Linear models for regression and classification

Grzegorz Chrupała

Saarland University

October 19, 2012

Outline

Outline

Regression analysis

- Model relationships between variables

Regression analysis

- Model relationships between variables
- Specifically: model the dependent (output) variable as a function of the independent (input) variables

Regression analysis

- Model relationships between variables
- Specifically: model the dependent (output) variable as a function of the independent (input) variables
- Example:
  - **Describe** how people's weight depends on their height

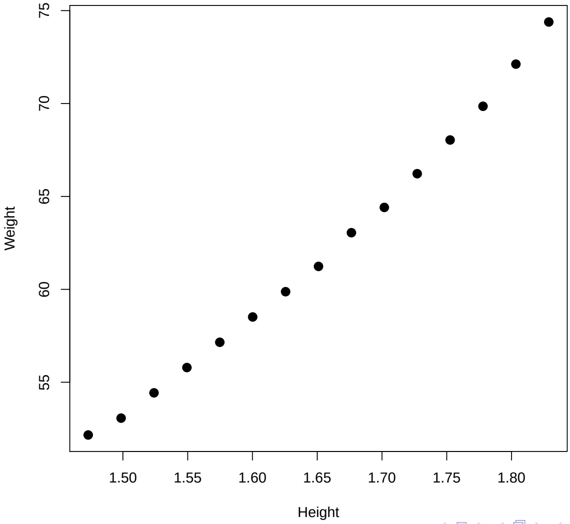
Regression analysis

- Model relationships between variables
- Specifically: model the dependent (output) variable as a function of the independent (input) variables
- Example:
  - **Describe** how people's weight depends on their height
  - **Predict** people's weight given their height

Sample data

	Height	Weight
1	1.47	52.2
2	1.50	53.1
3	1.52	54.4
4	1.55	55.8
5	1.57	57.2
6	1.60	58.5
7	1.63	59.9
8	1.65	61.2
9	1.68	63.0
10	1.70	64.4
11	1.73	66.2
12	1.75	68.0
13	1.78	69.9
14	1.80	72.1
15	1.83	74.4

Scatter plot



Model

- Single independent variable x
- Dependent variable y
- Model the relationship as a parametrized function $y = f(x)$:
  - $f(x) = ax^2 + bx + c$

Model

- Single independent variable x
- Dependent variable y
- Model the relationship as a parametrized function $y = f(x)$:
  - $f(x) = ax^2 + bx + c$
  - $f(x) = a\sin(x) + b$

Model

- Single independent variable x
- Dependent variable y
- Model the relationship as a parametrized function $y = f(x)$:
  - $f(x) = ax^2 + bx + c$
  - $f(x) = a\sin(x) + b$
  - $f(x) = ax + b$
- We focus on **linear** regression

Linear Regression

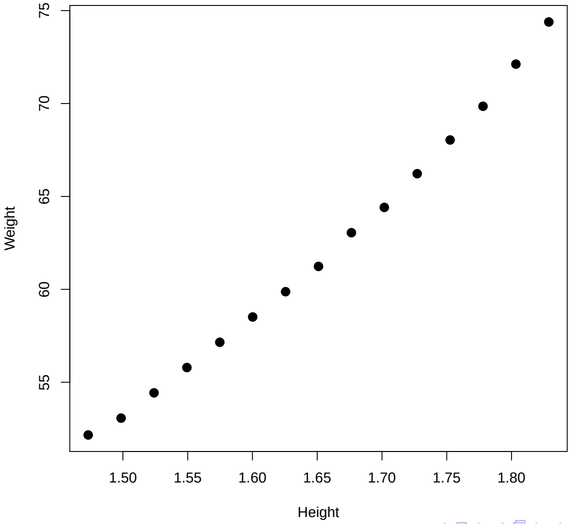
- Training data: observations paired with outcomes
- Observations are described by independent variables (features, predictors)
- The model is a regression line $y = ax + b$ which best fits the observations
  - a is the slope
  - b is the intercept (bias)
  - This model has two parameters (weigths, coefficients)
  - There is only one independent variable $= x$

Best fit

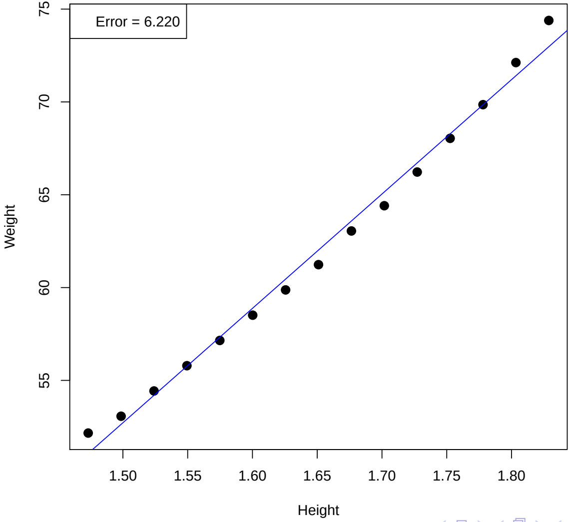
- Residual: difference between true value y and predicted value $f(x)$
- Find a line which minimizes sum of squared residuals:

$$\text{Error} = \sum_{i=0}^{N} (y^{(i)} - f(x^{(i)}))^2$$

Scatter plot



Prediction of weight from height



Is the choice of a linear relationship appropriate for this data?

Is the choice of a linear relationship appropriate for this data?

- Simplify: model a subject as a solid ball of radius r

Is the choice of a linear relationship appropriate for this data?

- Simplify: model a subject as a solid ball of radius r
- How will weight depend on radius?

Is the choice of a linear relationship appropriate for this data?

