Classification: Linear Models

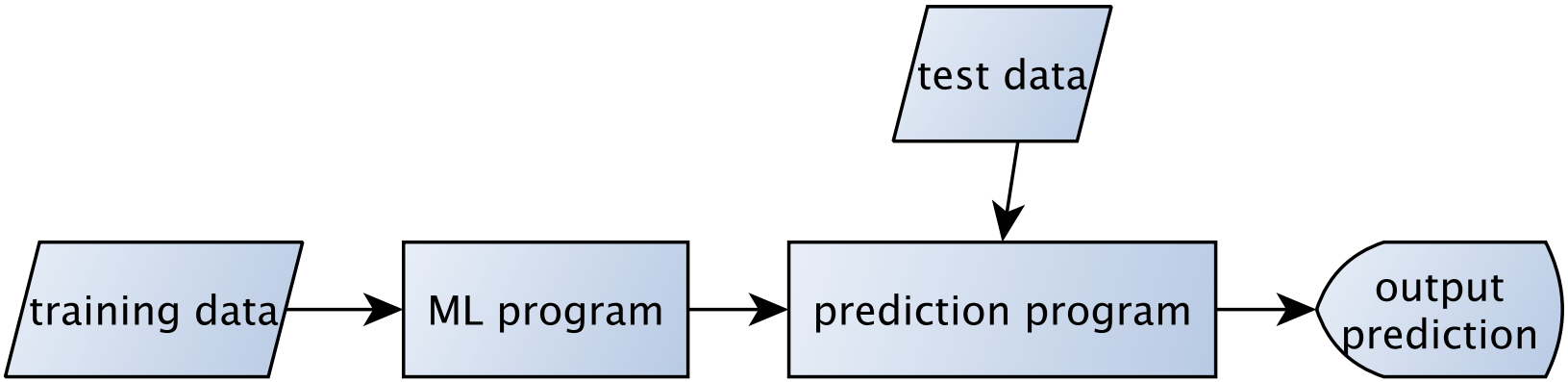
Oliver Schulte

Deep Learning

Classification Problems

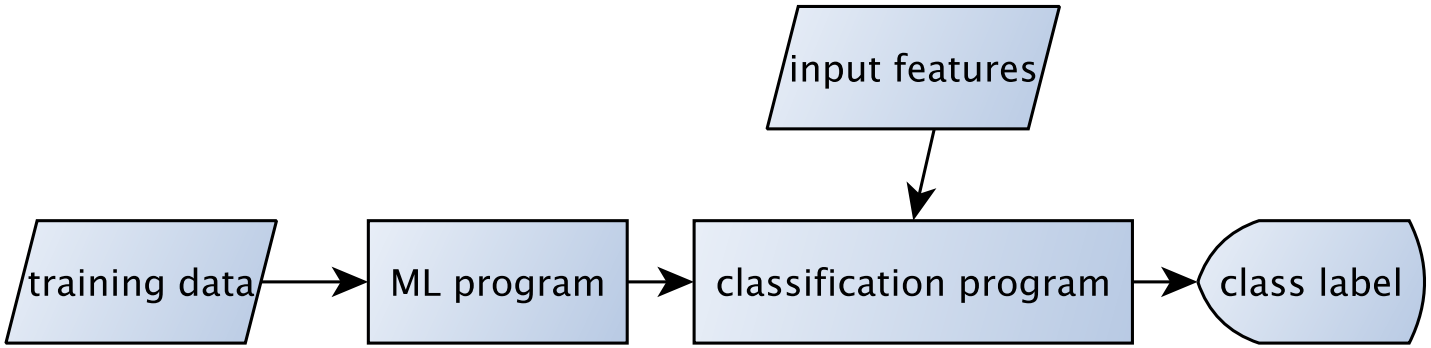
Machine Learning as Program Synthesis

- We can think of a machine learning system as a program that produces a program



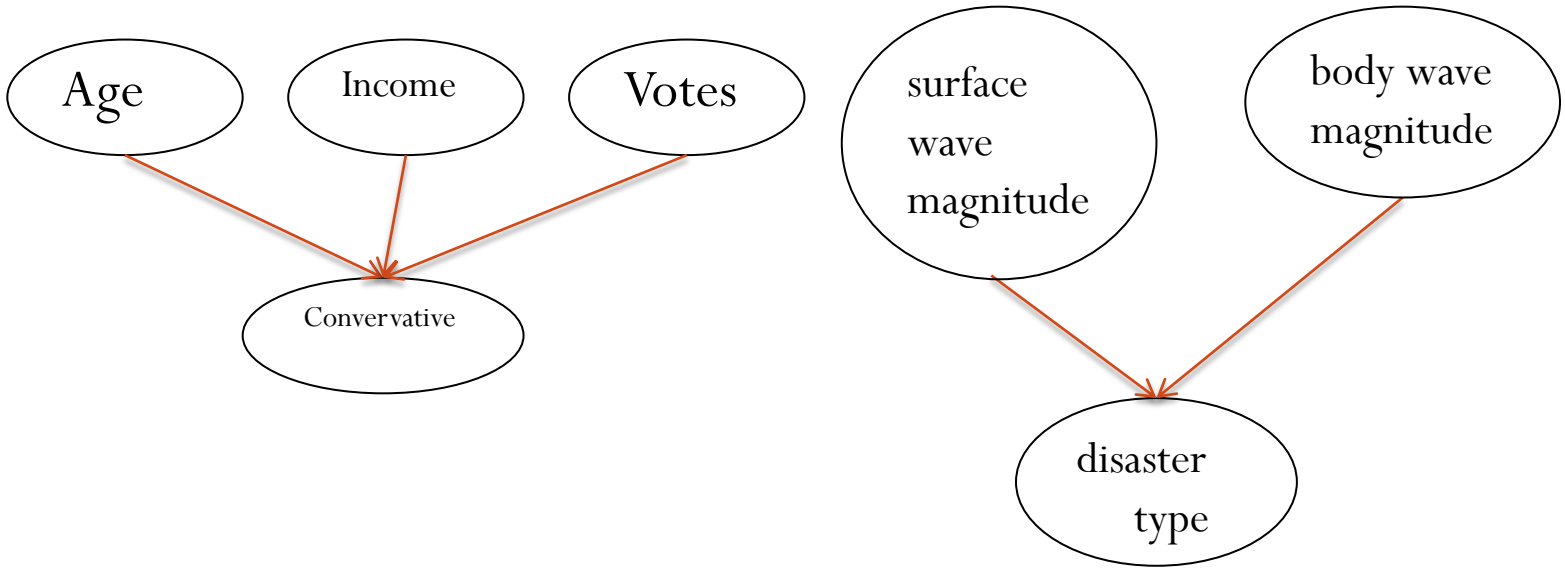
Classifying Examples

- Many predictive problems in machine learning seek to build a program with the following I/O specs.
- Input: A list/tuple/vector of **features** $x_1,..,x_m$
- Output: A discrete class label.
  - If there are only two possible classes/labels, we have a **binary** classification problem.



Examples

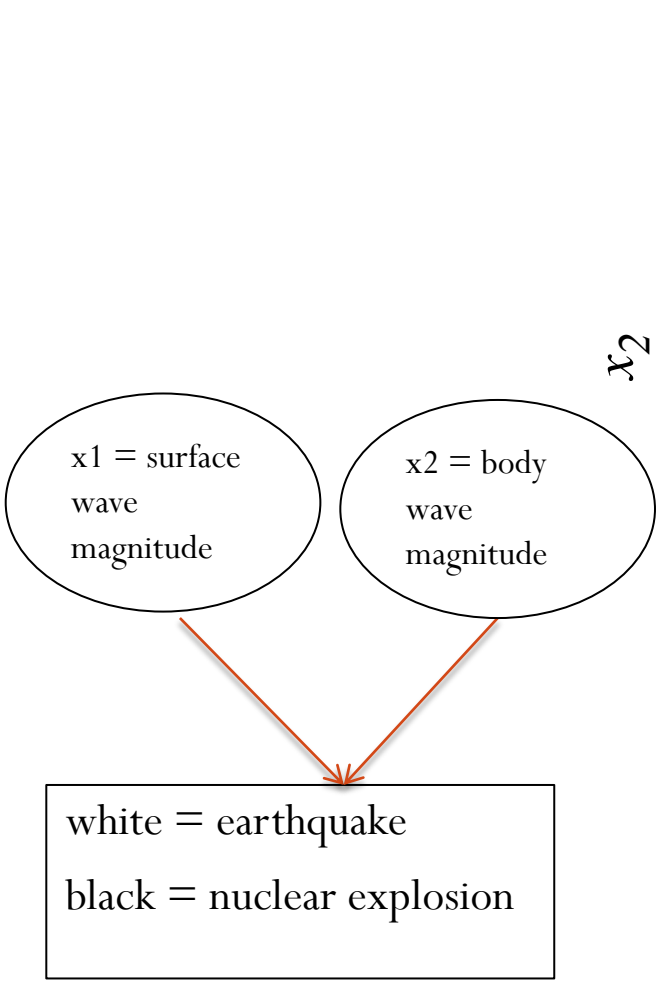
- Will the person vote conservative, given age, income, previous votes?
- Is the patient at risk of diabetes given body mass, age, blood test measurements?
- Predict Earthquake vs. nuclear explosion given body wave magnitude and surface wave magnitude.

