. Otherwise goto 2.

Note: Can you imagine this algorithm to end up at a local maximum?

Clustering, Soft Assignment, Relation to K-means (2)

An extreme of the previous example is that **no** data have the x -coordinate (car/truck).

$$\mu_c = \frac{\sum_{i=1}^M q(z_i = \text{car}) y_i^\bullet}{\sum_{i=1}^M q(z_i = \text{car})} \quad (45)$$

$$\mu_t = \frac{\sum_{i=1}^M q(z_i = \text{truck}) y_i^\bullet}{\sum_{i=1}^M q(z_i = \text{truck})} \quad (46)$$

EM algorithm:

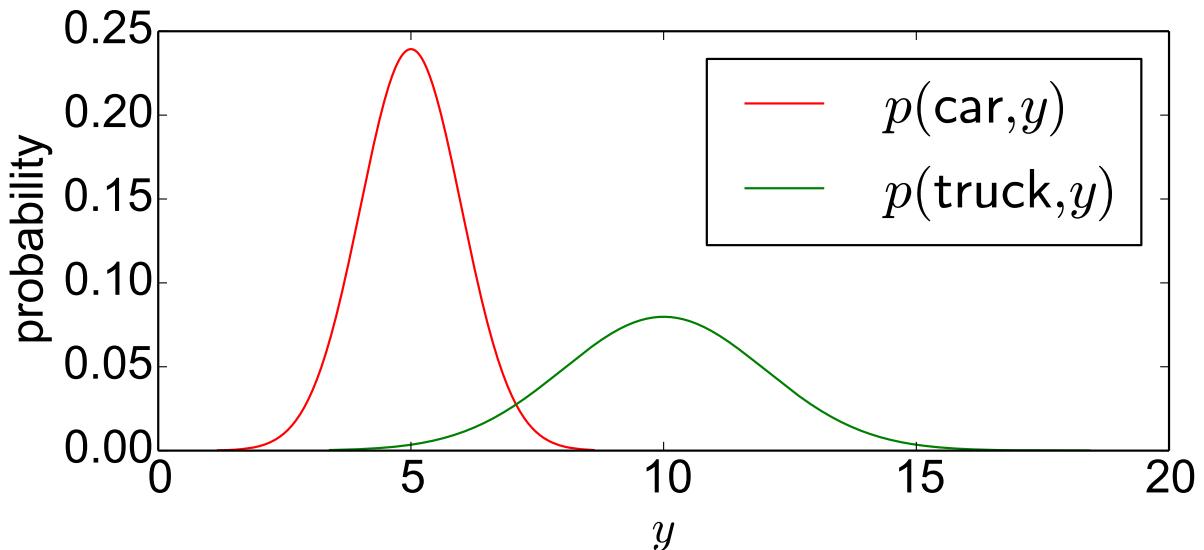
1. Initialize μ_c, μ_t
 2. Compute $q(z_i) = p(z_i | y_i^\bullet, \mu_c, \mu_t)$
for all $i = 1, 2, \dots, M$
 3. Recompute μ_c, μ_t according to Eqs.(45, 46)
 4. If termination condition is met, finish.
Otherwise goto 2.
1. ditto
 2. $q(z_i = \text{car}) = \llbracket |y_i^\bullet - \mu_c| < |y_i^\bullet - \mu_t| \rrbracket$
 $q(z_i = \text{truck}) = \llbracket |y_i^\bullet - \mu_t| \leq |y_i^\bullet - \mu_c| \rrbracket$
for all $i = 1, 2, \dots, M$
 3. ditto
 4. ditto

K-means:

EM-based clustering uses soft assignment. K-means can be interpreted as an EM-based clustering with hard assignment.

Example 1 - Setting

$$\pi_c = 0.6, \pi_t = 0.4, \sigma_c = 1, \sigma_t = 2, \mu_c = 5, \mu_t = 10$$



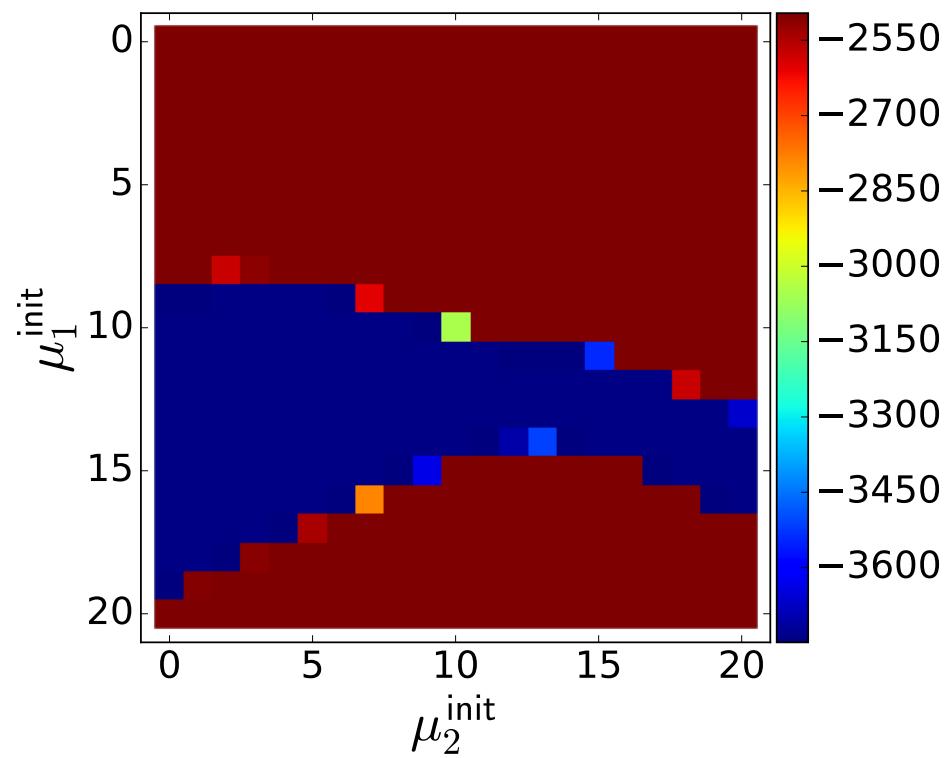
Data:

- ◆ 50 points from car distribution,
50 points from truck d.,
1000 points from mixed
distribution (car/truck
coordinate unknown)

Experiment:

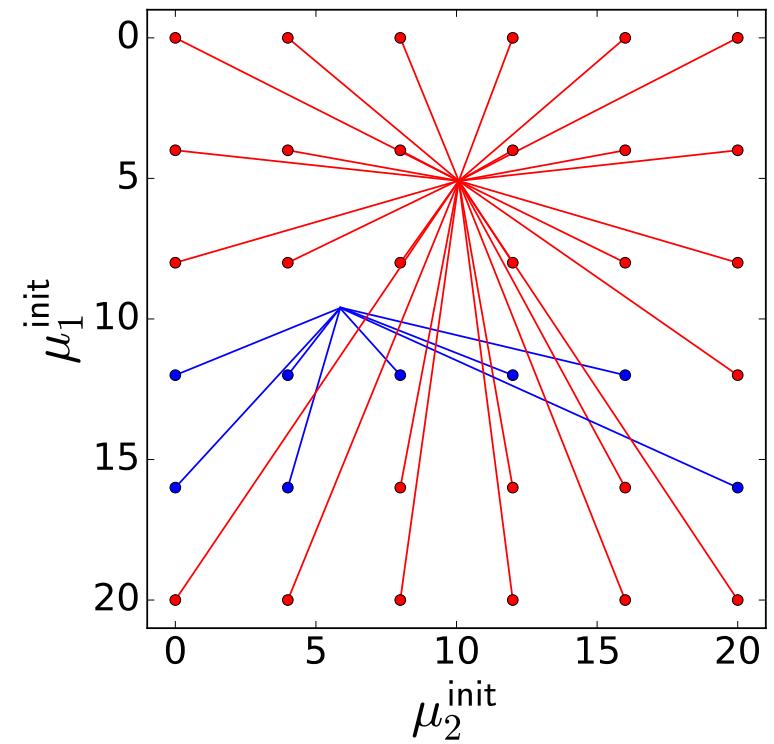
Employ EM algorithm for estimating μ_1, μ_2 . Use different initializations.

Example 1 - Result



Log-likelihood ℓ after 10 iterations of EM,
depending on initialization $(\mu_1^{\text{init}}, \mu_2^{\text{init}})$.

Convergence in this case is quite fast
(3 iterations are enough for most of the
initialization values.)



Value of (μ_1, μ_2) after 10 iterations,
depending on initialization $(\mu_1^{\text{init}}, \mu_2^{\text{init}})$. The
first point of convergence corresponds to
the ground truth values $(\mu_1, \mu_2) = (5, 10)$.
The **second** point is a only a local
maximum of log-likelihood. It corresponds
to car distribution approximating truck
sample points, and vice versa.

Mixture Models

Generalization of the Motivation example with missing values.

$$\mu_c = \frac{\sum_{i=1}^M q(z_i = \text{car}) y_i^\bullet}{\sum_{i=1}^M q(z_i = \text{car})} \quad (47)$$

$$\sigma_c^2 = \frac{\sum_{i=1}^M q(z_i = \text{car}) (y_i^\bullet - \mu_c)^2}{\sum_{i=1}^M q(z_i = \text{car})} \quad (48)$$

$$\pi_c = \frac{\sum_{i=1}^M q(z_i = \text{car})}{M} \quad (49)$$

Toy Example 2: (Temperature \times Snow) Model Estimation

You are measuring temperature and amount of snow in the mountains in the month of January. Both the temperature t and the snow s observations are binary:



$$t \in \{t_0 = \text{low temperature}, t_1 = \text{high temperature}\} \quad (50)$$

$$s \in \{s_0 = \text{little snow}, s_1 = \text{lot of snow}\} \quad (51)$$

Your own long-term research suggests that the model for the joint probability $p(t, s)$ can be parametrized by two scalars a and b and written as

$$\begin{array}{c}
 p(t, s | a, b) \\
 \begin{array}{|c|c|c|} \hline
 t_0 & a & 5a \\ \hline
 t_1 & 3b & b \\ \hline
 \end{array} \\
 \begin{array}{|c|c|} \hline
 & s_0 & s_1 \\ \hline
 \end{array}
 \end{array} \quad (52)$$

At a big ski-center, you have N measurements in total, with counts for individual possibilities for t and s as follows:

observation counts		
t_0	N_{00}	N_{01}
t_1	N_{10}	N_{11}
	s_0	s_1

(53)

What is the ML estimate for a and b ?

Toy Example 2: Model Estimation

$$p(t, s | a, b)$$

t_0	a	$5a$
t_1	$3b$	b
	s_0	s_1

observation counts

t_0	N_{00}	N_{01}
t_1	N_{10}	N_{11}
	s_0	s_1

Likelihood is $P(\mathcal{T}|a, b) = a^{N_{00}}(5a)^{N_{01}}(3b)^{N_{10}}(b)^{N_{11}}$.

Log-likelihood is $\ell(\mathcal{T}|a, b) = N_{00} \ln a + N_{01} \ln 5a + N_{10} \ln 3b + N_{11} \ln b$. Maximize this log-likelihood s.t. $6a + 4b = 1$. The Lagrangian is

$L(a, b, \lambda) = N_{00} \ln a + N_{01} \ln 5a + N_{10} \ln 3b + N_{11} \ln b + \lambda(6a + 4b - 1)$. Conditions of optimality are:

$$\frac{\partial L}{\partial a} = N_{00} \frac{1}{a} + N_{01} \frac{1}{a} + 6\lambda = 0 \quad (54)$$

$$\frac{\partial L}{\partial b} = N_{10} \frac{1}{b} + N_{11} \frac{1}{b} + 4\lambda = 0 \quad (55)$$

$$6a + 4b = 1 \quad (56)$$

and they have the solution ($N = N_{00} + N_{01} + N_{10} + N_{11}$):

$$a = \frac{N_{00} + N_{01}}{6N} \quad b = \frac{N_{10} + N_{11}}{4N} \quad (57)$$

Toy Example 2: Model Estimation (Incomplete Data)

Now imagine you have data from little village in the mountains. Unfortunately, there is *no* measurement for which both temperature and snow amount would be available. The data consist only of T_0 reports of low temperature, T_1 of high temperature, S_0 of little snow and S_1 of lots of snow.

$p(t, s a, b)$		
t_0	a	$5a$
t_1	$3b$	b
	s_0	s_1

\Rightarrow

$p(t_0)$	$6a$
$p(t_1)$	$4b$
$p(s_0)$	$a + 3b$
$p(s_1)$	$5a + b$

observation counts

t_0	T_0
t_1	T_1
s_0	S_0
s_1	S_1

Log-likelihood is $\ell(\mathcal{T}|a, b) = T_0 \ln 6a + T_1 \ln 4b + S_0 \ln(a + 3b) + S_1 \ln(5a + b)$.

Maximize this log-likelihood s.t. $6a + 4b = 1$. The Lagrangian is

$$L(a, b, \lambda) = T_0 \ln 6a + T_1 \ln 4b + S_0 \ln(a + 3b) + S_1 \ln(5a + b) + \lambda(6a + 4b - 1).$$

Conditions of optimality:

$$\frac{\partial L}{\partial a} = \frac{T_0}{a} + \frac{S_0}{a + 3b} + \frac{5S_1}{5a + b} + 6\lambda = 0 \quad (58)$$

$$\frac{\partial L}{\partial b} = \frac{T_1}{b} + \frac{3S_0}{a + 3b} + \frac{3S_1}{5a + b} + 4\lambda = 0 \quad (59)$$

$$6a + 4b = 1 \quad (60)$$

→ Not as easy to solve as in the previous case!

Toy Example 2: Model Estimation using EM algorithm

This is what EM algorithm would do to maximize likelihood for these incomplete data.

24/32

1. Make initial estimate of a and b
2. **E-step:** For each observation, compute the distribution over the missing value, given the observed value and current estimate of a and b .
E.g. observation (\bullet, s_0) where ' \bullet ' is the unknown temperature t and s_0 is the observed low amount of snow. The distrib. $q(t) = p(t|s_0, a, b)$ is computed as follows:

$p(t, s a, b)$		
t_0	a	$5a$
t_1	$3b$	b
	s_0	s_1

$$q(t_0) = p(t_0|s_0, a, b) = \frac{a}{a + 3b} \quad (61)$$

$$q(t_1) = p(t_1|s_0, a, b) = \frac{3b}{a + 3b} \quad (62)$$

3. **M-step:** Recompute parameters a , b :

Use the distribution q computed in the previous step as weights.

I.e. the considered incomplete observation (\bullet, s_0) produces two complete observations: (t_0, s_0) with weight $q(t_0)$, and (t_1, s_0) with weight $q(t_1)$.

Let w_{ij} be the sum of weights for observations (t_i, s_j) across the entire dataset. Then a and b are computed (using the result for complete data) as:

$$a = \frac{w_{00} + w_{01}}{6N}, \quad b = \frac{w_{10} + w_{11}}{4N} \quad (63)$$

4. Iterate (go to 2.)

EM algorithm - Derivation

- ◆ \mathcal{T} : training set
- ◆ \mathbf{o} : all observed values (no essential difference between \mathcal{T} and \mathbf{o} , just notational convenience)
- ◆ \mathbf{z} : all unobserved values
- ◆ $\boldsymbol{\theta}$: model parameters to be estimated.

Goal: Find $\boldsymbol{\theta}^*$ using the Maximum Likelihood approach:

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \operatorname{argmax}_{\boldsymbol{\theta}} \ln p(\mathbf{o} | \boldsymbol{\theta}) \quad (64)$$

Line of thought

Assume that solving this:

$$\operatorname{argmax}_{\boldsymbol{\theta}} \ln p(\mathbf{o}, \mathbf{z} | \boldsymbol{\theta}) \quad (65)$$

is easy (that is, estimation of optimal parameters had the data been complete.)

Our goal will be to rewrite Eq. (64) in a way which will involve optimization terms of kind as in Eq. (65).

Lower Bound on the Log Likelihood

$$\ln p(\mathbf{o}|\boldsymbol{\theta}) = \ln \sum_{\mathbf{z}} p(\mathbf{o}, \mathbf{z}|\boldsymbol{\theta}) \quad (66)$$

$$= \ln \sum_{\mathbf{z}} q(\mathbf{z}) \frac{p(\mathbf{o}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z})}$$

$$\geq \sum_{\mathbf{z}} q(\mathbf{z}) \ln \frac{p(\mathbf{o}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z})}$$

Introduction of distribution $q(\mathbf{z})$ (67)

As $\forall \mathbf{z} : 0 \leq q(\mathbf{z}) \leq 1$ and $\sum_{\mathbf{z}} q(\mathbf{z}) = 1$, the sum is now a convex combination of $p(\mathbf{o}, \mathbf{z}|\boldsymbol{\theta})/q(\mathbf{z})$.

Jensen's inequality. Here inequality holds because logarithm is a concave function. (68)

Define

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\mathbf{z}} q(\mathbf{z}) \ln \frac{p(\mathbf{o}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z})}. \quad (69)$$

This $\mathcal{L}(q, \boldsymbol{\theta})$ is the lower bound for $\ln p(\mathbf{o}|\boldsymbol{\theta})$ due to Eq. (68), for any distribution q .

Maximizing $\mathcal{L}(q, \boldsymbol{\theta})$ will also push the log likelihood $\ln p(\mathbf{o}|\boldsymbol{\theta})$ upwards.

How Tight Is This Bound? (1)

$$\ln p(\mathbf{o}|\boldsymbol{\theta}) - \mathcal{L}(q, \boldsymbol{\theta}) = \ln p(\mathbf{o}|\boldsymbol{\theta}) - \sum_{\mathbf{z}} q(\mathbf{z}) \ln \frac{p(\mathbf{o}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z})} \quad (70)$$

$$= \ln p(\mathbf{o}|\boldsymbol{\theta}) - \sum_{\mathbf{z}} q(\mathbf{z}) \left\{ \underbrace{\ln p(\mathbf{o}, \mathbf{z}|\boldsymbol{\theta})}_{p(\mathbf{z}|\mathbf{o}, \boldsymbol{\theta})p(\mathbf{o}|\boldsymbol{\theta})} - \ln q(\mathbf{z}) \right\} \quad (71)$$

$$= \ln p(\mathbf{o}|\boldsymbol{\theta}) - \sum_{\mathbf{z}} q(\mathbf{z}) \left\{ \ln p(\mathbf{z}|\mathbf{o}, \boldsymbol{\theta}) + \ln p(\mathbf{o}|\boldsymbol{\theta}) - \ln q(\mathbf{z}) \right\} \quad (72)$$

$$= \ln p(\mathbf{o}|\boldsymbol{\theta}) - \underbrace{\sum_{\mathbf{z}} q(\mathbf{z}) \ln p(\mathbf{o}|\boldsymbol{\theta})}_{1} - \sum_{\mathbf{z}} q(\mathbf{z}) \left\{ \ln p(\mathbf{z}|\mathbf{o}, \boldsymbol{\theta}) - \ln q(\mathbf{z}) \right\} \quad (73)$$

$$= - \sum_{\mathbf{z}} q(\mathbf{z}) \ln \frac{p(\mathbf{z}|\mathbf{o}, \boldsymbol{\theta})}{q(\mathbf{z})} \quad (74)$$

This is the Kullback Leibler divergence between the two distributions $q(\mathbf{z})$ and $p(\mathbf{z}|\mathbf{o}, \boldsymbol{\theta})$:

$$D_{\text{KL}}(q||p) = \sum_{\mathbf{z}} q(\mathbf{z}) \ln \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{o}, \boldsymbol{\theta})} = - \sum_{\mathbf{z}} q(\mathbf{z}) \ln \frac{p(\mathbf{z}|\mathbf{o}, \boldsymbol{\theta})}{q(\mathbf{z})} \quad (75)$$

How Tight Is This Bound? (2)

$$\ln p(\mathbf{o}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + D_{\text{KL}}(q||p) \quad (76)$$

↑ ↑ ↑

log likelihood lower bound gap

We already know that due to Jensen's inequality, $\mathcal{L}(q, \boldsymbol{\theta})$ is indeed the lower bound. This is confirmed by the fact that $D_{\text{KL}}(q||p) \geq 0$ for any q, p . Additionally,

$$D_{\text{KL}}(q||p) = 0 \quad \Leftrightarrow \quad p = q. \quad (77)$$

When $q = p$, the bound is tight.

EM algorithm

$$\ln p(\mathbf{o}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + D_{\text{KL}}(q||p) \quad (78)$$

↑ ↑ ↑

log likelihood lower bound gap

EM algorithm attempts to maximize the log-likelihood by instead maximizing the lower bound (why 'attempts'? Because it may end up in local maximum).

1. Initialize $\boldsymbol{\theta} = \boldsymbol{\theta}^{(0)}$ ($t = 0$)

2. **E-step** (Expectation):

$$q^{(t+1)} = \underset{q}{\operatorname{argmax}} \mathcal{L}(q, \boldsymbol{\theta}^{(t)}) \quad (79)$$

3. **M-step** (Maximization):

$$\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \mathcal{L}(q^{(t+1)}, \boldsymbol{\theta}) \quad (80)$$

4. If termination condition is not met, goto 2.

Expectation step

E-step: $\theta^{(t)}$ is fixed

$$q^{(t+1)} = \underset{q}{\operatorname{argmax}} \mathcal{L}(q, \theta^{(t)}) \quad (81)$$

$$\mathcal{L}(q, \theta^{(t)}) = \underbrace{\ln p(\mathbf{o} | \theta^{(t)})}_{\text{const.}} - D_{\text{KL}}(q || p) \quad (82)$$

Note: The distribution q maximizing this term is the one which minimizes the KL divergence. KL divergence is minimized when the two distributions are the same. Thus, the distribution maximizing Eq. (81) is

$$q^{(t+1)}(\mathbf{z}) = p(\mathbf{z} | \mathbf{o}, \theta^{(t)}). \quad \left[D_{\text{KL}}(q || p) = - \sum_{\mathbf{z}} q(\mathbf{z}) \ln \frac{p(\mathbf{z} | \mathbf{o}, \theta)}{q(\mathbf{z})} \right] \quad (83)$$

Note that this corresponds to what we've obtained e.g. in our car/truck example,

$$q_i^{(t+1)}(\text{car}) = p(\text{car} | y_i^\bullet, \mu_c, \mu_t), \quad q_i^{(t+1)}(\text{truck}) = p(\text{truck} | y_i^\bullet, \mu_c, \mu_t) \quad (84)$$

Maximization step

M-step: $q^{(t+1)}$ is fixed

$$\boldsymbol{\theta}^{(t+1)} = \operatorname{argmax}_{\boldsymbol{\theta}} \mathcal{L}(q^{(t+1)}, \boldsymbol{\theta}) \quad (85)$$

$$\mathcal{L}(q^{(t+1)}, \boldsymbol{\theta}) = \sum_{\mathbf{z}} q^{(t+1)}(\mathbf{z}) \ln \frac{p(\mathbf{o}, \mathbf{z} | \boldsymbol{\theta})}{q^{(t+1)}(\mathbf{z})} \quad (86)$$

$$= \sum_{\mathbf{z}} q^{(t+1)}(\mathbf{z}) \ln p(\mathbf{o}, \mathbf{z} | \boldsymbol{\theta}) - \underbrace{\sum_{\mathbf{z}} q^{(t+1)}(\mathbf{z}) \ln q^{(t+1)}(\mathbf{z})}_{\text{const.}} \quad (87)$$

Result: The parameters $\boldsymbol{\theta}$ maximizing Eq. (85) are

$$\boldsymbol{\theta}^{(t+1)} = \operatorname{argmax}_{\boldsymbol{\theta}} \sum_{\mathbf{z}} q^{(t+1)}(\mathbf{z}) \ln p(\mathbf{o}, \mathbf{z} | \boldsymbol{\theta}) . \quad (88)$$

Note that this maximization is done as if all data were known (observed) and thus is often easy (has analytic solution.) E.g. in the case of estimating mean of Gaussian mixture component, it leads to weighted average of data.

Summary

- ◆ EM's most known application is estimating Gaussian Mixtures. M-step computes probabilities that a given point is generated by given components, and E-step computes the unknown parameters effectively (analytic solution). EM algorithm is similarly useful and effective for more exponential family distributions.
- ◆ EM cleverly maximizes likelihood by pushing its lower bound upwards.
- ◆ It is an iterative method and may not end up in the global maximum.
- ◆ Attention needs to be applied to parameter initialization, like with other methods we've already encountered (K-means, NNs, . . .)

Principal Component Analysis

Lecturer:
Jiří Matas

Authors:
Ondřej Drbohlav, Jiří Matas

Centre for Machine Perception
Czech Technical University, Prague
<http://cmp.felk.cvut.cz>

1.1.2017



Introduction

- ◆ Alternative name: Karhunene Loeve transform
- ◆ Used for: data approximation, identifying sources of variance in the data

Maximum variance formulation (1/3)

Let the data be $\{\mathbf{x}_i \mid i = 1, 2, \dots, N\}$, with sample mean $\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$.

Let us find the unit vector \mathbf{u}_1 to project to such that the variance $J(\mathbf{u}_1)$ of the projected data is *maximized*. The projection $\mathbf{x}_n^{(p)}$ of an \mathbf{x}_n to one-dimensional subspace generated by \mathbf{u}_1 is given by

$$\mathbf{x}_n^{(p)} = \mathbf{u}_1 (\mathbf{u}_1^T \mathbf{x}_n), \quad \mathbf{u}_1^T \mathbf{u}_1 = 1. \quad (1)$$

The variance $J(\mathbf{u}_1)$ of projected data is

$$J(\mathbf{u}_1) = \frac{1}{N} \sum_{n=1}^N (\mathbf{u}_1^T \mathbf{x}_n - \mathbf{u}_1^T \bar{\mathbf{x}})^2 = \frac{1}{N} \sum_{n=1}^N \mathbf{u}_1^T (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T \mathbf{u}_1 = \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1, \quad (2)$$

where \mathbf{S} is the normalized scatter matrix:

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T. \quad (3)$$



The Lagrangian of this optimization problem is

$$L(\mathbf{u}_1, \lambda_1) = J(\mathbf{u}_1) + \lambda_1 \underbrace{(1 - \mathbf{u}_1^T \mathbf{u}_1)}_{\text{constraint}} = \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^T \mathbf{u}_1), \quad (4)$$

where λ_1 is the Lagrange multiplier. Taking the derivative w.r.t. the vector \mathbf{u}_1 and setting it to zero gives

$$\frac{\partial L(\mathbf{u}_1, \lambda_1)}{\partial \mathbf{u}_1} = \mathbf{S} \mathbf{u}_1 - \lambda_1 \mathbf{u}_1 = 0, \quad (5)$$

and thus

$$\mathbf{S} \mathbf{u}_1 = \lambda_1 \mathbf{u}_1. \quad (6)$$

This is the characteristic equation for the covariance matrix \mathbf{S} . Any eigenvalue λ_1 and its corresponding eigenvector \mathbf{v}_1 solves this equation, with variance $J(\mathbf{u}_1)$ equal to:

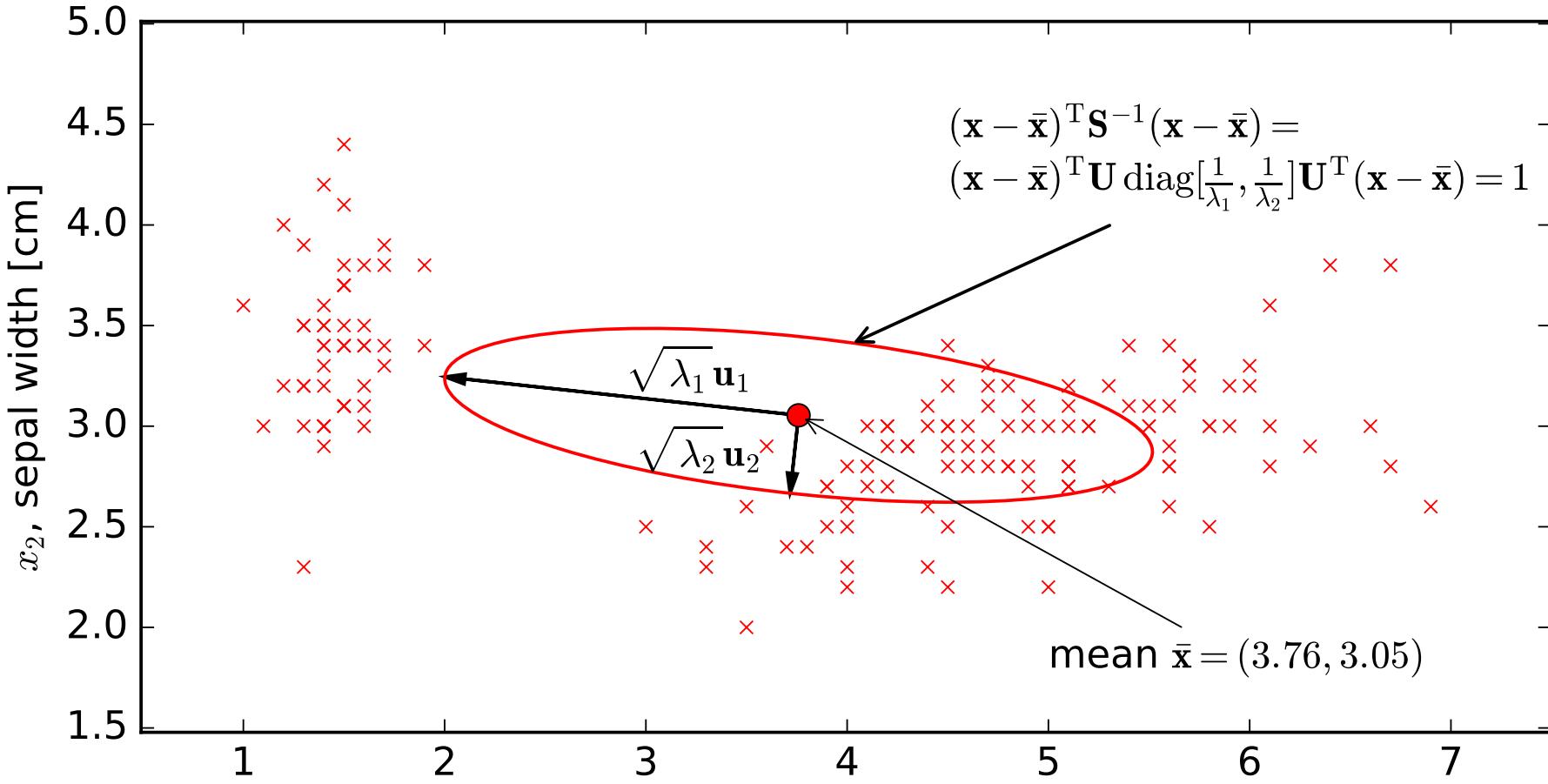
$$J(\mathbf{u}_1) = \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 = \mathbf{u}_1^T \lambda_1 \mathbf{u}_1 = \lambda_1. \quad (7)$$

The maximum is attained if λ_1 is the largest eigenvalue of the matrix \mathbf{S} and \mathbf{u}_1 is its corresponding eigenvector.



Example 1 - Iris dataset

Iris dataset: feature vectors are 4-dimensional, here dimensions 2 and 3 used (petal length and sepal width). Data shown as crosses \times .



$$\mathbf{S} = \begin{bmatrix} 3.09 & -0.32 \\ -0.32 & 0.19 \end{bmatrix} = [\mathbf{u}_1, \mathbf{u}_2] \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix} [\mathbf{u}_1, \mathbf{u}_2]^T$$

Eigenvectors: $[\mathbf{u}_1, \mathbf{u}_2] = \begin{bmatrix} -0.99 & -0.11 \\ 0.11 & -0.99 \end{bmatrix}$, eigenvalues: $\lambda_1 = 3.13$, $\lambda_2 = 0.15$

Variance is maximized when data are projected to direction \mathbf{u}_1 .

1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

17 18

19 20

21 22

23 24

25 26

27 28

29 30

Recall: The variance of a 1-D projection is maximized when data are projected to the direction of the eigenvector of \mathbf{S} corresponding to the largest eigenvalue.

\mathbf{S} is symmetric and positive semidefinite. The eigenvectors corresponding to different eigenvalues are orthogonal.

It follows that the D -dimensional subspace maximizing the variance of the data is the one formed by D eigenvectors of \mathbf{S} corresponding to the D largest eigenvalues.

Note: "Variance" in the above sentence is the sum of variances in individual orthogonal directions. For a 2-D subspace,

$$J(\mathbf{u}_1, \mathbf{u}_2) = \frac{1}{N} \sum_{n=1}^N [\mathbf{u}_1^T (\mathbf{x}_n - \bar{\mathbf{x}})]^2 + [\mathbf{u}_2^T (\mathbf{x}_n - \bar{\mathbf{x}})]^2. \quad (8)$$

1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

17 18

19 20

21 22

23 24

25 26

27 28

29 30

Consider the complete orthogonal basis $\{\mathbf{u}_i\}$ where $i = 1, \dots, D$. Thus

$$\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij} \quad (9)$$

Each point can be represented as

$$\mathbf{x}_n = \sum_{i=1}^D \alpha_{ni} \mathbf{u}_i, \quad (10)$$

and

$$\mathbf{x}_n = \sum_{i=1}^D (\mathbf{x}_n^T \mathbf{u}_i) \mathbf{u}_i. \quad (11)$$

This is just expressing \mathbf{x}_n in a rotated coordinate system given by orthonormal system $\{\mathbf{u}_i\}$. Let us create an approximation to each \mathbf{x}_n by truncating this expansion to only M components, the remaining $D - M$ components approximated by constants b_i .

The approximation $\tilde{\mathbf{x}}_n$:

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^M (\mathbf{x}_n^T \mathbf{u}_i) \mathbf{u}_i + \sum_{i=M+1}^D b_i \mathbf{u}_i \quad (12)$$

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^M (\mathbf{x}_n^T \mathbf{u}_i) \mathbf{u}_i + \sum_{i=M+1}^D b_i \mathbf{u}_i \quad (12)$$

Clearly,

$$b_i = \bar{\mathbf{x}}^T \mathbf{u}_i, i = M + 1, \dots, D \quad (13)$$

The task is to find the optimal orthonormal basis $\{\mathbf{u}_i\}$ which produces the best approximation measured by

$$J(\{\mathbf{u}_i\}) = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 \quad (14)$$

The minimum error criterion is the complement of the maximum variance criterion, and thus the solution to the set $\{\mathbf{u}_i\}$ is the same.

Recall that the ML estimate of the Multivariate Normal Distribution is defined by sample mean $\bar{\mathbf{x}}$ and sample covariance matrix \mathbf{S} . The model is

$$p(\mathbf{x} | \bar{\mathbf{x}}, \mathbf{S}) = \frac{1}{\sqrt{|2\pi\mathbf{S}|}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{S}^{-1} (\mathbf{x} - \bar{\mathbf{x}}) \right\} \quad (15)$$

Denote stacked eigenvectors in descending order of their eigenvalues as \mathbf{U} ,

$$\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_D\} \quad (16)$$

Therefore (characteristic equation)

$$\mathbf{S}\mathbf{U} = \mathbf{U}\Lambda = \mathbf{U} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_D \end{bmatrix}, \quad (17)$$

and

$$\mathbf{S} = \mathbf{U}\Lambda\mathbf{U}^T. \quad (18)$$



We approximate the data, as before, by projecting to first M eigenvectors. Thus, given data point \mathbf{x} we have

$$\mathbf{x} - \bar{\mathbf{x}} = (\delta_1, \delta_2, \dots, \delta_M, \delta_{M+1}, \dots, \delta_D) \quad (19)$$

Note that we only can compute $\delta_1 \dots \delta_M$, as often we don't or can't store all eigenvectors for computing all δ 's. However, we can easily compute

$$\Delta = \delta_{M+1}^2 + \delta_{M+2}^2 + \dots + \delta_D^2 = \|\mathbf{x} - \bar{\mathbf{x}}\|^2 - \delta_1^2 - \delta_2^2 - \dots - \delta_M^2 \quad (20)$$

and the exponent is then approximated as

$$-\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{S}^{-1}(\mathbf{x} - \bar{\mathbf{x}}) \simeq -\frac{1}{2} \left(\frac{\delta_1^2}{\lambda_1} + \frac{\delta_2^2}{\lambda_2} + \frac{\delta_3^2}{\lambda_3} + \dots + \frac{\delta_M^2}{\lambda_M} + \frac{\Delta}{\lambda} \right) \quad (21)$$

Common choice: $\lambda = \lambda_{M+1}$

1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

17 18

19 20

21 22

23 24

25 26

27 28

29 30



Dimensionality of data can be high, and even higher than number of samples.

Consider dimensionality $D = 1M$ (one million) and number of samples $N = 100$. All analysis still applies, but it would be wasteful to compute eigenvectors for the $1M \times 1M$ matrix, as its rank will anyway be at most N (thus 100). Let us define \mathbf{X} to be a matrix formed by stacking all the data vectors (after having subtracted the mean from them): $\mathbf{X} = [\mathbf{x}_1 - \bar{\mathbf{x}}, \mathbf{x}_2 - \bar{\mathbf{x}}, \dots, \mathbf{x}_N - \bar{\mathbf{x}}]$.