- Simplify: model a subject as a solid ball of radius r
- How will weight depend on radius?

$$V = \frac{4}{3}\pi r^3$$

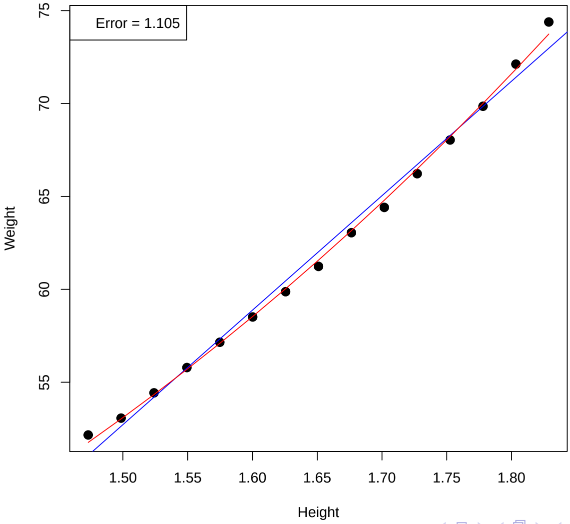
Is the choice of a linear relationship appropriate for this data?

- Simplify: model a subject as a solid ball of radius r
- How will weight depend on radius?

$$V = \frac{4}{3}\pi r^3$$

- How can we test if this carries over to the real subjects?

Prediction of weight from height cubed



Multiple linear regression

- More generally $y = w_0 + \sum_{i=1}^{d} w_i x_i$, where
  - y = outcome
  - w_0 = intercept
  - $x_1..x_d$ = features vector and $w_1..w_d$ weight vector
  - Get rid of bias:

$$g(\mathbf{x}) = \sum_{i=0}^{d} w_i x_i = \mathbf{w} \cdot \mathbf{x}$$

Learning linear regression

- Minimize sum squared error over N training examples

$$\mathrm{Err}(\mathbf{w}) = \sum_{n=1}^{N} (g(\mathbf{x}^{(n)}) - y^{(n)})^2$$

- Closed-form formula for choosing the best weights $\mathbf{w}$:

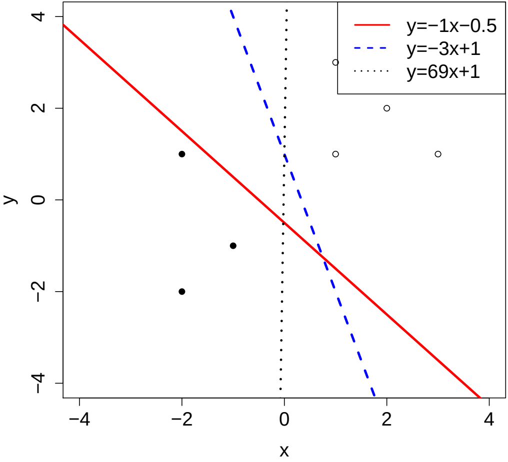
$$\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y}$$

where the matrix X contains training example features, and $\mathbf{y}$ is the vector of outcomes.

Outline

Classification: An example

Positive examples are blank, negative are filled



Linear models

Think of training examples as points in d-dimensional space. Each dimension corresponds to one feature.

Linear models

Think of training examples as points in d-dimensional space. Each dimension corresponds to one feature.

A linear binary classifier defines a plane in the space which separates positive from negative examples.

Linear decision boundary

- A hyperplane is a generalization of a straight line to > 2 dimensions

Linear decision boundary

- A hyperplane is a generalization of a straight line to > 2 dimensions
- A hyperplane contains all the points in a d dimensional space satisfying the following equation:

$$w_1 x_1 + w_2 x_2, \ldots, + w_d x_d + w_0 = 0$$

Linear decision boundary

- A hyperplane is a generalization of a straight line to > 2 dimensions
- A hyperplane contains all the points in a d dimensional space satisfying the following equation:

$$w_1 x_1 + w_2 x_2, \ldots, + w_d x_d + w_0 = 0$$

- Each coefficient w_i can be thought of as a weight on the corresponding feature

Linear decision boundary

- A hyperplane is a generalization of a straight line to > 2 dimensions
- A hyperplane contains all the points in a d dimensional space satisfying the following equation:

$$w_1 x_1 + w_2 x_2, \ldots, + w_d x_d + w_0 = 0$$

- Each coefficient w_i can be thought of as a weight on the corresponding feature
- The vector containing all the weights $\mathbf{w} = (w_0, \ldots, w_d)$ is the parameter vector or weigth vector

Normal vector

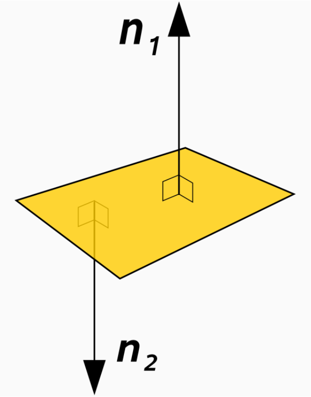
- Geometrically, the weight vector $\mathbf{w}$ is a normal vector of the separating hyperplane

Normal vector

- Geometrically, the weight vector $\mathbf{w}$ is a normal vector of the separating hyperplane
- A normal vector of a surface is any vector which is perpendicular to it

Normal vector

- Geometrically, the weight vector $\mathbf{w}$ is a normal vector of the separating hyperplane
- A normal vector of a surface is any vector which is perpendicular to it