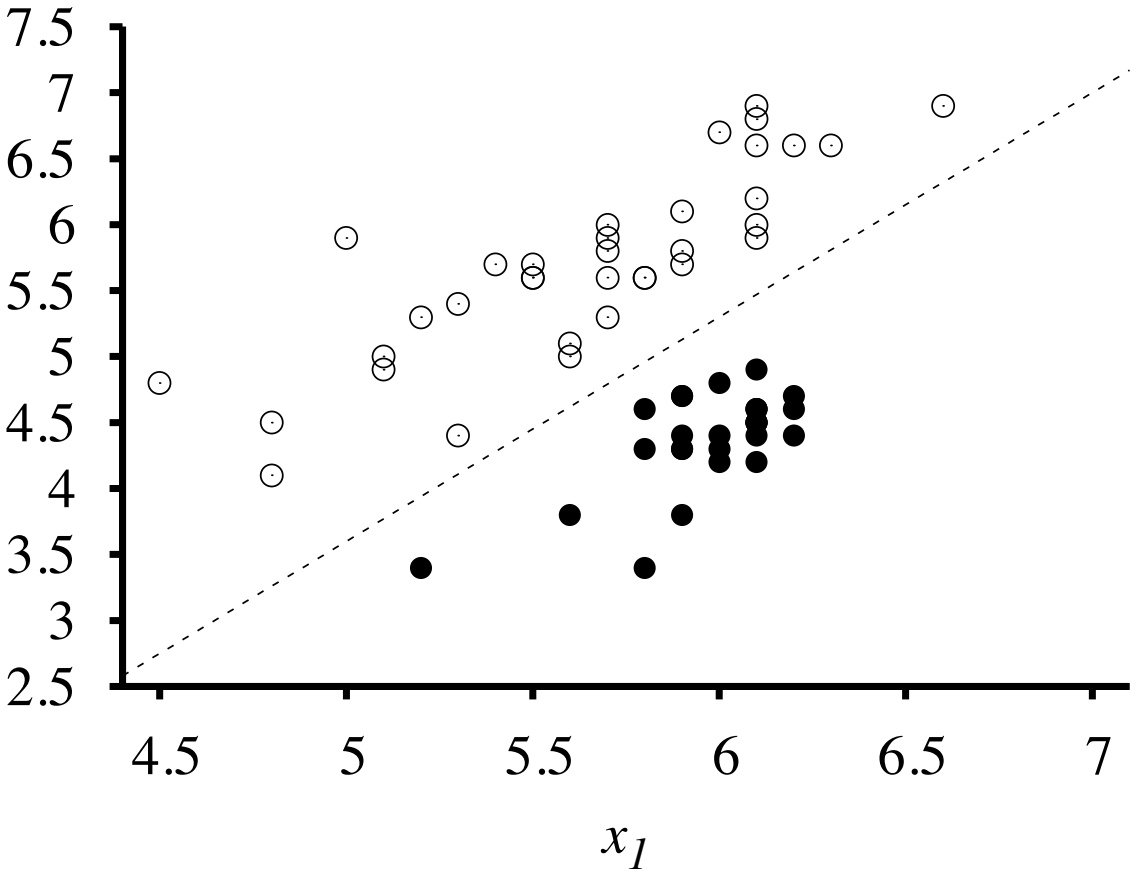


Learning to Classify

- For classification, there are many datasets of the form $(\text{input}_1, \text{output}_1), \ldots, (\text{input}_n, \text{output}_n)$
  - Mathematical notation: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$.
    - If y is discrete → classification problem
    - If y is continuous → regression problem
  - ➢ Supervised Learning: the data tell us the right answer
  - The basis of most neural net methods

Example Data

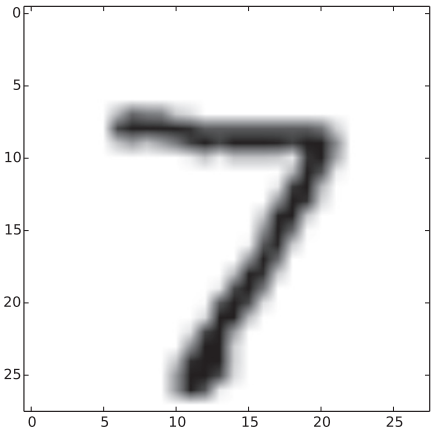




Russell and Norvig Figure 18.15

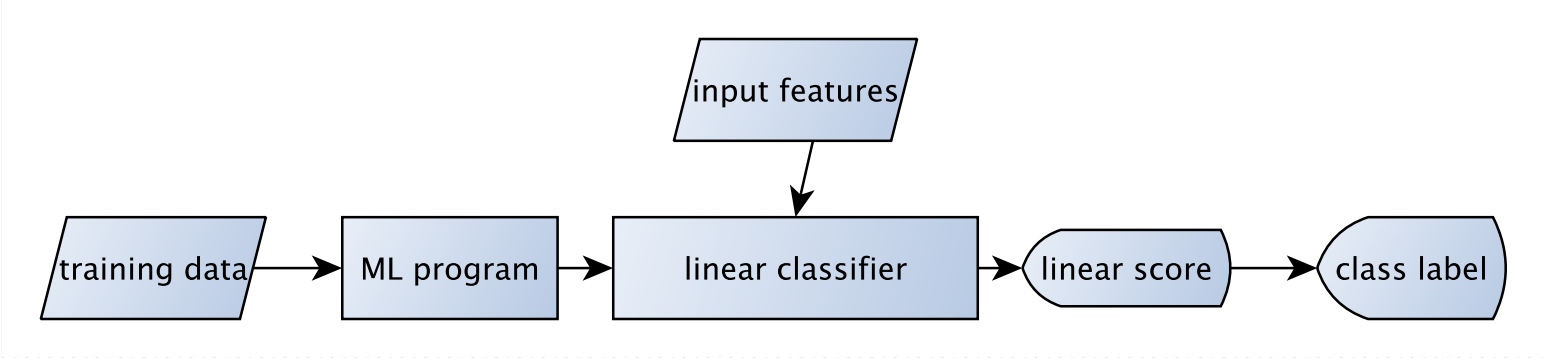
Example: Classifying Digits

- Classify image as "7" vs. "not 7".
- Represent input image as vector **x** with 28x28 =784 numbers.
- Discussion: is this representation a good idea?