Thus,

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T = \frac{1}{N} \mathbf{X} \mathbf{X}^T. \quad (22)$$

The characteristic equation is then

$$\frac{1}{N} \mathbf{X} \mathbf{X}^T \mathbf{u} = \lambda \mathbf{u}. \quad (23)$$

Left-multiplying both sides by \mathbf{X}^T gives

$$\frac{1}{N} \mathbf{X}^T \mathbf{X} \overbrace{(\mathbf{X}^T \mathbf{u})}^{\mathbf{w}} = \lambda \overbrace{(\mathbf{X}^T \mathbf{u})}^{\mathbf{w}}. \quad (24)$$

1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

17 18

19 20

21 22

23 24

25 26

27 28

29 30



Thus, $\mathbf{X}^T \mathbf{X}$, which is only 100×100 , has exactly the same set of eigenvalues:

$$\frac{1}{N} \mathbf{X}^T \mathbf{X} \mathbf{w} = \lambda \mathbf{w}. \quad (25)$$

Left-multiplying now by \mathbf{X} , we get

$$\frac{1}{N} \mathbf{X} \mathbf{X}^T (\mathbf{X} \mathbf{w}) = \lambda (\mathbf{X} \mathbf{w}). \quad (26)$$

Conclusion: If $D \gg N$, form the matrix $\mathbf{T} = \frac{1}{N} \mathbf{X}^T \mathbf{X}$ and compute its eigenvalues λ 's and eigenvectors \mathbf{w} . Compute the eigenvectors of $\mathbf{S} = \frac{1}{N} \mathbf{X} \mathbf{X}^T$ as

$$\mathbf{v} = \frac{\mathbf{X} \mathbf{w}}{\|\mathbf{X} \mathbf{w}\|}. \quad (27)$$

1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

17 18

19 20

21 22

23 24

25 26

27 28

29 30

Example 2 - Yale database (1/5)



m p

images of 38 subjects, each under 64 different illumination conditions:



Subject 1, 64 illumination conditions

1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

17 18

19 20

21 22

23 24

25 26

27 28

29 30

Example 2 - Yale database (2/5)



m p

images of 38 subjects, each under 64 different illumination conditions:



38 subjects

1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

17 18

19 20

21 22

23 24

25 26

27 28

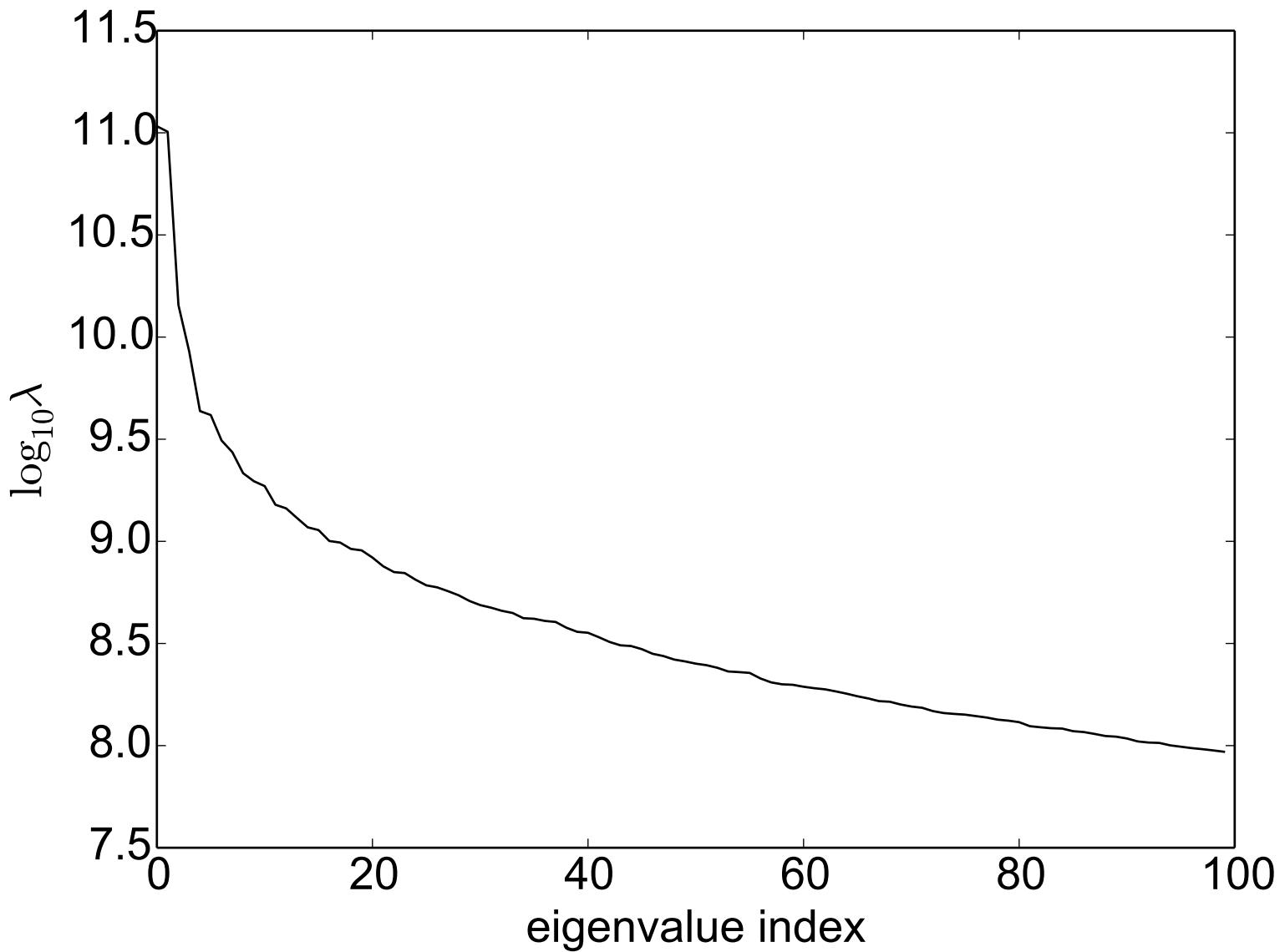
29 30

Example 2 - Yale database (3/5)



m p

images of 38 subjects, each under 64 different illumination conditions. Thus, there is $38 \times 64 = 2432$ images in total. Each of them is a feature vector with $192 \times 168 = 32256$ dimensions (pixels). PCA gives the following eigenvalues:



1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

17 18

19 20

21 22

23 24

25 26

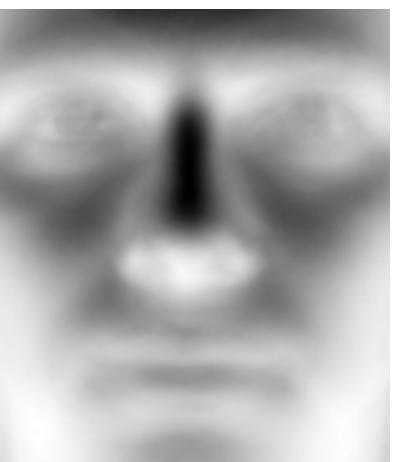
27 28

29 30

Example 2 - Yale database (4/5)



m p



mean

1st ev

2nd ev

3rd ev



first 72 eigenvectors

1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

17 18

19 20

21 22

23 24

25 26

27 28

29 30

Example 2 - Yale database (5/5)



m p

Reconstruction of original vector using eigenvectors



original



mean and 3 evs



mean and 10 evs



mean and 50 evs



mean and 100 evs



mean and 300 evs

1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

17 18

19 20

21 22

23 24

25 26

27 28

29 30

Linear Discriminant Analysis (LDA)



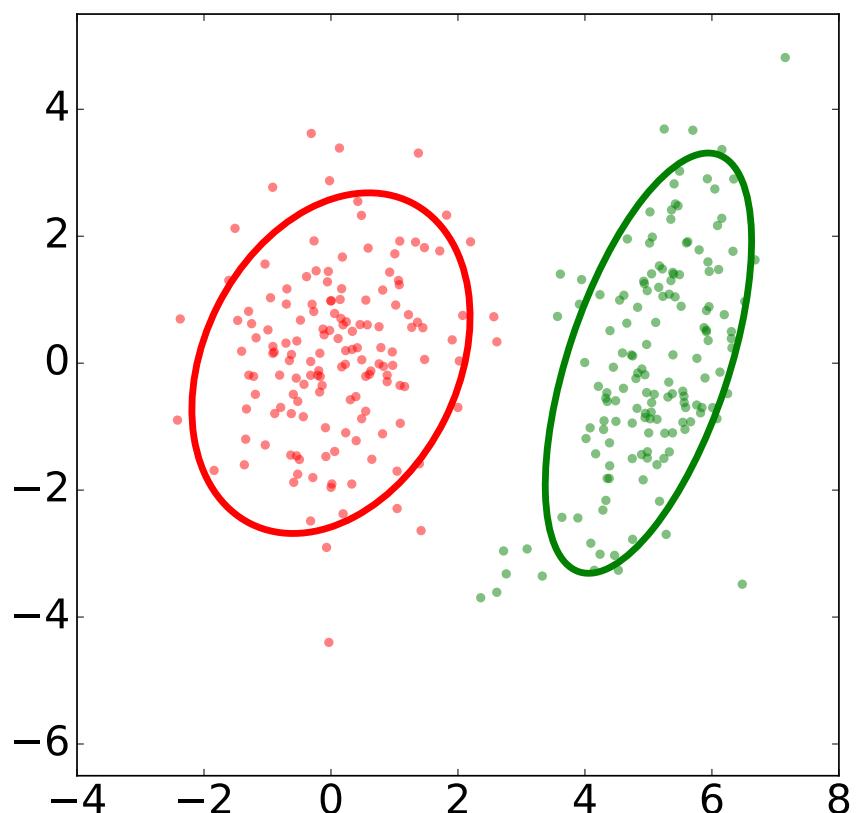
m p

Setting: Classification, training set: N_1 points (class 1) and N_2 points (class 2)

Goal: Project data to a 1D subspace such that a low-error classifier can be constructed.

Approach: Find a direction to project the data to such that the two classes are well separated in this projection.

Example: Data as shown



1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

17 18

19 20

21 22

23 24

25 26

27 28

29 30

Linear Discriminant Analysis (LDA)



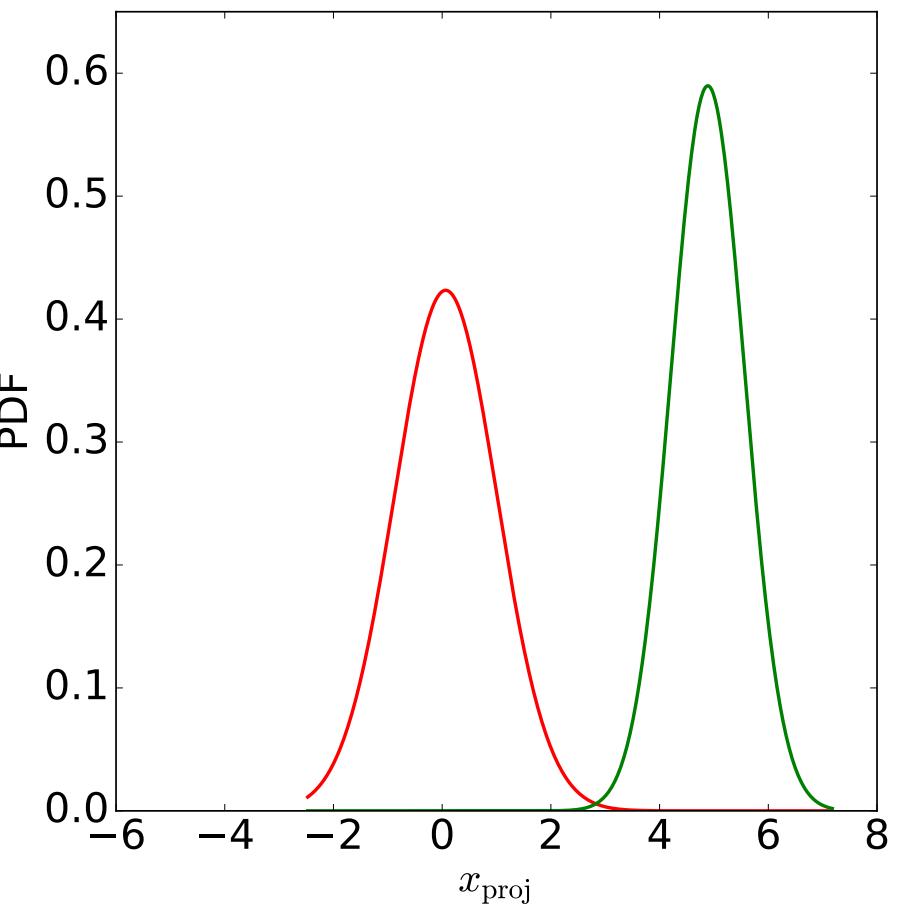
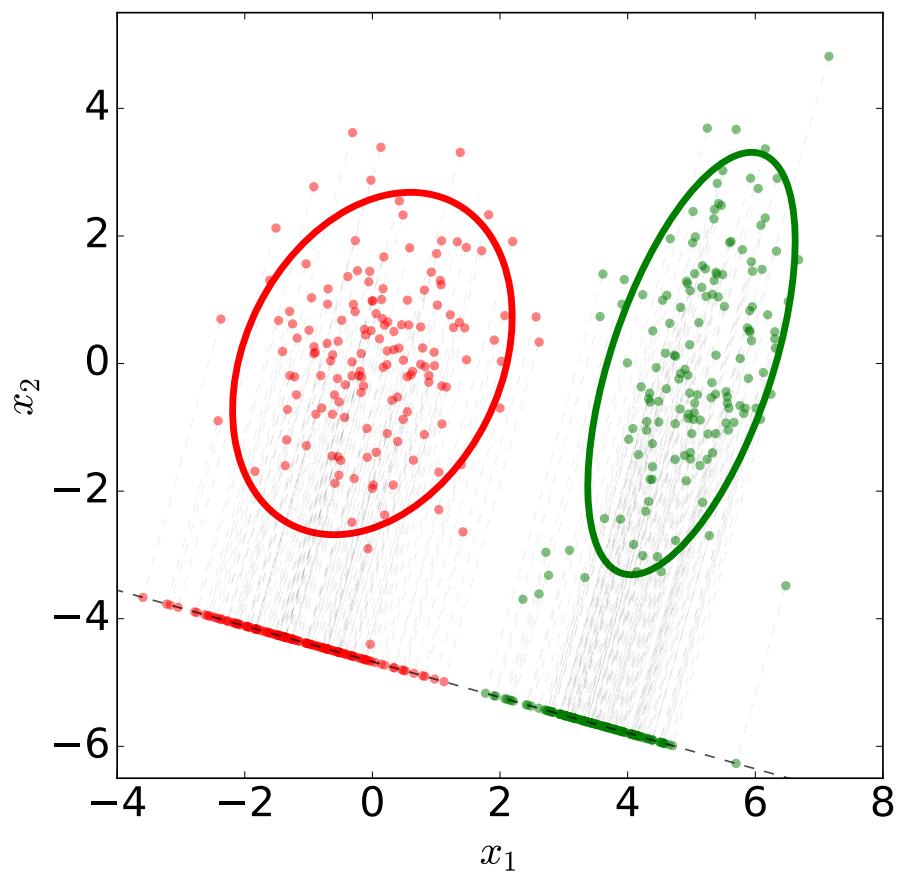
m p

Setting: Classification, training set: N_1 points (class 1) and N_2 points (class 2)

Goal: Project data to a 1D subspace such that a low-error classifier can be constructed.

Approach: Find a direction to project the data to such that the two classes are well separated in this projection.

Example: Projection direction producing good separation



1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

17 18

19 20

21 22

23 24

25 26

27 28

29 30

Linear Discriminant Analysis (LDA)



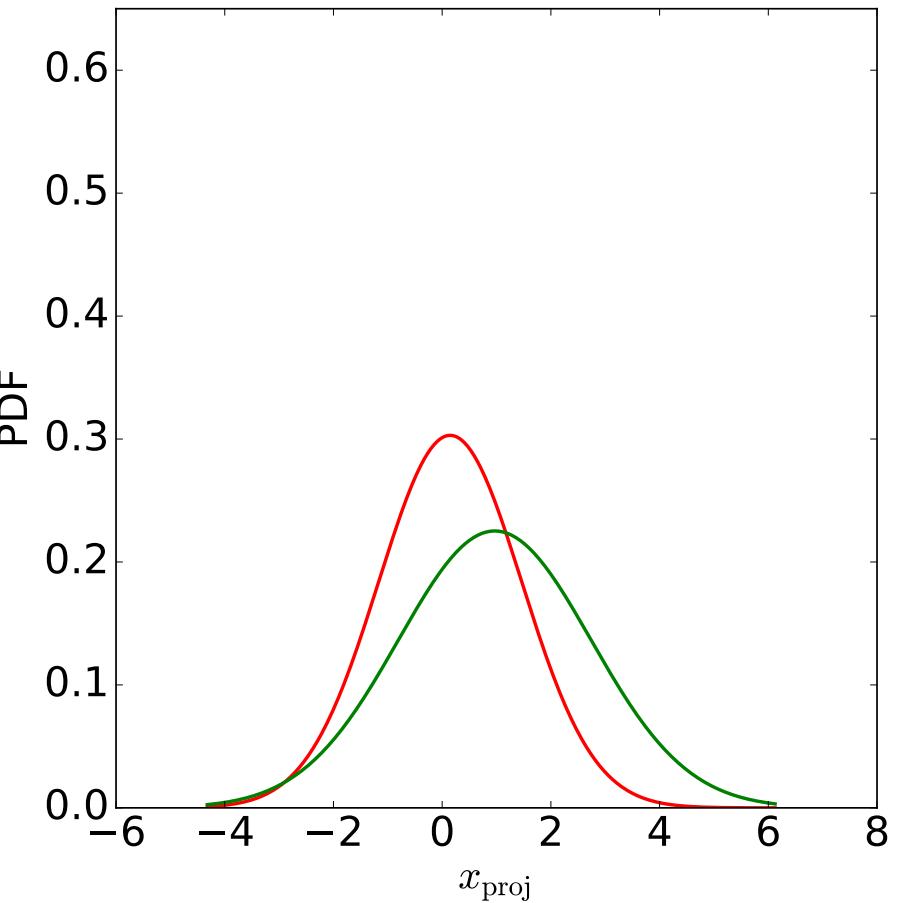
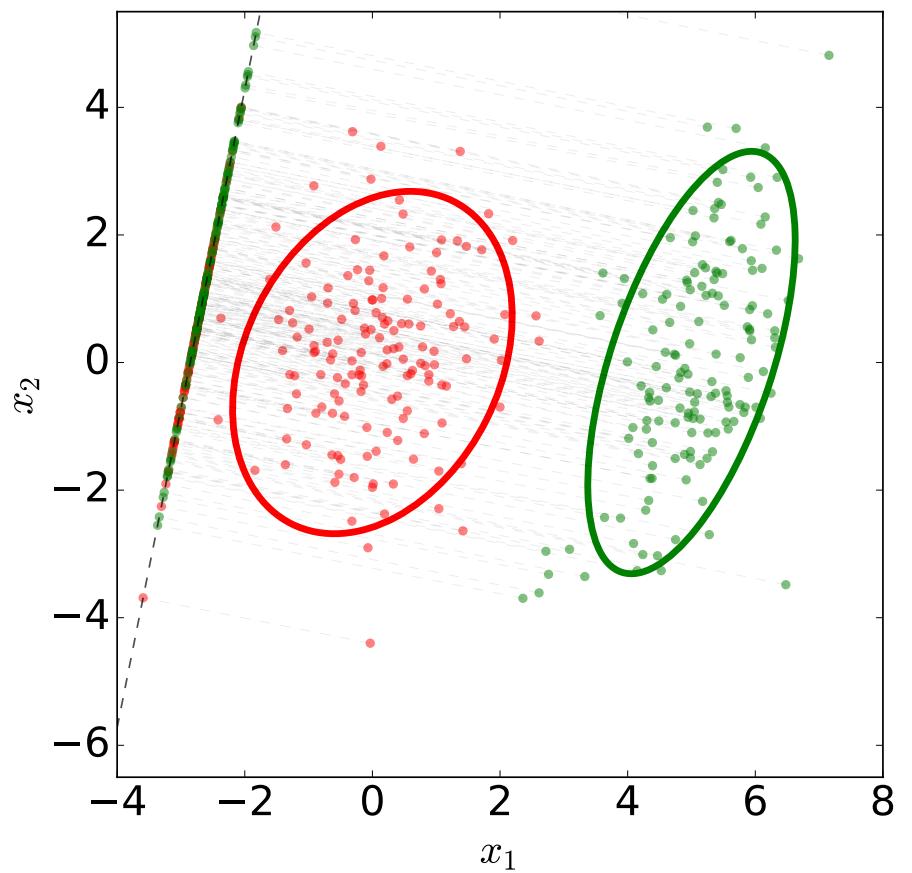
m p

Setting: Classification, training set: N_1 points (class 1) and N_2 points (class 2)

Goal: Project data to a 1D subspace such that a low-error classifier can be constructed.

Approach: Find a direction to project the data to such that the two classes are well separated in this projection.

Example: Projection direction producing bad separation



1	2
3	4
5	6
7	8
9	10
11	12
13	14
15	16
17	18
19	20
21	22
23	24
25	26
27	28
29	30

LDA: What makes a good separation?



m p

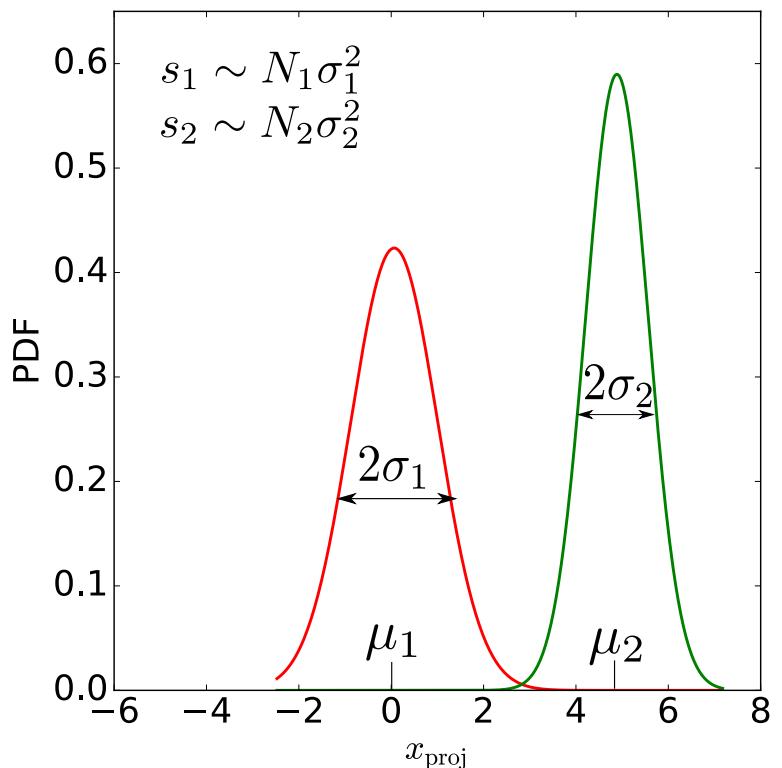
Training set: $\mathbf{x}_1^1, \dots, \mathbf{x}_{N_1}^1$ (class 1), $\mathbf{x}_1^2, \dots, \mathbf{x}_{N_2}^2$ (class 2).

Separation is higher when:

- ◆ the means of projected data are farther apart, and/or
- ◆ the scatters of the projected data are smaller.

These two observations combined suggest the following criterion to optimize:

$$\frac{(\mu_1 - \mu_2)^2}{s_1 + s_2} \rightarrow \max \quad (28)$$



$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \mathbf{v}^T \mathbf{x}_i^k \quad (k = 1, 2) \quad (29)$$

$$s_k = \sum_{i=1}^{N_k} (\mathbf{v}^T \mathbf{x}_i^k - \mu_k)^2 \quad (k = 1, 2) \quad (30)$$



$$\frac{(\mu_1 - \mu_2)^2}{s_1 + s_2} \rightarrow \max, \quad \mu_k = \mathbf{v}^T \bar{\mathbf{x}}_k, \quad s_k = \sum_{i=1}^{N_k} (\mathbf{v}^T \mathbf{x}_i^k - \mu_k)^2 \quad (k = 1, 2) \quad (31)$$

Let us rewrite the criterion in terms of unprojected entities. The nominator:

$$(\mu_1 - \mu_2)^2 = [\mathbf{v}^T (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)]^2 = \mathbf{v}^T \underbrace{(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T}_{\mathbf{S}_b} \mathbf{v} \quad (32)$$

The scatters:

$$s_1 = \sum_{i=1}^{N_1} (\mathbf{v}^T \mathbf{x}_i - \mathbf{v}^T \bar{\mathbf{x}}_1)^2 = \sum_{i=1}^{N_1} \mathbf{v}^T (\mathbf{x}_i - \bar{\mathbf{x}}_1)(\mathbf{x}_i - \bar{\mathbf{x}}_1)^T \mathbf{v} \quad (33)$$

$$= \mathbf{v}^T \underbrace{\left(\sum_{i=1}^{N_1} (\mathbf{x}_i - \bar{\mathbf{x}}_1)(\mathbf{x}_i - \bar{\mathbf{x}}_1)^T \right)}_{\mathbf{S}_1} \mathbf{v} \quad (34)$$

$$s_2 = \mathbf{v}^T \mathbf{S}_2 \mathbf{v} \quad \mathbf{S}_1, \mathbf{S}_2 : \text{scatter matrices for classes 1, 2} \quad (35)$$



$$\frac{(\mu_1 - \mu_2)^2}{s_1 + s_2} \rightarrow \max, \quad \mu_k = \mathbf{v}^T \bar{\mathbf{x}}_k, \quad s_k = \sum_{i=1}^{N_k} (\mathbf{v}^T \mathbf{x}_i^k - \mu_k)^2 \quad (k = 1, 2) \quad (36)$$

Therefore, the criterion can be rewritten as

$$\frac{(\mu_1 - \mu_2)^2}{s_1 + s_2} = \frac{\mathbf{v}^T \mathbf{S}_b \mathbf{v}}{\mathbf{v}^T (\mathbf{S}_1 + \mathbf{S}_2) \mathbf{v}} = \frac{\mathbf{v}^T \mathbf{S}_b \mathbf{v}}{\mathbf{v}^T \mathbf{S}_w \mathbf{v}}, \quad (37)$$

where everything except the to-be-found vector \mathbf{v} is computed from the training data:

$$\mathbf{S}_b : \text{between-class scatter matrix}, \quad \mathbf{S}_b = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T \quad (38)$$

$$\mathbf{S}_w : \text{within-class scatter matrix}, \quad \mathbf{S}_w = \mathbf{S}_1 + \mathbf{S}_2 \quad (39)$$

$$\mathbf{S}_k = \sum_{i=1}^{N_k} (\mathbf{x}_i^k - \bar{\mathbf{x}}_k)(\mathbf{x}_i^k - \bar{\mathbf{x}}_k)^T, \quad (k = 1, 2) \quad (40)$$

Let us now solve the maximization task:

$$\mathbf{v}_1 = \operatorname{argmax}_{\mathbf{v}} \frac{\mathbf{v}^T \mathbf{S}_b \mathbf{v}}{\mathbf{v}^T \mathbf{S}_w \mathbf{v}} \quad (41)$$

Note that there is no need to constrain \mathbf{v} to e.g. unit length, as the scaling in denominator and nominator cancels out.



$$\mathbf{v}_1 = \underset{\mathbf{v}}{\operatorname{argmax}} \frac{\mathbf{v}^T \mathbf{S}_b \mathbf{v}}{\mathbf{v}^T \mathbf{S}_w \mathbf{v}} \quad (42)$$

Note that \mathbf{S}_b is symmetric, positive semi-definite (rank 1) matrix.

Matrix \mathbf{S}_w is symmetric, positive semi-definite.

Assume that \mathbf{S}_w has full rank, thus \mathbf{S}_w^{-1} exists. Let $\mathbf{S}_w^{\frac{1}{2}}$ be the symmetric, positive-definite matrix such that $\mathbf{S}_w = \mathbf{S}_w^{\frac{1}{2}} \mathbf{S}_w^{\frac{1}{2}}$. Let its inverse be denoted $\mathbf{S}_w^{-\frac{1}{2}}$. Define a substitution

$$\mathbf{z} = \mathbf{S}_w^{\frac{1}{2}} \mathbf{v}. \quad (43)$$

Using the variable \mathbf{z} , the criterion becomes

$$\frac{\mathbf{v}^T \mathbf{S}_b \mathbf{v}}{\mathbf{v}^T \mathbf{S}_w \mathbf{v}} = \frac{\mathbf{z}^T \mathbf{S}_w^{-\frac{1}{2}} \mathbf{S}_b \mathbf{S}_w^{-\frac{1}{2}} \mathbf{z}}{\mathbf{z}^T \mathbf{z}} \quad (44)$$

Let us fix the length of \mathbf{z} to 1 ($\mathbf{z}^T \mathbf{z} = 1$). The denominator is then a constant, and the criterion is maximized when the numerator is maximized. The latter achieves maximum for the largest eigenvalue λ_1 of matrix $\mathbf{S}_w^{-\frac{1}{2}} \mathbf{S}_b \mathbf{S}_w^{-\frac{1}{2}}$ and the corresponding eigenvector \mathbf{z}_1 :

$$\mathbf{S}_w^{-\frac{1}{2}} \mathbf{S}_b \mathbf{S}_w^{-\frac{1}{2}} \mathbf{z}_1 = \lambda_1 \mathbf{z}_1 \quad (45)$$

Symmetric, positive definite \mathbf{S} :

$$\mathbf{S} = \mathbf{U} \operatorname{diag}[\lambda_1, \dots, \lambda_D] \mathbf{U}^T$$

\mathbf{U} : orthogonal, unit columns

$$\mathbf{S}^{\frac{1}{2}} = \mathbf{U} \operatorname{diag}[\sqrt{\lambda_1}, \dots, \sqrt{\lambda_D}] \mathbf{U}^T$$

$$\mathbf{S}^{-\frac{1}{2}} = \mathbf{U} \operatorname{diag}\left[\frac{1}{\sqrt{\lambda_1}}, \dots, \frac{1}{\sqrt{\lambda_D}}\right] \mathbf{U}^T$$

$$\mathbf{S}^{-1} = \mathbf{U} \operatorname{diag}\left[\frac{1}{\lambda_1}, \dots, \frac{1}{\lambda_D}\right] \mathbf{U}^T$$



(copied from previous slide:)

$$\mathbf{S}_w^{-\frac{1}{2}} \mathbf{S}_b \mathbf{S}_w^{-\frac{1}{2}} \mathbf{z}_1 = \lambda_1 \mathbf{z}_1 \quad (46)$$

Taking this \mathbf{z}_1 , and substituting back, gives the solution $\mathbf{v}_1 = \mathbf{S}_w^{-\frac{1}{2}} \mathbf{z}_1$. But left-multiplying the previous equation by $\mathbf{S}_w^{-\frac{1}{2}}$, we see that

$$\mathbf{S}_w^{-1} \mathbf{S}_b (\mathbf{S}_w^{-\frac{1}{2}} \mathbf{z}_1) = \lambda_1 (\mathbf{S}_w^{-\frac{1}{2}} \mathbf{z}_1), \Rightarrow \mathbf{S}_w^{-1} \mathbf{S}_b \mathbf{v}_1 = \lambda_1 \mathbf{v}_1. \quad (47)$$

Thus \mathbf{v}_1 can be computed directly as the eigenvector of $\mathbf{S}_w^{-1} \mathbf{S}_b$ corresponding to the highest eigenvalue, λ_1 (note that $\mathbf{S}_w^{-1} \mathbf{S}_b$ and $\mathbf{S}_w^{-\frac{1}{2}} \mathbf{S}_b \mathbf{S}_w^{-\frac{1}{2}}$ share the eigenvalues).

Moreover, \mathbf{S}_b has rank 1. There holds $\mathbf{S}_b = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T$, and

$$\mathbf{S}_w^{-1} \mathbf{S}_b \mathbf{z} = \mathbf{S}_w^{-1} (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2) \underbrace{(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T \mathbf{z}}_{\text{a scalar}}, \quad (48)$$

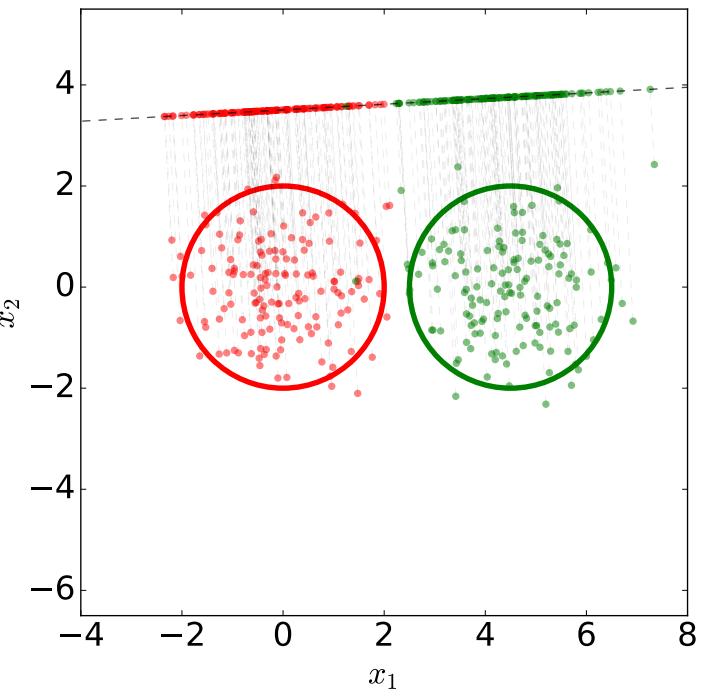
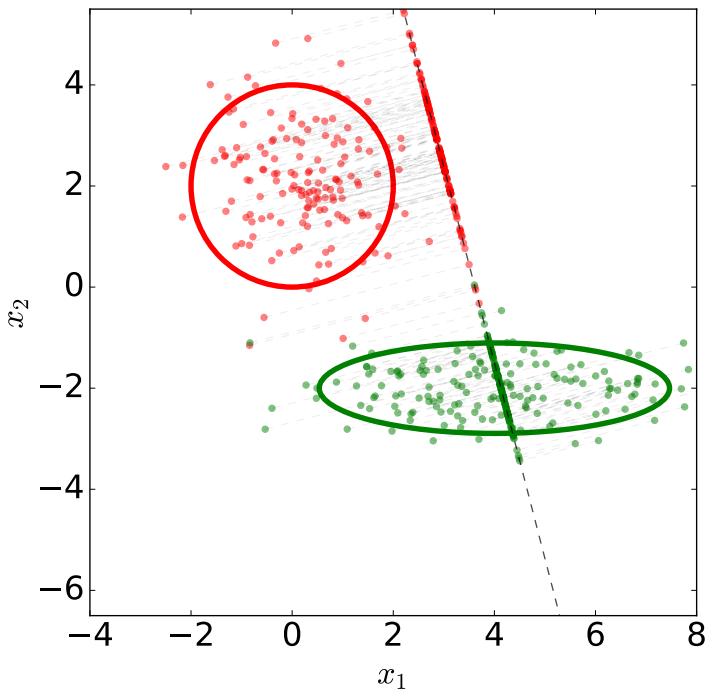
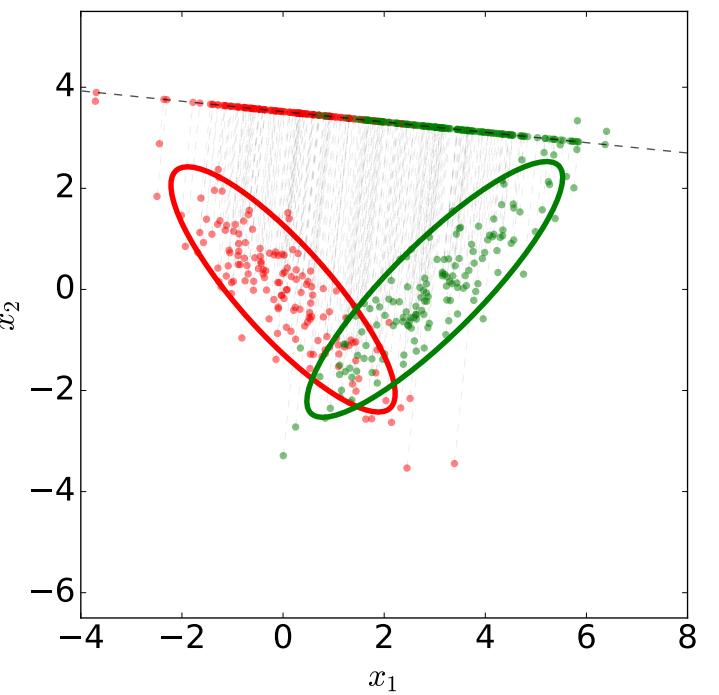
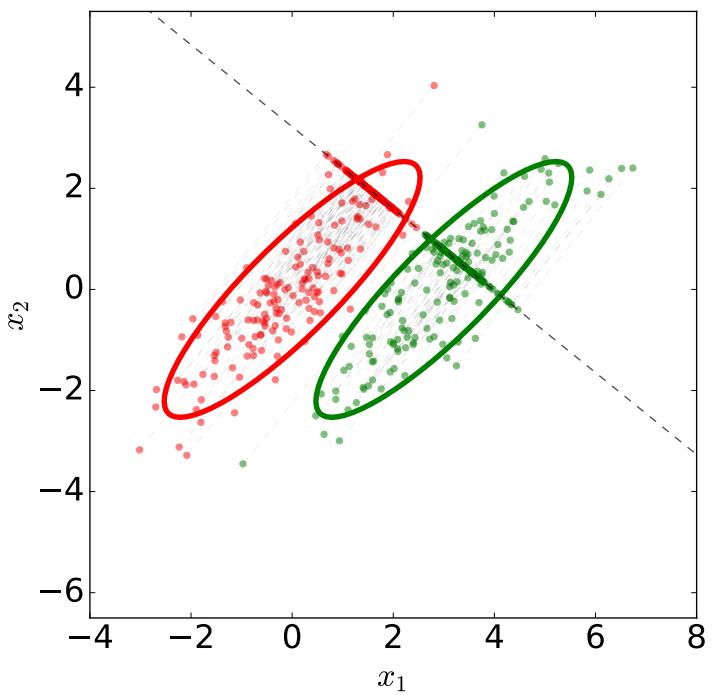
thus the dominant eigenvector (the only one with non-zero eigenvalue) must be

$$\mathbf{v}_1 = \mathbf{S}_w^{-1} (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2). \quad (49)$$