Hyperplane as a classifier

- Let

$$g(\mathbf{x}) = w_1 x_1 + w_2 x_2, \ldots, + w_d x_d + w_0$$

Hyperplane as a classifier

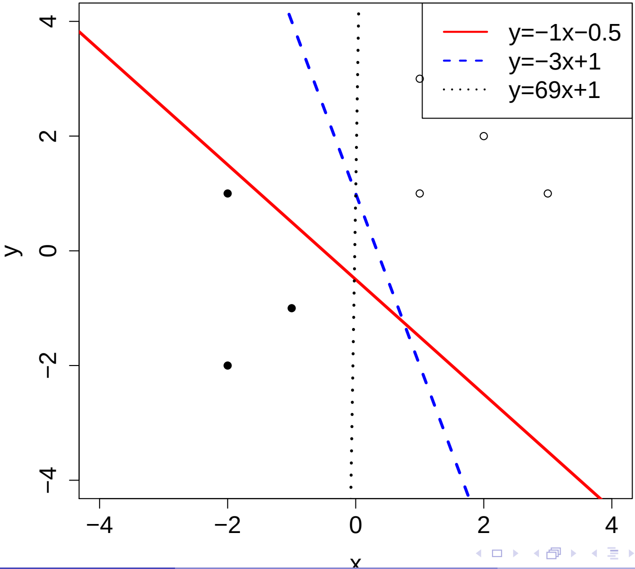
- Let

$$g(\mathbf{x}) = w_1 x_1 + w_2 x_2, \ldots, + w_d x_d + w_0$$

- Then

$$y = \operatorname{sign}(g(\mathbf{x})) = \begin{cases} +1 & \text{if } g(\mathbf{x}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Separating hyperplanes in 2 dimensions



Learning

- The goal of the learning process is to come up with a **good** weight vector $\mathbf{w}$

Learning

- The goal of the learning process is to come up with a **good** weight vector $\mathbf{w}$
- The learning process will use examples to guide the search of a **good** $\mathbf{w}$

Learning

- The goal of the learning process is to come up with a **good** weight vector $\mathbf{w}$
- The learning process will use examples to guide the search of a **good** $\mathbf{w}$
- Different notions of **goodness** exist, which yield different learning algorithms

Outline

Perceptron training

- How do we find a set of weights that separate our classes?

Perceptron training

- How do we find a set of weights that separate our classes?
- **Perceptron**: A simple mistake-driven online algorithm

Perceptron training

- How do we find a set of weights that separate our classes?
- **Perceptron**: A simple mistake-driven online algorithm

  - Start with a zero weight vector and process each training example in turn.

Perceptron training

- How do we find a set of weights that separate our classes?
- **Perceptron**: A simple mistake-driven online algorithm

  - Start with a zero weight vector and process each training example in turn.
  - If the current weight vector classifies the current example incorrectly, move the weight vector in the right direction.

Perceptron training

- How do we find a set of weights that separate our classes?
- **Perceptron**: A simple mistake-driven online algorithm

  - Start with a zero weight vector and process each training example in turn.
  - If the current weight vector classifies the current example incorrectly, move the weight vector in the right direction.
  - If weights stop changing, stop

Perceptron training

- How do we find a set of weights that separate our classes?
- **Perceptron**: A simple mistake-driven online algorithm
  - Start with a zero weight vector and process each training example in turn.
  - If the current weight vector classifies the current example incorrectly, move the weight vector in the right direction.
  - If weights stop changing, stop
- If examples are linearly separable, then this algorithm is guaranteed to converge to the solution vector

Update rule

- Binary classification, with classes $+1$ and -1

Update rule

- Binary classification, with classes $+1$ and -1
- Decision function $y' = \text{sign}(\mathbf{w} \cdot \mathbf{x})$

Update rule

- Binary classification, with classes $+1$ and -1
- Decision function $y' = \text{sign}(\mathbf{w} \cdot \mathbf{x})$
- How should we change $\mathbf{w}$ to make $\mathbf{w} \cdot \mathbf{x}$ higher?

Update rule

- Binary classification, with classes $+1$ and -1
- Decision function $y' = \text{sign}(\mathbf{w} \cdot \mathbf{x})$
- How should we change $\mathbf{w}$ to make $\mathbf{w} \cdot \mathbf{x}$ higher?
- Or lower?

Update rule

- Binary classification, with classes $+1$ and -1
- Decision function $y' = \text{sign}(\mathbf{w} \cdot \mathbf{x})$
- How should we change $\mathbf{w}$ to make $\mathbf{w} \cdot \mathbf{x}$ higher?
- Or lower?
- Add or subtract $\mathbf{x}$

Fixed increment online perceptron algorithm

1: $\mathbf{w} \leftarrow \mathbf{0}$

Fixed increment online perceptron algorithm

$\mathbf{w} \leftarrow \mathbf{0}$
for $n = 1...$ **do**

Fixed increment online perceptron algorithm

```
w ← 0
for n = 1... do
    if y^(n) = sign(w · x^(n)) then
        pass
```

Fixed increment online perceptron algorithm

$\mathbf{w} \leftarrow \mathbf{0}$
for $n = 1...$ **do**
  if $y^{(n)} = \text{sign}(\mathbf{w} \cdot \mathbf{x}^{(n)})$ **then**
    pass
  else if $y^{(n)} = +1 \wedge \text{sign}(\mathbf{w} \cdot \mathbf{x}^{(n)}) = -1$ **then**

Fixed increment online perceptron algorithm

$\mathbf{w} \leftarrow \mathbf{0}$
for $n = 1...$ **do**
    if $y^{(n)} = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x}^{(n)})$ **then**
        pass
    else if $y^{(n)} = +1 \wedge \mathrm{sign}(\mathbf{w} \cdot \mathbf{x}^{(n)}) = -1$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^{(n)}$