	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	185	159	151	60	36	0	0	0	0	0	0	0	0	0
8	254	254	254	254	241	198	198	198	198	198	198	198	198	170
9	114	72	114	163	227	254	225	254	254	254	250	229	254	254
10	0	0	0	0	17	66	14	67	67	67	59	21	236	254
11	0	0	0	0	0	0	0	0	0	0	0	83	253	209
12	0	0	0	0	0	0	0	0	0	0	22	233	255	83
13	0	0	0	0	0	0	0	0	0	0	129	254	238	44
14	0	0	0	0	0	0	0	0	0	59	249	254	62	0
15	0	0	0	0	0	0	0	0	0	133	254	187	5	0
16	0	0	0	0	0	0	0	0	9	205	248	58	0	0
17	0	0	0	0	0	0	0	0	126	254	182	0	0	0
18	0	0	0	0	0	0	0	75	251	240	57	0	0	0
19	0	0	0	0	0	0	19	221	254	166	0	0	0	0
20	0	0	0	0	0	3	203	254	219	35	0	0	0	0
21	0	0	0	0	0	38	254	254	77	0	0	0	0	0
22	0	0	0	0	31	224	254	115	1	0	0	0	0	0
23	0	0	0	0	133	254	254	52	0	0	0	0	0	0
24	0	0	0	61	242	254	254	52	0	0	0	0	0	0
25	0	0	0	121	254	254	219	40	0	0	0	0	0	0
26	0	0	0	121	254	207	18	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0

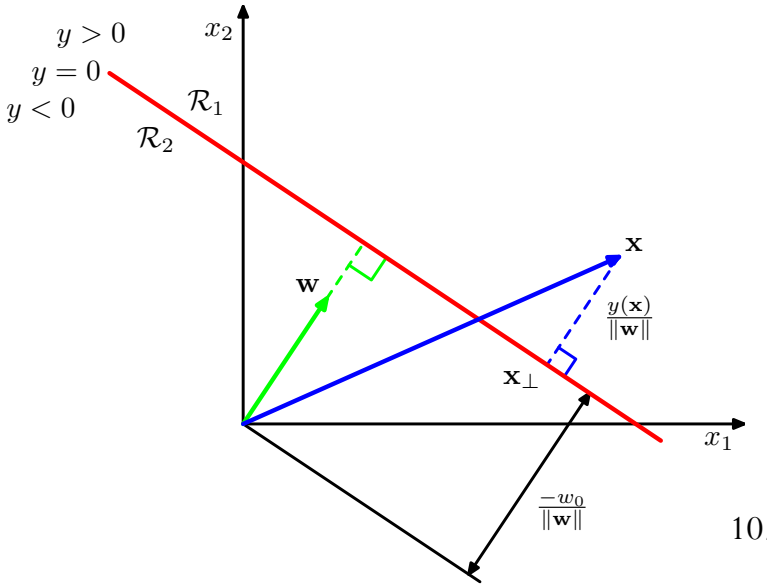
Linear Classification Models



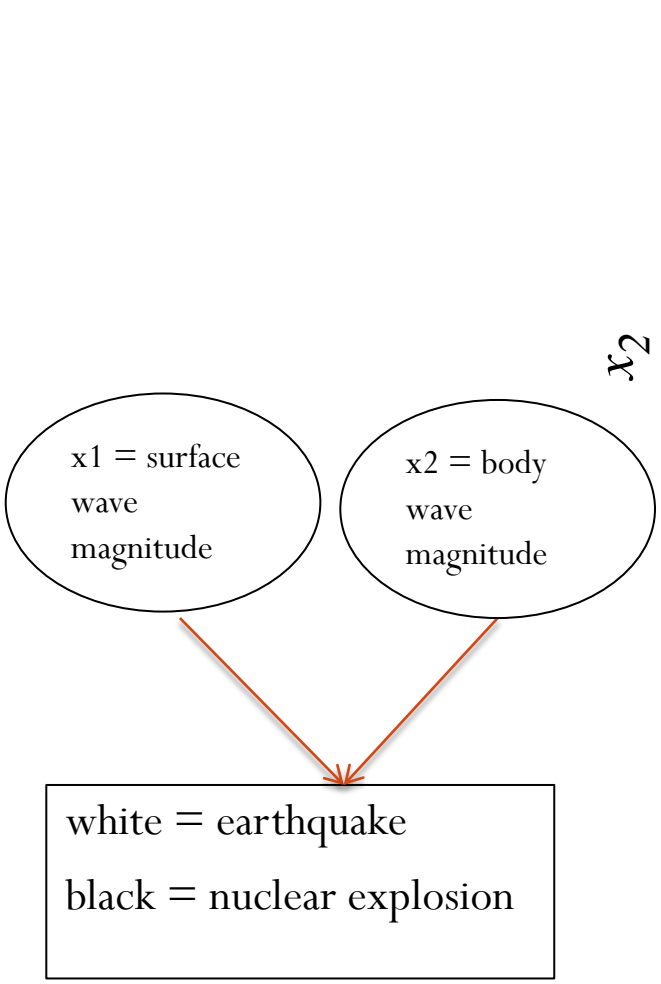
Linear Classification Models

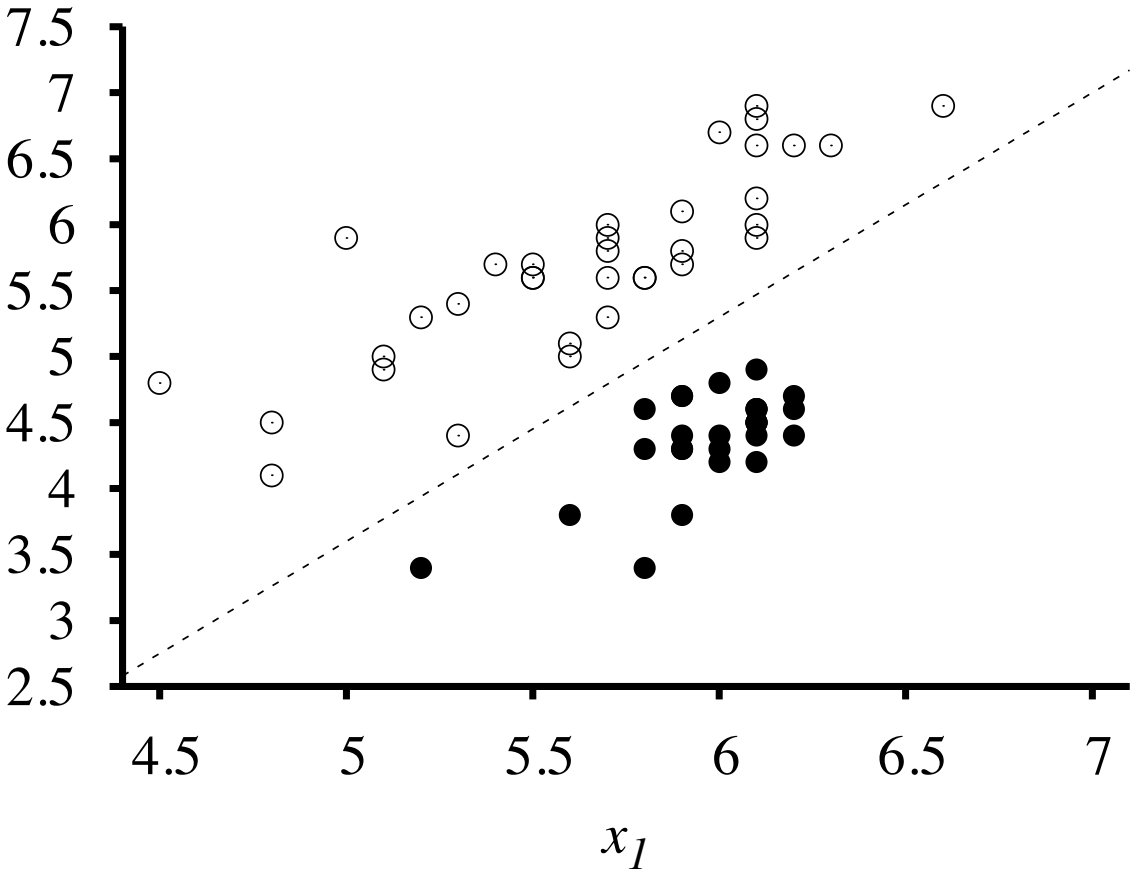
- General Idea:

1. Simple linear model $y(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$
   - **w** is a list of weight parameters to be learned
2. Classify as positive if $y(x)$ crosses a threshold, typically 0.
   - The **decision boundary** $y(x)=0$ defines a line for 2 input features, a hyperplane for > 2.



Example: Linear Separation

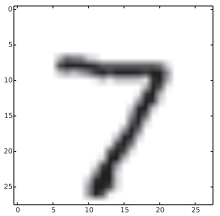




Russell and Norvig Figure 18.15

Example: Classifying Digits

- Classify input vector as "7" vs. "not 7".
- Represent input image as vector $\mathbf{x}$ with 28x28 =784 numbers.
- Target $t = 1$ for "positive", -1 for "negative".
- Prediction function $y: R^{784} \rightarrow R$.
- Classify $\mathbf{x}$ as positive if $y(\mathbf{x}) > 0$, negative o.w.
- Discussion: how can we handle multiple classes, e.g. digits from 1..10?