LDA: Examples (1)



m p

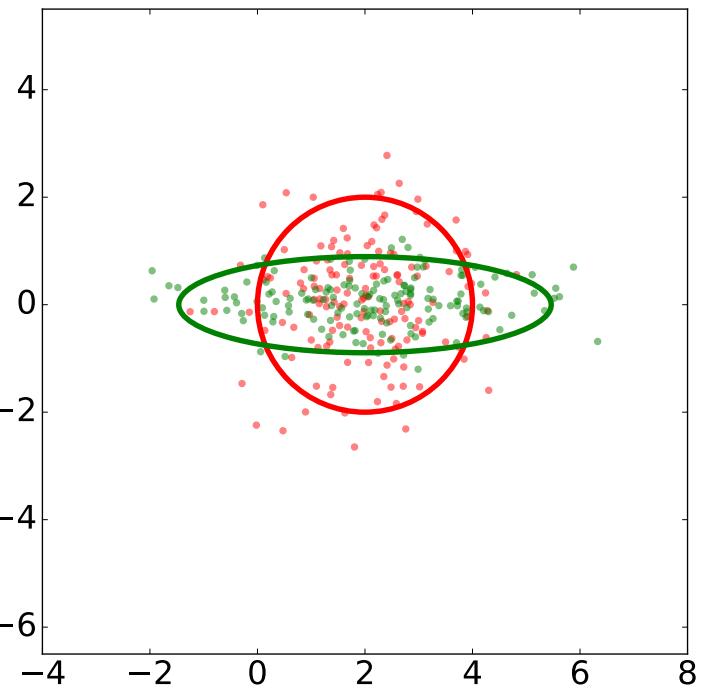
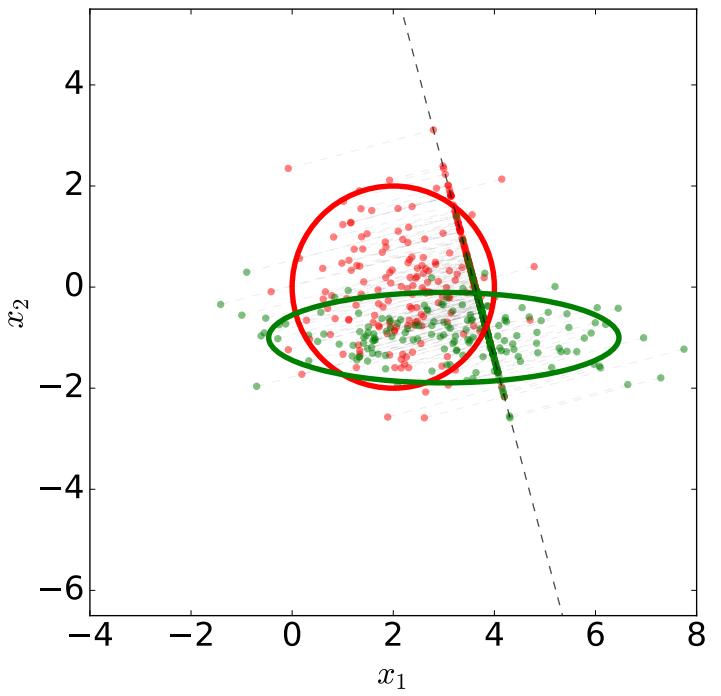
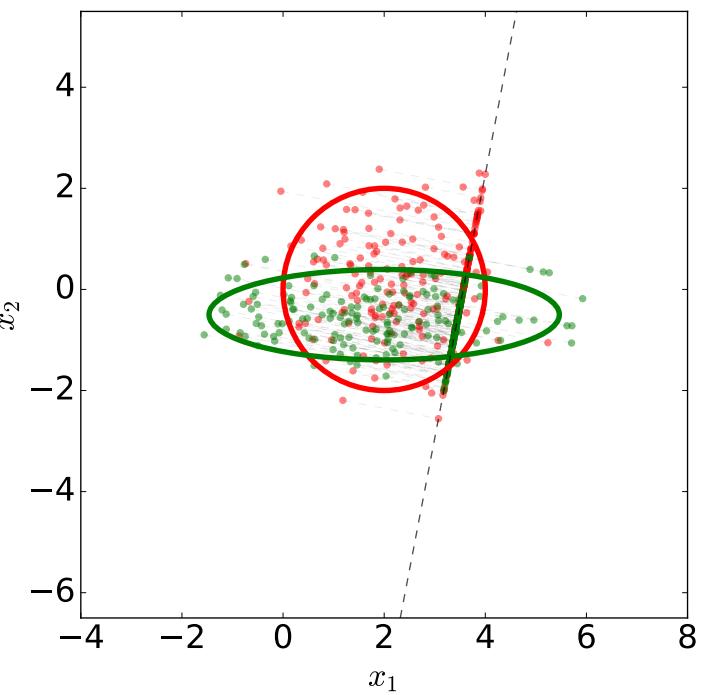
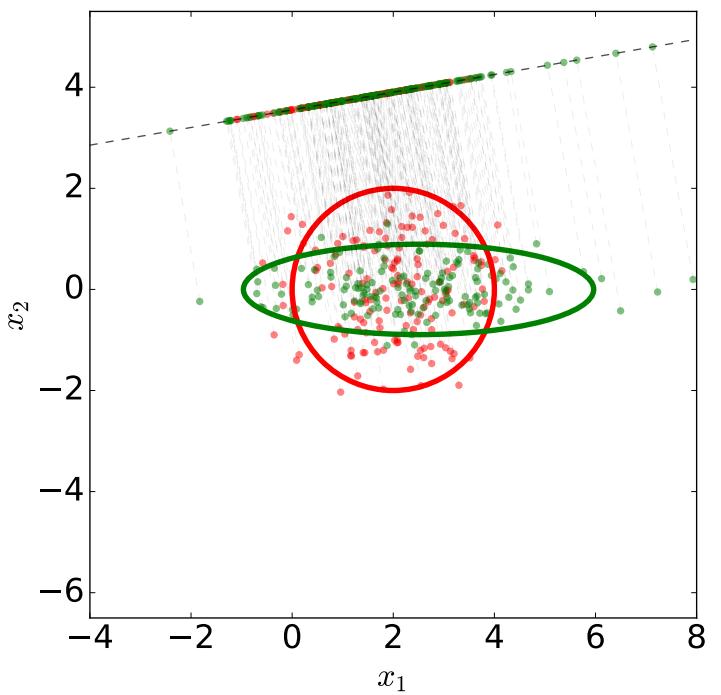


1	2
3	4
5	6
7	8
9	10
11	12
13	14
15	16
17	18
19	20
21	22
23	24
25	26
27	28
29	30

LDA: Examples (2)



m p



1	2
3	4
5	6
7	8
9	10
11	12
13	14
15	16
17	18
19	20
21	22
23	24
25	26
27	28
29	30



Consider the case that the data points \mathbf{x} 's are transformed by a non-singular linear transformation \mathbf{A} . The entities appearing in formulation and solution of LDA are then transformed as follows:

	points	scatter matrix	inv. scatter m.
original	\mathbf{x}	\mathbf{S}	\mathbf{S}^{-1}
transformed	\mathbf{Ax}	\mathbf{ASA}^T	$\mathbf{A}^{-T}\mathbf{S}^{-1}\mathbf{A}^{-1}$

Thus, $\mathbf{v}_1 = \mathbf{S}_w^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$ transforms to

$$\mathbf{v}'_1 = \mathbf{A}^{-T}\mathbf{S}_w^{-1}\mathbf{A}^{-1}\mathbf{A}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2) = \mathbf{A}^{-T}\mathbf{S}_w^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2). \quad (50)$$

The original projected coordinates are

$$\mathbf{v}_1^T \mathbf{x} = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T \mathbf{S}_w^{-1} \mathbf{x}, \quad (51)$$

and do not change under \mathbf{A} , as

$$\mathbf{v}'_1^T \mathbf{x}' = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T \mathbf{S}_w^{-1} \mathbf{A}^{-1} \mathbf{A} \mathbf{x} = (\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)^T \mathbf{S}_w^{-1} \mathbf{x} = \mathbf{v}_1^T \mathbf{x}. \quad (52)$$

1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

17 18

19 20

21 22

23 24

25 26

27 28

29 30

Multiple Discriminant Analysis (MDA)



m p

Generalization of LDA to multiple classes K

Define:

$$\mathbf{S}_w = \sum_{i=1}^K \mathbf{S}_i \quad (\text{sum of class scatters}) \quad (53)$$

$$\mathbf{S}_b = \sum_{i=1}^K N_i (\bar{\mathbf{x}}_i - \bar{\mathbf{x}})(\bar{\mathbf{x}}_i - \bar{\mathbf{x}})^T \quad (54)$$

$$\bar{\mathbf{x}}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \mathbf{x}_i^k \quad (\text{mean of class } k \text{ data}) \quad (55)$$

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \quad (\text{mean of all data}) \quad (56)$$

Goal: find matrix \mathbf{V} stacking $L < K$ vectors such that

$$\frac{\det(\mathbf{V}^T \mathbf{S}_b \mathbf{V})}{\det(\mathbf{V}^T \mathbf{S}_w \mathbf{V})} \rightarrow \max \quad (57)$$



Solution: L most significant eigenvectors for the generalized eigenvalue problem:

$$\mathbf{S}_b \mathbf{v} = \lambda \mathbf{S}_w \mathbf{v} \quad (58)$$

Note: \mathbf{S}_b can have rank at most $K - 1$, thus at most $K - 1$ projection directions will be produced.

Employing MDA:

Useful e.g. when the number of classes K and/or number of data is very high and thus the only information about data which can be used is stored in means and scatters of classes. These are computed in incremental fashion.

1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

17 18

19 20

21 22

23 24

25 26

27 28

29 30

Decision Trees

Lecturer:
Jiří Matas

Authors:
Jiří Matas, Ondřej Drbohlav

Centre for Machine Perception
Czech Technical University, Prague
<http://cmp.felk.cvut.cz>

9.1.2017



Pros and Cons of Decision Trees

Pros

1. fast (in decision time)
2. sublinear in the number of classes K
3. intuitive, interpretable
4. no restrictions on the feature space:

$$X_1 \times X_2 \times \dots \times X_D, \quad (D \gg 1) \quad (1)$$

X_i can be discrete, continuous, categorical, ordinal, . . .

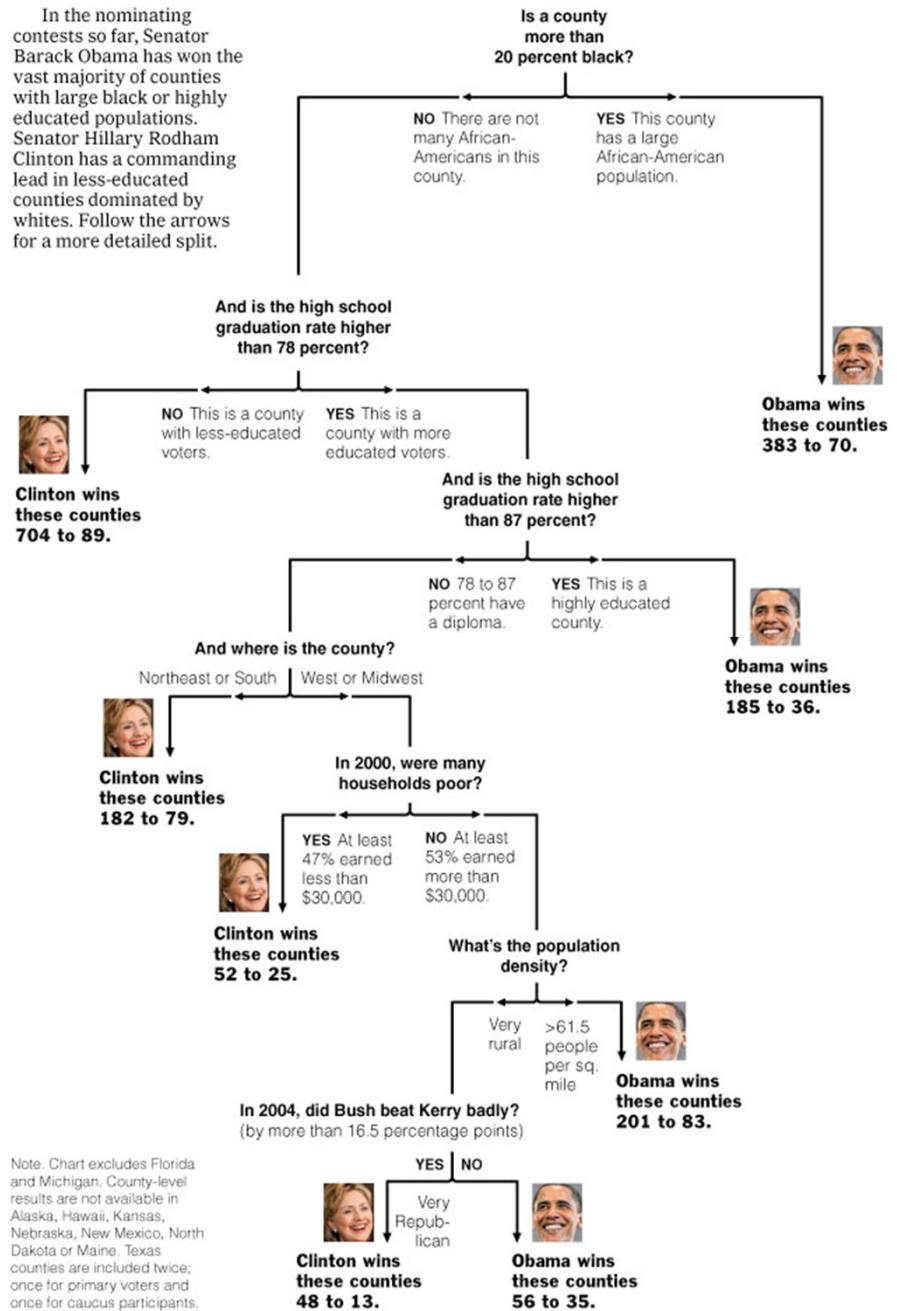
5. to classify, not all X_i may be actually needed
6. sequential decisions
7. high degree of robustness to outliers (unlike e.g. linear classifiers)
8. different scales between features are not a problem
9. deals with additional information
10. classified samples can be kept at tree nodes/leaves; this enables to compute class probabilities (therefore decision error), and use statistics of the samples for further analysis
11. can be used in regression problems

Cons

1. not well justified learning algorithm
2. overfitting is a problem for standard methods, but this can be fixed (randomized d. trees, decision forests)

Decision Tree: The Obama-Clinton Divide

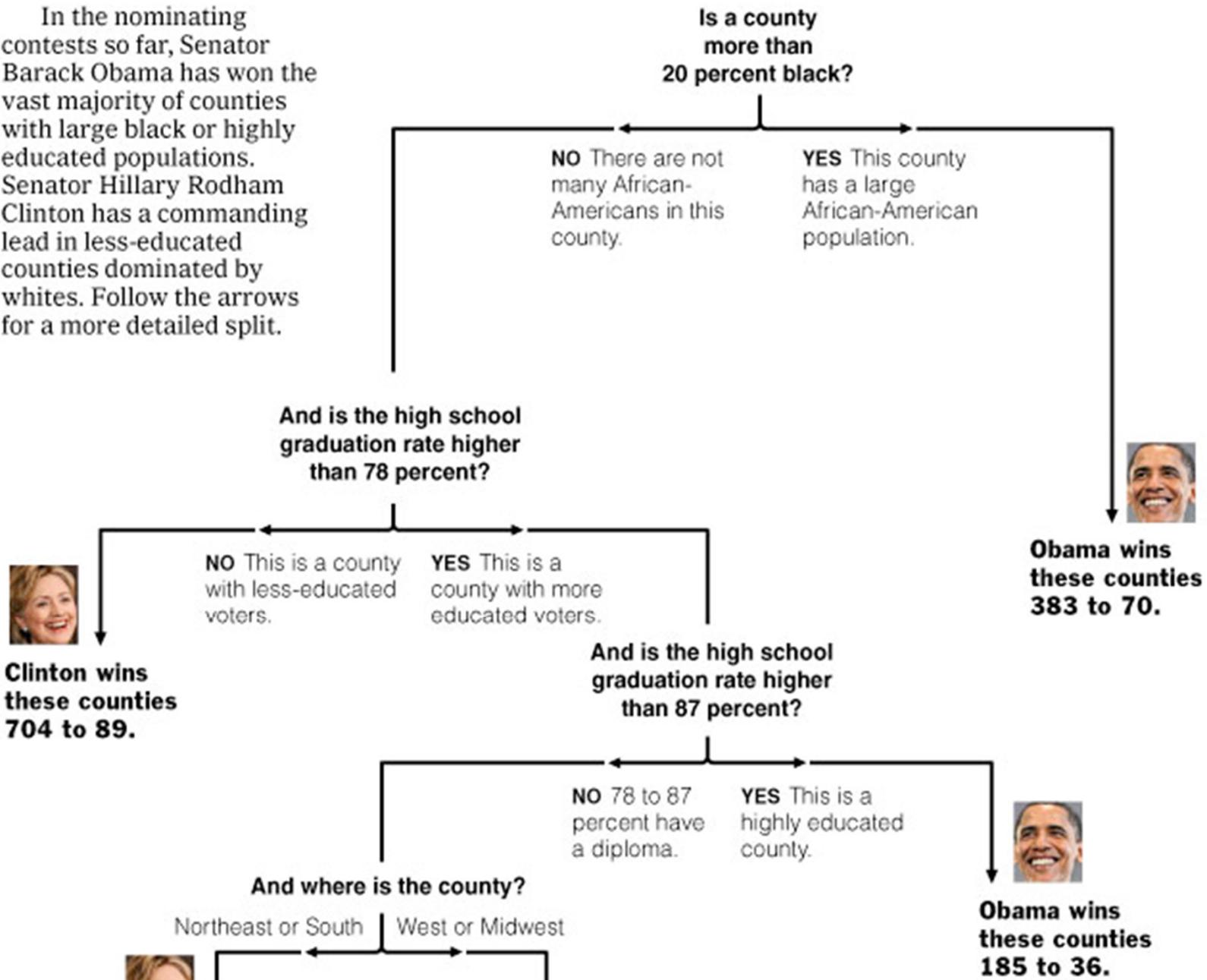
In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.

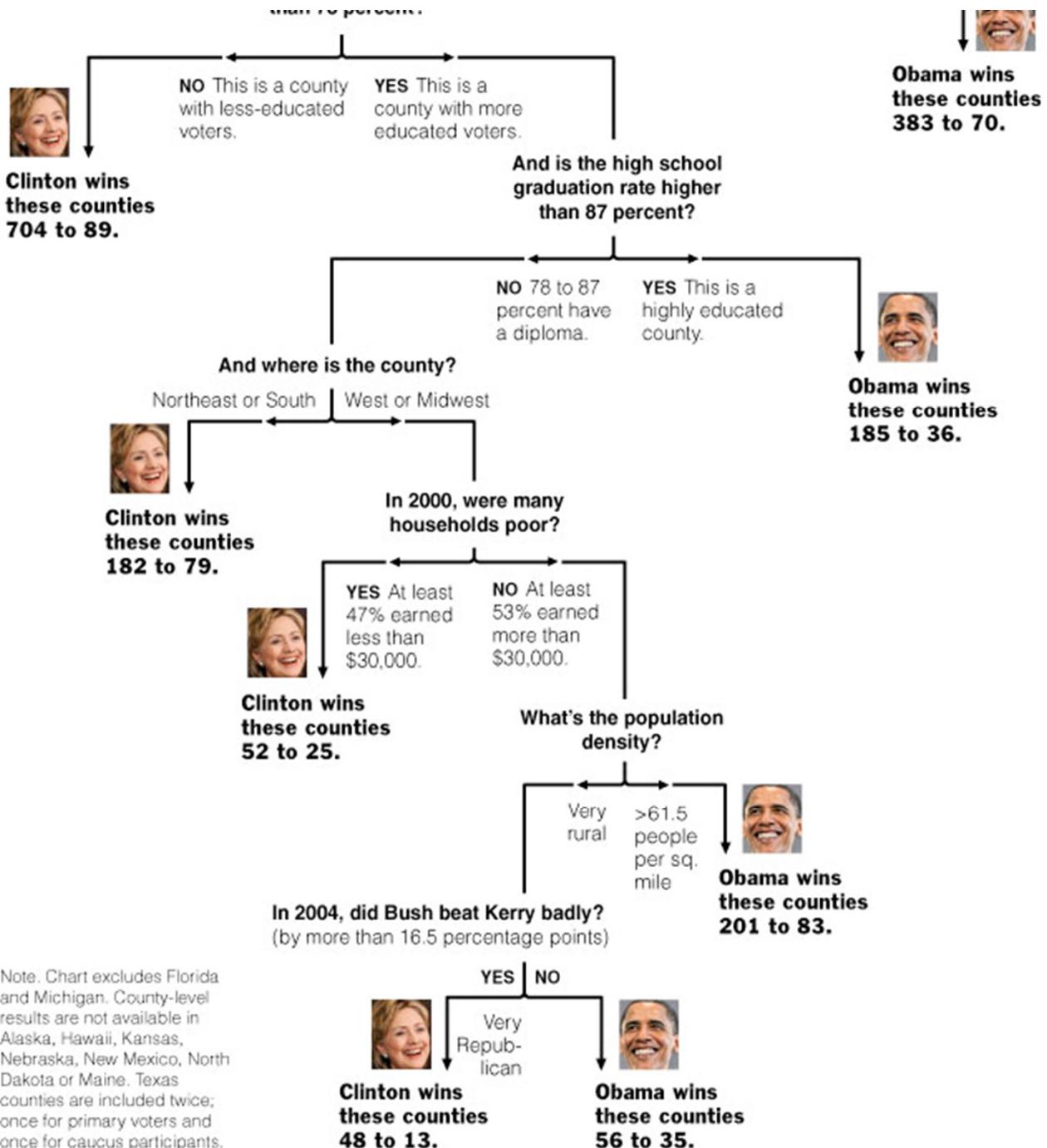


Note: Chart excludes Florida and Michigan. County-level results are not available in Alaska, Hawaii, Kansas, Nebraska, New Mexico, North Dakota or Maine. Texas counties are included twice; once for primary voters and once for caucus participants.

Decision Tree: The Obama-Clinton Divide

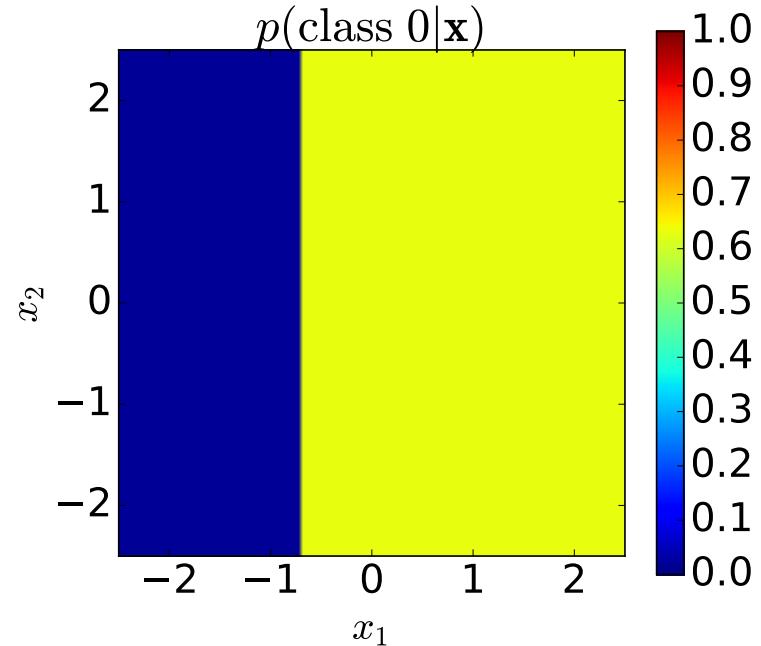
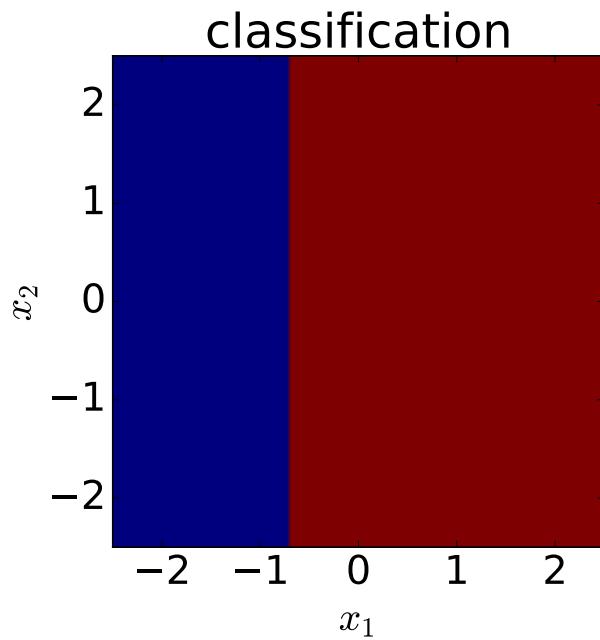
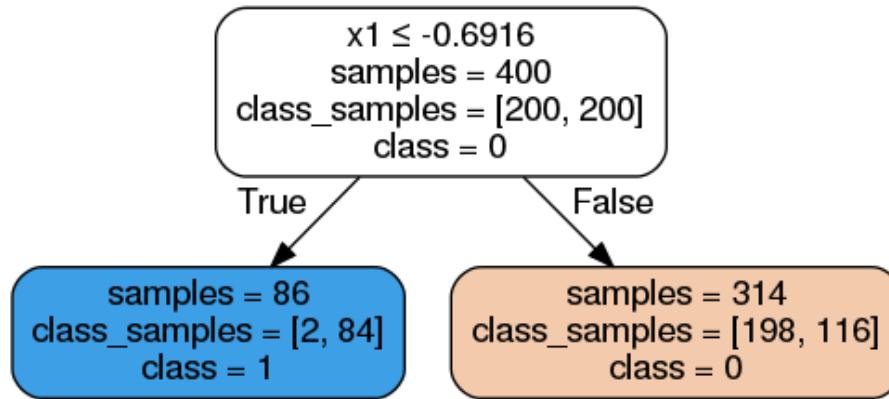
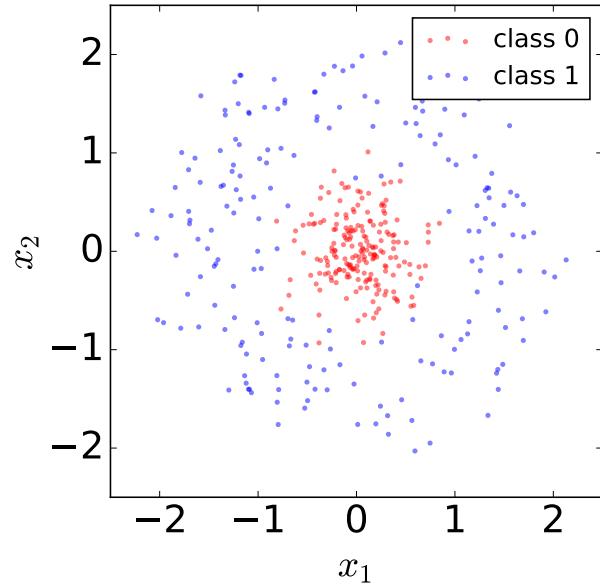
In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.



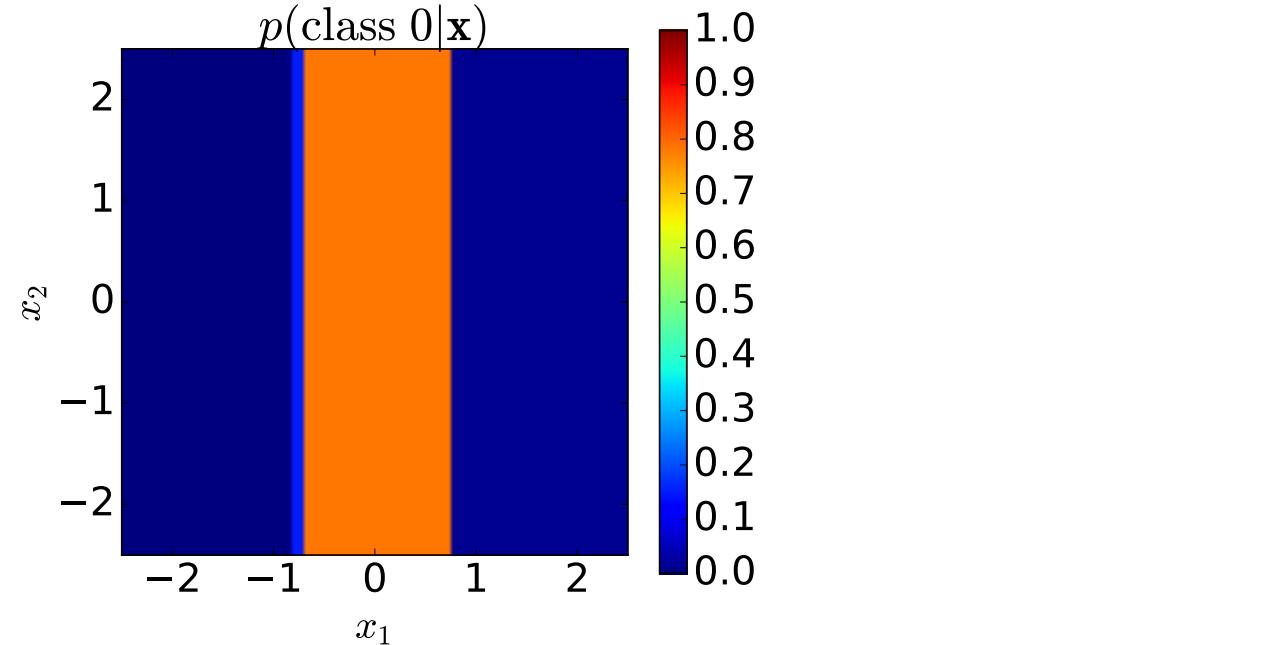
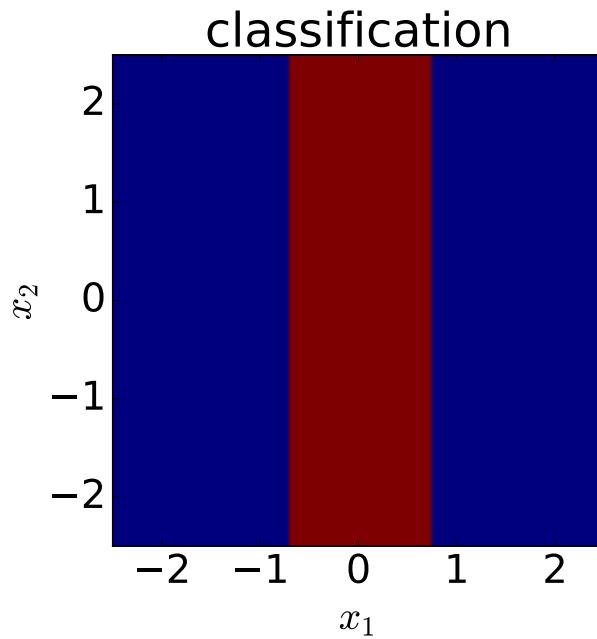
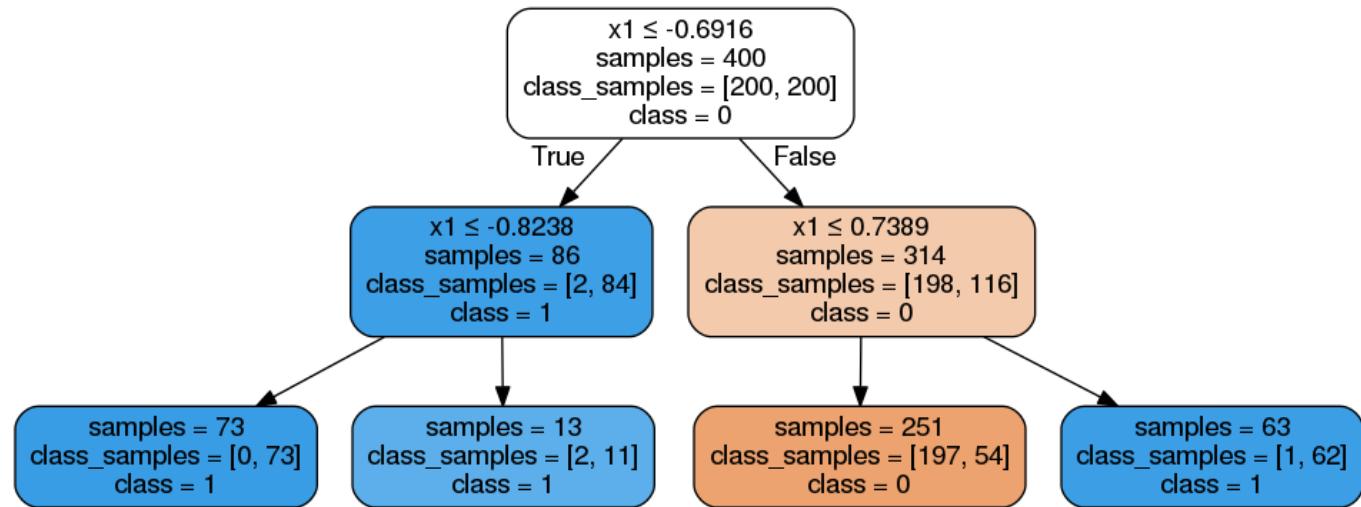
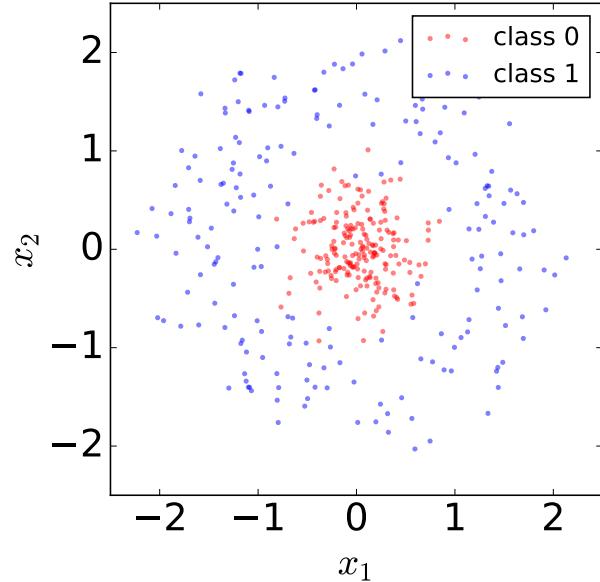


Note. Chart excludes Florida and Michigan. County-level results are not available in Alaska, Hawaii, Kansas, Nebraska, New Mexico, North Dakota or Maine. Texas counties are included twice; once for primary voters and once for caucus participants.

Decision tree: Example, 2D data



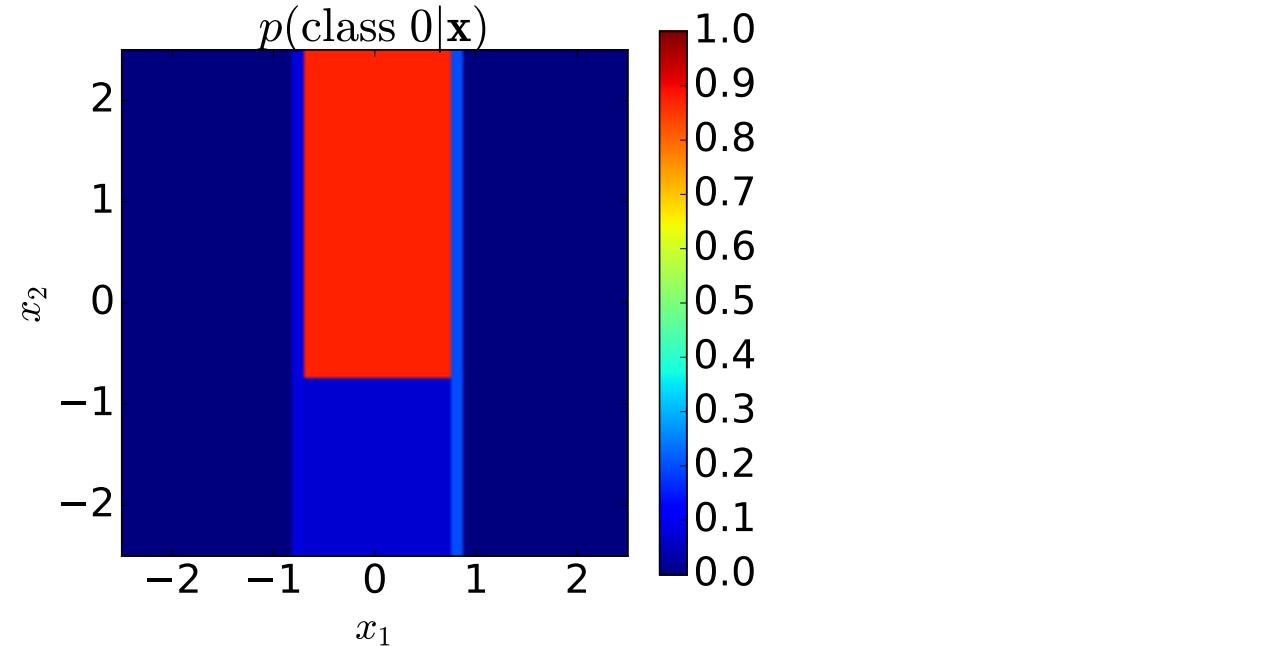
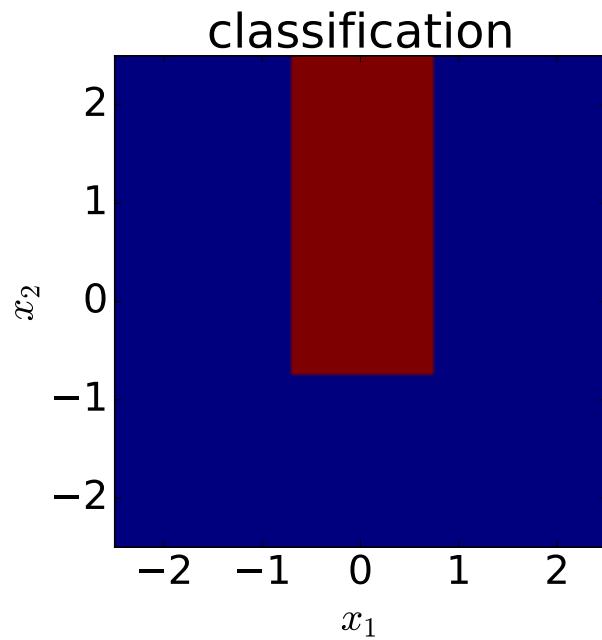
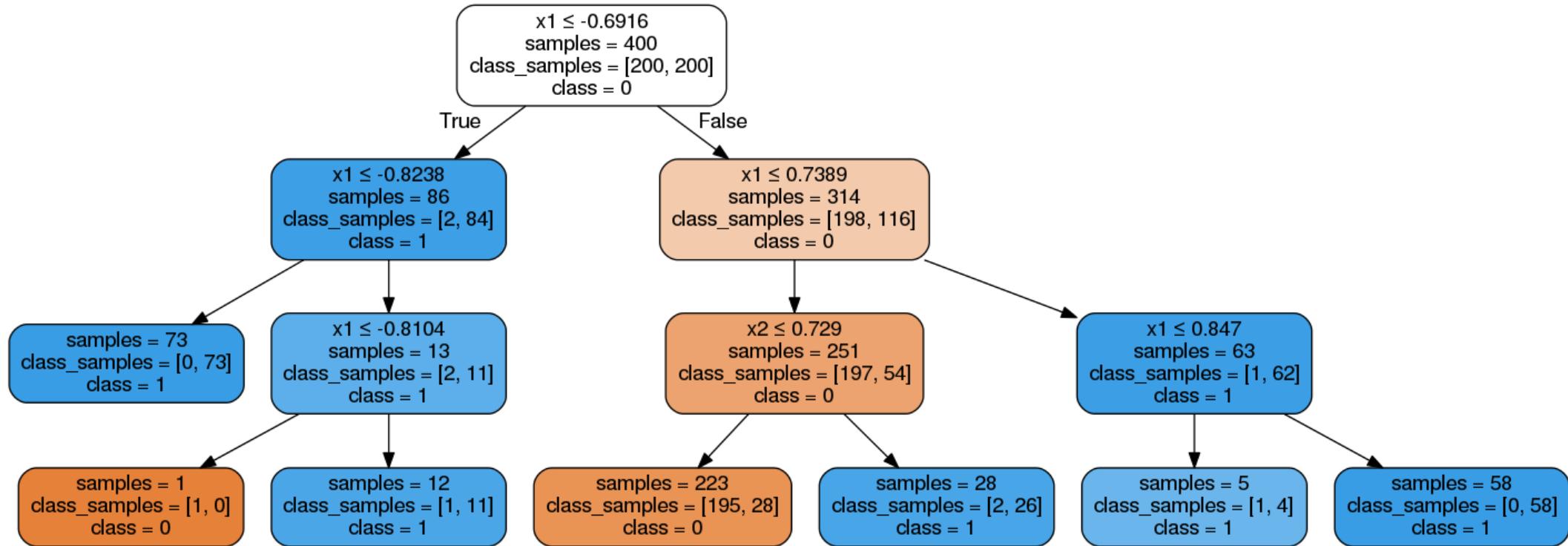
Decision tree: Example, 2D data