Fixed increment online perceptron algorithm

$\mathbf{w} \leftarrow \mathbf{0}$
for $n = 1...$ **do**
    if $y^{(n)} = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x}^{(n)})$ **then**
        pass
    else if $y^{(n)} = +1 \wedge \mathrm{sign}(\mathbf{w} \cdot \mathbf{x}^{(n)}) = -1$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^{(n)}$
    else if $y^{(n)} = -1 \wedge \mathrm{sign}(\mathbf{w} \cdot \mathbf{x}^{(n)}) = +1$ **then**

Fixed increment online perceptron algorithm

$\mathbf{w} \leftarrow \mathbf{0}$
for $n = 1...$ **do**
    if $y^{(n)} = \text{sign}(\mathbf{w} \cdot \mathbf{x}^{(n)})$ **then**
        pass
    else if $y^{(n)} = +1 \wedge \text{sign}(\mathbf{w} \cdot \mathbf{x}^{(n)}) = -1$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^{(n)}$
    else if $y^{(n)} = -1 \wedge \text{sign}(\mathbf{w} \cdot \mathbf{x}^{(n)}) = +1$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}^{(n)}$

Fixed increment online perceptron algorithm

$\mathbf{w} \leftarrow \mathbf{0}$
for $n = 1...$ **do**
  if $y^{(n)} = \text{sign}(\mathbf{w} \cdot \mathbf{x}^{(n)})$ **then**
    pass
  else if $y^{(n)} = +1 \wedge \text{sign}(\mathbf{w} \cdot \mathbf{x}^{(n)}) = -1$ **then**
    $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}^{(n)}$
  else if $y^{(n)} = -1 \wedge \text{sign}(\mathbf{w} \cdot \mathbf{x}^{(n)}) = +1$ **then**
    $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}^{(n)}$
return $\mathbf{w}$

Or more compactly

PERCEPTRON($x^{1:N}, y^{1:N}, I$):

```
w ← 0
for i = 1...I do
    for n = 1...N do
        if y^(n)(w · x^(n)) ≤ 0 then
            w ← w + y^(n) x^(n)
return w
```

$\mathbf{w} \leftarrow \mathbf{0}$

for $i = 1...I$ **do**

  for $n = 1...N$ **do**

    if $y^{(n)}(\mathbf{w} \cdot \mathbf{x}^{(n)}) \leq 0$ **then**

      $\mathbf{w} \leftarrow \mathbf{w} + y^{(n)}\mathbf{x}^{(n)}$

return $\mathbf{w}$

Weight averaging

- Although the algorithm is guaranteed to converge, the solution is not unique

Weight averaging

- Although the algorithm is guaranteed to converge, the solution is not unique
- Empirically, better generalization performance with weight averaging

Weight averaging

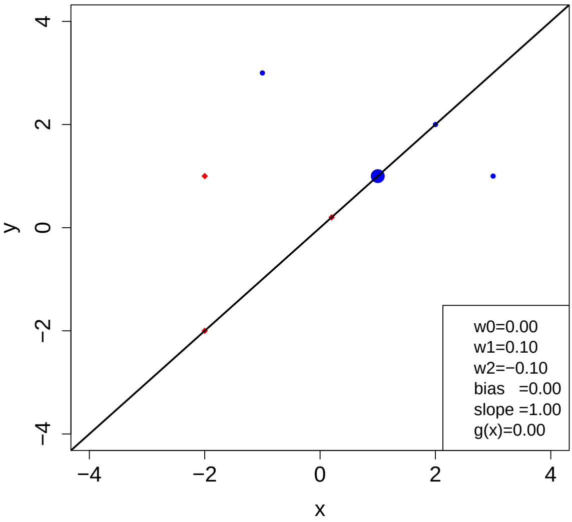
- Although the algorithm is guaranteed to converge, the solution is not unique
- Empirically, better generalization performance with weight averaging
  - A method of avoiding overfitting

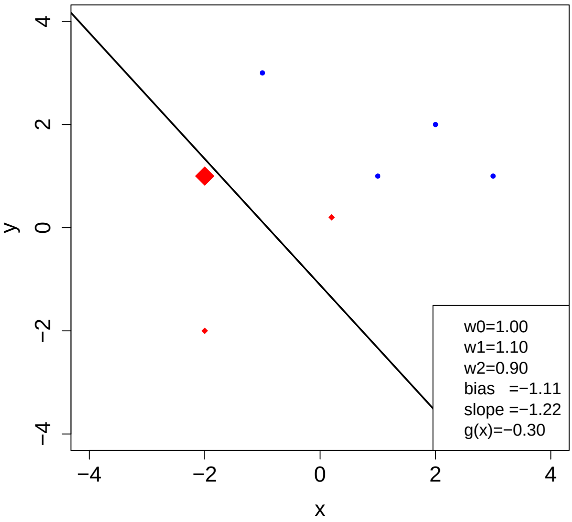
Weight averaging

- Although the algorithm is guaranteed to converge, the solution is not unique
- Empirically, better generalization performance with weight averaging
  - A method of avoiding overfitting
  - As final weight vector, use the mean of all the weight vector values for each step of the algorithm