The Bias Weight

- Usually a linear includes a bias term (aka intercept, baseline) b.
  - E.g. $y(x_1,x_2) = \mathrm{b} + w_1 x_1 + w_2 x_2$
- Equivalently: add an imaginary constant 1 input $x_0 = 1$ and set $b = w_0$
  - E.g. $\mathrm{y}(x_0 = 1, x_1, x_2) = \mathbf{w} \cdot \mathbf{x} = w_0 x_0 + w_1 x_1 + w_2 x_2 = w_0 1 + w_1 x_1 + w_2 x_2$

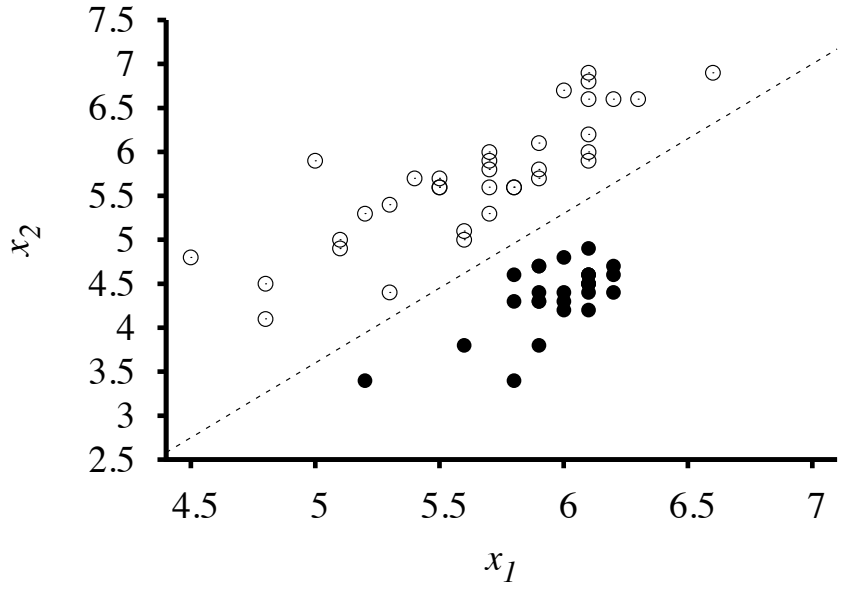
Strengths of Linear Classifiers

- Efficient to learn
- Interpretable
  - in many applications, the "effects" are most important – which features receive the biggest weight.
- Can *quantify predictive uncertainty*
  - derive confidence bounds on accuracy of predictions.
- In machine learning, interpretability and quantifying uncertainty often go together, but trade off against accuracy.
- Science manages to combine all three.

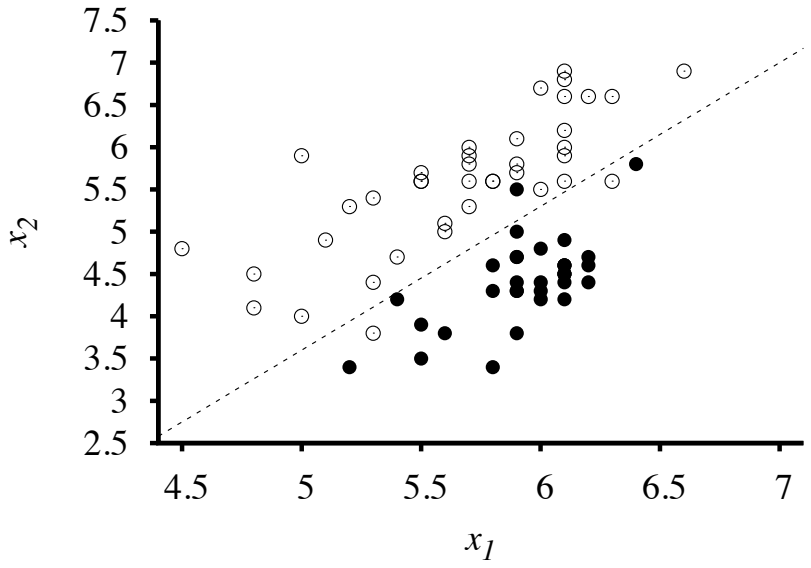
Convexity and Linear Separability

- A set of points C is convex if for any two points $\mathbf{x},\mathbf{y}$ in C, fraction $0 \leq \alpha \leq 1$ we have $\alpha\mathbf{x}+(1-\alpha)\mathbf{y}$ is also in C.
- If two classes are linearly separable, they are convex.
- Separating Hyperplane Theorem: There exists a linear separator for two disjoint sets (classes) if and only if each each set (class) is convex.

Linear Separability and Convexity



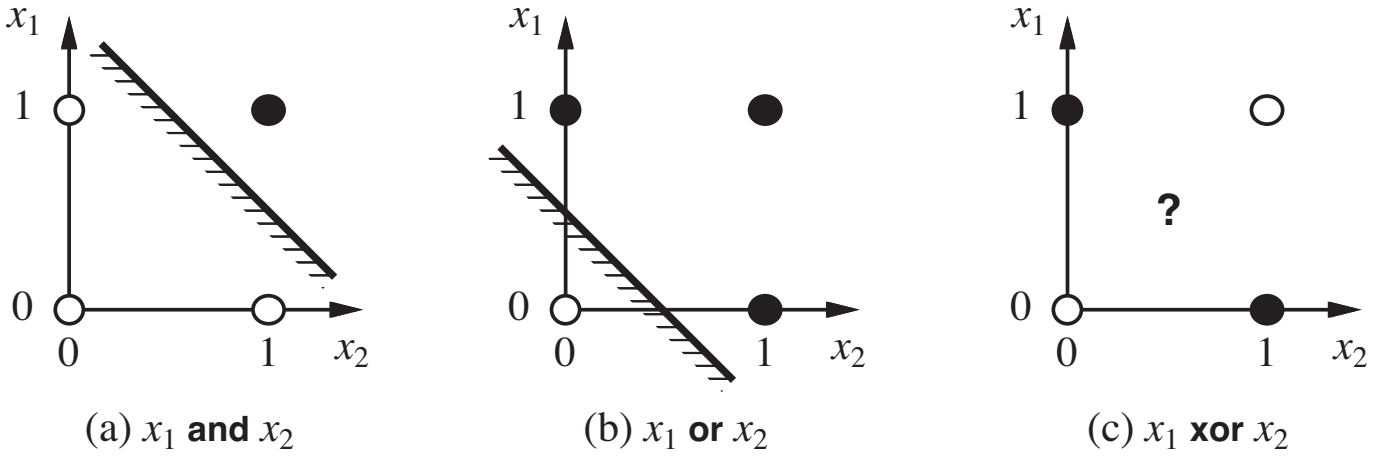
Data with outliers removed:
convex, separable