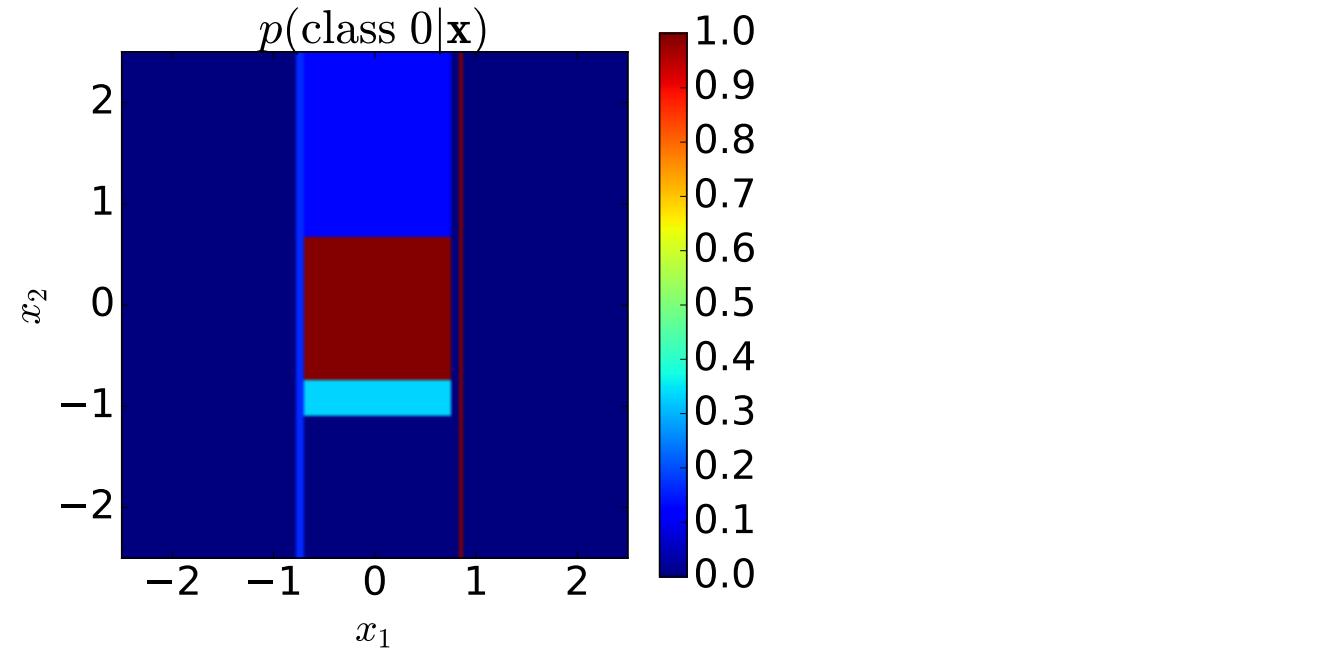
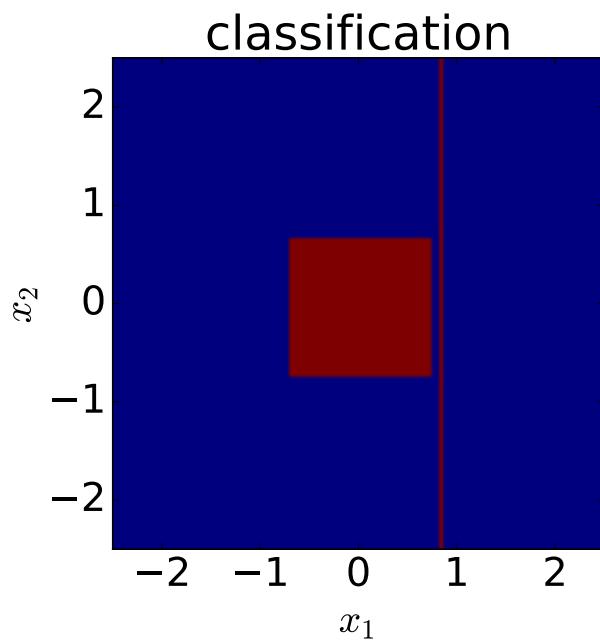
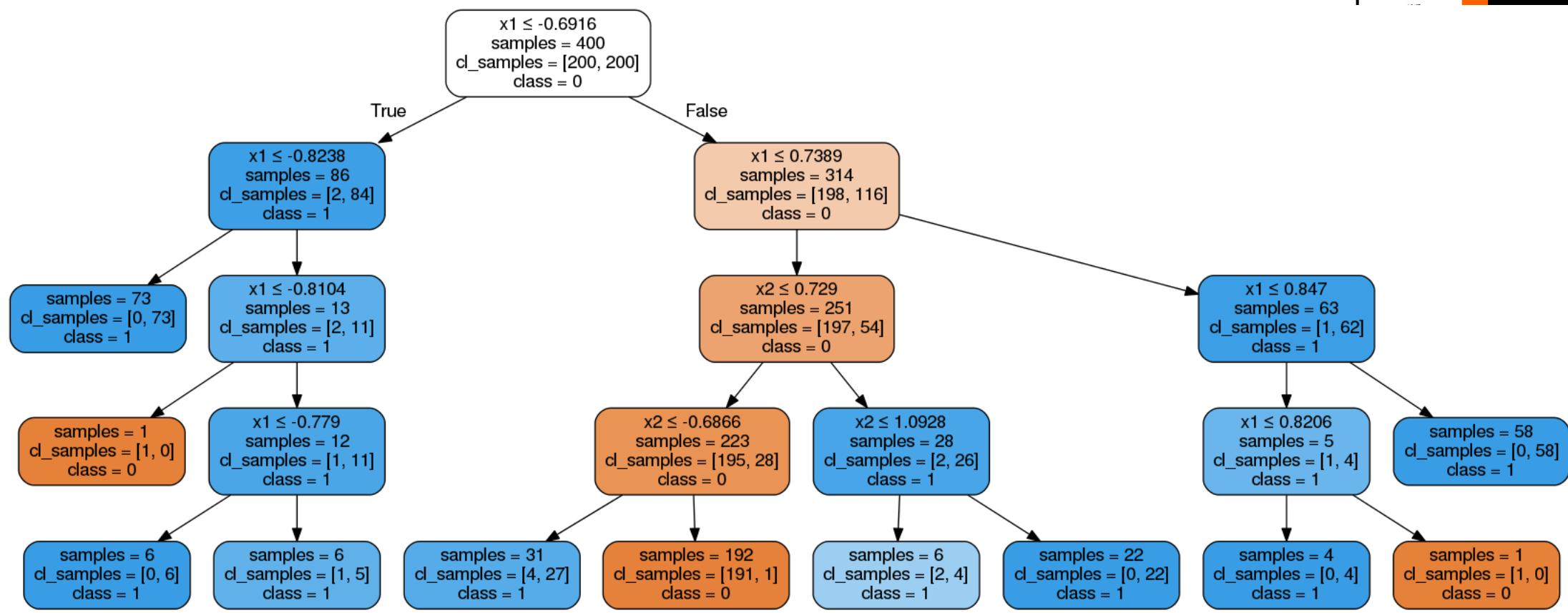




m p

Decision tree: Example, 2D data





Learning decision trees: An example

Problem: decide whether to wait for a table at a restaurant.

Attributes used

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range (\$, \$\$, \$\$\$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Goal predicate: WillWait?

Attribute-based representations

Examples described by **attribute values** (Boolean, discrete, continuous)

Example	Attributes										Target Wait
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30–60	T

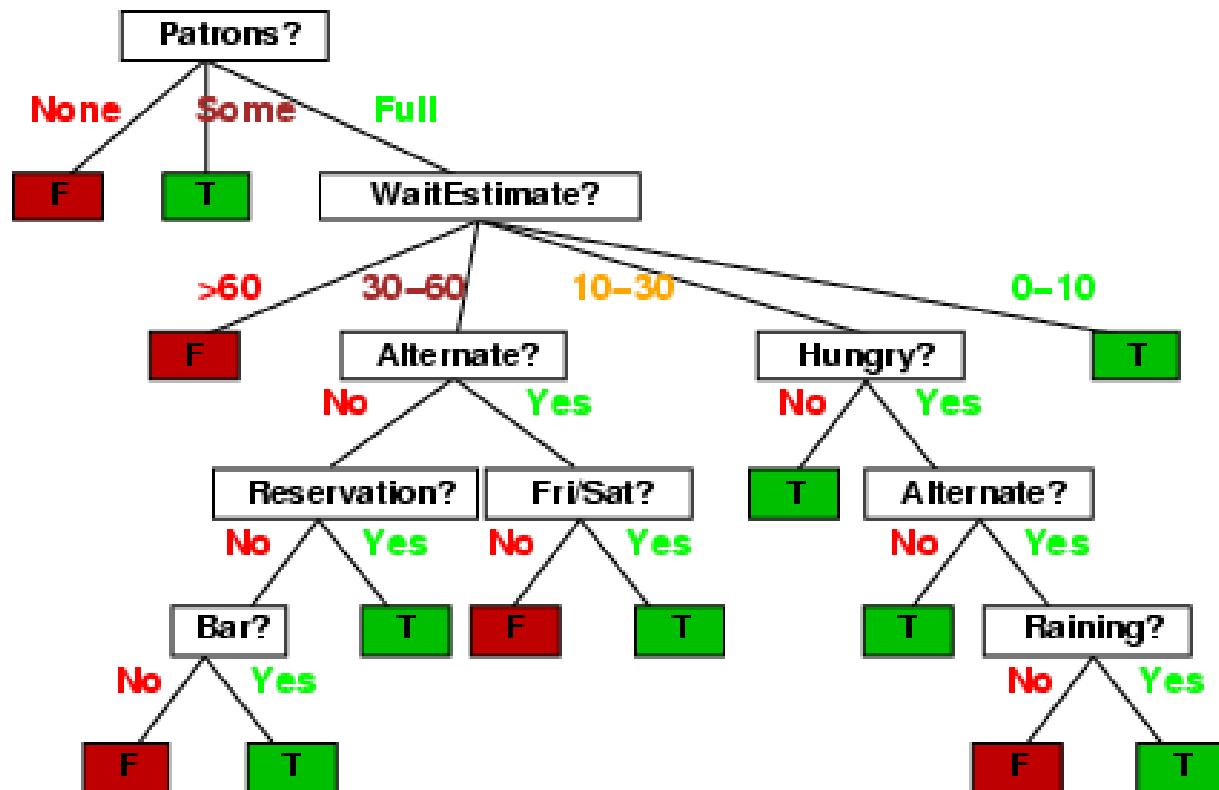
12 examples
(6 T, 6 F)

Classification of examples is **positive** (T) or **negative** (F)

Decision trees

One possible representation for hypotheses

E.g., here is a tree for deciding whether to wait:



Top-Down Induction of DT (TDIDT)

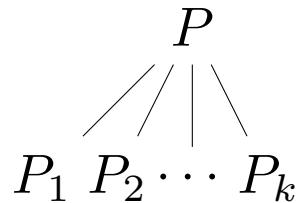
Growth Phase: The tree is constructed top-down.

- Find the “best” attribute.
- Partition examples based on the attribute’s values.
- Apply recursively to each partition.

Pruning Phase: The tree is pruned bottom-up

- For each node, keep subtree or change to leaf.
- Choose by comparing estimated error.

Splitting a Node to Branches



- ◆ Define a splitting function $S(P, \theta)$ as a mapping which takes a set P (data points at a node) and splitting parameters θ and produces the partition of P to $\{P_1, P_2, \dots, P_k\}$. That is,

$$S : P, \theta \rightarrow \{P_1, P_2, \dots, P_k\}, \quad (2)$$

$$P_i \cap P_j = \emptyset \Leftrightarrow i \neq j \quad (3)$$

$$\bigcup_{i=1}^k P_i = P \quad (4)$$

- ◆ θ : may contain the branching factor k , the index i of the splitting dimension X_i
- ◆ $k = 2$: binary split
- ◆ $k > 2$: multiway split
- ◆ When X_i is continuous, θ contains the range(s) of values for individual partitions (when $k = 2$, typically a threshold. Example: [height $\leq 1.75m$?])

How to Split: Greedy Maximization of Information Gain

Information Gain (IG) measures the expected reduction in entropy when a set P is partitioned to subsets P_1, P_2, \dots, P_k . The information gain $IG(A)$ (A is the attribute used for splitting P) is

$$IG(A) = H(P) - \sum_{i=1}^k \frac{|P_i|}{|P|} H(P_i) \quad (5)$$

$$H(P) = - \sum_j \frac{|P^{(j)}|}{|P|} \log \frac{|P^{(j)}|}{|P|} \quad (\text{entropy of } P) \quad (6)$$

$(P^{(j)} : \text{number of points of class } j \text{ in } P)$

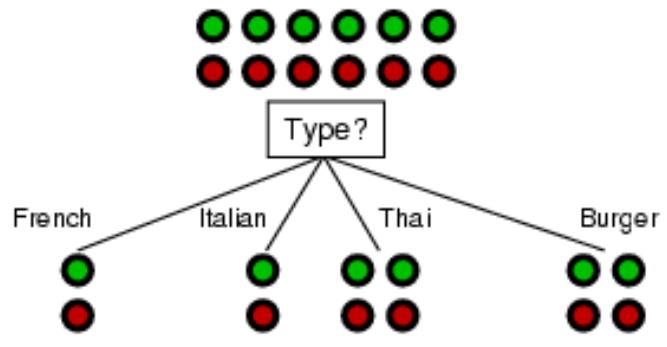
$$H(P_i) = - \sum_j \frac{|P_i^{(j)}|}{|P_i|} \log \frac{|P_i^{(j)}|}{|P_i|} \quad (\text{entropy of } P_i) \quad (7)$$

$(P_i^{(j)} : \text{number of points of class } j \text{ in } P_i)$

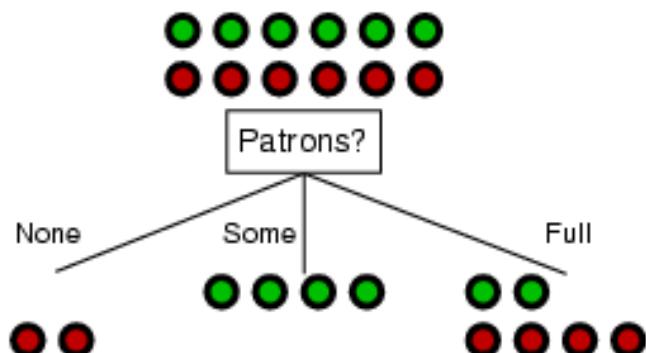
Information gain

For the training set, $p = n = 6$, entropy $I(6/12, 6/12) = 1$ bit

Consider the attributes *Type* and *Patrons*:



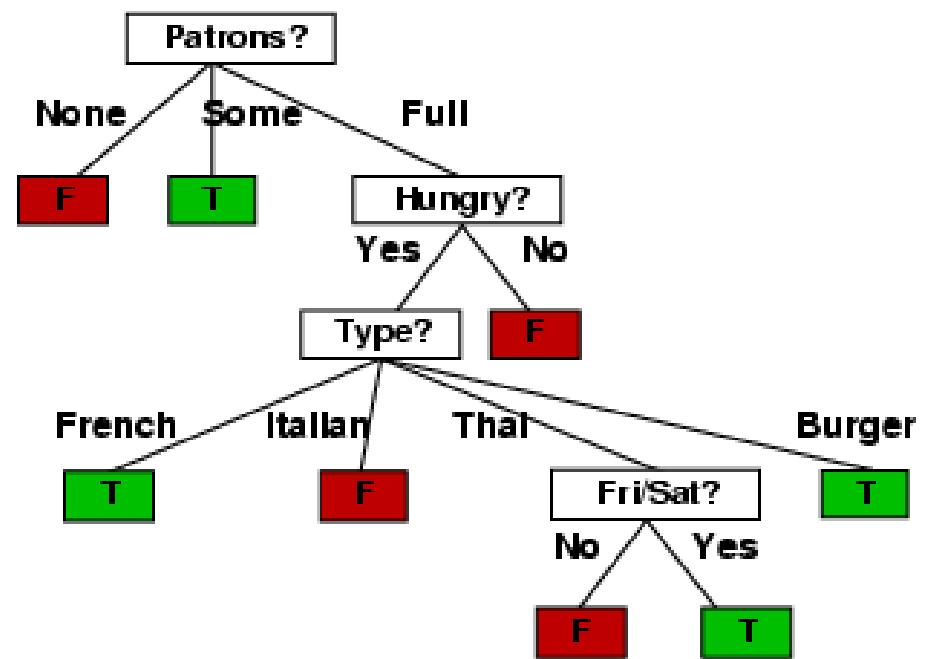
$$IG(\text{Type}) = 1 - \left[\frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) \right] = 0 \text{ bits}$$



$$IG(\text{Patrons}) = 1 - \left[\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right) \right] = .0541 \text{ bits}$$

Example contd.

Decision tree learned from the 12 examples:



How to Split: Greedy Maximization of Information Gain (reformulation)

Input:

- ◆ training set $\mathcal{T} = \{(\mathbf{x}_1, k_1), (\mathbf{x}_2, k_2), \dots, (\mathbf{x}_N, k_N)\}$.
- ◆ class \mathcal{S} of allowed splitting parameters

Do:

1. $P = \mathcal{T}$
2. Find the best splitting function (best is measured by **Information Gain** which is the difference of entropies H of data before and after the split. Note: To maximize IG means to minimize entropies after the split.)

$$S = \operatorname{argmin}_{\theta \in \mathcal{S}} \left(\frac{|P_1|}{|P|} H(P_1) + \frac{|P_2|}{|P|} H(P_2) + \dots + \frac{|P_k|}{|P|} H(P_k) \right) \quad (8)$$

$$\{P_1, P_2, \dots, P_k\} = S(P, \theta) \quad (9)$$

$$H(P_i) = - \sum_j \frac{|P_i^{(j)}|}{|P_i|} \log \frac{|P_i^{(j)}|}{|P_i|} \quad (\text{entropy of } P_i) \quad (10)$$

$(P_i^{(j)} : \text{number of points of class } j \text{ in } P_i)$

3. Go recursively to branches

How to Split: Greedy Minimization of Training Error

Input:

- ◆ training set $\mathcal{T} = \{(\mathbf{x}_1, k_1), (\mathbf{x}_2, k_2), \dots, (\mathbf{x}_N, k_N)\}$.
- ◆ class \mathcal{S} of allowed splitting parameters

Do:

1. $P = \mathcal{T}$
2. Find the best splitting function (best measured by training error ϵ):

$$S = \operatorname{argmin}_{\theta \in \mathcal{S}} \left(\frac{|P_1|}{|P|} \epsilon(P_1) + \frac{|P_2|}{|P|} \epsilon(P_2) + \dots + \frac{|P_k|}{|P|} \epsilon(P_k) \right) \quad (11)$$

$$\{P_1, P_2, \dots, P_k\} = S(P, \theta) \quad (12)$$

$$\epsilon(P_i) = 1 - \max_j \frac{|P_i^{(j)}|}{|P_i|} \quad (13)$$

$(P_i^{(j)} : \text{number of points of class } j \text{ in } P_i)$

3. Go recursively to branches

Random Decision Forests

Decision trees tend to overfit. Combining multiple DTs offers a solution to this problem.

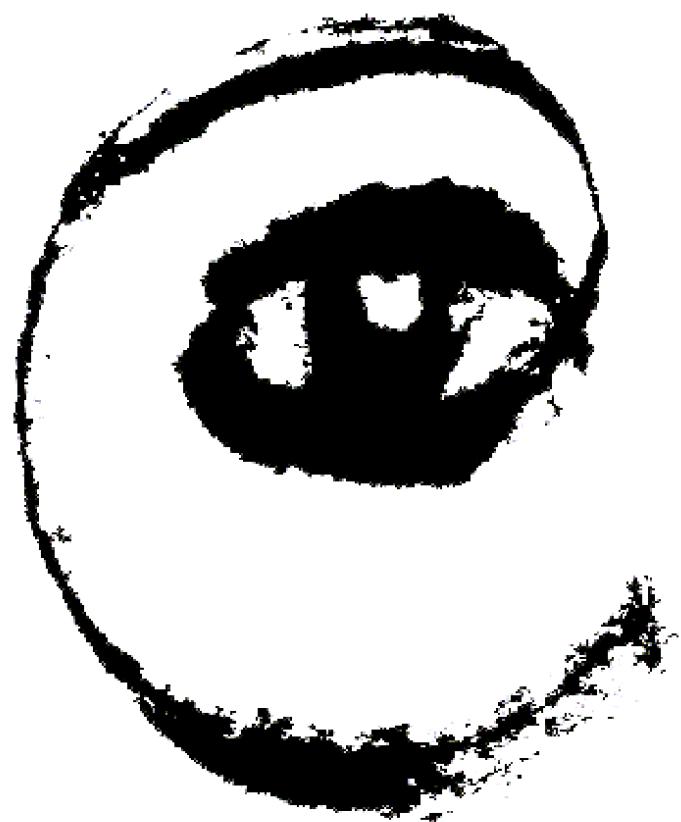
But:

- ◆ How should multiple trees be combined?
- ◆ How should multiple trees be constructed? Note that so far, the procedure to obtain a DT has been deterministic. Repeating this procedure N times will result in N identical trees.

Combining trees: Often, majority of votes for a given class is used (like in NNs).

Constructing multiple different trees requires randomization. Randomization can be employed for:

- ◆ Training set generation for individual DTs, using **bagging**. For a training set with N_t samples, this means selecting N_t samples from this training set by sampling *with replacement*. Some training data will be selected multiple times. Different training sets will be produced due to random nature of sampling.
- ◆ **Random subset of features**. Bagging cannot produce uncorrelated trees if some features strongly dominate in the DT construction. Selecting randomly a subset of features to consider during the construction of individual DTs helps.

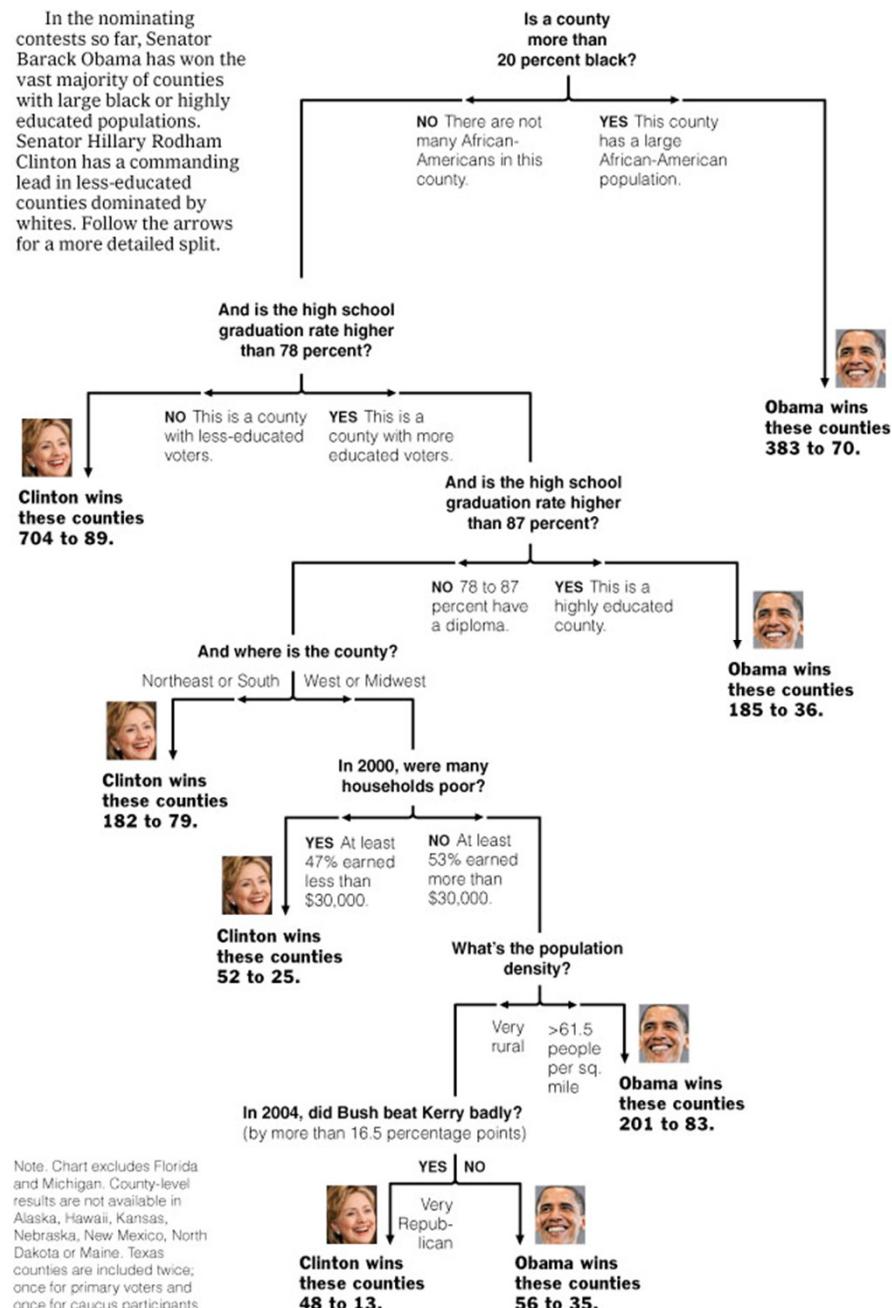


m p

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



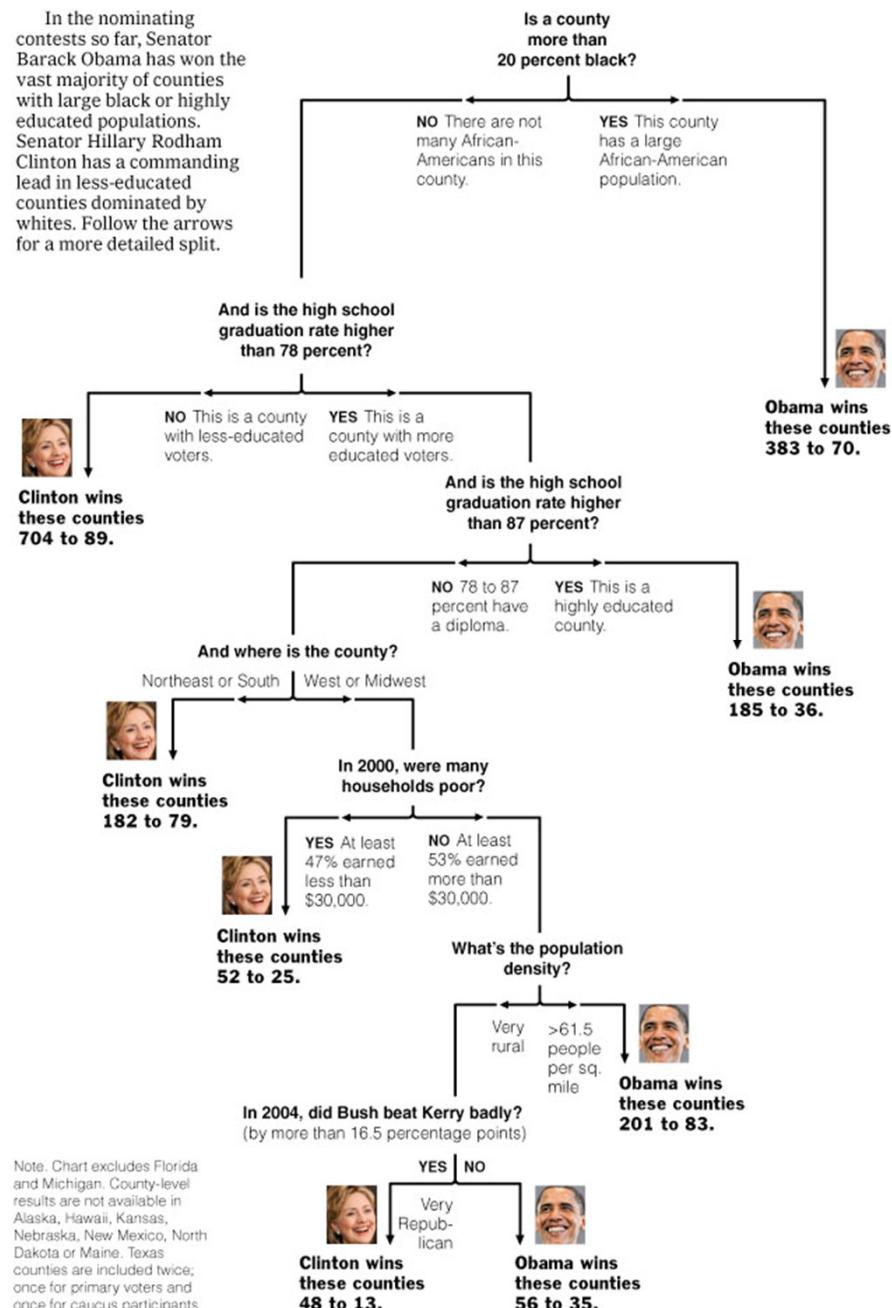
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



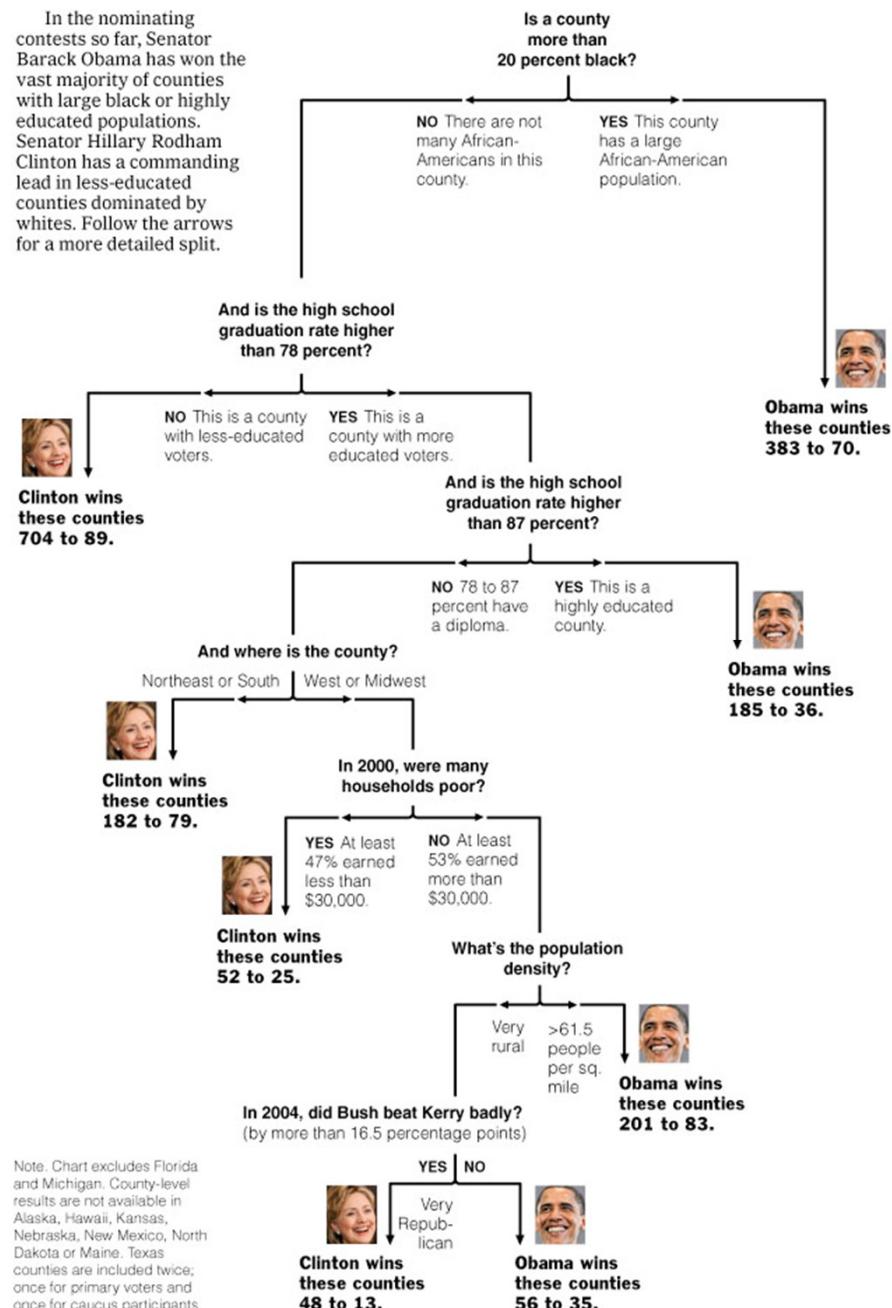
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



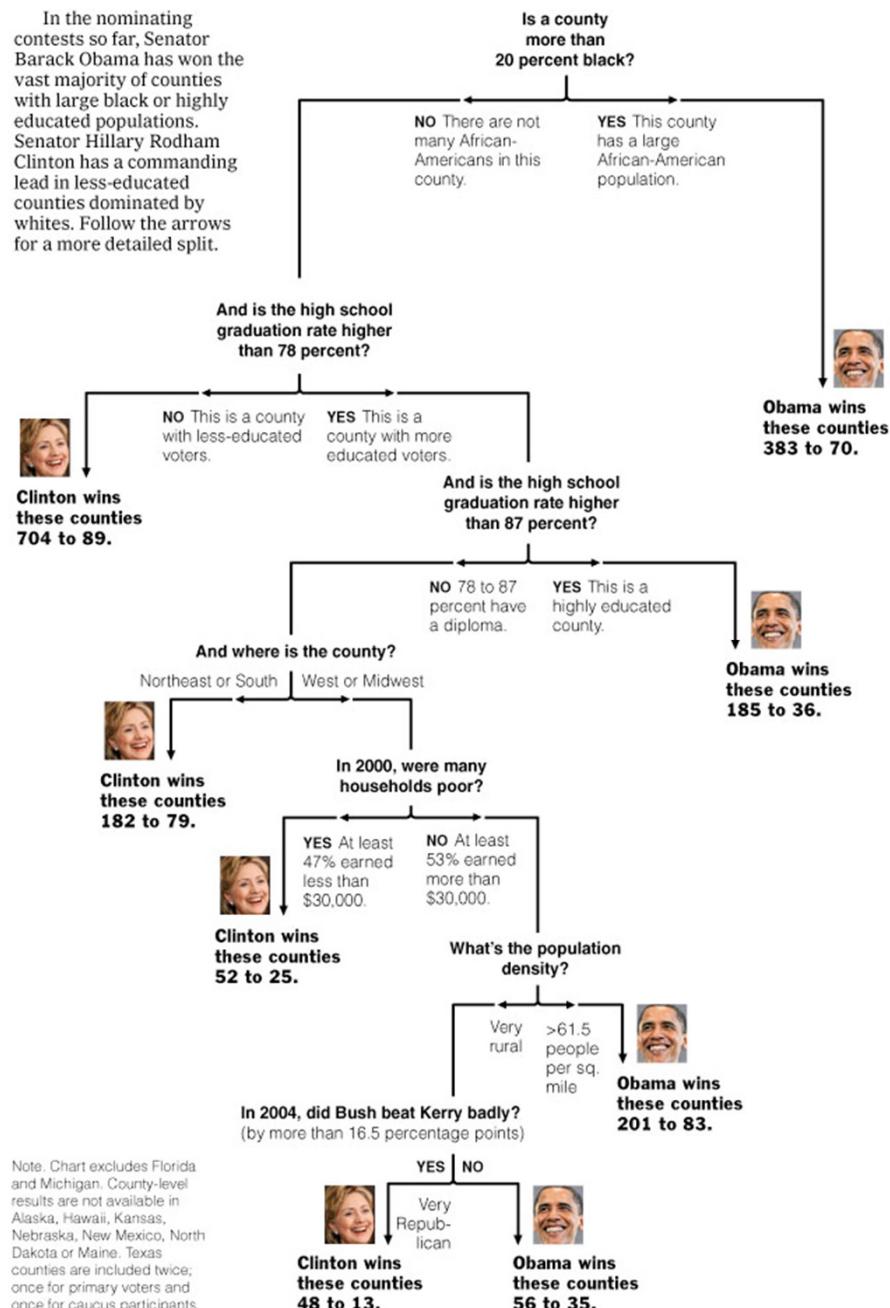
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



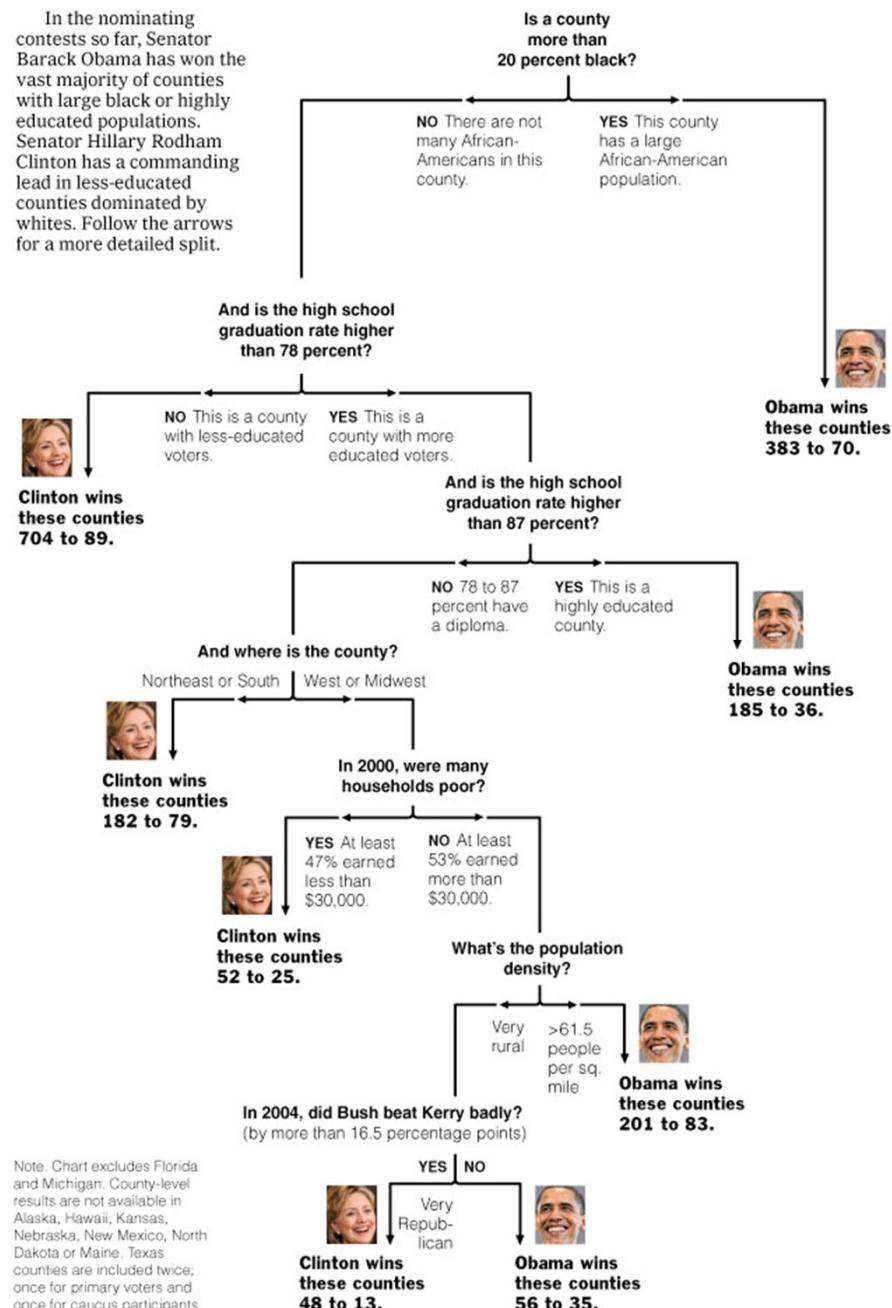
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