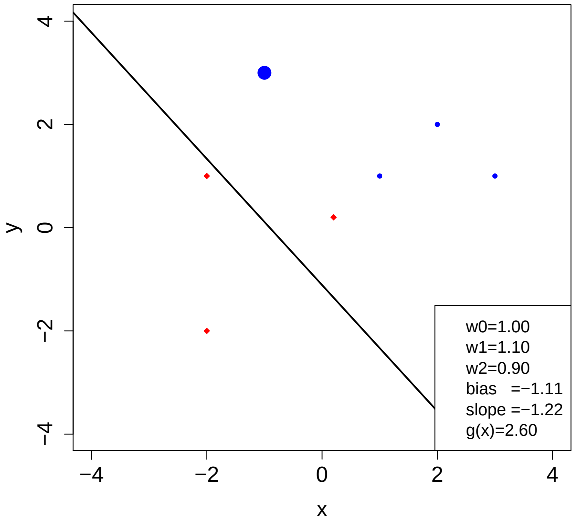
Weight averaging

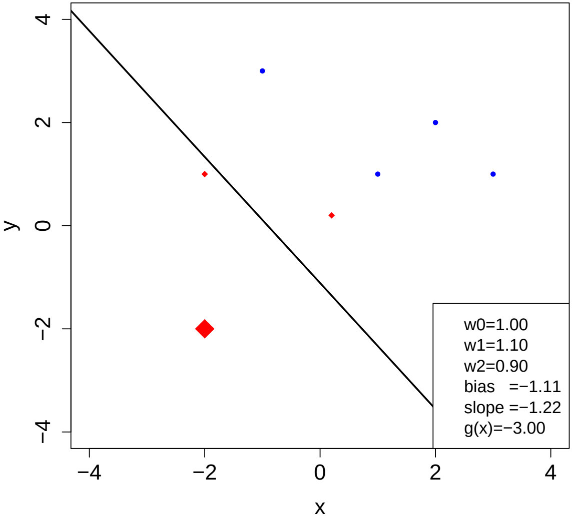
- Although the algorithm is guaranteed to converge, the solution is not unique
- Empirically, better generalization performance with weight averaging
  - A method of avoiding overfitting
  - As final weight vector, use the mean of all the weight vector values for each step of the algorithm
  - (cf. regularization in a following session)


w0=0.00
w1=0.10
w2=−0.10
bias =0.00
slope =1.00
g(x)=0.00
x
y
−4
−2
0
2
4

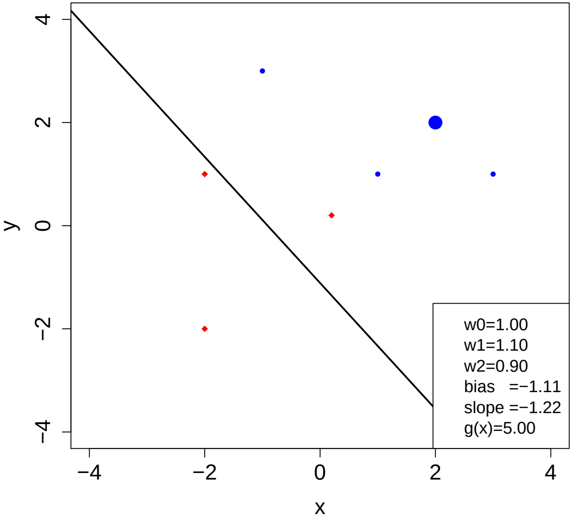


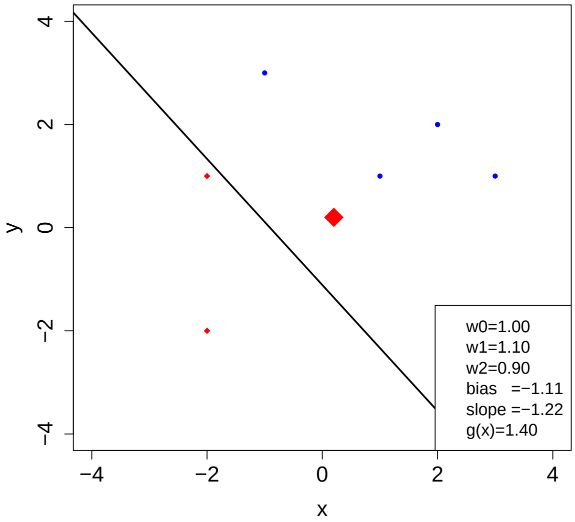
w0=1.00
w1=1.10
w2=0.90
bias =−1.11
slope =−1.22
g(x)=−0.30
x
y
−4
−2
0
2
4

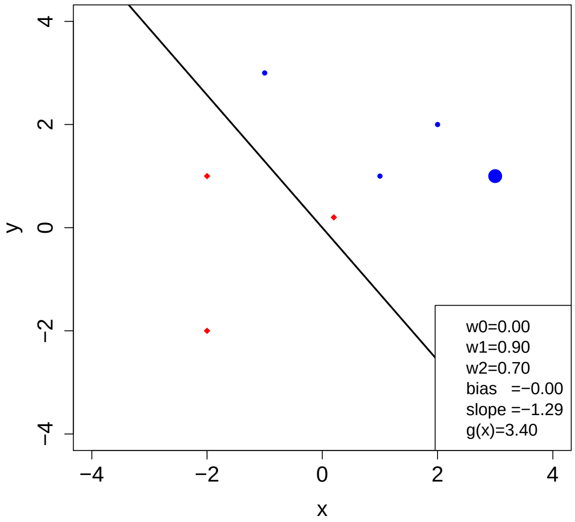

w0=1.00
w1=1.10
w2=0.90
bias =−1.11
slope =−1.22
g(x)=2.60
x
y
−4
−2
0
2
4



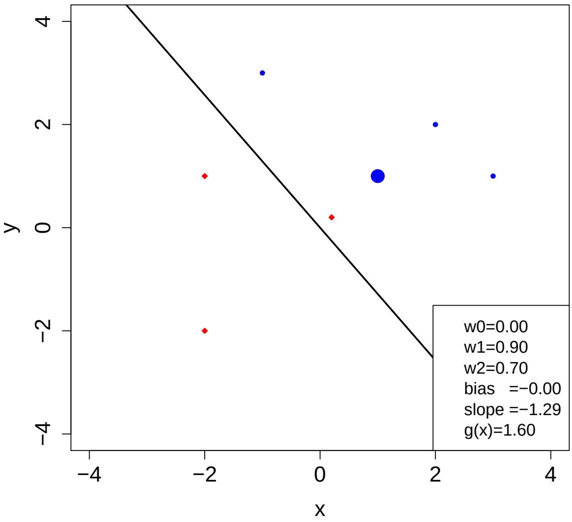
w0=1.00
w1=1.10
w2=0.90
bias =−1.11
slope =−1.22
g(x)=−3.00
y
x
−4
−2
0
2
4