Original data:
non-convex, non-separable

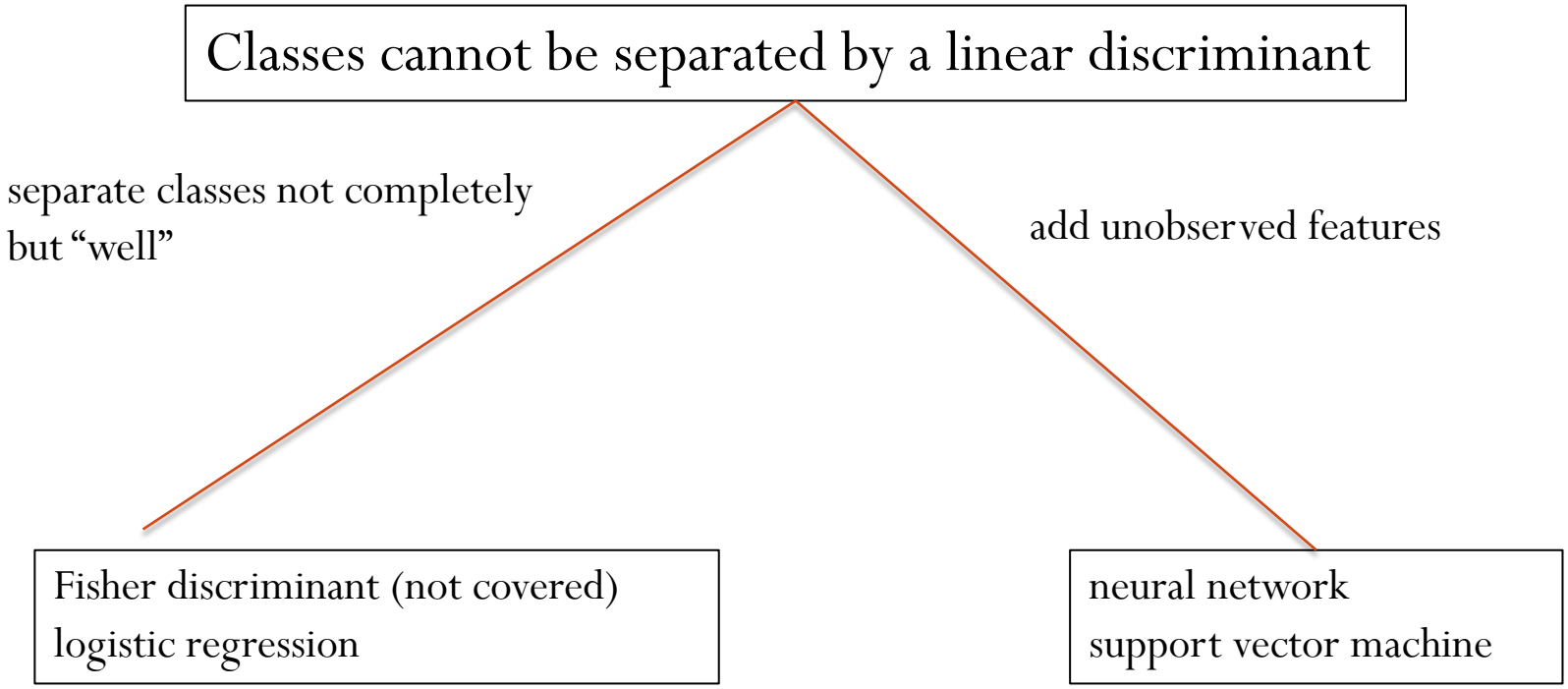
Nonseparability

- Linear discriminants can solve problems only if the classes can be separated by a line (hyperplane).
- Canonical example of non-separable problem is X-OR.
- Illustration



(a) x_1 **and** x_2 (b) x_1 **or** x_2 (c) x_1 **xor** x_2

Responses to Nonseparability



Gradient Descent Learning

Learning a Linear Classifier

- Most learning for continuous data follows a decision-theoretic (Bayesian) approach to learning.

1. For a given data set D, define an **error function** E(**w**,D) that measures how well the weights **w** fit the data D.
2. Find **w** that minimize the error for a given input data set D.
3. In neural net learning, the basic minimization algorithm is **gradient descent**.
   Illustration

Gradient Descent: Choosing a direction

- Intuition: think about trying to find street number 1000 on a block. You stop and see that you are at number 100. Which direction should you go, left or right?
- You initially check every 50 houses or so where you are. What happens when you get closer to the goal 1000?
- The fly and the window: the fly sees that the wall is darker, so the light gradient goes down: bad direction.
- See here for illustration (around 38 sec)

Gradient Descent Scheme

1. Initialize weight vectors somehow.
   - Typically randomly, more on this later.
2. Update (with learning rate η)

$$w_{i+1} := w_i - \eta \frac{\partial E}{\partial w_i}$$

3. Until some convergence criterion is true

- Computational Complexity comment: assuming gradients are easy to compute, *every update step is linear in the number of weights.*

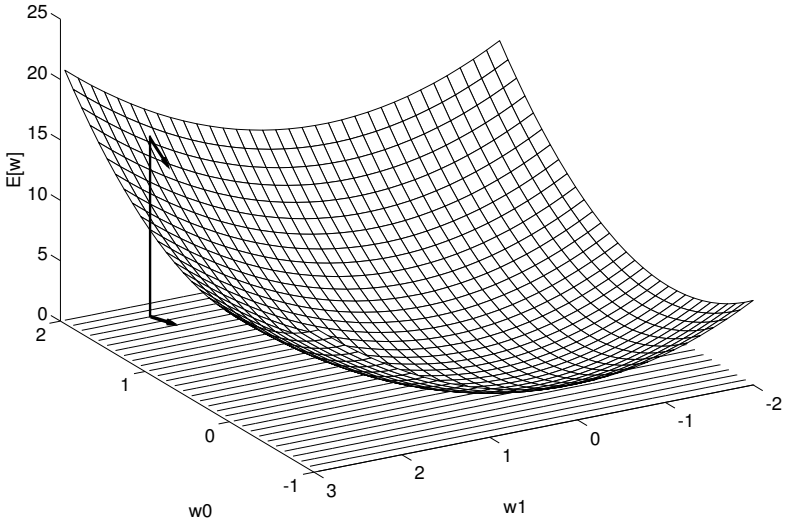
➢ *Learning time depends on number of steps required until convergence*

- Usually impossible to predict in advance

Gradient Descent in One Dimension

- Update Rule
  $\mathbf{x := x - \eta f'(x)}$ where η is a step size
  Example:
- Try to find y that minimizes $f(y) = y^2$.
- Your current location is $y = -3$.
- What is f'(y)?
- Answer: the derivative function is $2y$.
- Evaluated at the location -3, the derivative is $\nabla = -6$.
- To minimize, we move in the opposite direction $-\nabla$.
- Letting the step size $\eta = 1$, your new location is
  $-3 - (-6) = -3+6=3$

Gradient Descent In Multiple Dimensions



Gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Gradient Descent: Example.

- Try to find x,y that minimize
  $f(x,y) = 3x + y^2$.
- Your current location is $x = 10$, $y = -3$.
- What is $\frac{\partial f}{\partial x}$? $\frac{\partial f}{\partial y}$?
- Answer: the gradient vector is *(3, 2y)*.
- Evaluated at the location (10,-3), the gradient is $\nabla = (3, -6)$.
- To minimize, we move in the opposite direction $-\nabla$.
- Letting the step size $\eta = 1$, your new location is
  $(10,-3) - (3,-6) = (7, 3)$.

Gradient Descent: Exercise

- Try to find x,y that minimize
  $f(x,y) = 3x + y^2$.
- Your current location is $x = 7$, $y = 3$.
- The gradient vector is *(3, 2y)*.
- Letting the step size $\eta = 1$, what is your new location?
- Demos
  - Visualization
  - Linear classifier illustration

Perceptron Learning

Defining an Error Function

- General idea:

1. Encode class label using a real number t for target.
   - e.g., "positive" = 1, "negative" = 0 or "negative" = -1.
   - This is the first example we see of *embedding*.
2. Measure error or **loss** by comparing continuous linear output y and class label code t.
3. Obvious loss function is 0-1:
   0 if prediction correct, 1 otherwise.
4. In practice use *convex* upper bounds on 0-1 loss:
   $loss(y,t) \geq 0$ if prediction correct
   $loss(y,t) \geq 1$ if prediction false
5. Error function is sum of losses:

$$E(\boldsymbol{w}, D) = \sum_j loss\left(y(\boldsymbol{x}_j), t_j\right)$$

   where
   1. j=1,…N indexes data points with target labels t_j
   2. $y(\mathbf{x}_j)$ is the model output for data point $\mathbf{x}_j$ (the output depends on weights $\mathbf{w}$)