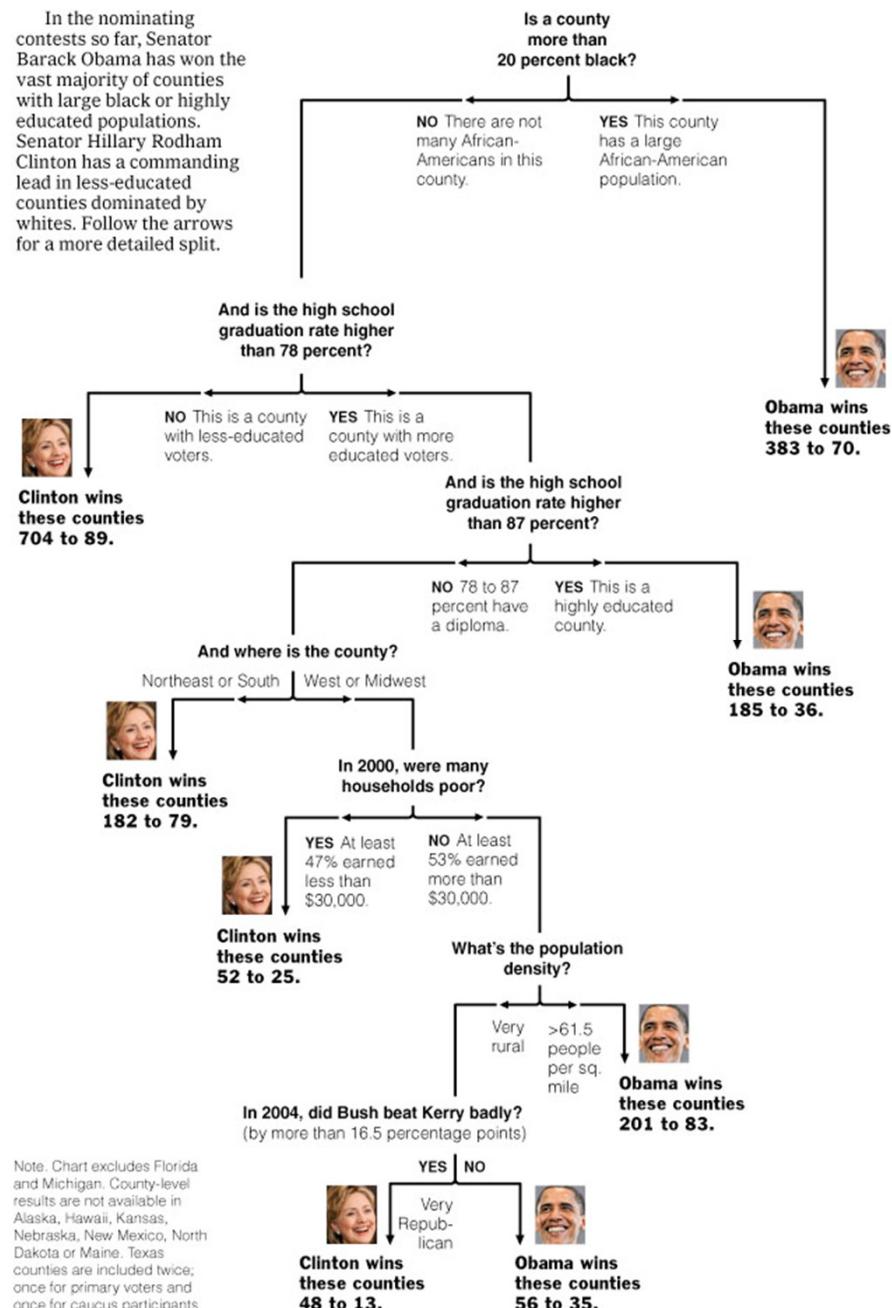
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



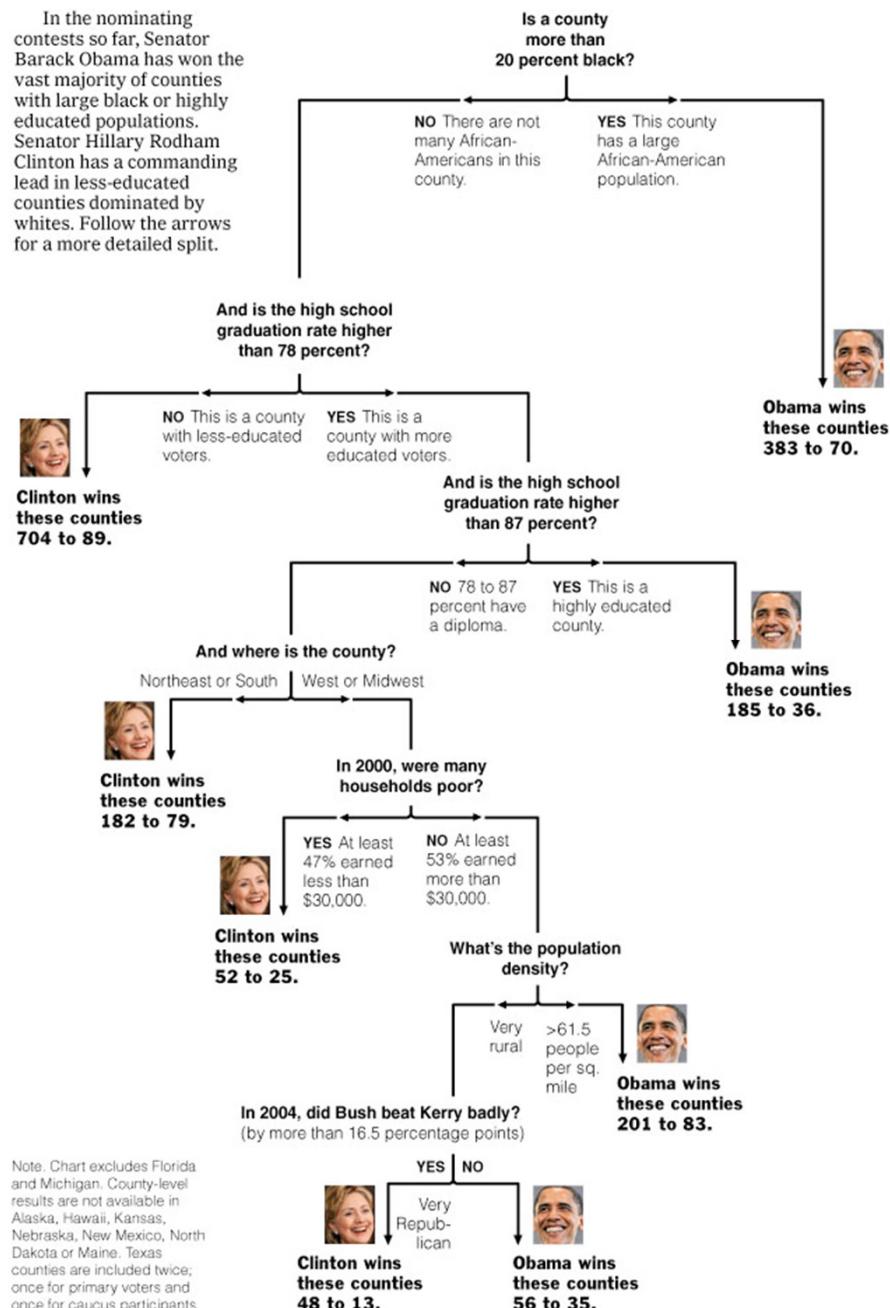
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



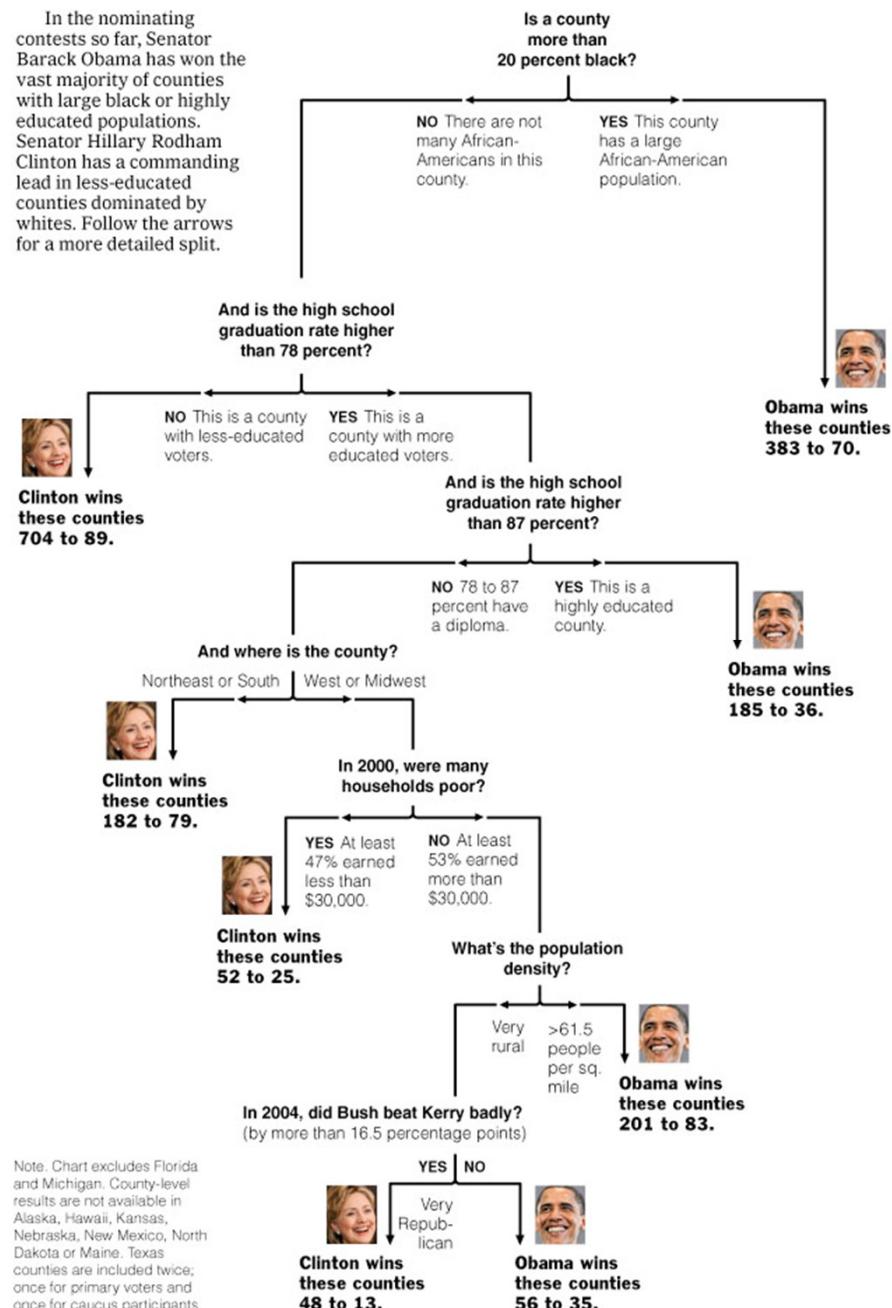
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



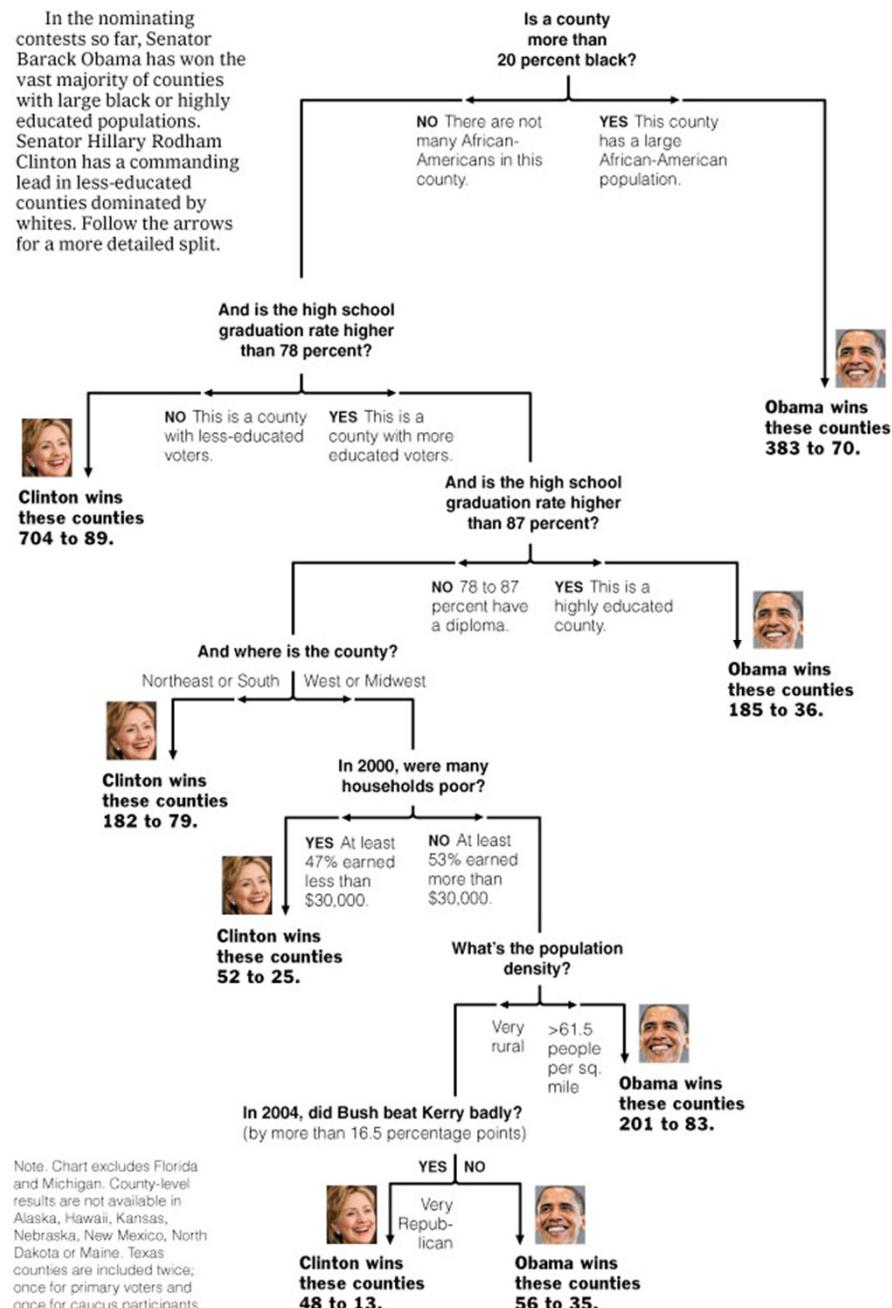
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



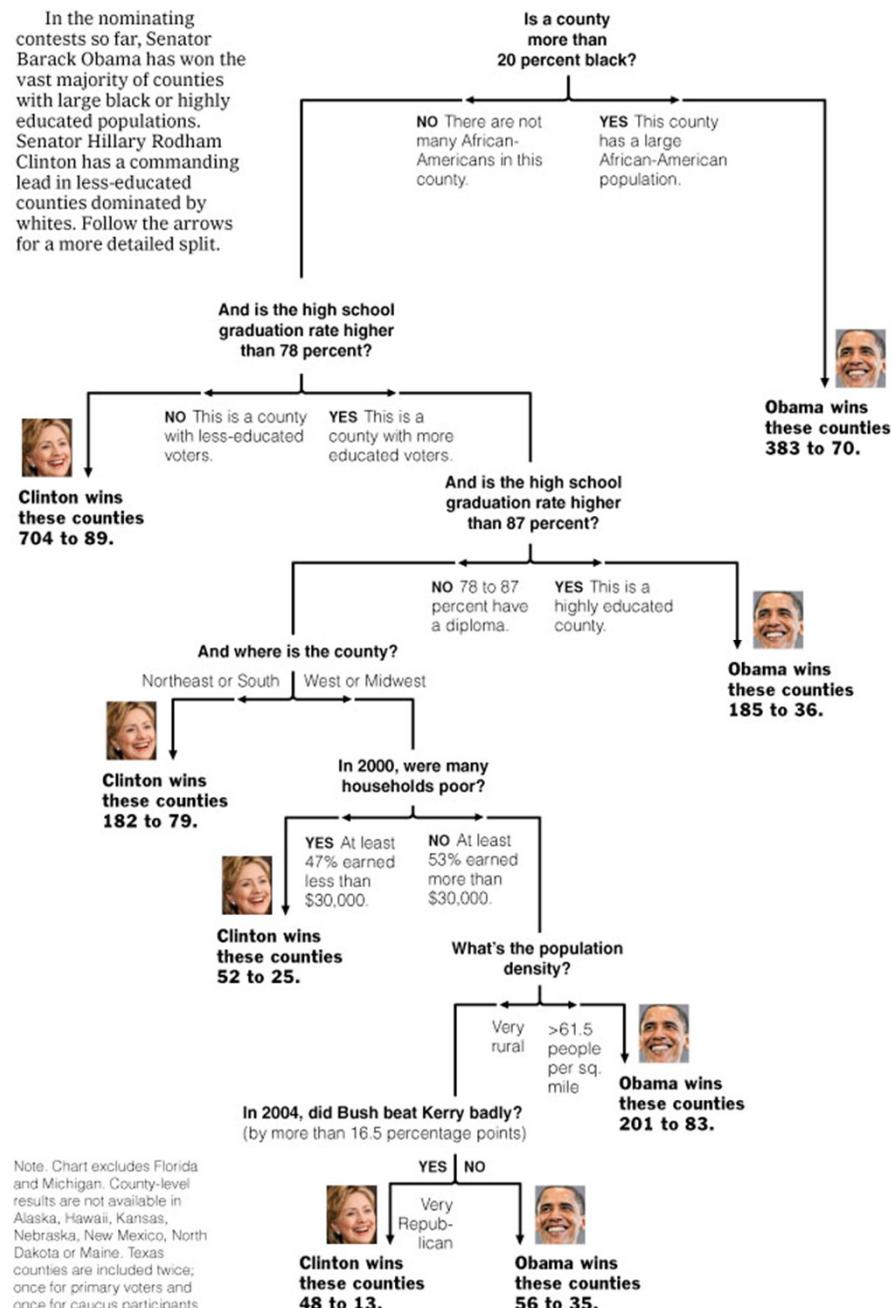
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



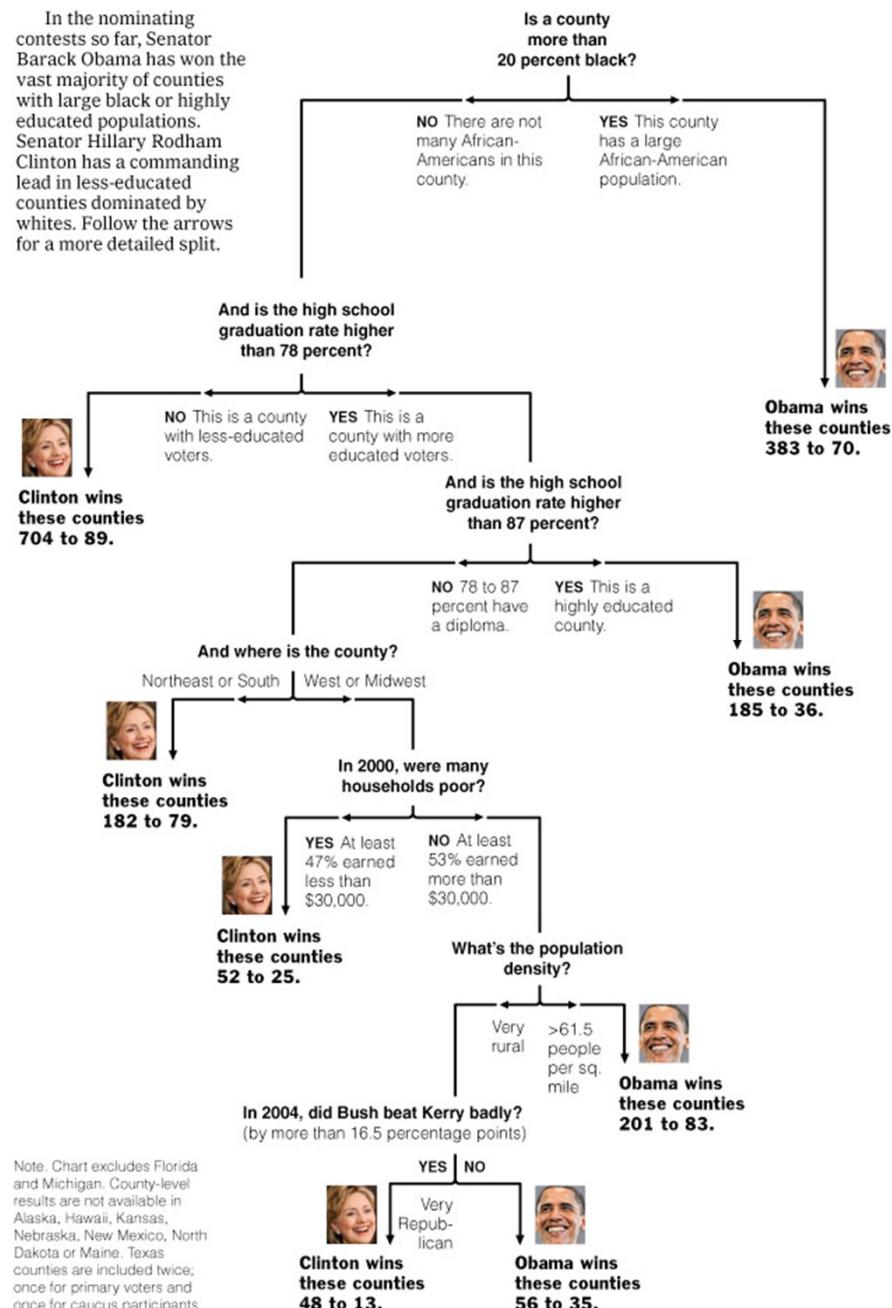
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

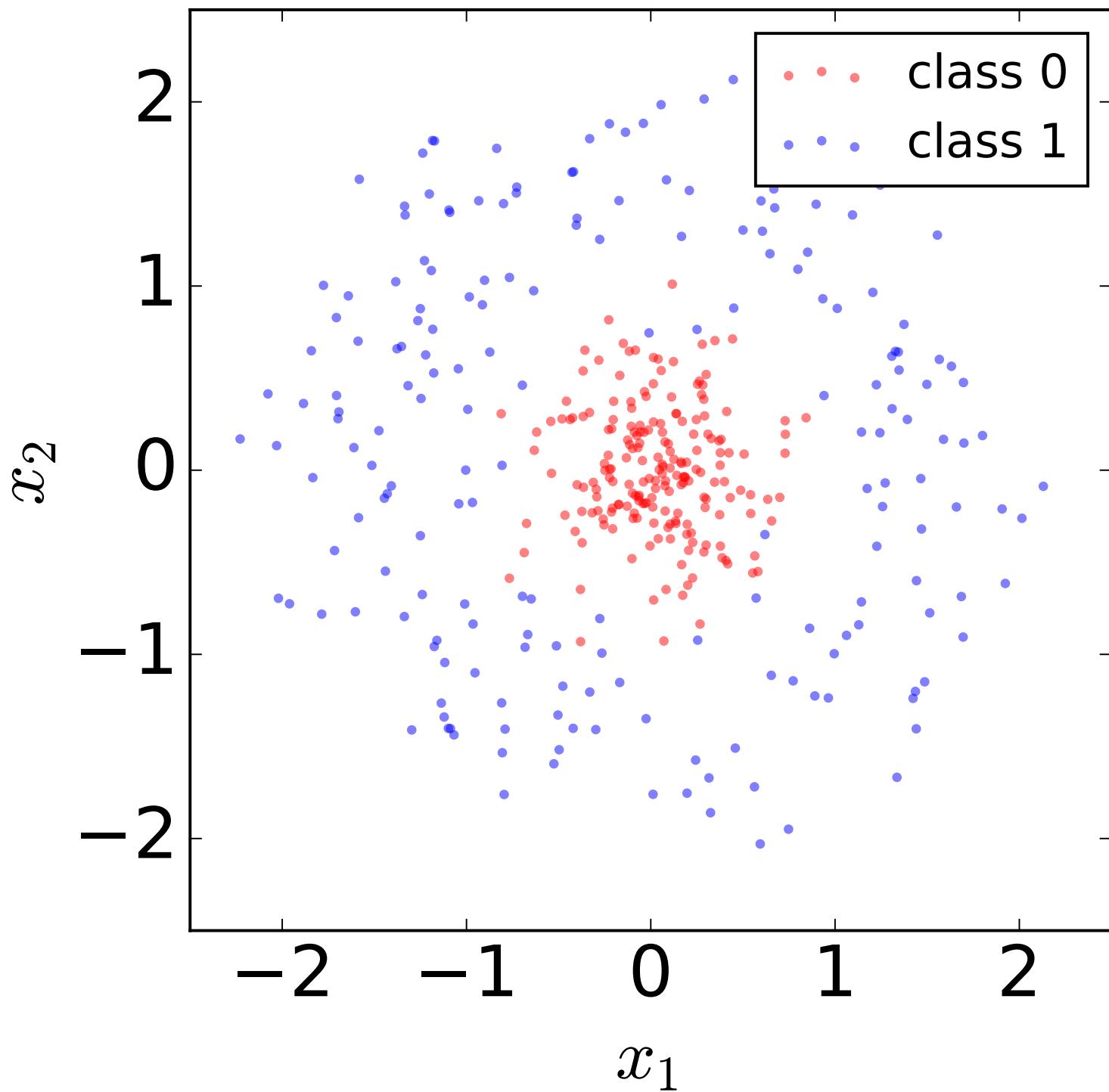
New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.

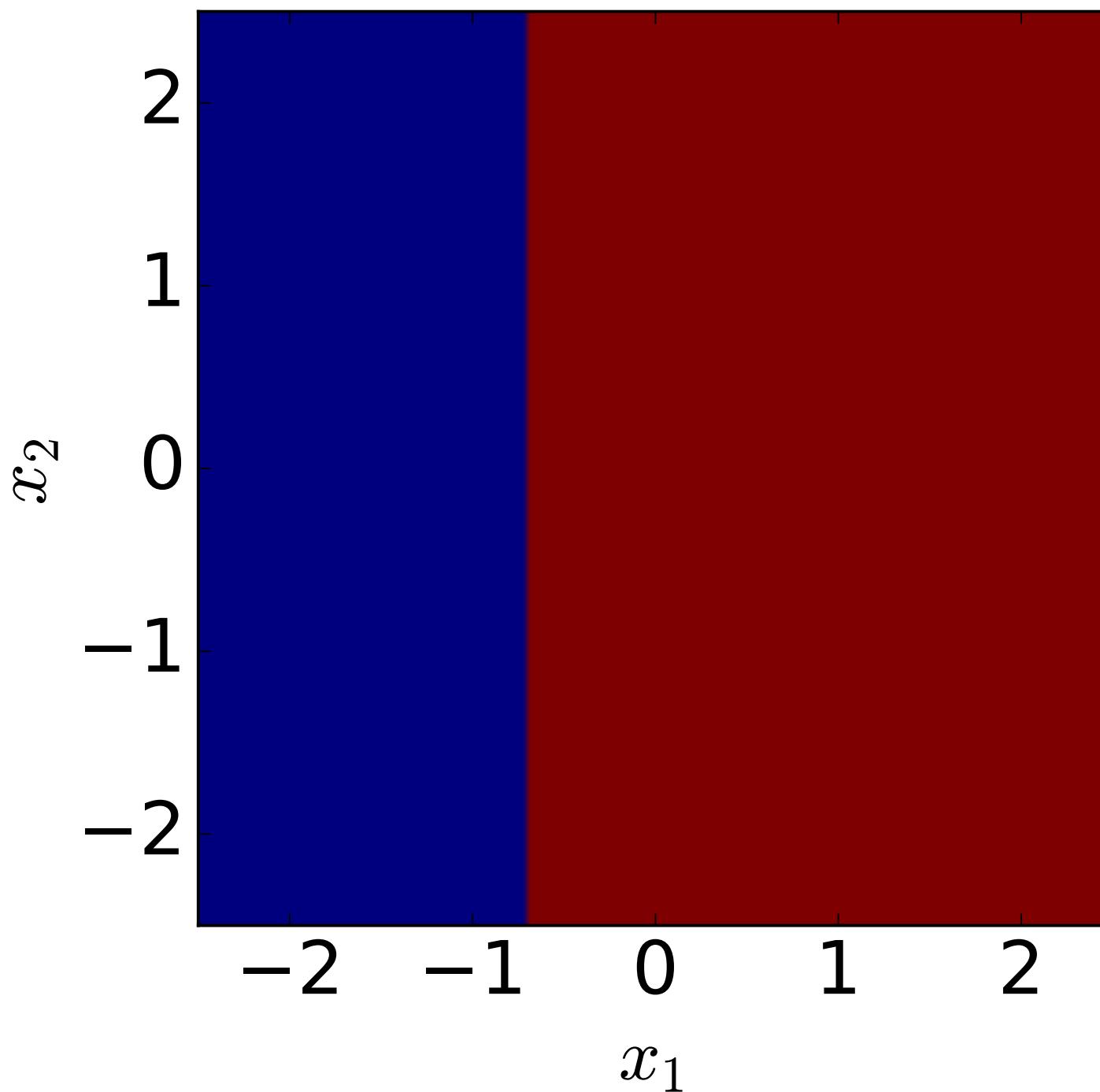


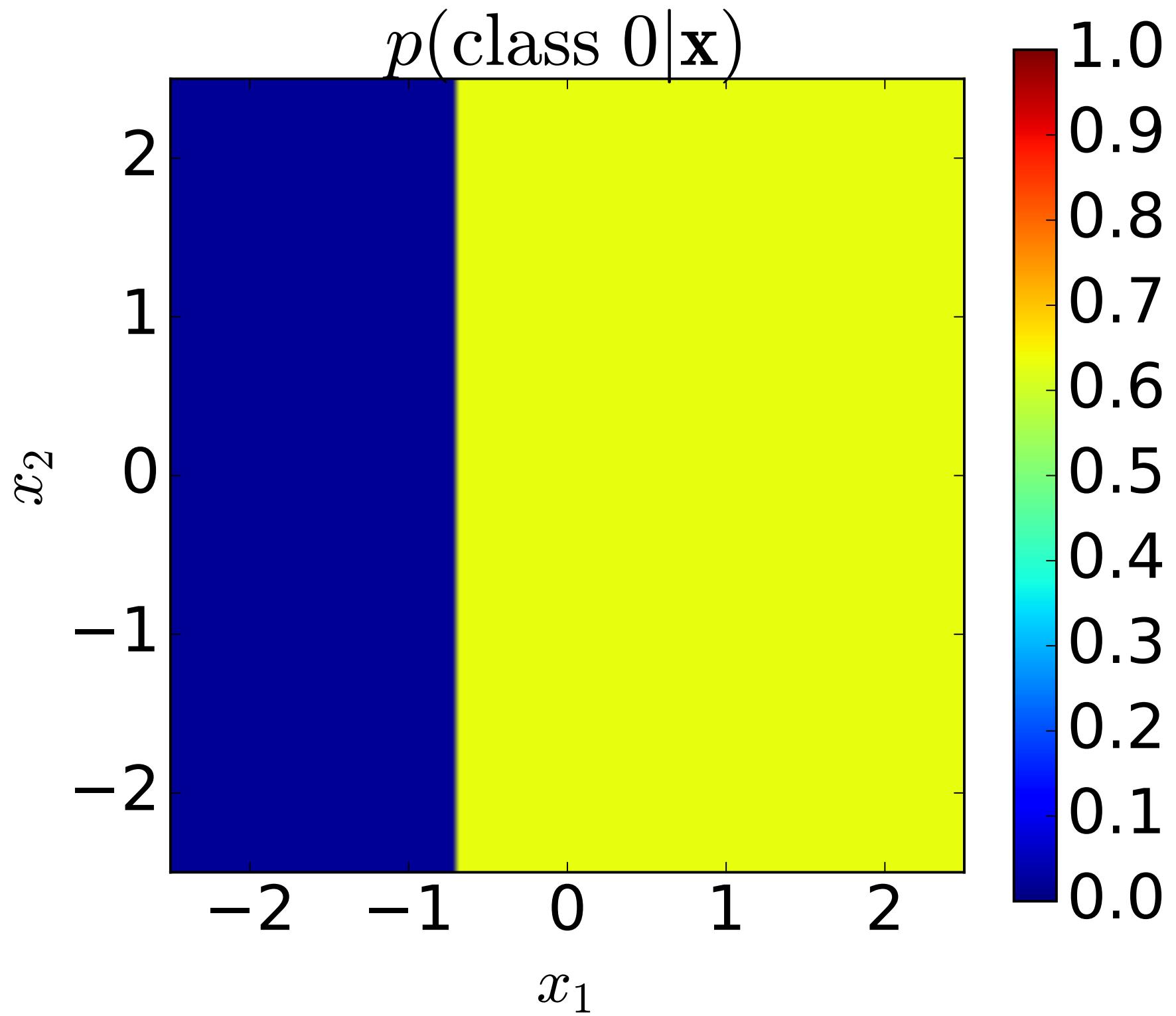
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES



classification



$p(\text{class } 0 | \mathbf{x})$ 

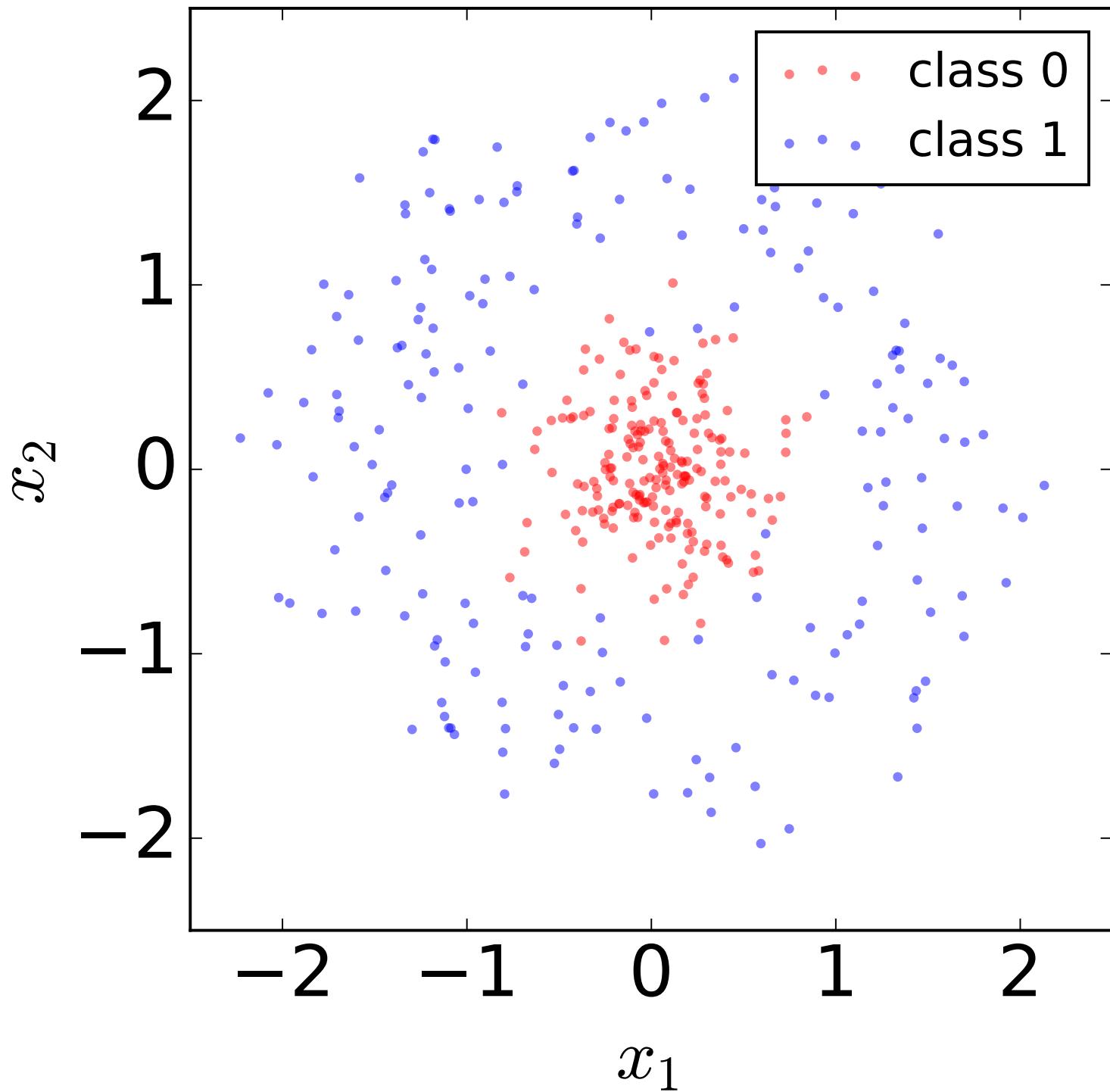
```
x1 ≤ -0.6916
samples = 400
class_samples = [200, 200]
class = 0
```