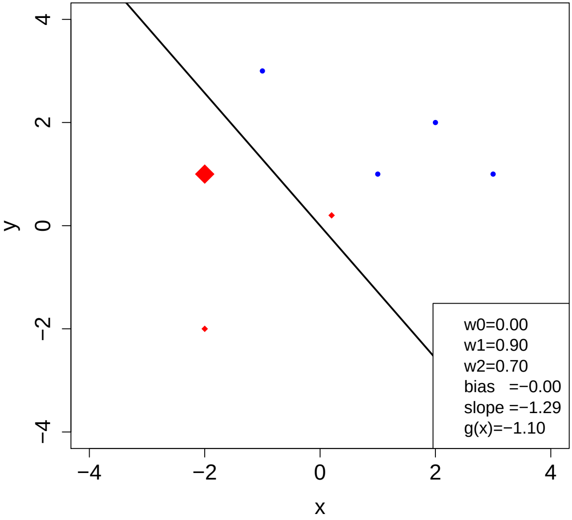

w0=1.00
w1=1.10
w2=0.90
bias =−1.11
slope =−1.22
g(x)=5.00
x
y
−4
−2
0
2
4

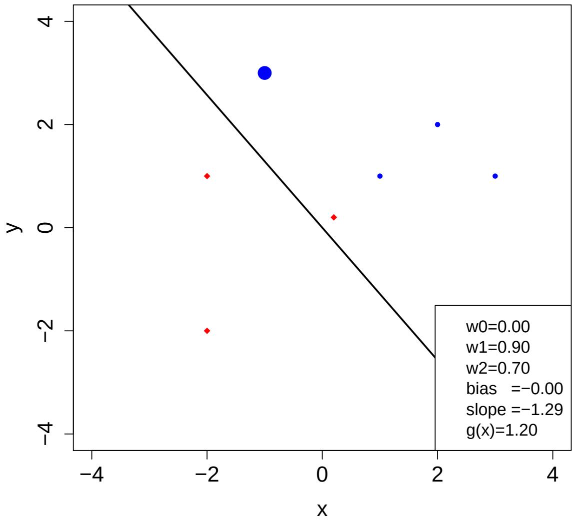

w0=1.00
w1=1.10
w2=0.90
bias =−1.11
slope =−1.22
g(x)=1.40
x
y
−4
−2
0
2
4



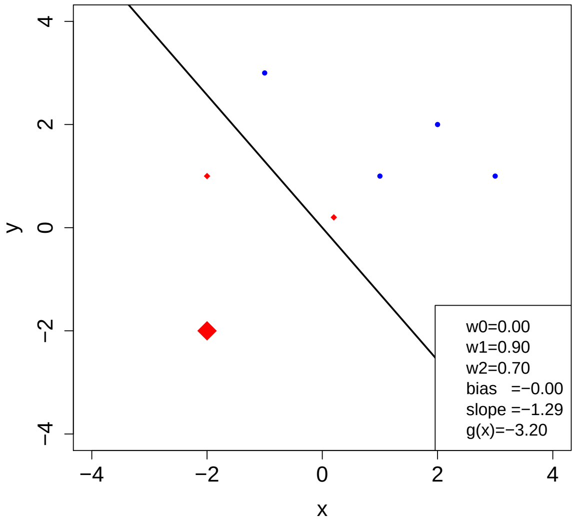
4
2
0
−2
−4
y
−4
−2
0
2
4
x
w0=0.00
w1=0.90
w2=0.70
bias =−0.00
slope =−1.29
g(x)=3.40