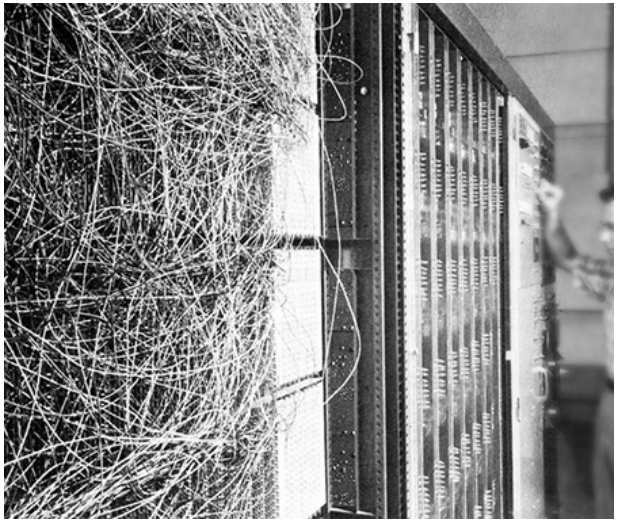
Dataset Error vs. Datapoint Loss

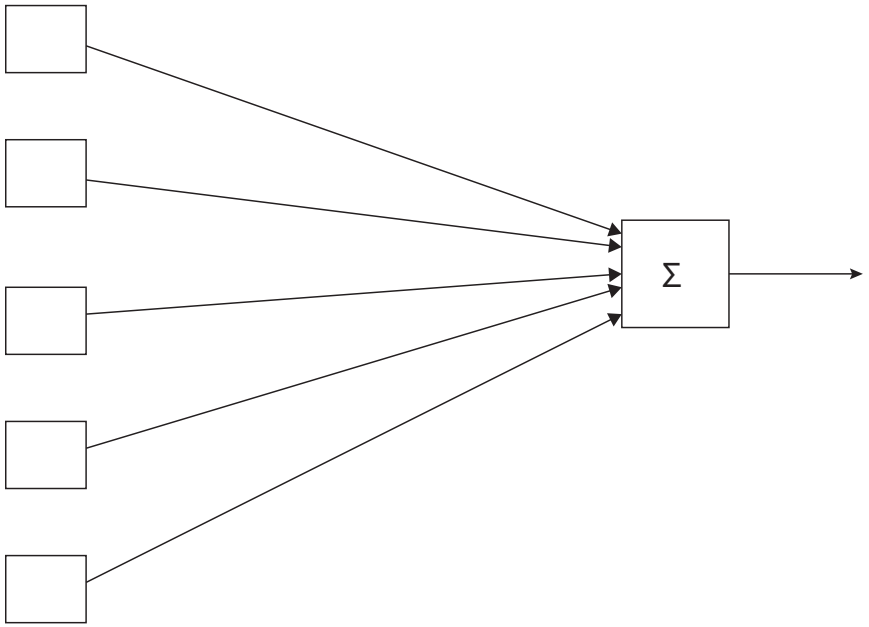
- The dataset error is the sum of datapoint loss.
  - The dataset error gradient is the sum of datapoint loss gradients:

$$\frac{\partial E(\boldsymbol{w}, D)}{\partial \boldsymbol{w}} = \sum_j \frac{\partial loss\left(y\left(\boldsymbol{x}_j\right), t_j\right)}{\partial \boldsymbol{w}}$$

Perceptrons

- Perceptrons are a precursor to neural nets.
  - Analog implementation by Rosenblatt in the 1950s





The Perceptron Error Function

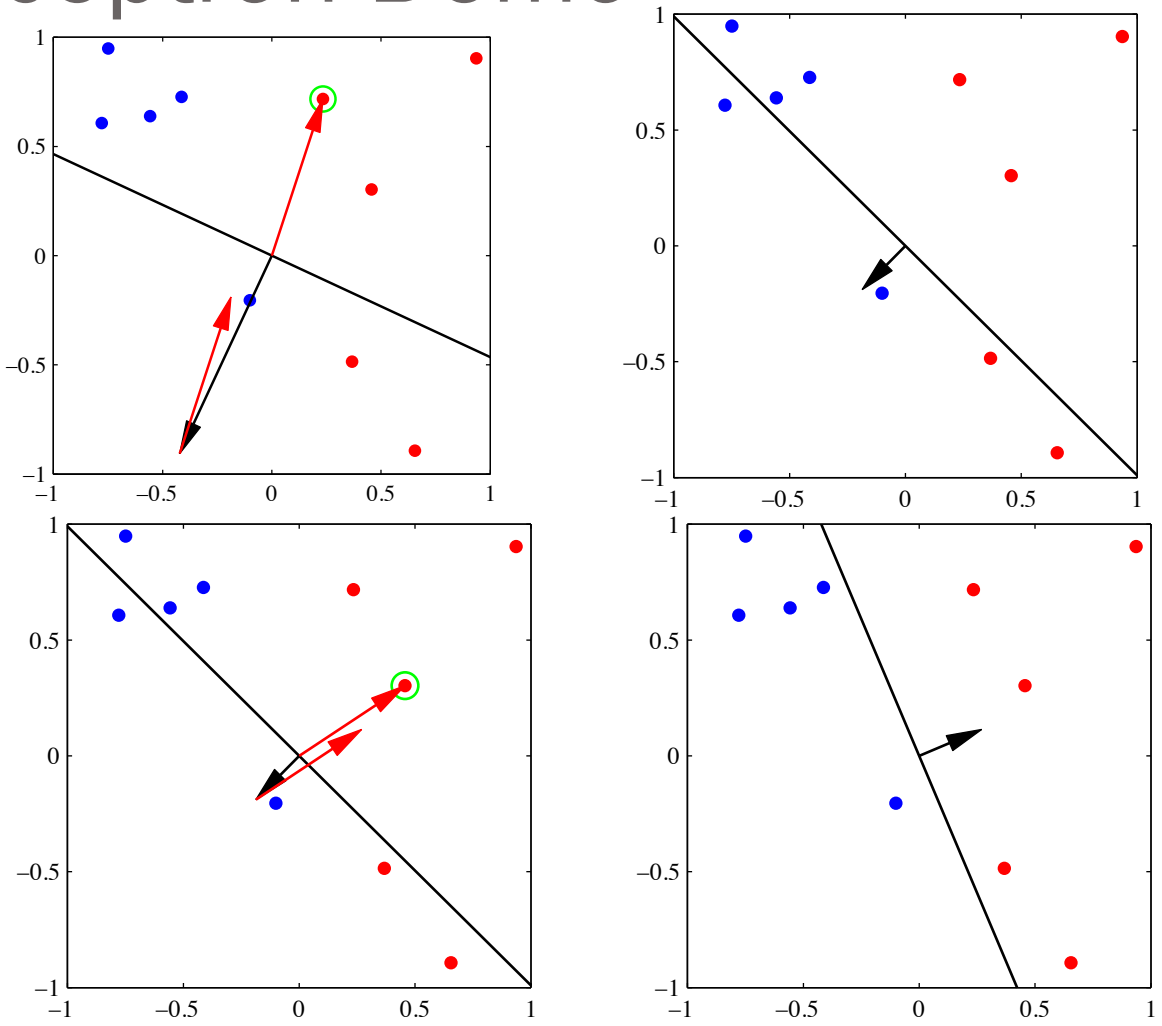
- Let y = 1 for positive class, y = -1 for negative
- An example is correctly classified if and only if $(\boldsymbol{x}_j \cdot \boldsymbol{w})t_j > 0$
  - Exercise: Take a moment to verify this.
- Perceptron Error (fixed dataset *D*)

$$E(\boldsymbol{w}) = -\sum_{j \in M} (\boldsymbol{x}_j \cdot \boldsymbol{w})t_j$$

  where *M* is the set of misclassified inputs, the **mistakes**.
- Exercise: find the gradient of the error function for a single input $\mathbf{x}_j$.
- Solution:
  - 0 if $\mathbf{x}_j$ is correctly classified.
  - $-\mathbf{x}_j t_j$ o.w.
- What is the gradient descent weight update formula?
- Excel Demo

Perceptron Demo

weight vector = black



Perceptron Learning Analysis

- **Theorem** If the classes are linearly separable, the perceptron learning algorithm converges to a weight vector that separates them.
- Number of steps to convergence has a theoretical upper bound.
  - In deep learning, usually no bound
- Sensitive to initialization.
- If classes are not linearly separable (e.g. X-OR), typically fails to converge.

Least-Squares Error Function

- Venerable idea (19th century)

1. For each data point, find the difference between predicted and observed value
2. Square the difference
3. Sum/average the squared differences

- Not popular for classification problems
- But widely used for regression

$$E(\boldsymbol{w}) = 1/2 \sum_j ((\boldsymbol{x}_j \cdot \boldsymbol{w}) - t_j)^2$$

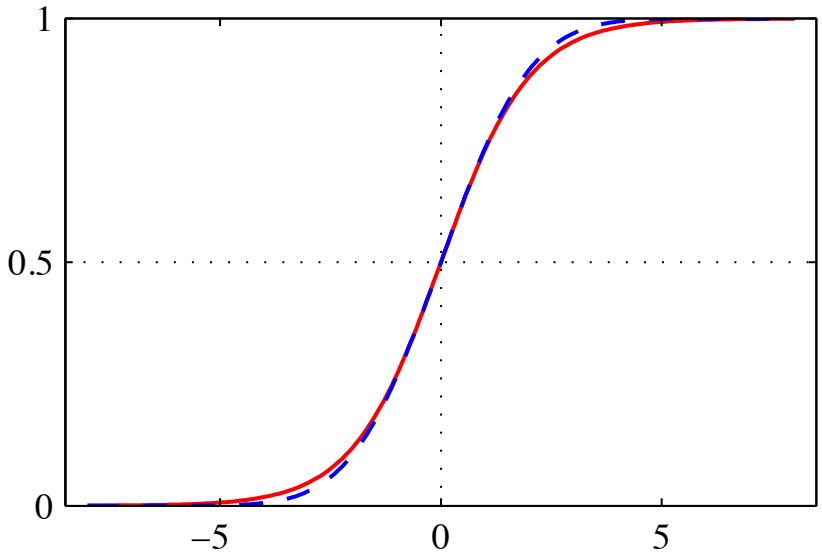
Logistic Regression

From Values to Probabilities

- Key idea: instead of predicting a class label, predict the *probability of a class label*.
- E.g., p+ = P(class is positive | features)
  p- = P(class is negative | features)
- Naturally a continuous quantity.
- How to turn a real number *y* into a probability p+?