True

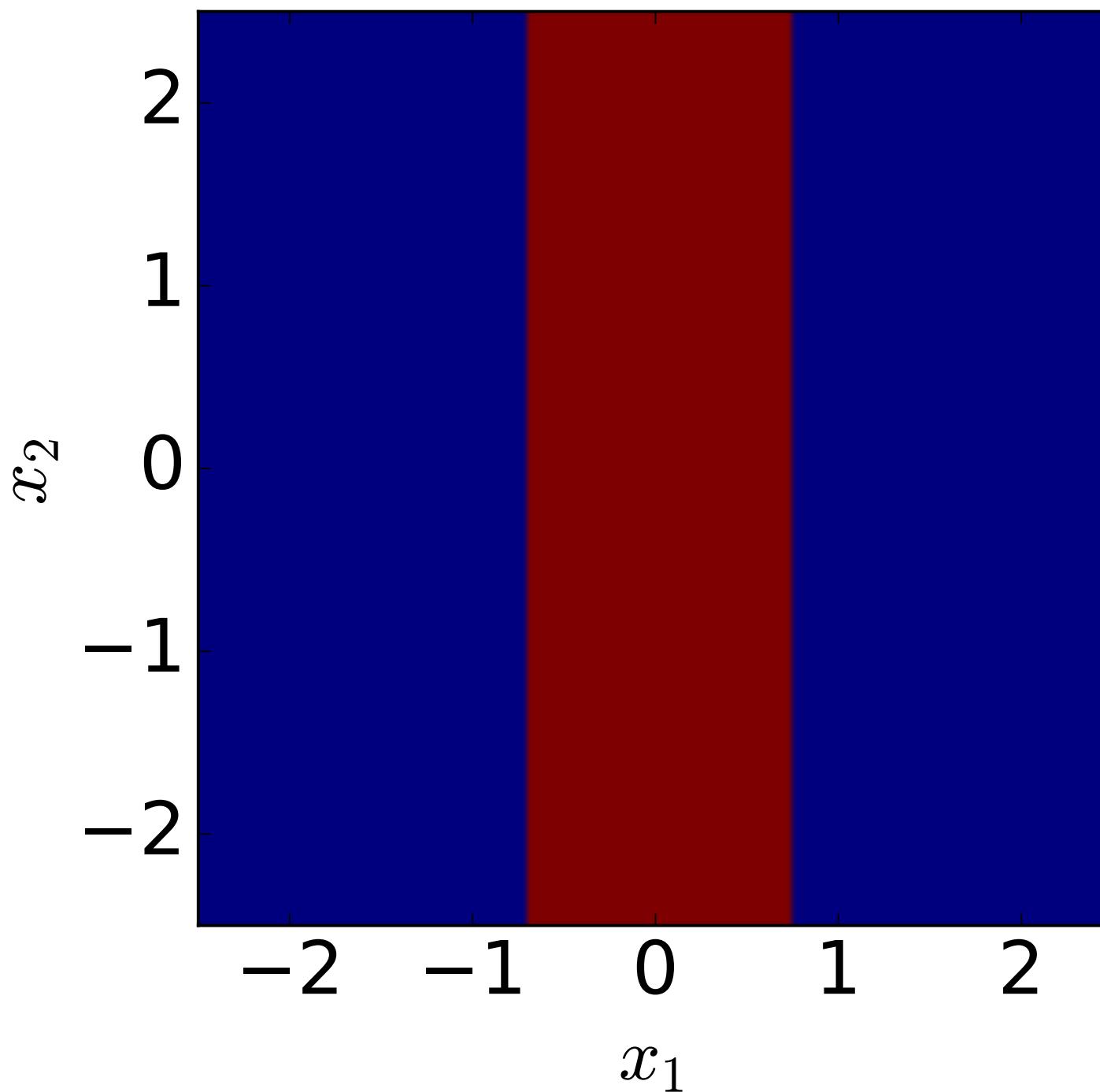
```
samples = 86
class_samples = [2, 84]
class = 1
```

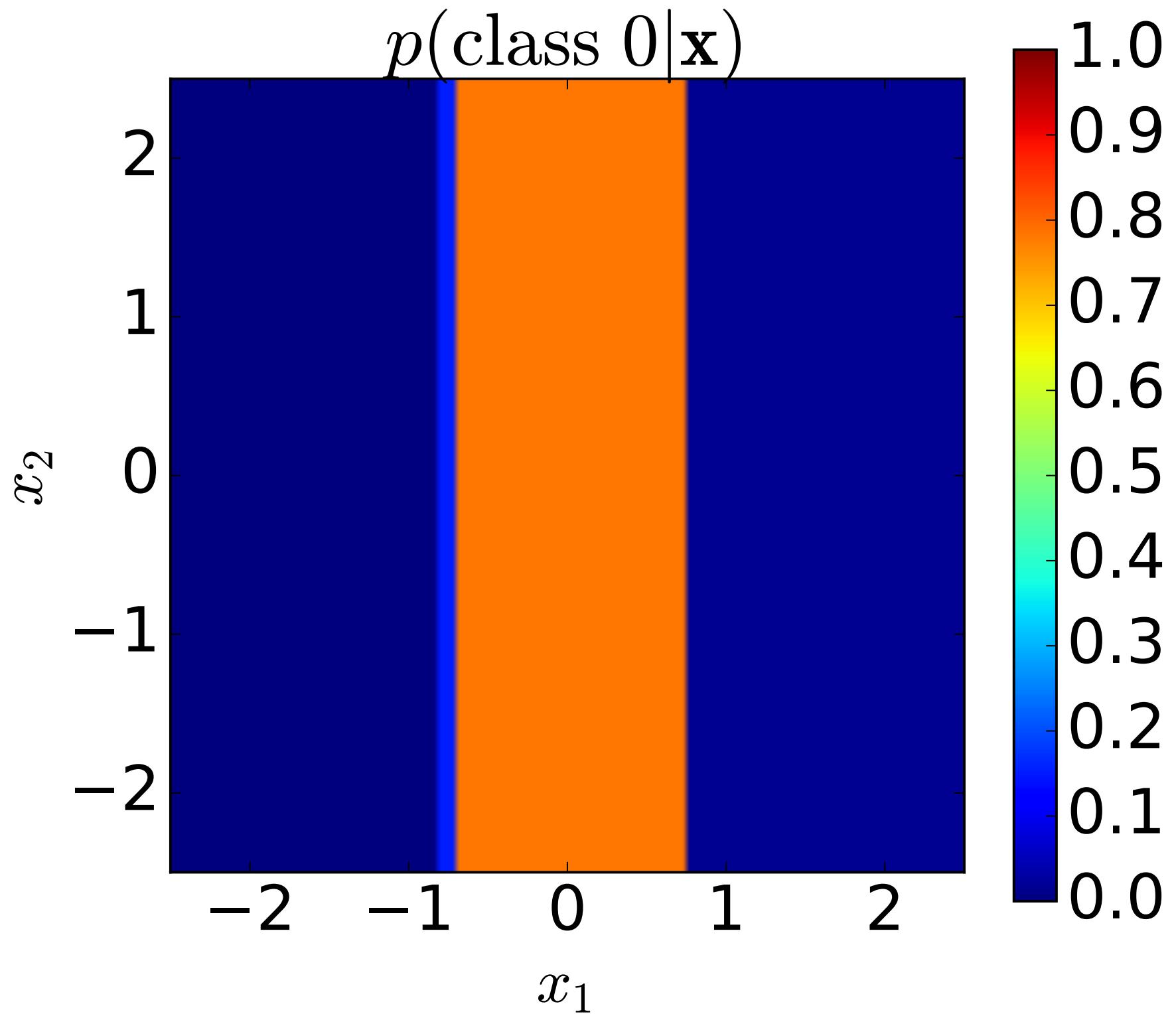
False

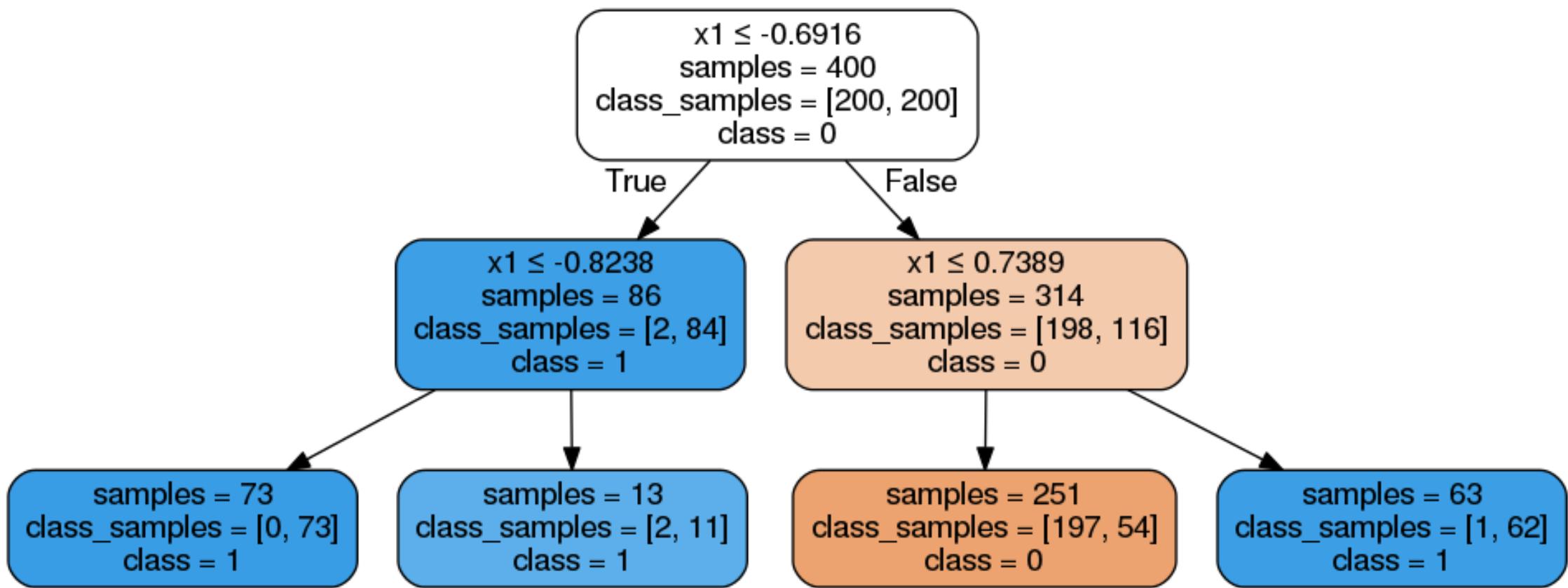
```
samples = 314
class_samples = [198, 116]
class = 0
```

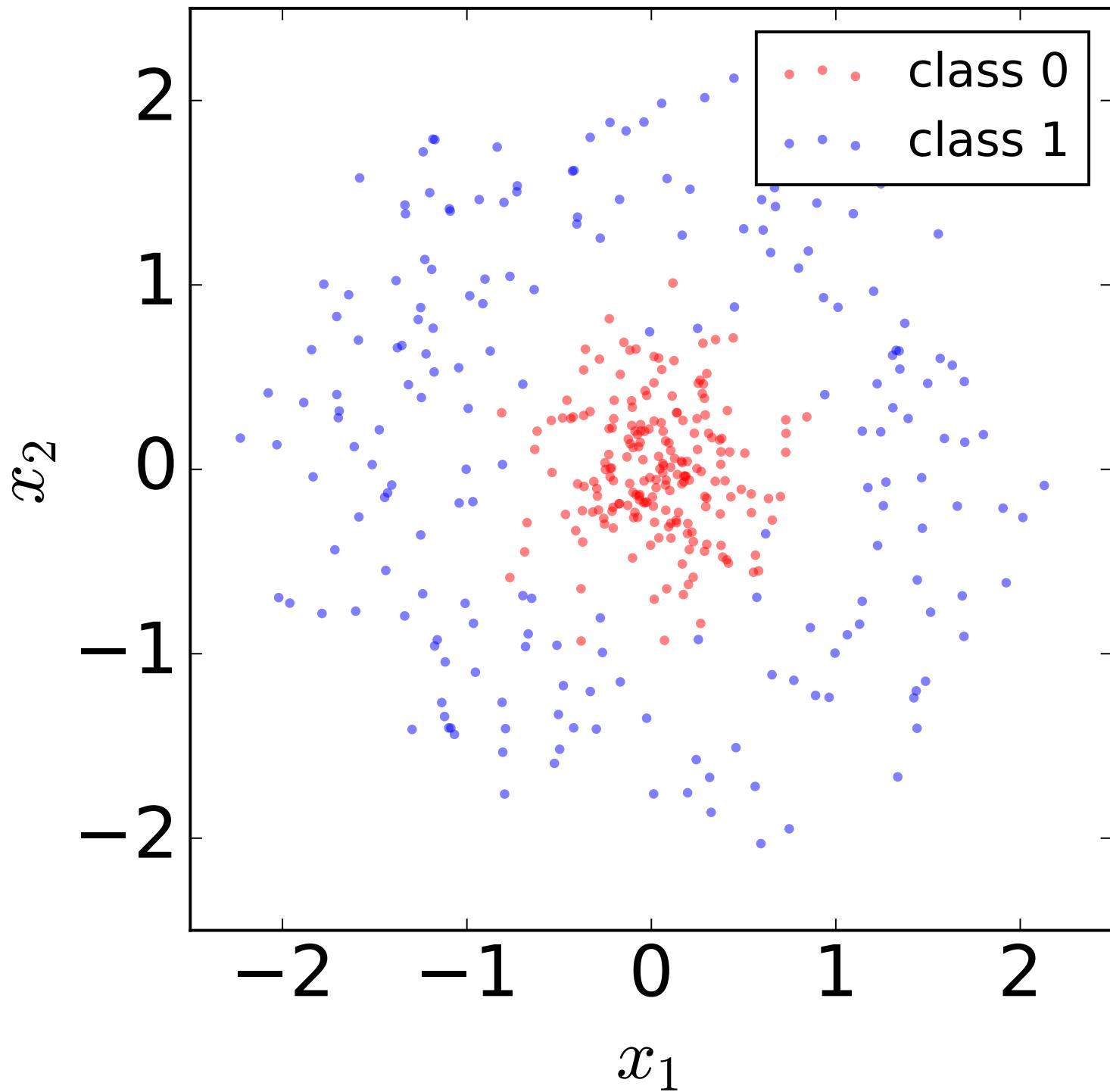


classification

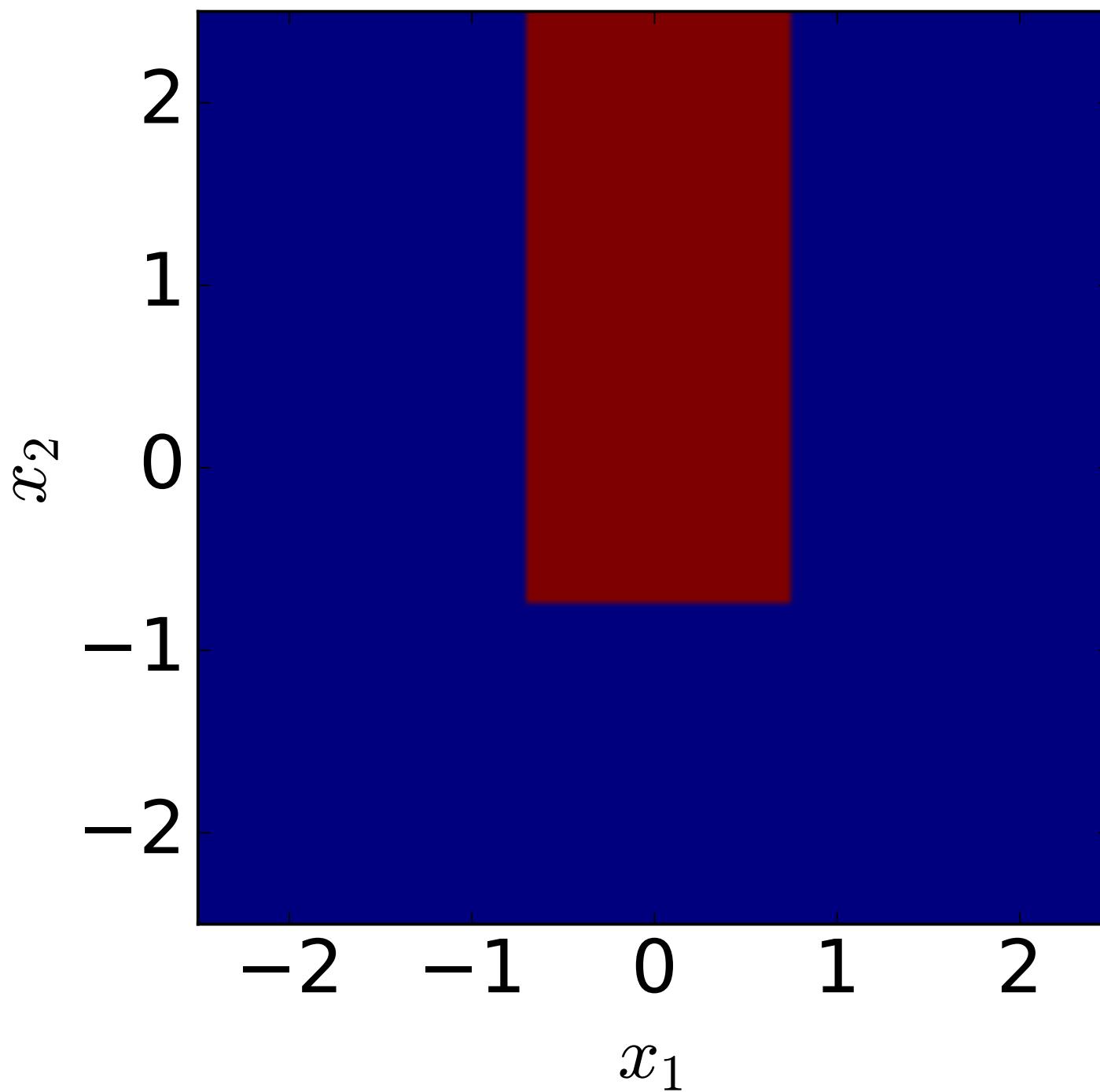


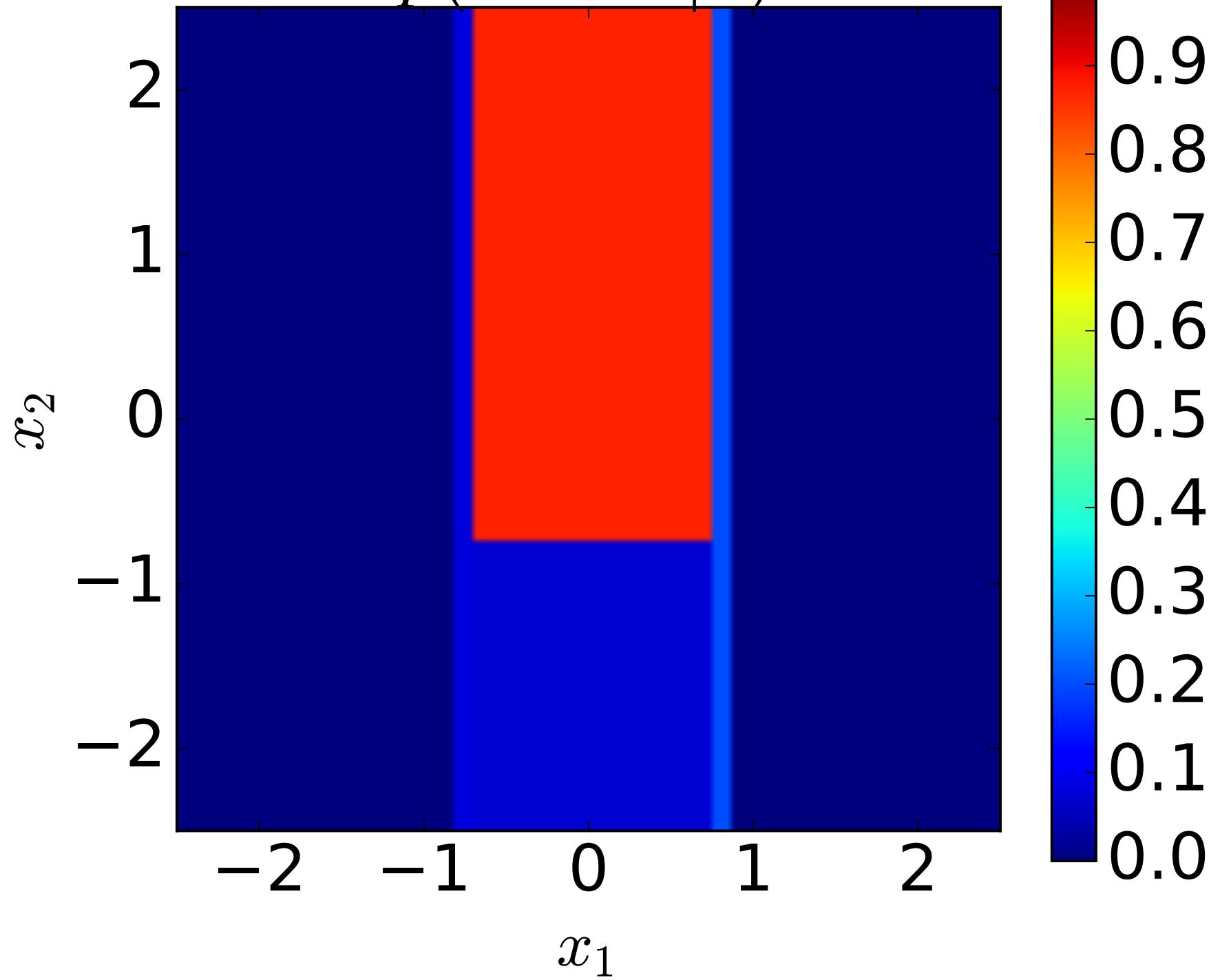
$p(\text{class } 0 | \mathbf{x})$ 

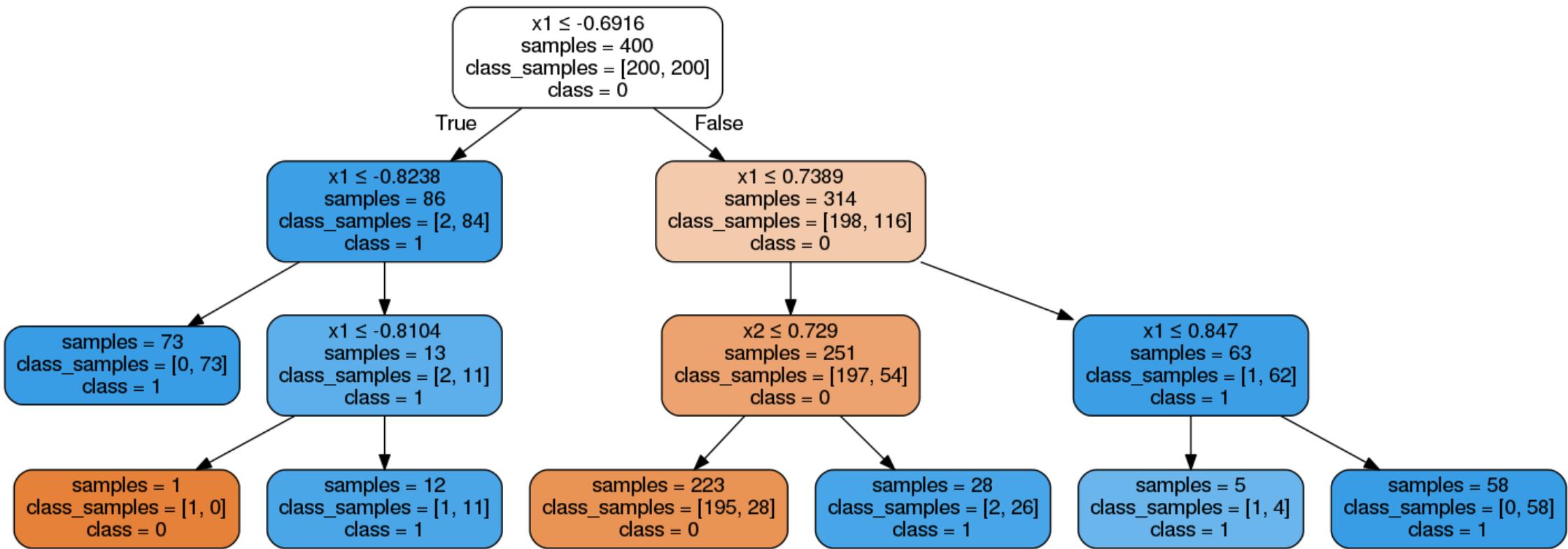


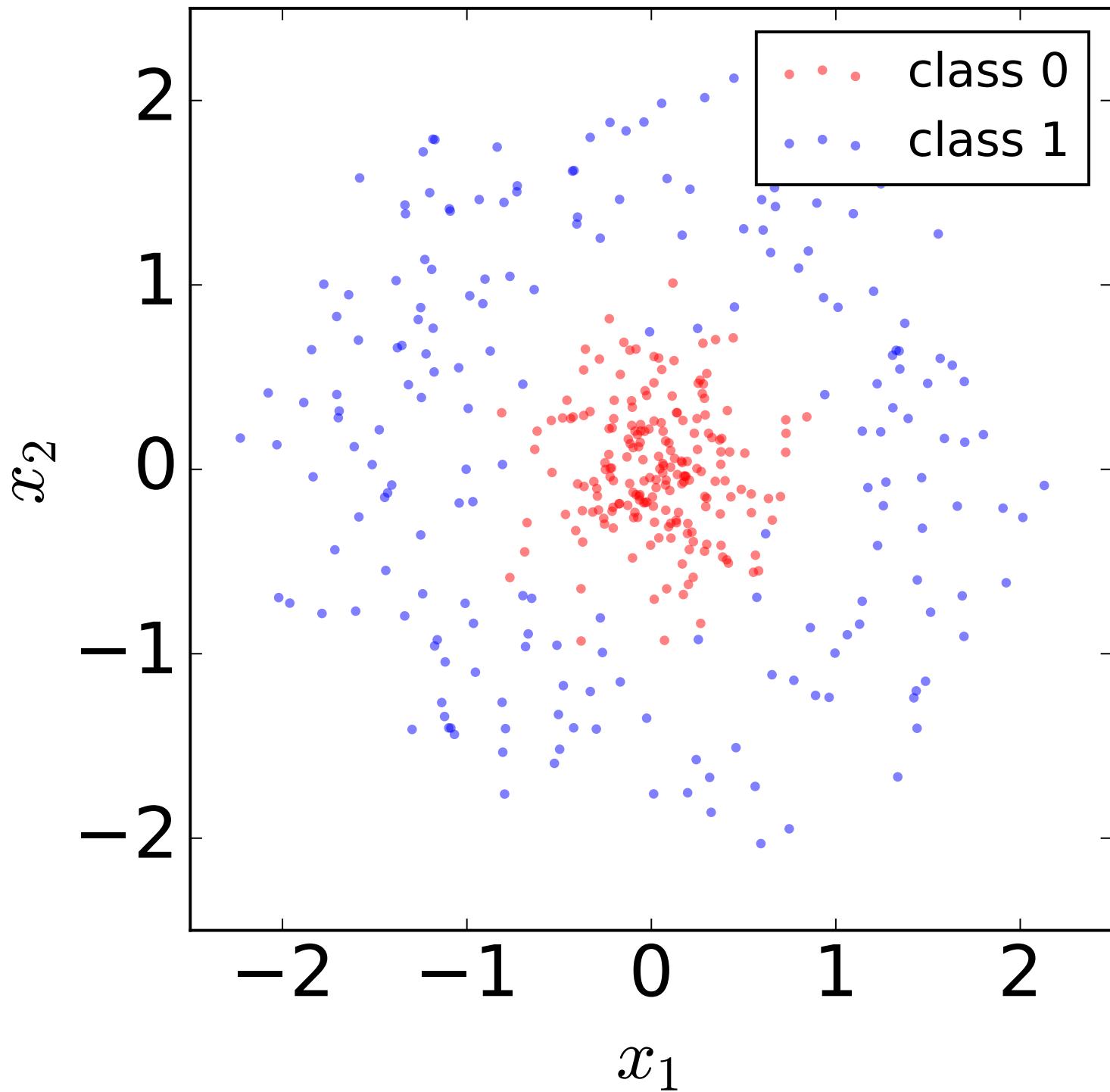


classification

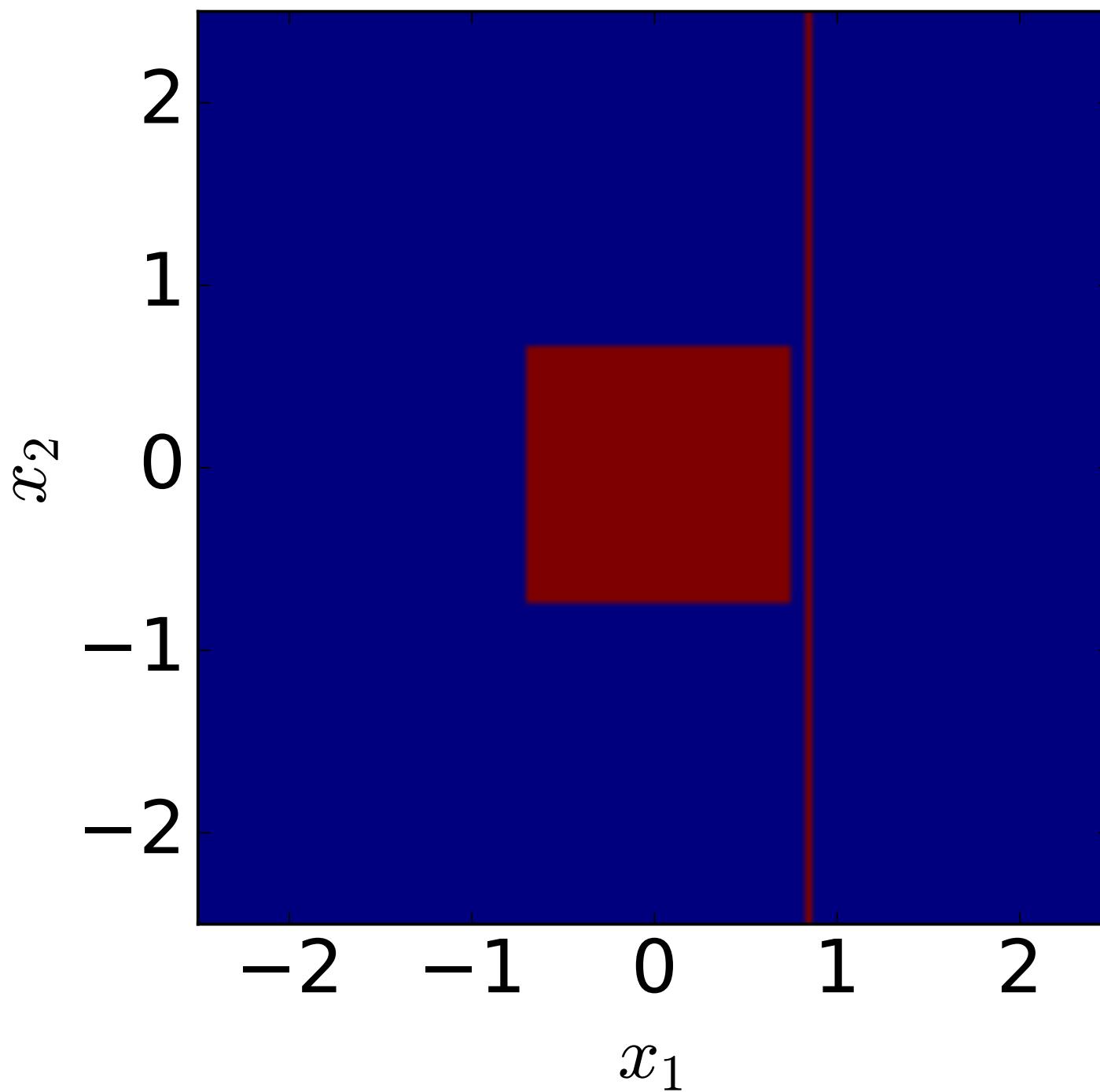


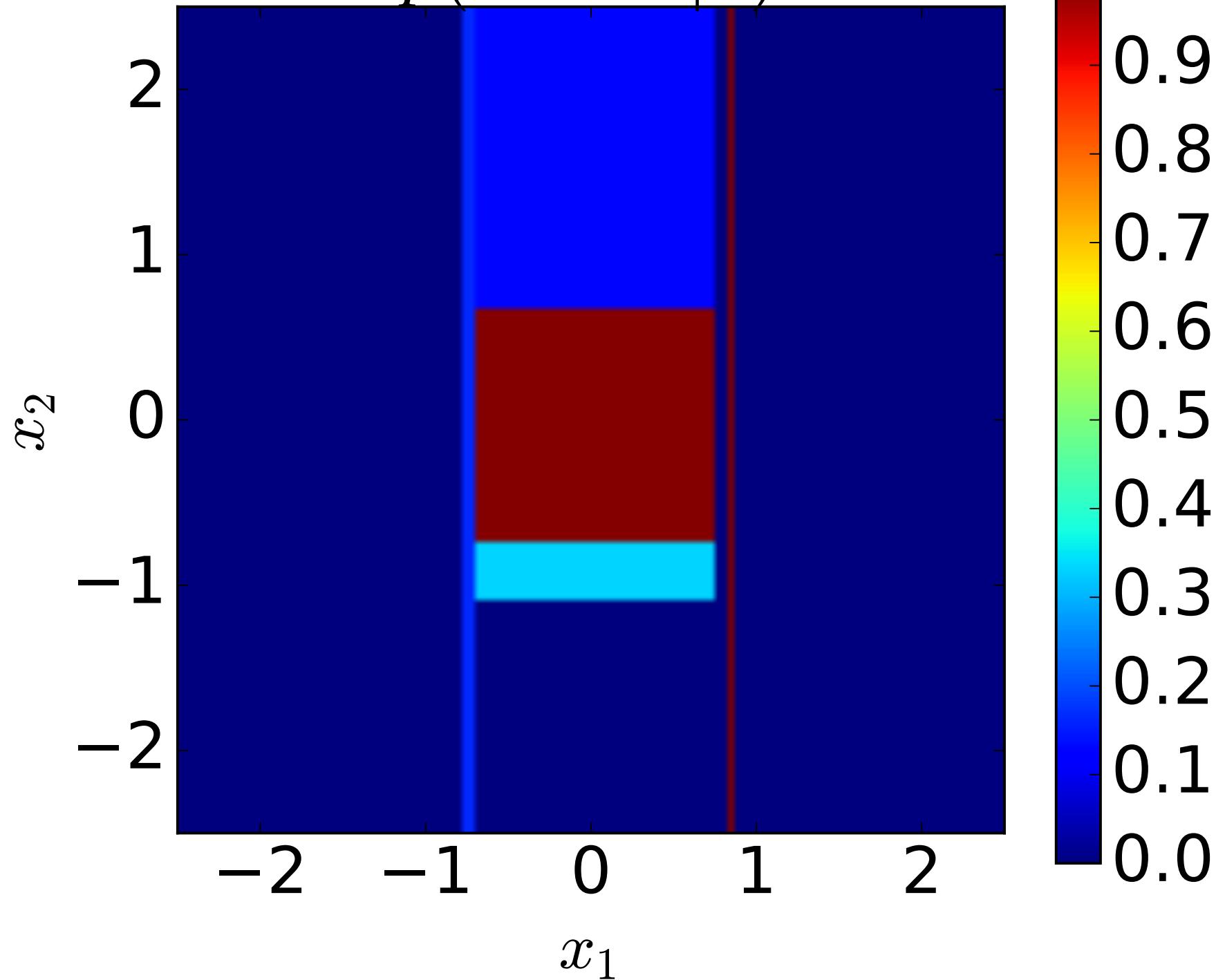
$p(\text{class } 0 | \mathbf{x})$ 

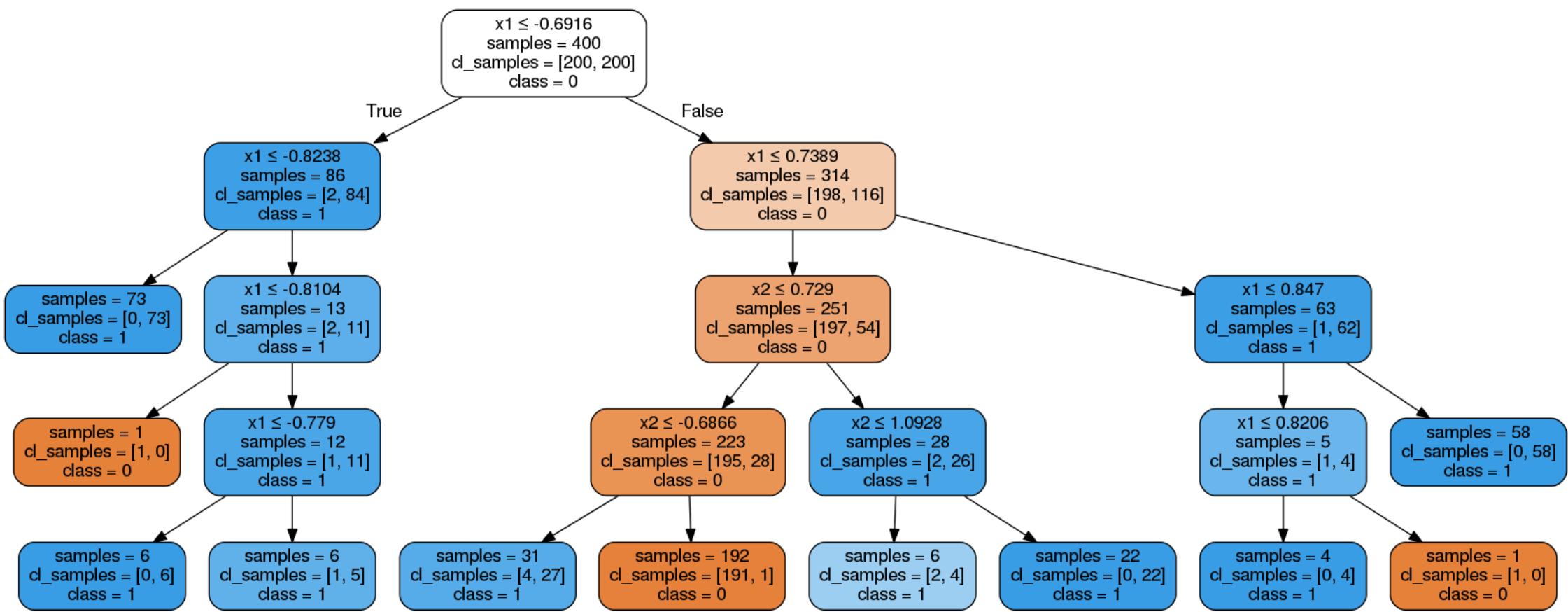




classification



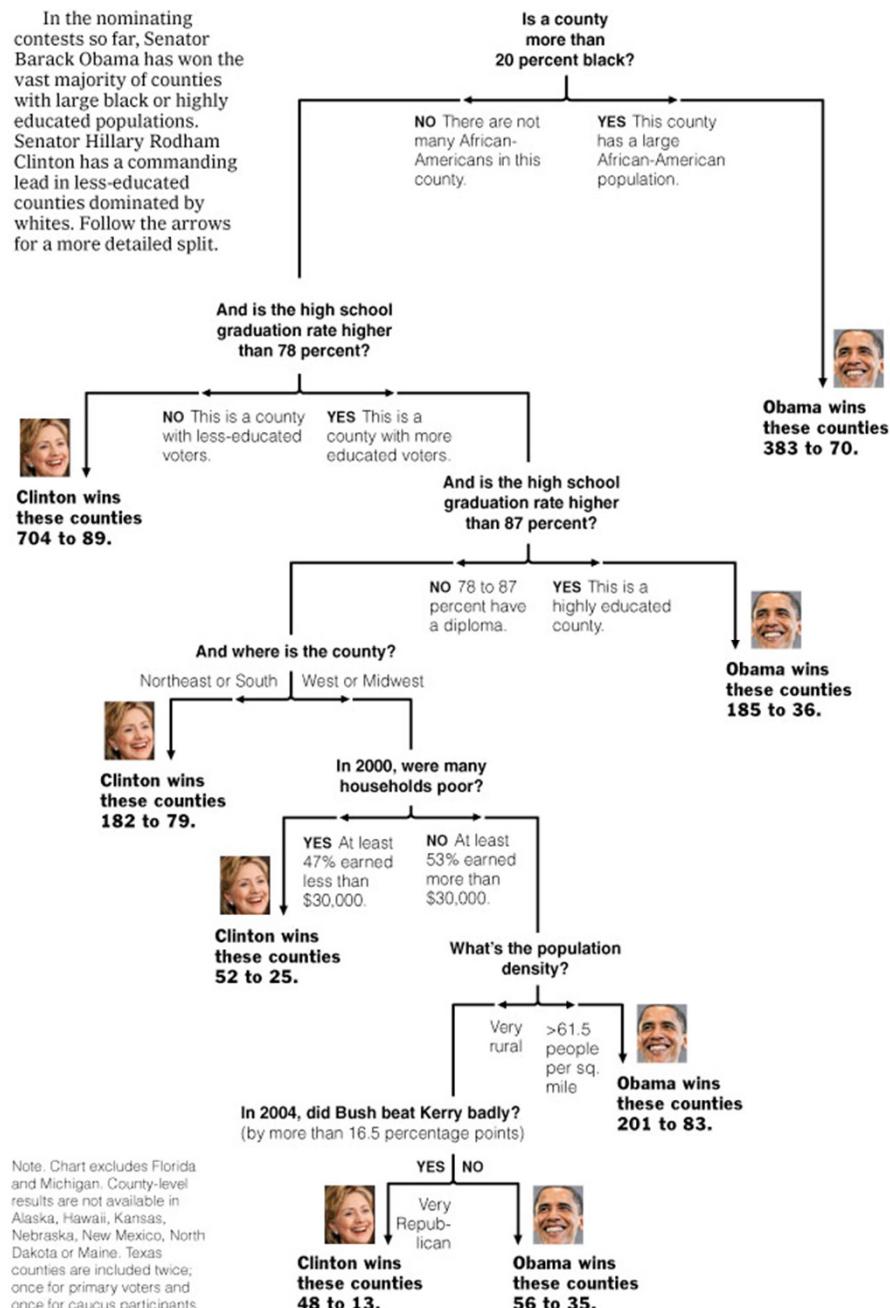
$p(\text{class } 0 | \mathbf{x})$ 



Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



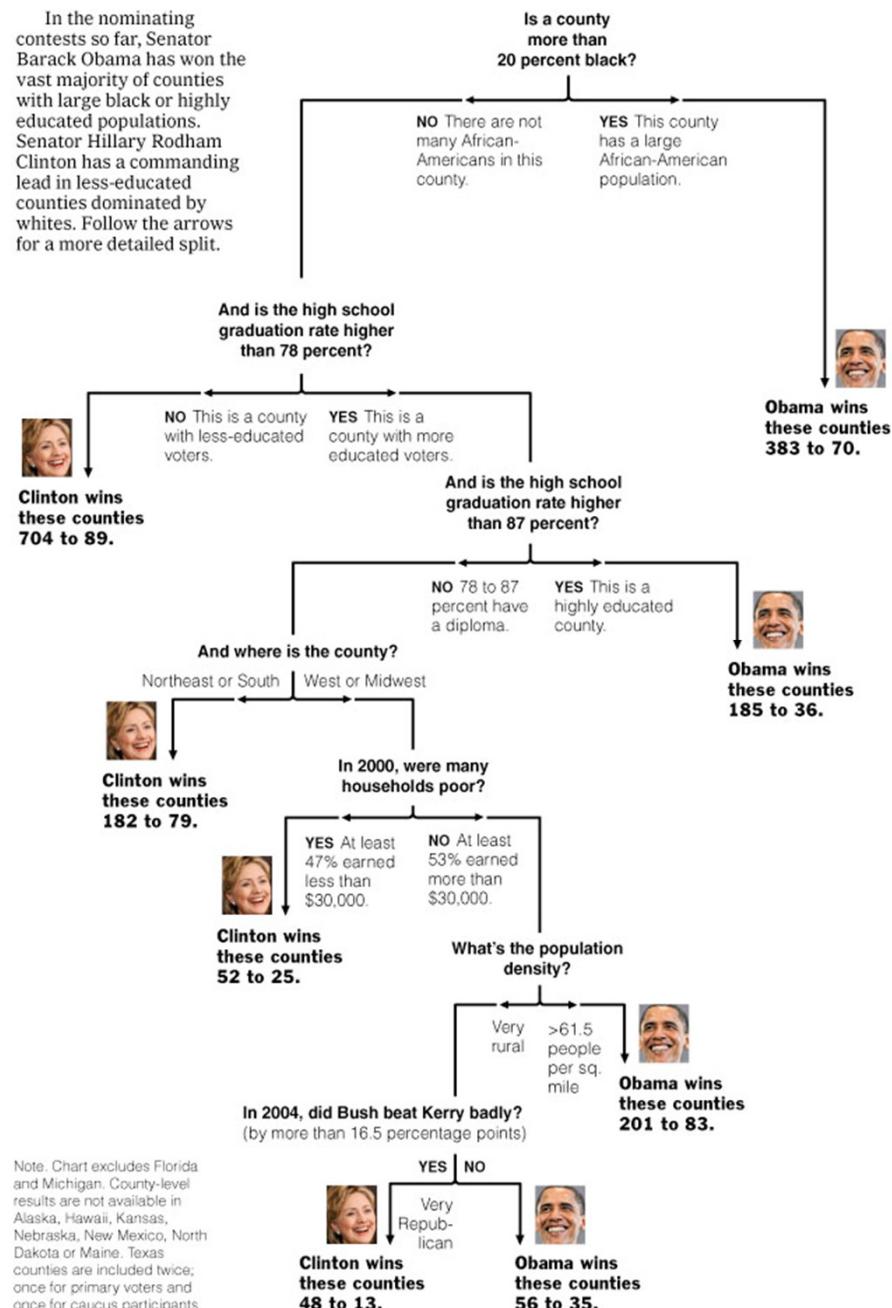
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