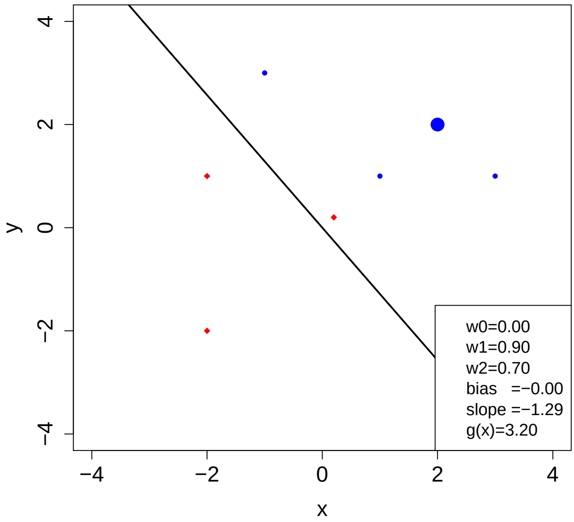

w0=0.00
w1=0.90
w2=0.70
bias =−0.00
slope =−1.29
g(x)=1.60
x
y
−4
−2
0
2
4

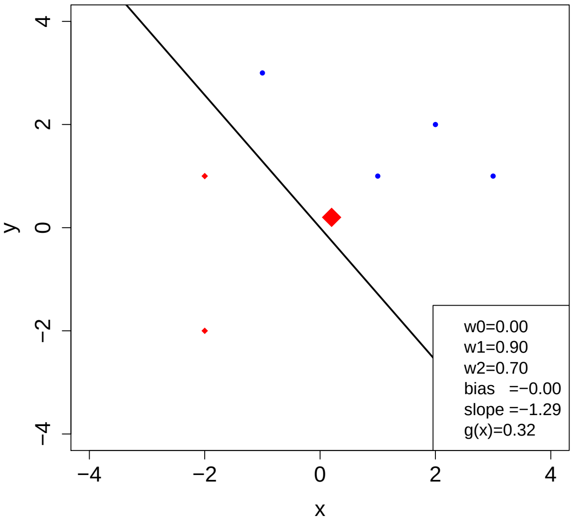

w0=0.00
w1=0.90
w2=0.70
bias =−0.00
slope =−1.29
g(x)=−1.10
x
y
−4
−2
0
2
4



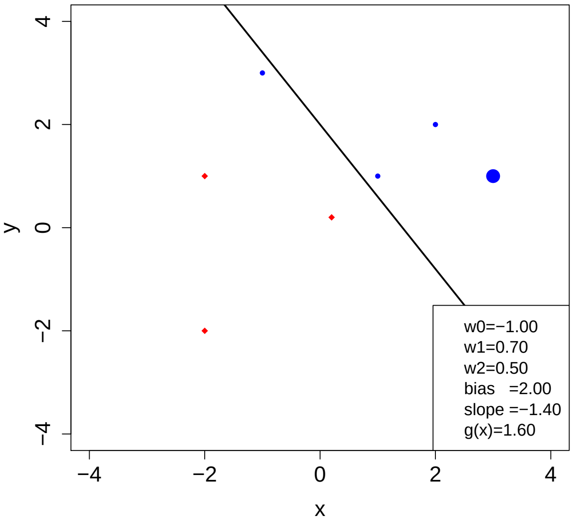
w0=0.00
w1=0.90
w2=0.70
bias =−0.00
slope =−1.29
g(x)=1.20
x
y

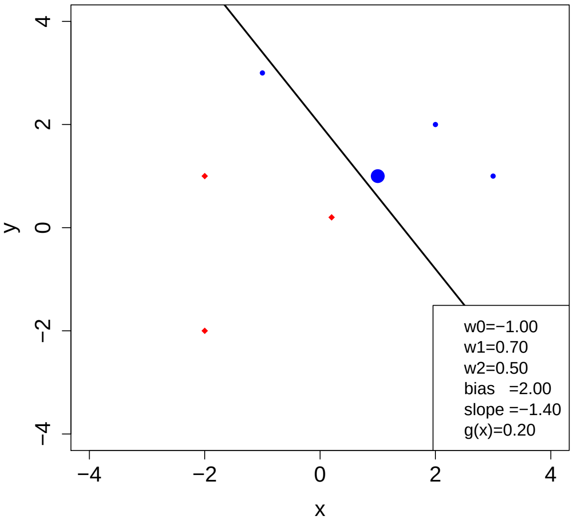

w0=0.00
w1=0.90
w2=0.70
bias =−0.00
slope =−1.29
g(x)=−3.20
x
y
−4
−2
0
2
4

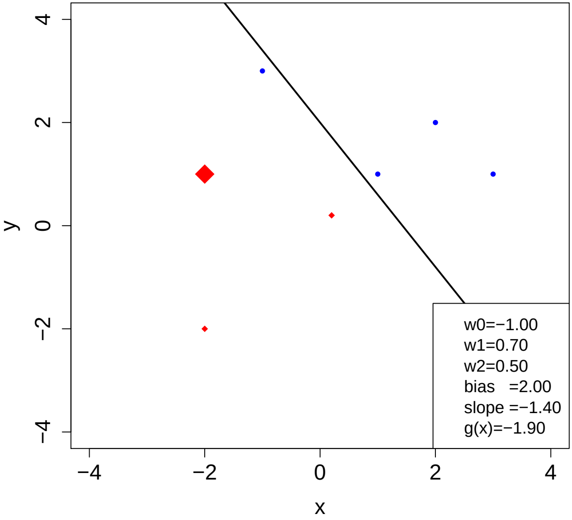

w0=0.00
w1=0.90
w2=0.70
bias =−0.00
slope =−1.29
g(x)=3.20
y
x
−4
−2
0
2
4

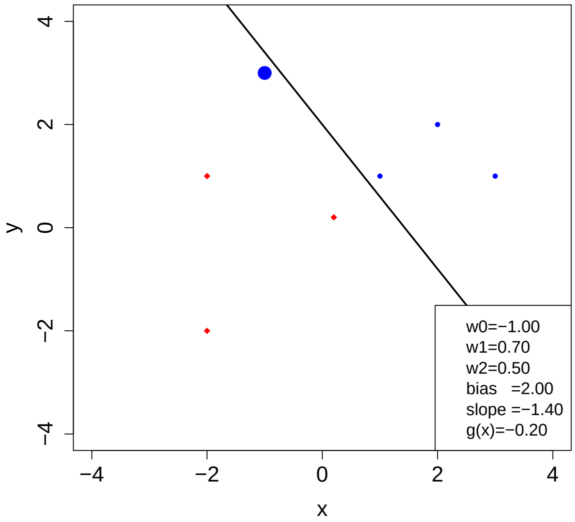


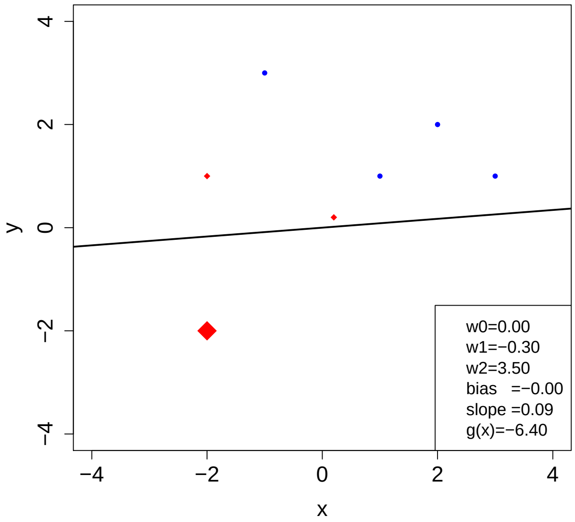
w0=0.00
w1=0.90
w2=0.70
bias =−0.00
slope =−1.29
g(x)=0.32
x
y
−4
−2
0
2
4