The Logistic Sigmoid Function

- Definition: $\sigma(y) = \dfrac{1}{1+\exp(-y)}$
- Squeezes the real line into $[0,1]$.
- Differentiable: $\dfrac{d\sigma}{dy} = \sigma(1-\sigma)$ (nice exercise)



Soft threshold interpretation

- If $y > 0$, $\sigma(y)$ goes to 1 very quickly.
- If $y < 0$, $\sigma(y)$ goes to 0 very quickly.

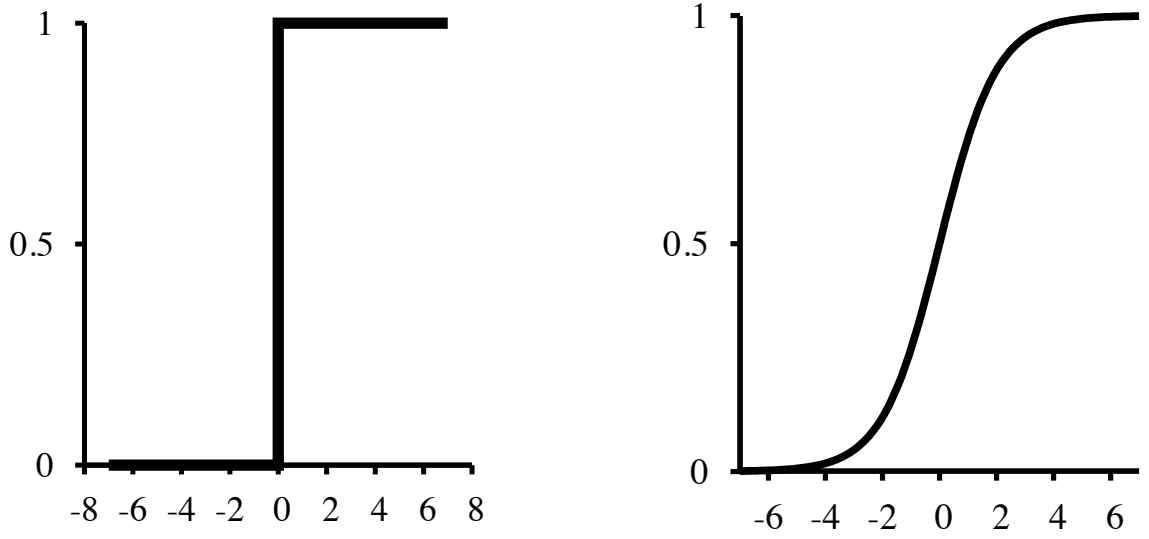


Figure Russell and Norvig 18.17

Probabilistic Interpretation

- The sigmoid can be interpreted in terms of the **class odds** p+/(1-p+).
- Exercise: Show the following implication for the class odds:

$$p^+ = \frac{1}{1+\exp(-y)} \Rightarrow \frac{p^+}{1-p^+} = \exp(y)$$

- Therefore $y = \ln(\frac{p^+}{1-p^+}) =$ the **log class odds**.

Logistic Regression

- In logistic regression, the log-class odds are a linear function of the input features:

$$\ln\left(\frac{p^+}{1-p^+}\right) = \mathbf{x} \bullet \mathbf{w}$$

- Learning logistic regression is conceptually similar to linear regression.
- Log-linear (or exponential) models are the "nicest" general family of statistical models.

Logistic Regression: Maximum Likelihood

- Notation: the probability that the n-th input example is positive = p_n^+ which depends on a weight vector $\mathbf{w}$.
- Positive example has $t_j = 1$, negative $t_j = 0$.
- Then the likelihood assigned to N independent training data is

$$p(\boldsymbol{y}; \boldsymbol{w}) = \prod_{j=1}^{N} (p_j^+)^{t_j} \, (1 - p_j^+)^{1-t_j}$$

➢ The cross-entropy error

$$\mathrm{E}(\boldsymbol{w}) = -\ln p(\boldsymbol{y}; \boldsymbol{w}) =$$
$$-\sum_{j=1}^{N} t_j ln(p_j^+) + (1 - t_j) ln(1 - p_j^+)$$

- Equivalent to minimizing the KL divergence between the predicted class probabilities and the observed class frequencies.

Weight Learning

- Homework Exercise: find the gradient of the cross-entropy error function wrt a single input $\mathbf{x}_n$

$$E(\boldsymbol{w}) = -\sum_{j=1}^{N} t_j ln(p_j^+) + (1 - t_j) ln(1 - p_j^+)$$

- Hint: recall that $\frac{d\sigma}{dy} = \sigma(1-\sigma)$
- No closed form minimum since p_j^+ is non-linear function of input features.
- ➢ Use gradient descent. See Stanford Demo with softmax.
- Better approach: Use Iterative Reweighted Least Squares (IRLS).

Example logistic regression model learned on non-separable data

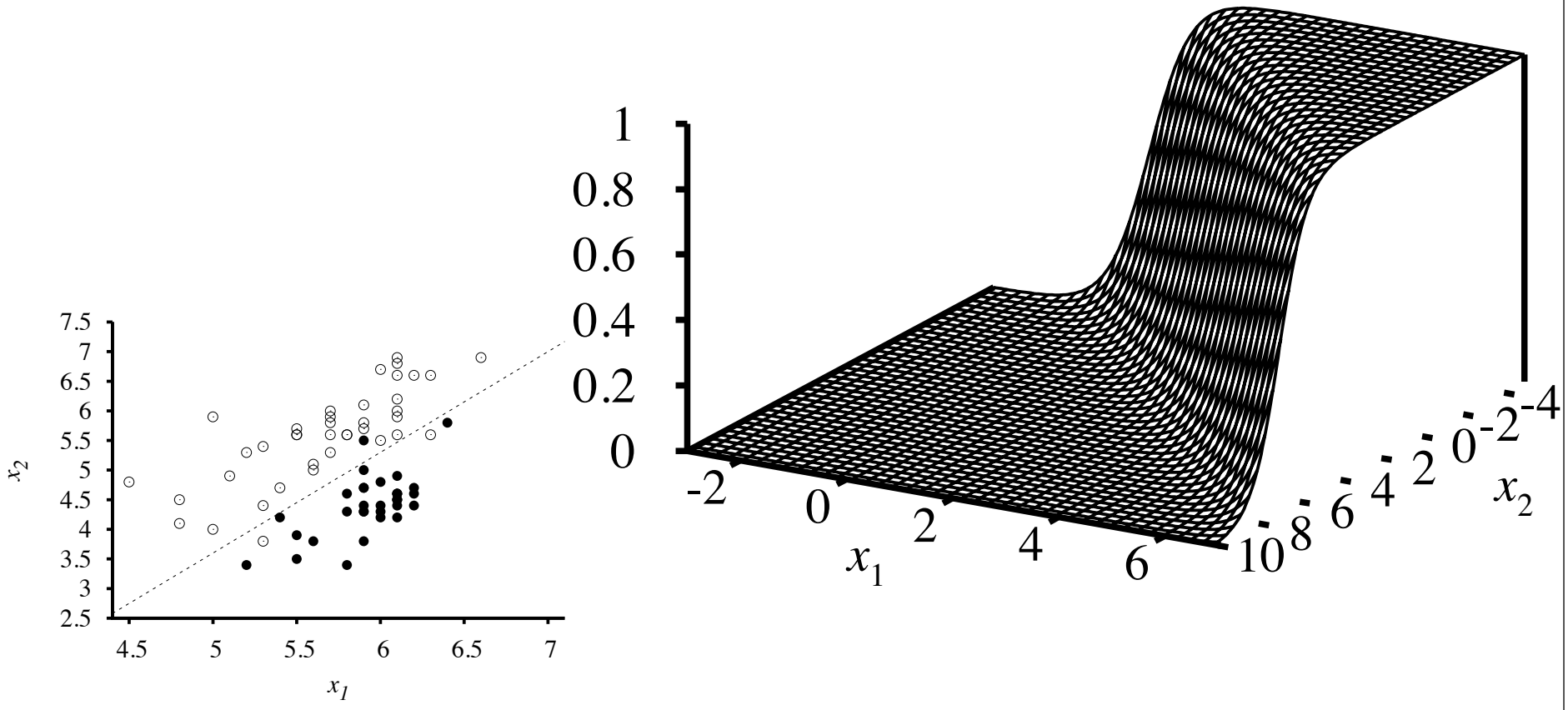
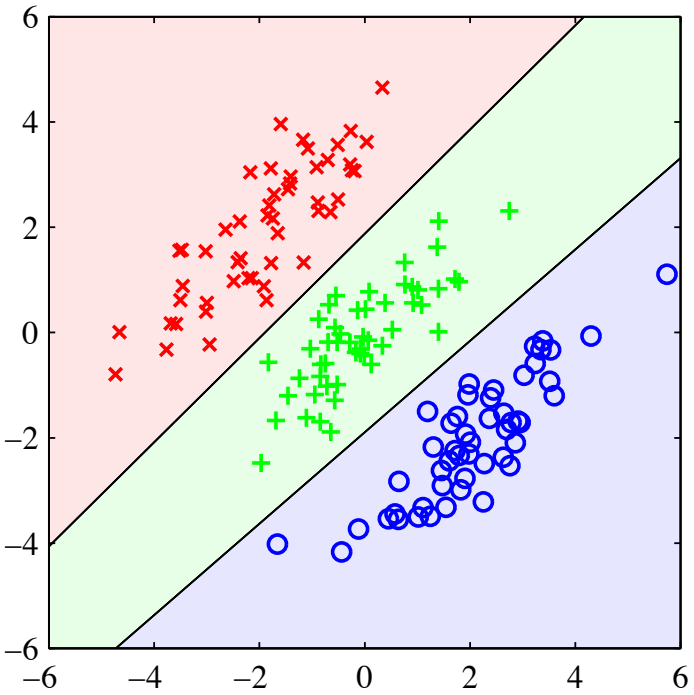