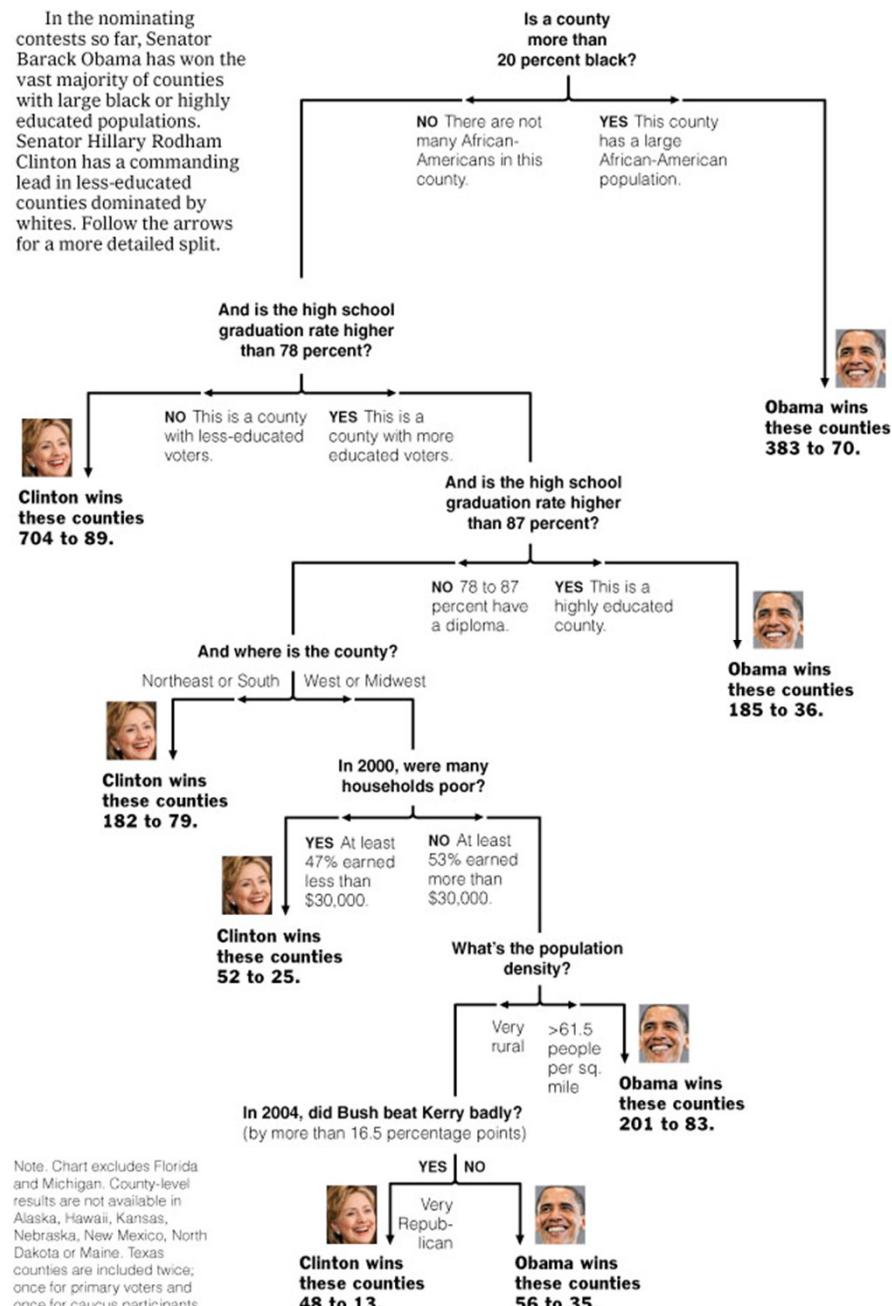
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



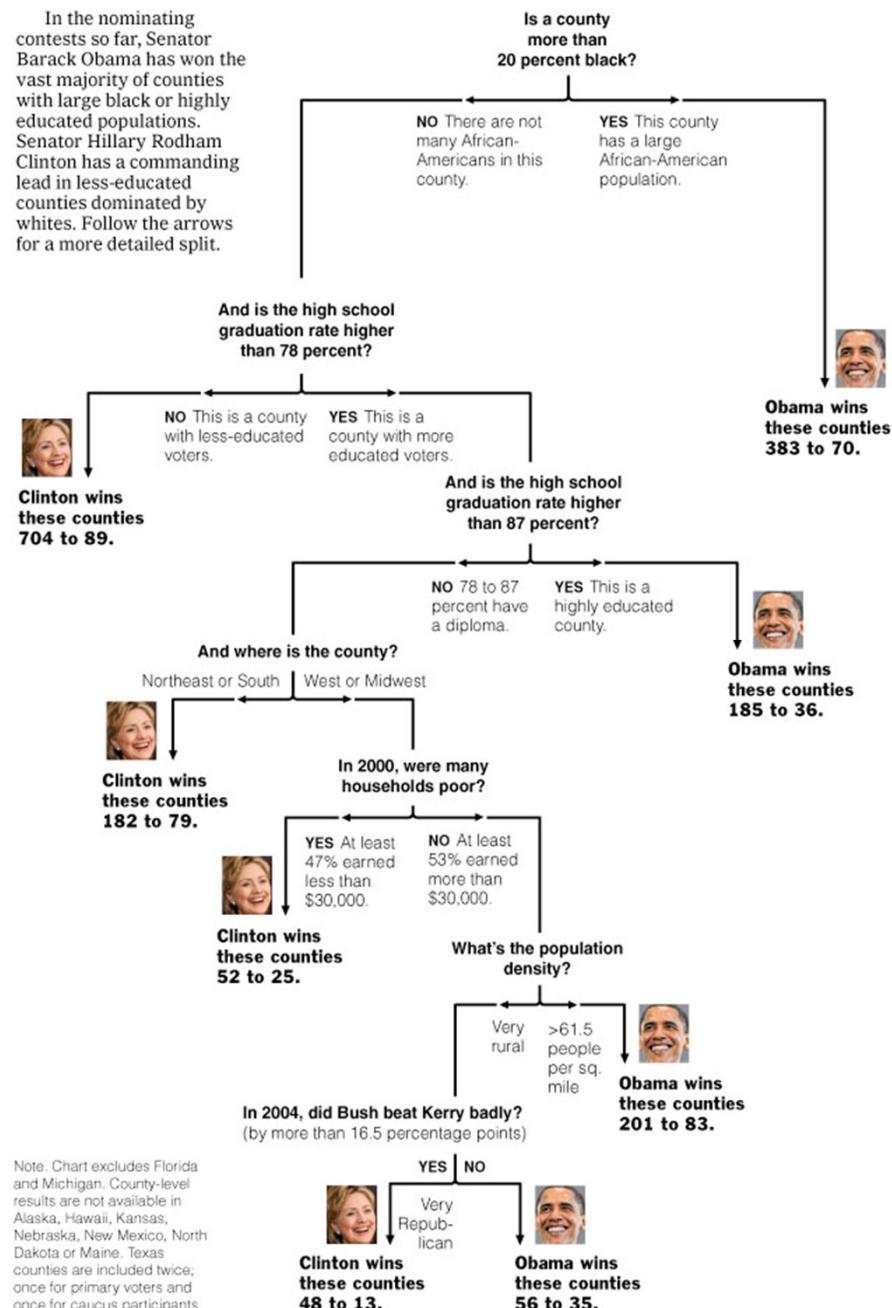
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



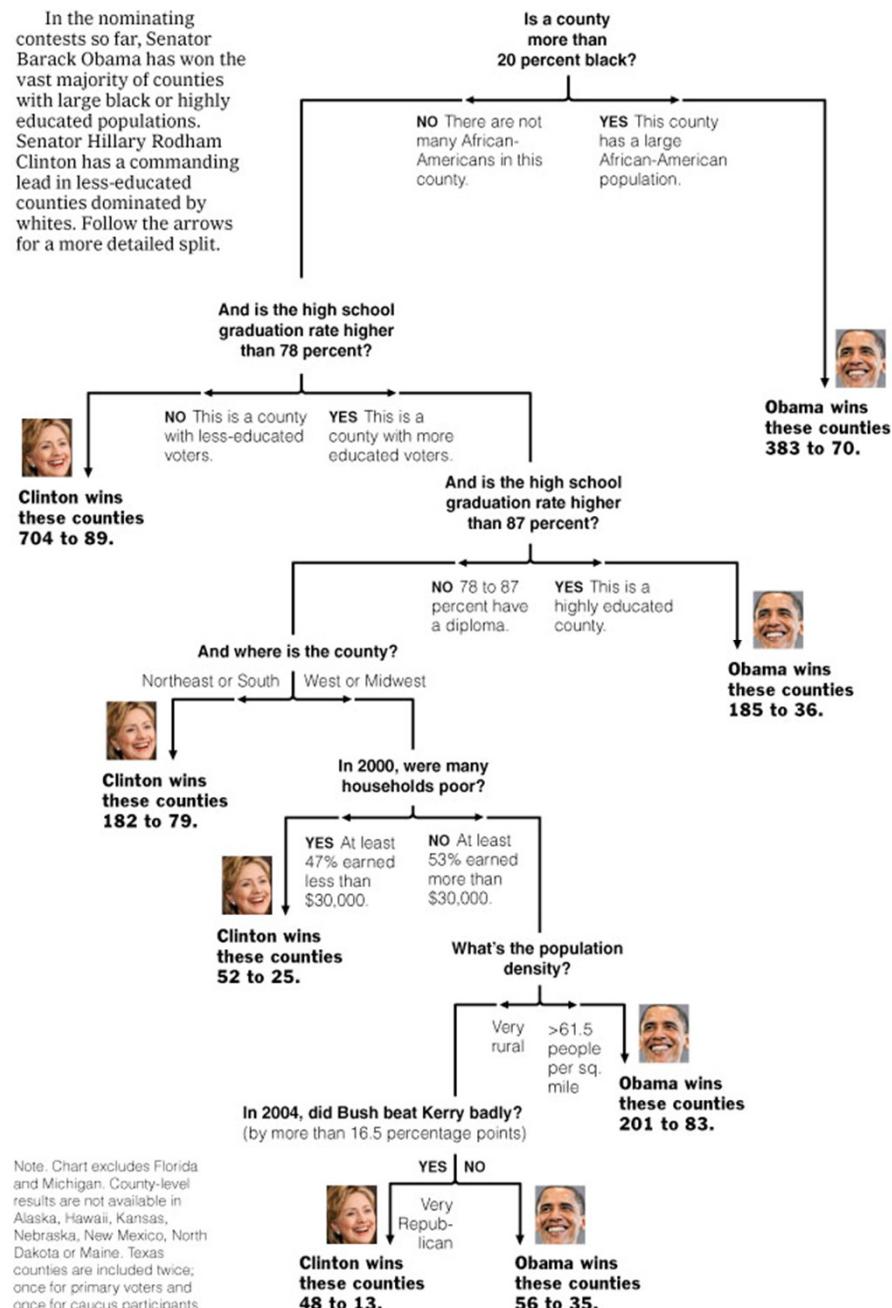
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



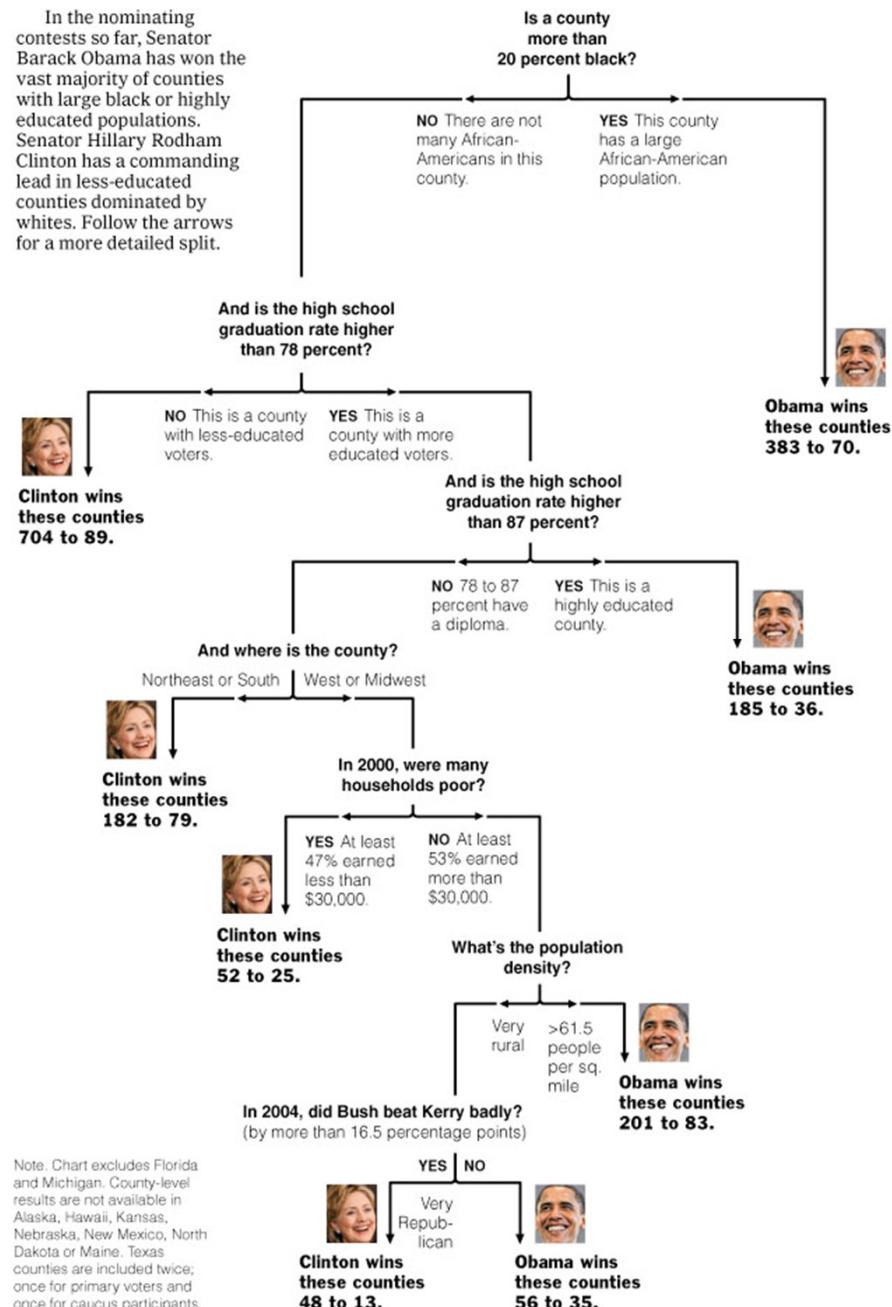
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



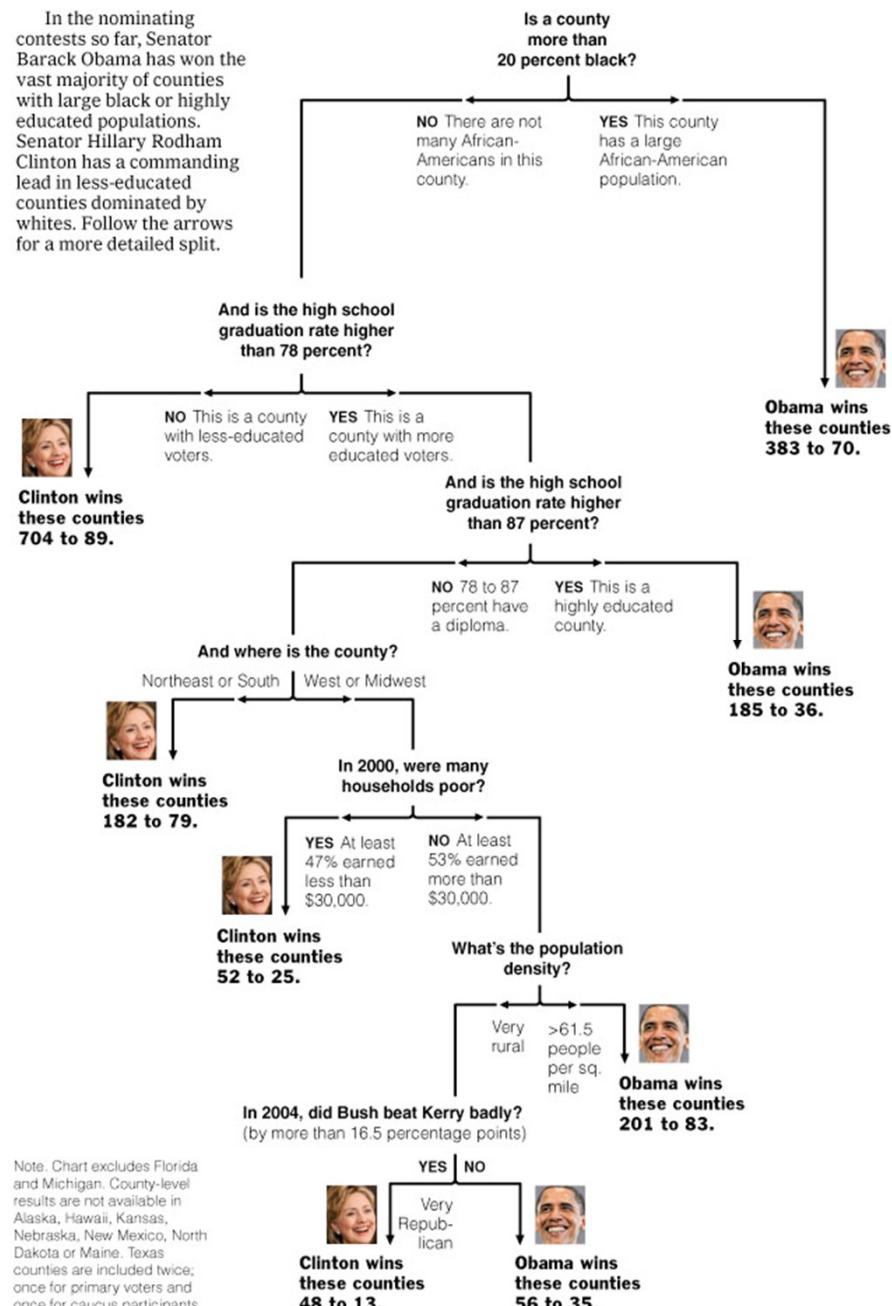
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



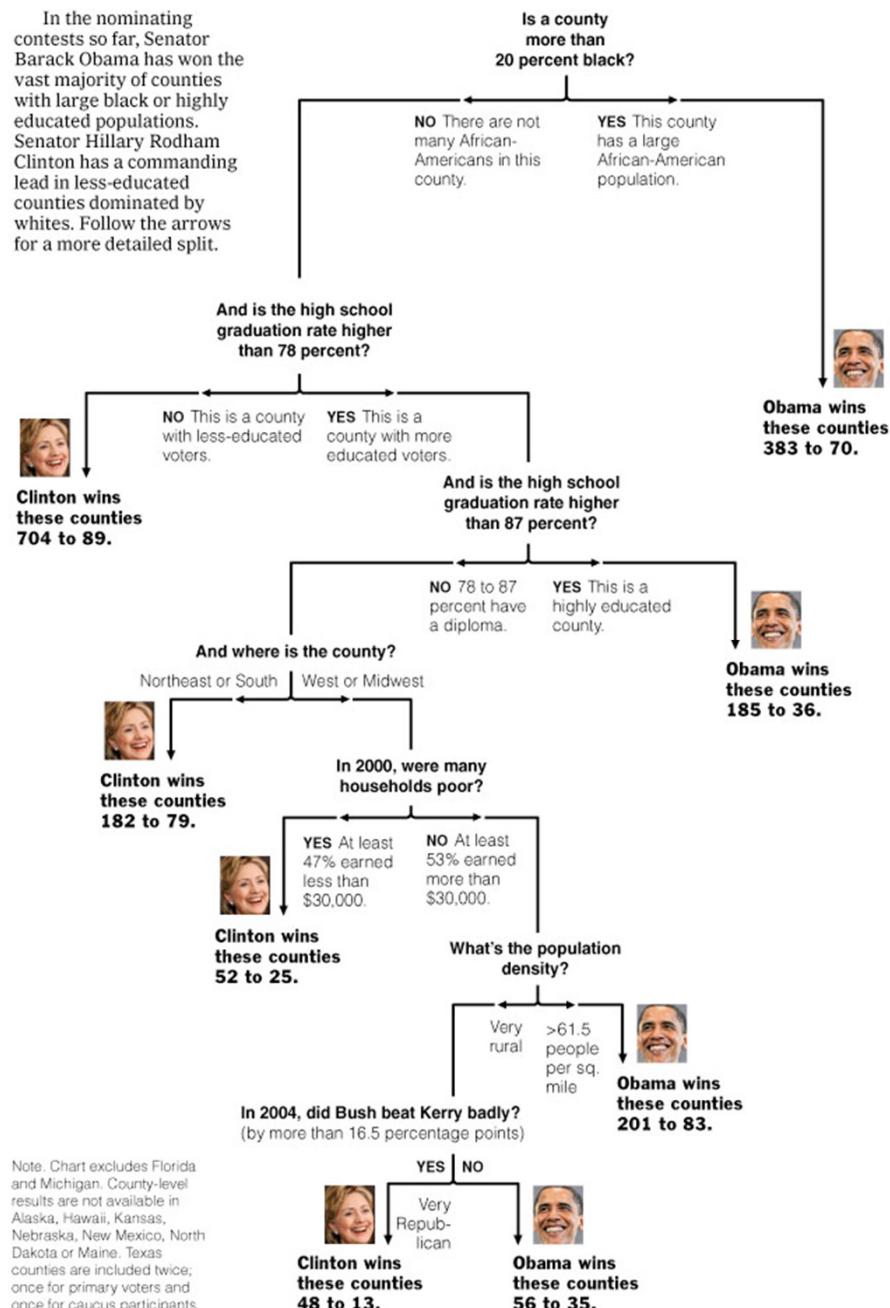
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



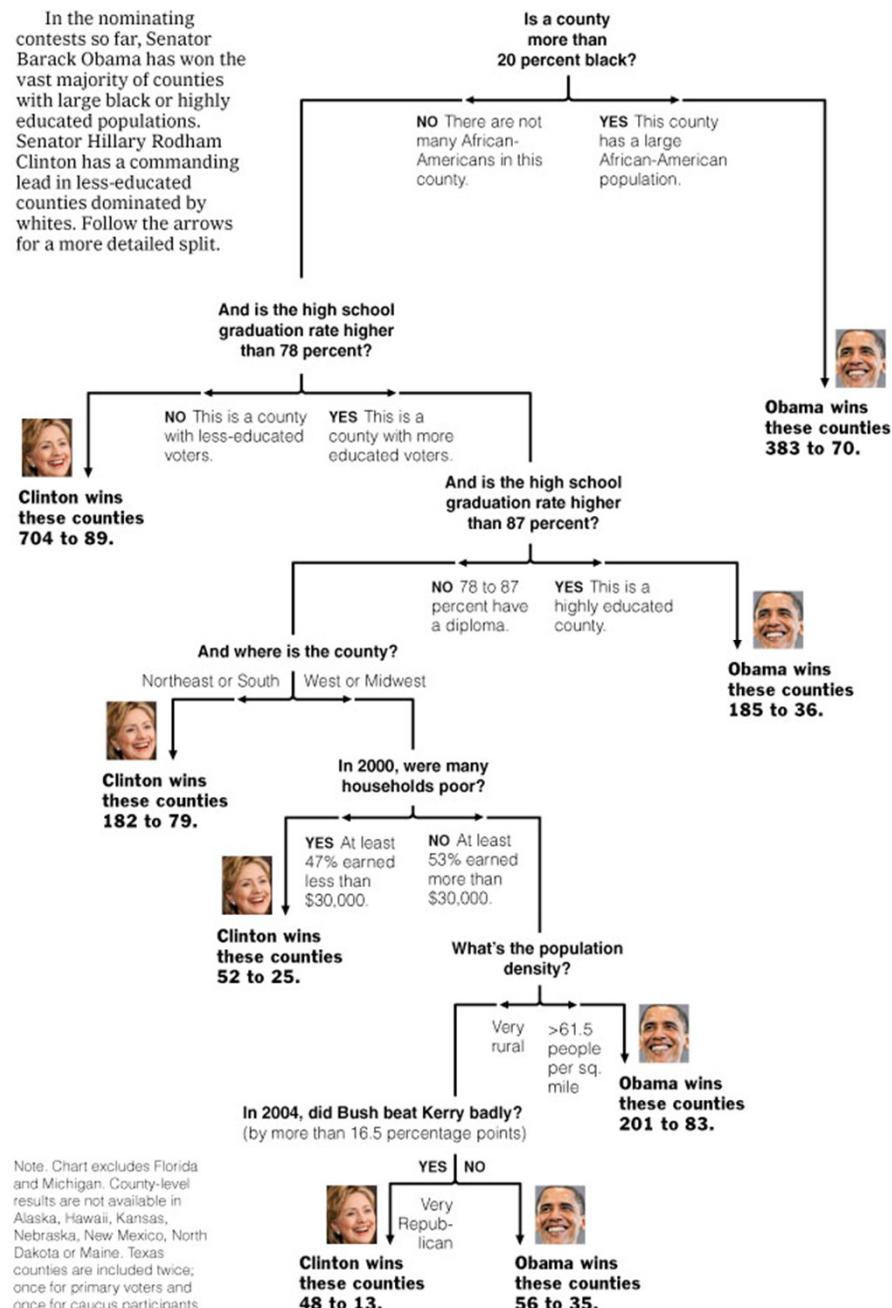
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



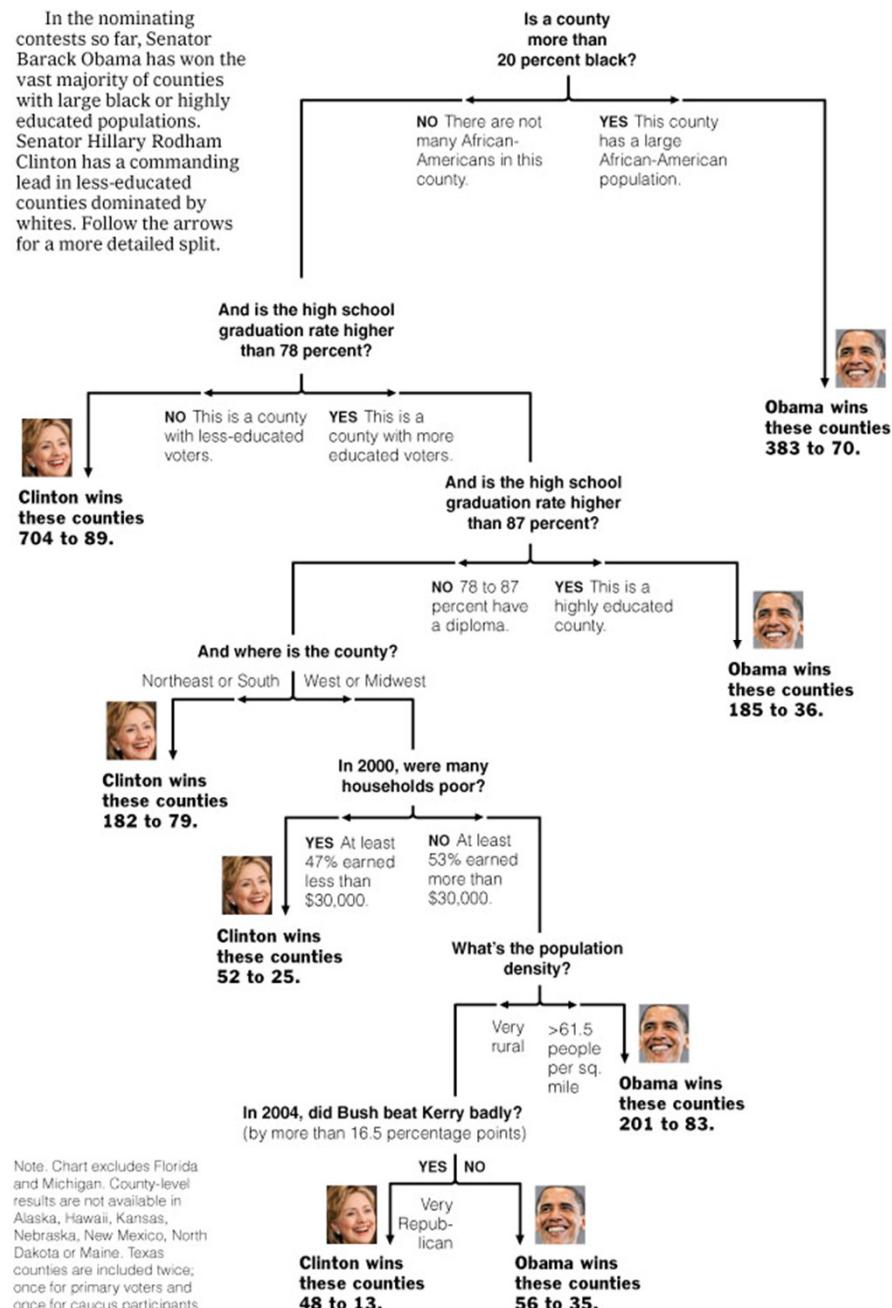
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



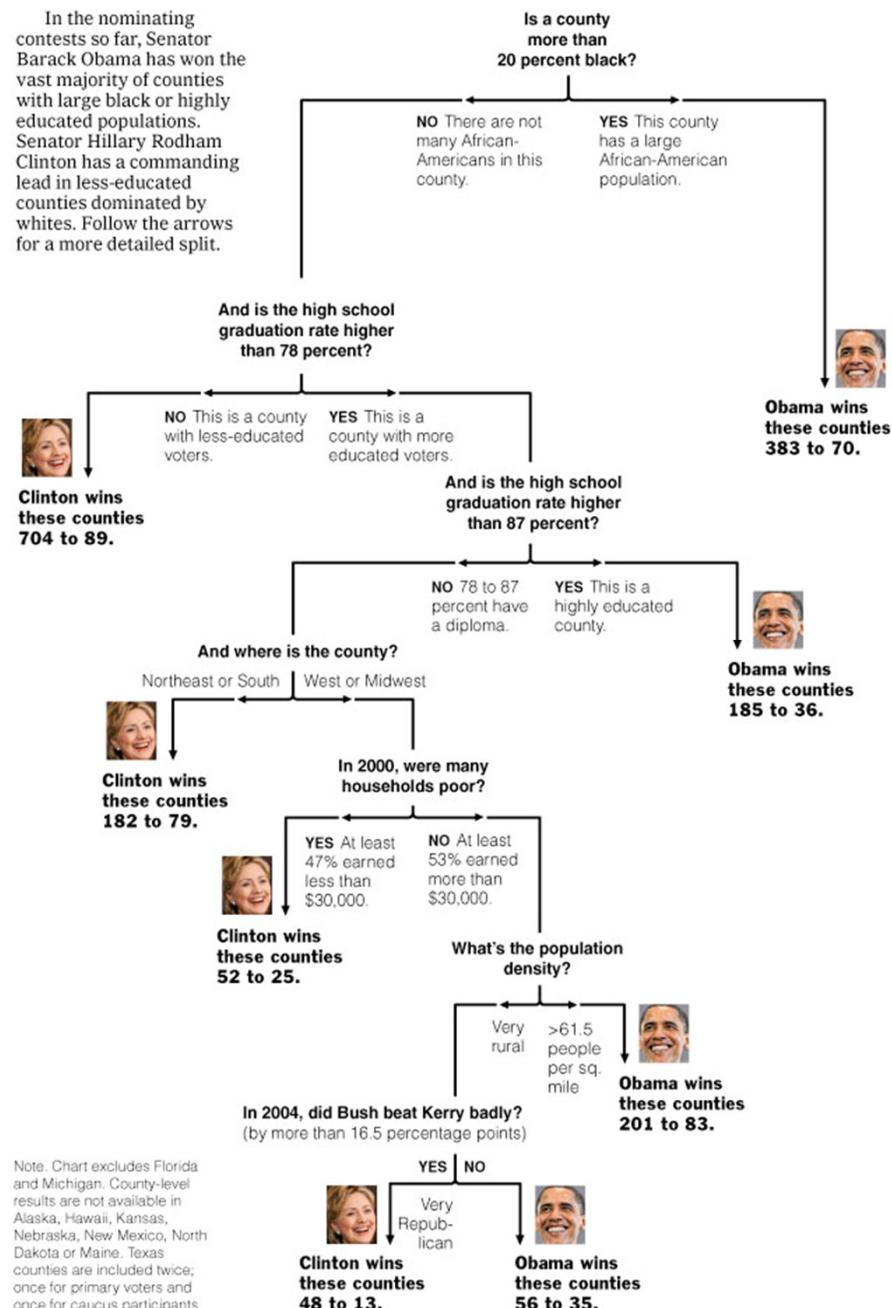
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



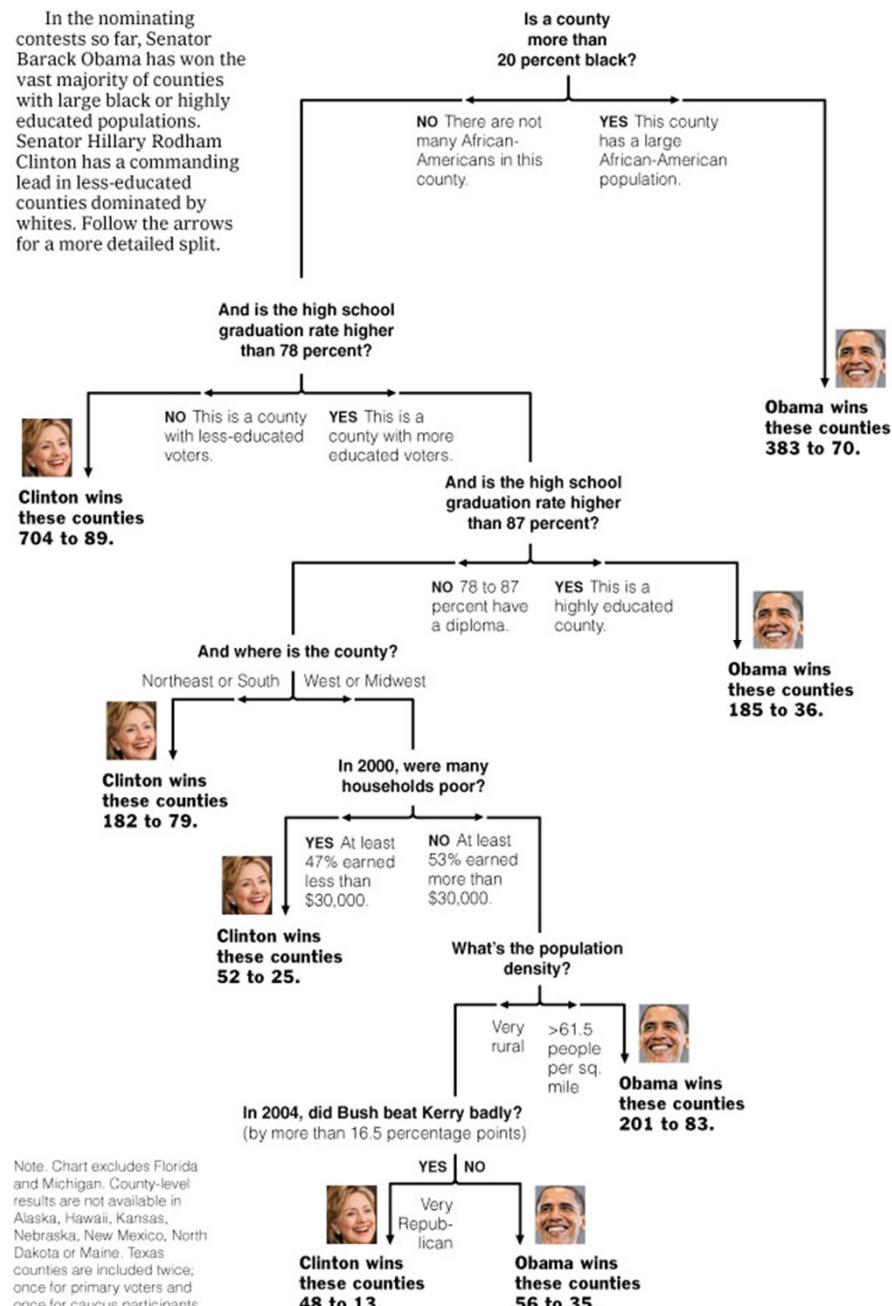
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



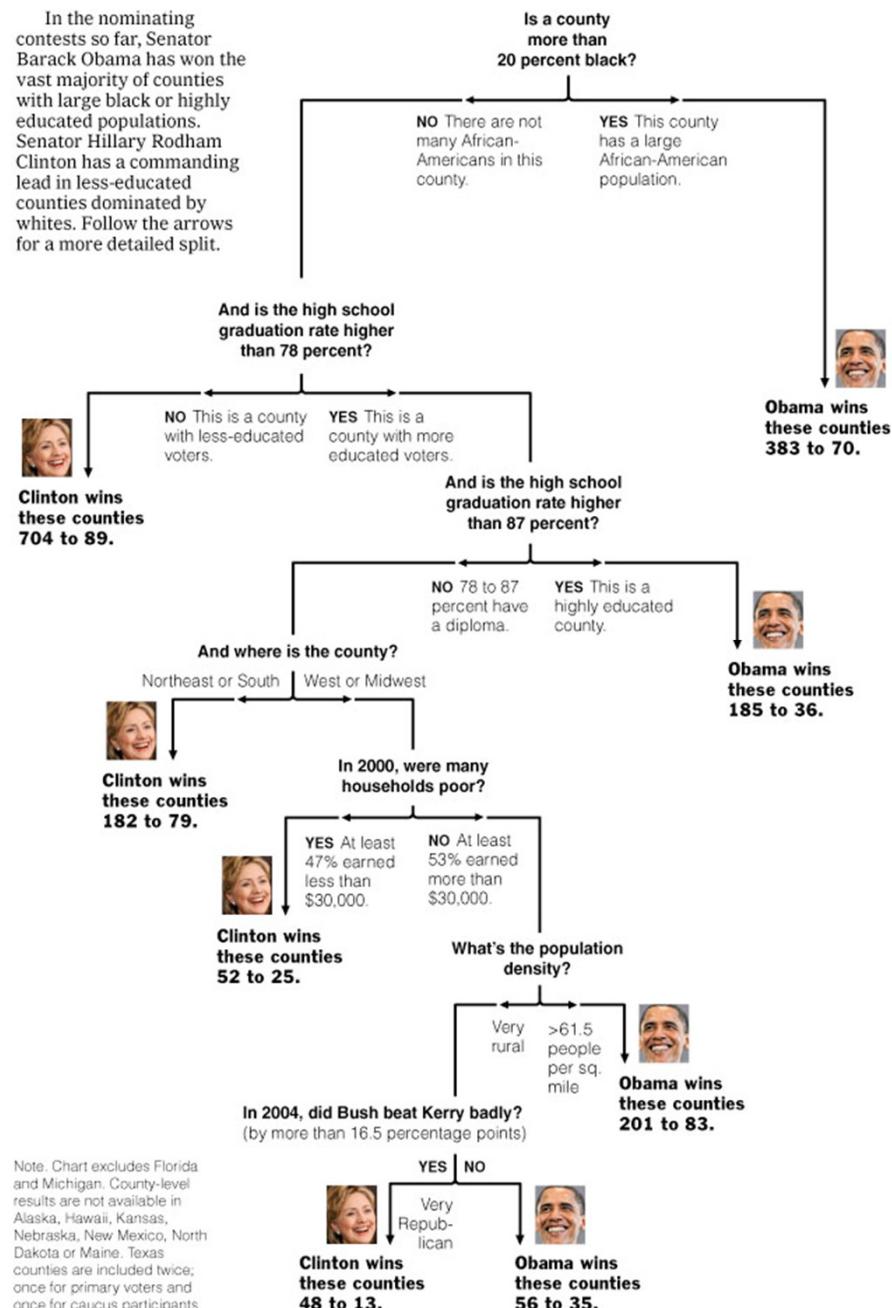
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



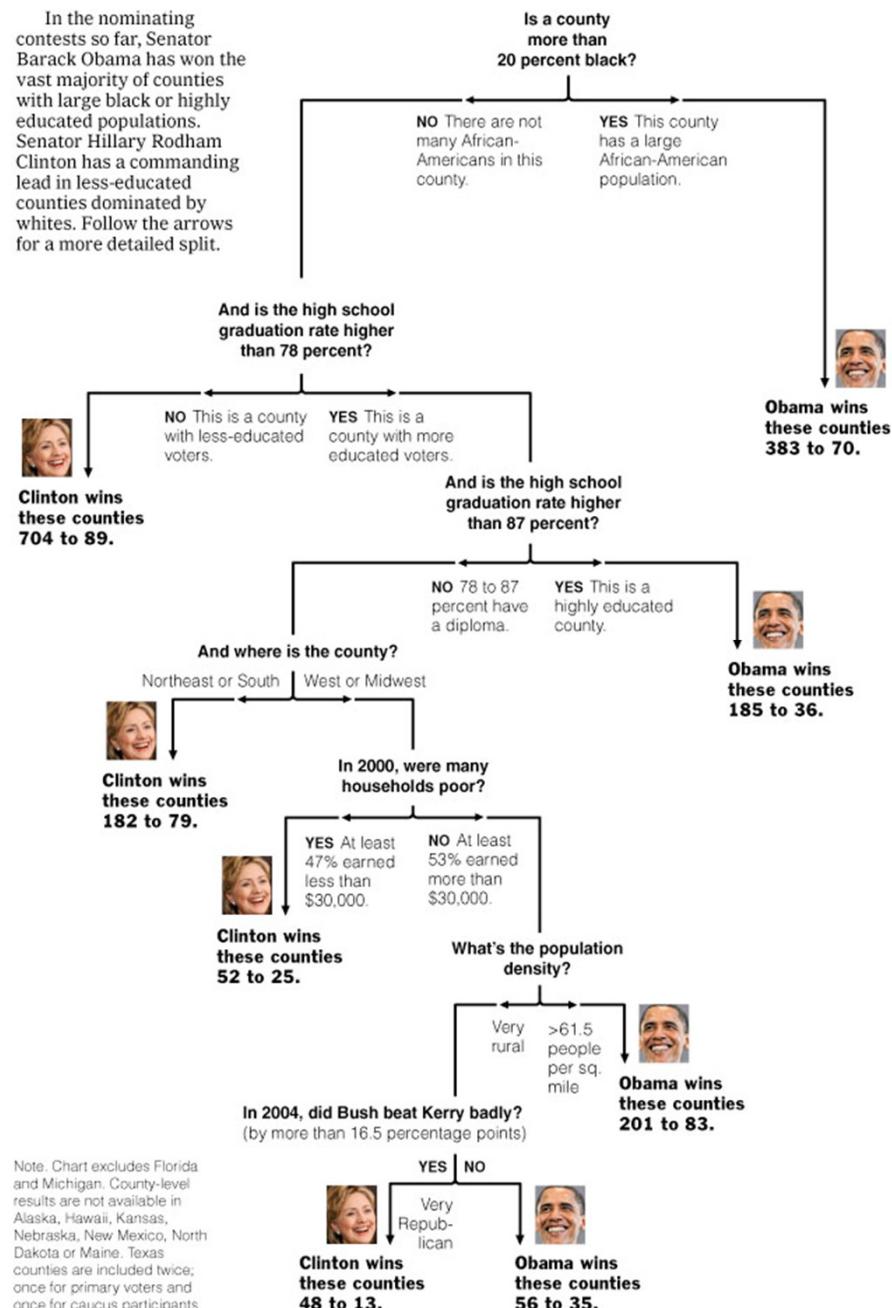
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



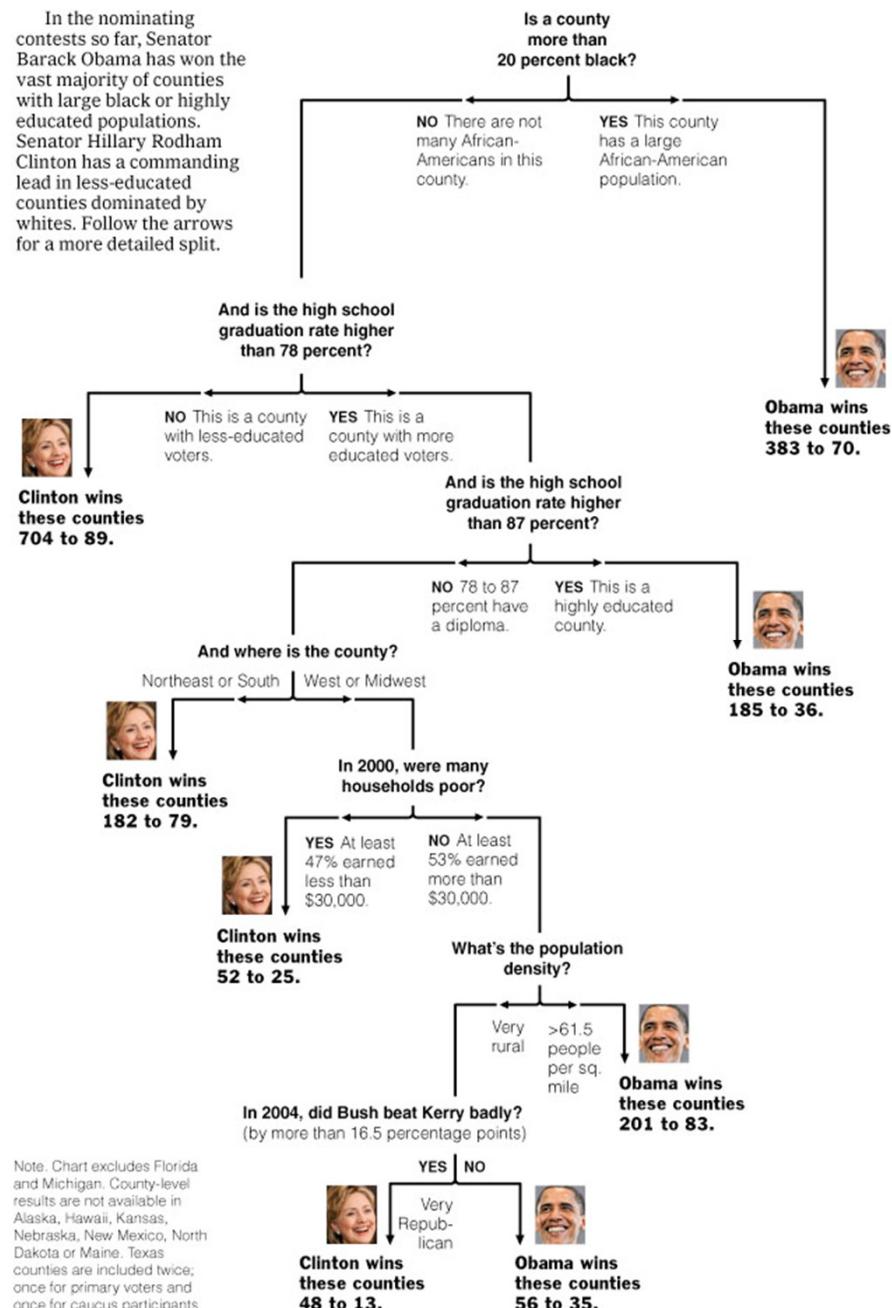
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



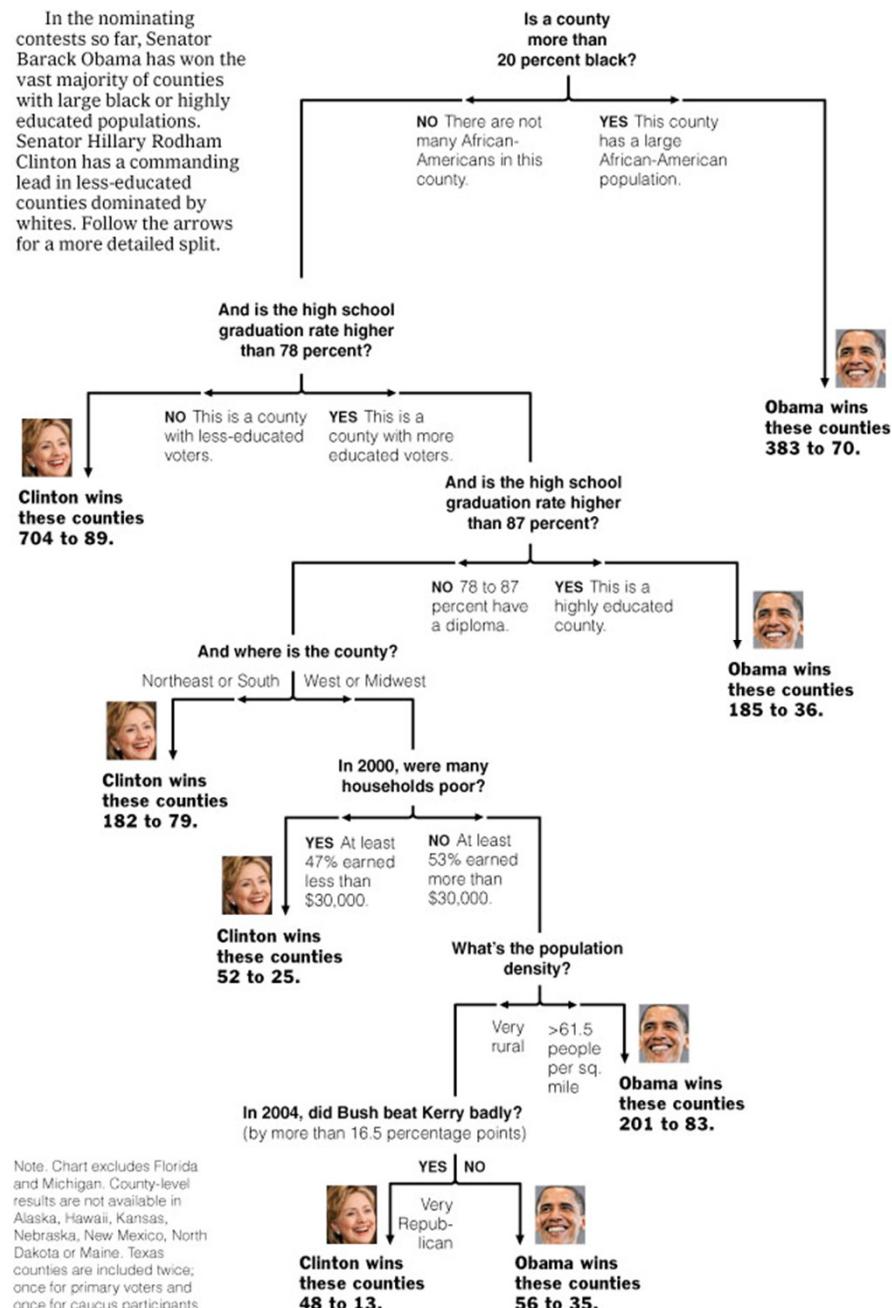
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



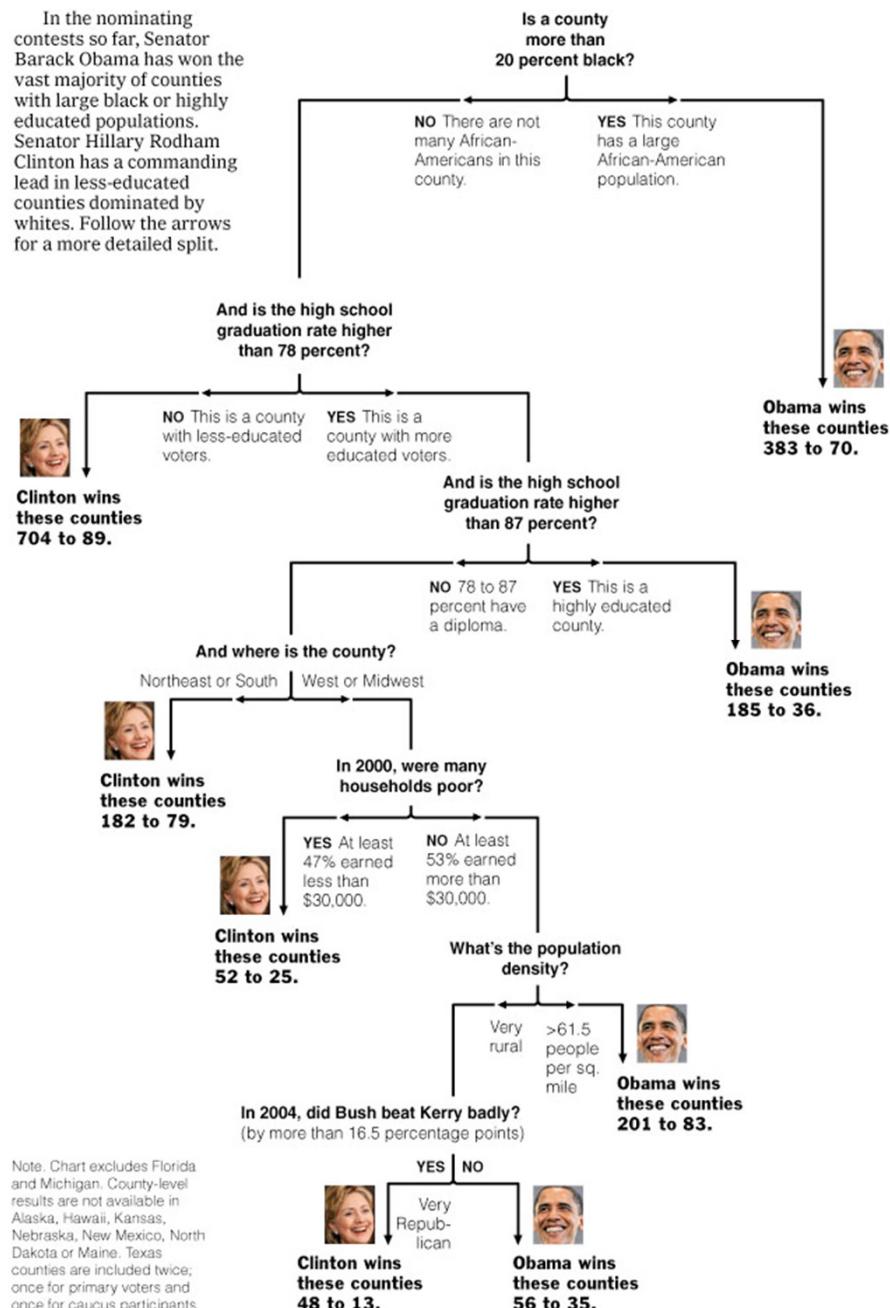
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



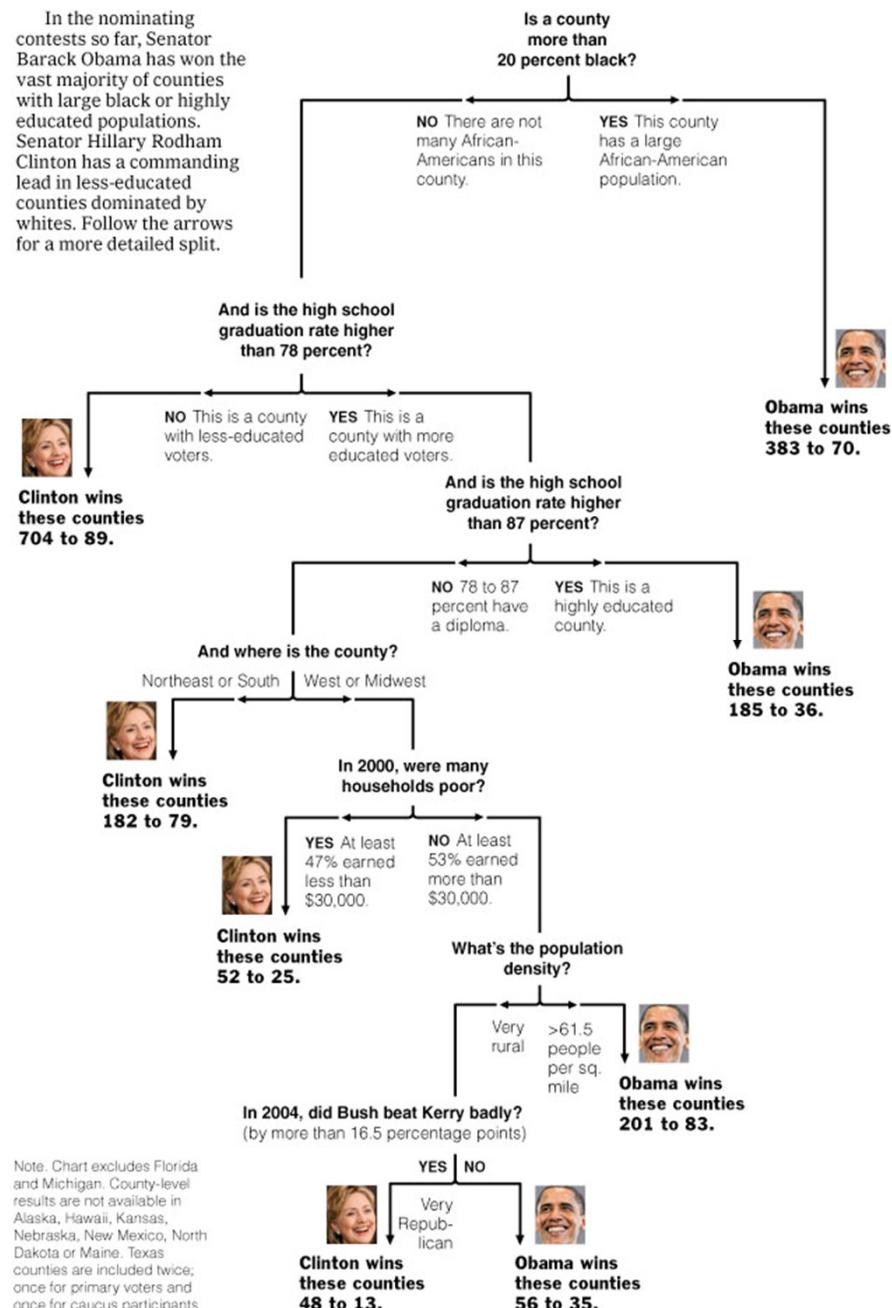
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



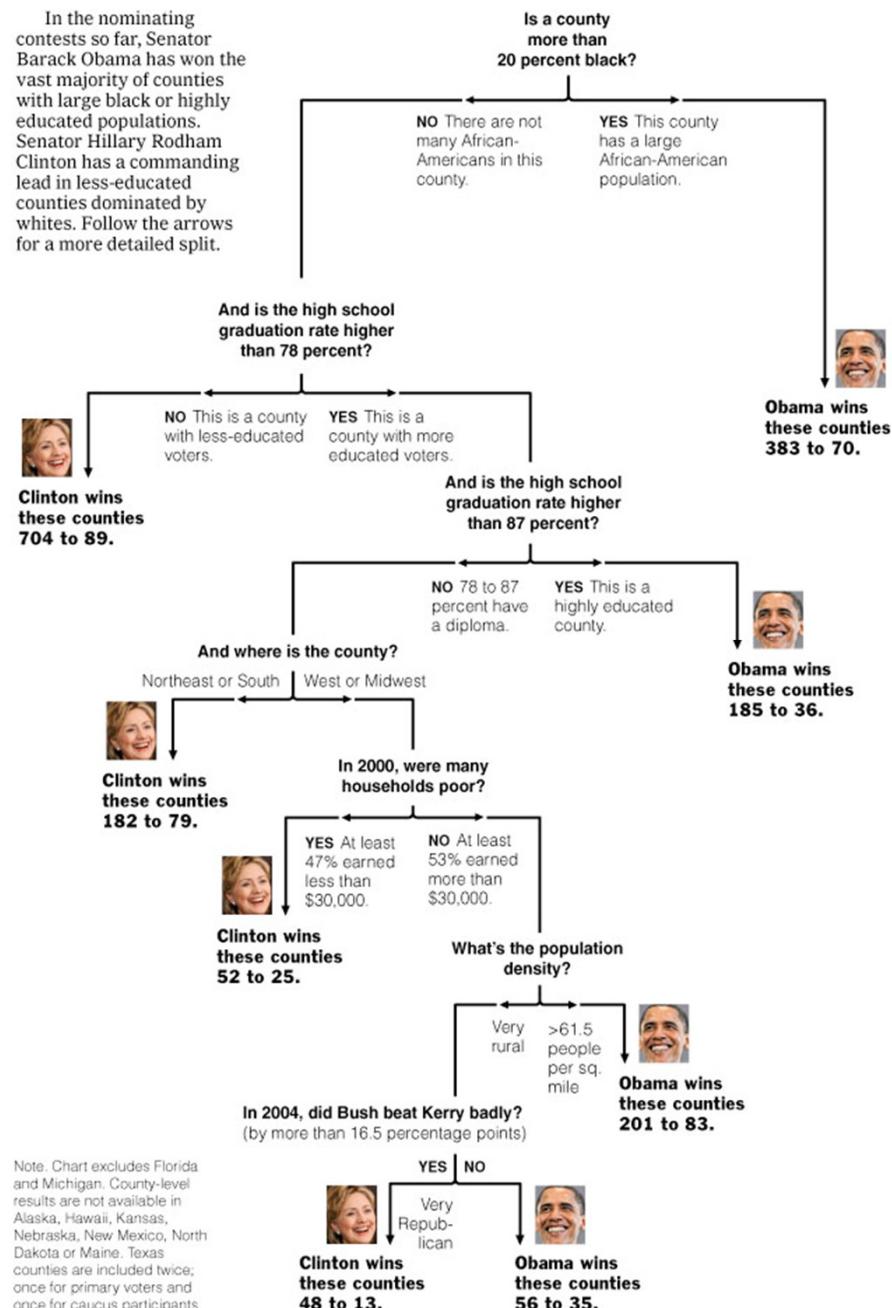
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



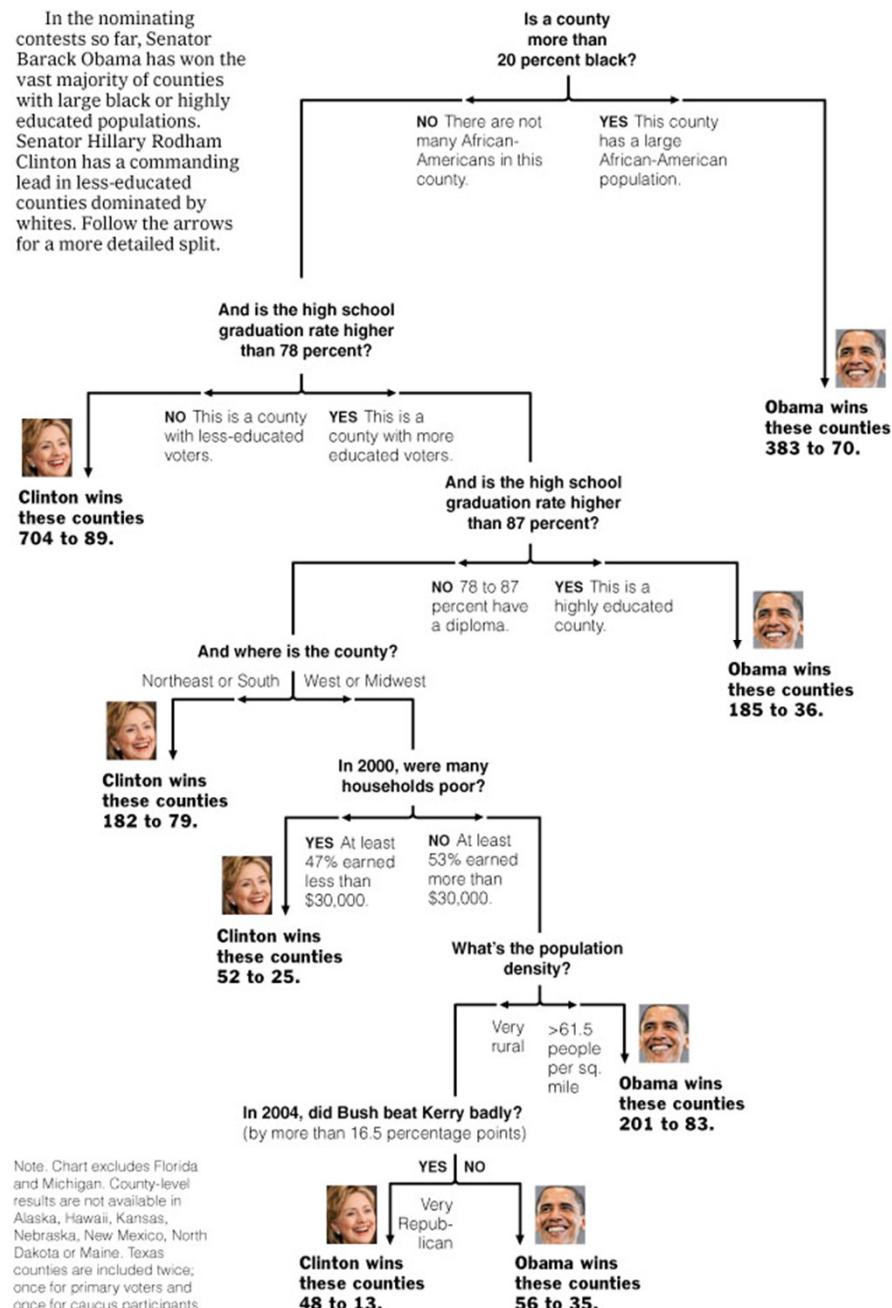
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



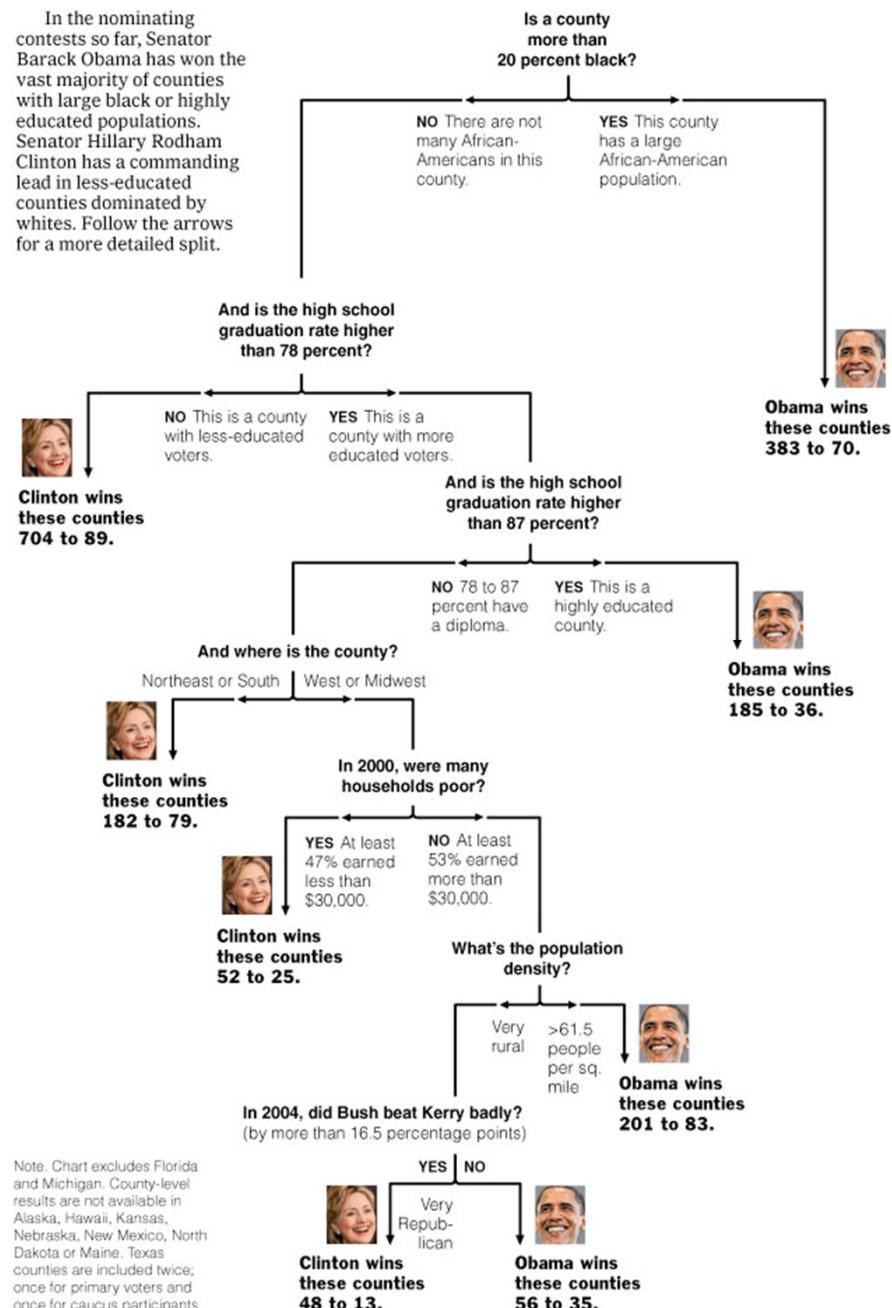
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



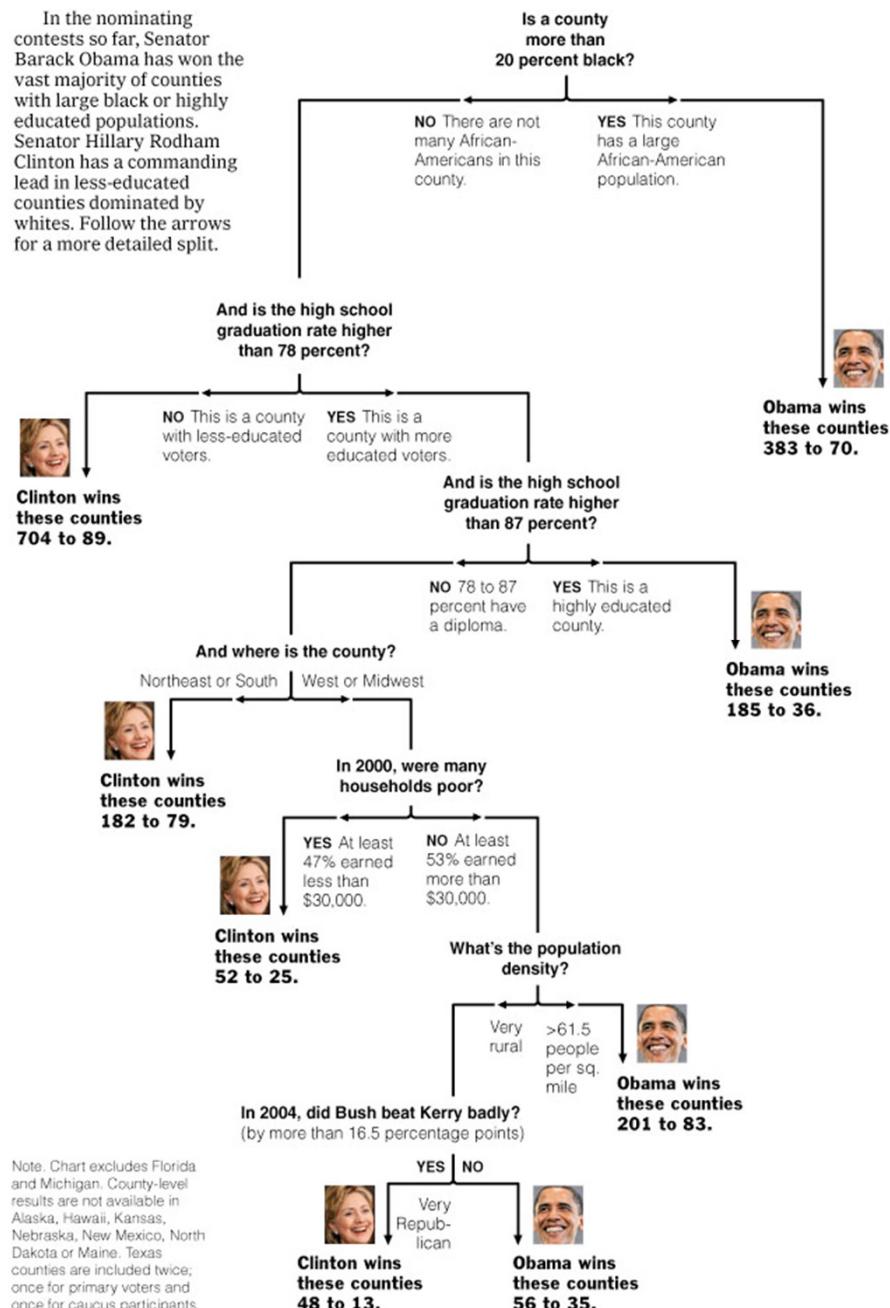
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

New York Times
April 16, 2008

Decision Trees:
 a sequence of tests.
Representation very natural
 for humans.
Style of many “How to”
 manuals and trouble-shooting
 procedures.



Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
 THE NEW YORK TIMES