Figure Russell and Norvig 18.17

Multi-Class Example

- Logistic regression can be extended to multiple classes.
- Here's a picture of what decision boundaries can look like.



Multi-Class Problems and the SoftMax Function

- Generic multi-class probabilistic prediction

1. Build prediction models $y_1(\mathbf{x}), y_2(\mathbf{x}), .. y_k(\mathbf{x})$, one for each of the k classes.
2. Transform the numbers into probabilities:
   1. Map each $y_i(\mathbf{x})$ to $\exp\{y_i(\mathbf{x})\}$
   2. Divide by the sum:
      $\sigma(\mathbf{x})_j = \exp\{y_j(\mathbf{x})\} / \Sigma i \exp\{y_i(\mathbf{x})\}$

Linear Algebra Notation

- Suppose we build 10 linear models $y_1(\mathbf{x}), y_2(\mathbf{x}), .. y_{10}(\mathbf{x})$, for 10 classes.
- For a data matrix $\mathbf{X}$, the linear model predictions for each data point can be written like this (including the bias terms) $\mathbf{Y = XW + B}$
- What are the dimensions of **X,W,Y,B** and what do they represent?
- n: data points. f features for each input

Feature Transformations

Basis Functions

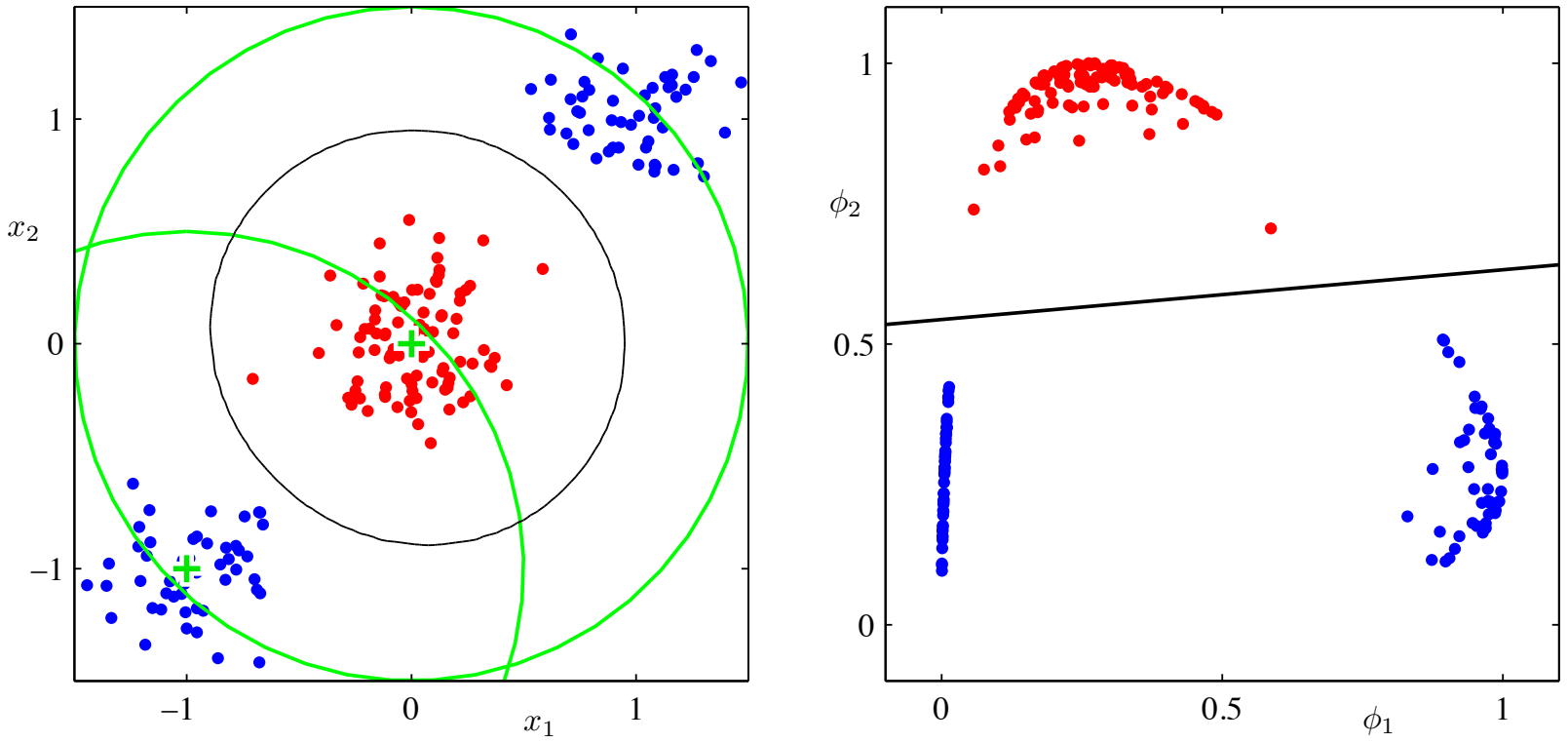
Basis Functions

- Powerful approach to non-separability
- Extract new features from the original data
  - Called basis functions in the context of regression
- Perform linear classification on the new features, not the original data

+ Instances may be linearly separable in the new feature space

– The new features may not be as meaningful or **interpretable** as the original ones.

Example



- $\varphi_1(x_1, x_2)$ measures distance from left green cross
- $\varphi_2(x_1, x_2)$ measures closeness to centre green cross
- 3D demo

Figure Bishop 4.12 see also

Visualize Transformation



original data



continuous function



transformed data

Transformations can keep/increase/decrease original dimensionality

Summary I

- Linear classification: learn a linear function of features that separates positive and negative classes.
- Find feature weights by minimizing an error function.
- Different error functions used:
  - perceptron
  - cross-entropy (logistic regression)
  - max-margin (support vector machines, not discussed)
  - Least squares (for regression not classification)
- If classes are not linearly separable, no linear classifier is 100% accurate.
- Options:
  - use the best line you can → logistic regression
  - use non-linear prediction function → neural nets

Looking Ahead to Deep Learning

- Neural nets:
  - Hidden layers transform the input features
  - Output nodes perform linear classification/regression on the transformed features
- Deep neural nets: transform input features, transform the transformed features, …., again and again
- The transformations are *learned* not fixed