w0=−1.00
w1=0.70
w2=0.50
bias =2.00
slope =−1.40
g(x)=1.60
x
y
−4
−2
0
2
4

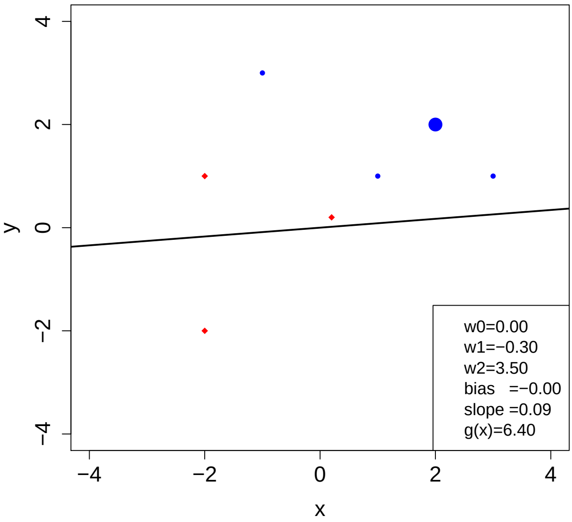

w0=−1.00
w1=0.70
w2=0.50
bias =2.00
slope =−1.40
g(x)=0.20
x
y
−4
−2
0
2
4

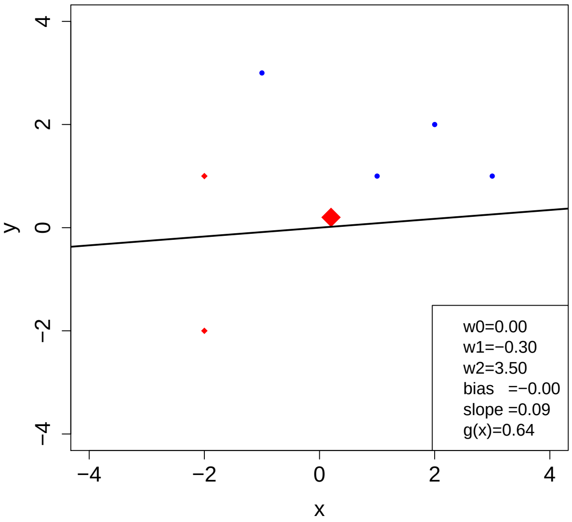

w0=−1.00
w1=0.70
w2=0.50
bias =2.00
slope =−1.40
g(x)=−1.90
y
x
4
2
0
−2
−4
−4
−2
0
2
4

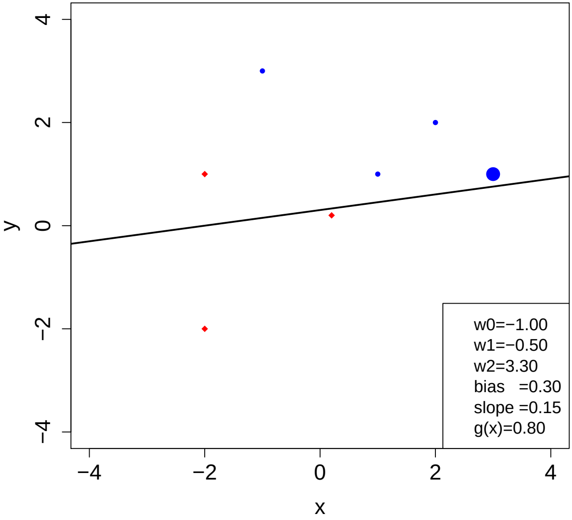

w0=−1.00
w1=0.70
w2=0.50
bias =2.00
slope =−1.40
g(x)=−0.20
x
y
−4
−2
0
2
4



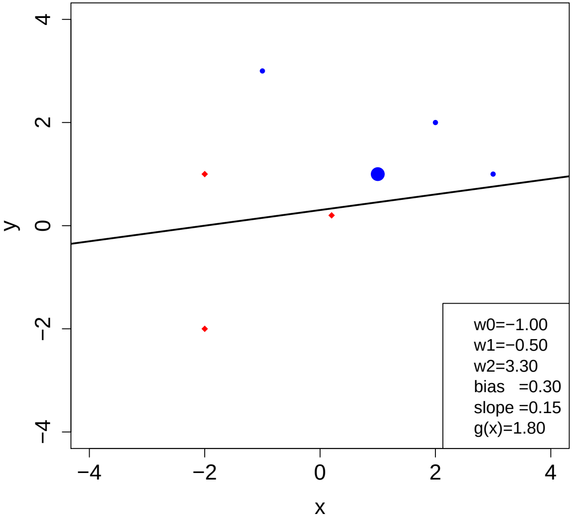
w0=0.00
w1=−0.30
w2=3.50
bias =−0.00
slope =0.09
g(x)=−6.40
x
y
−4
−2
0
2
4

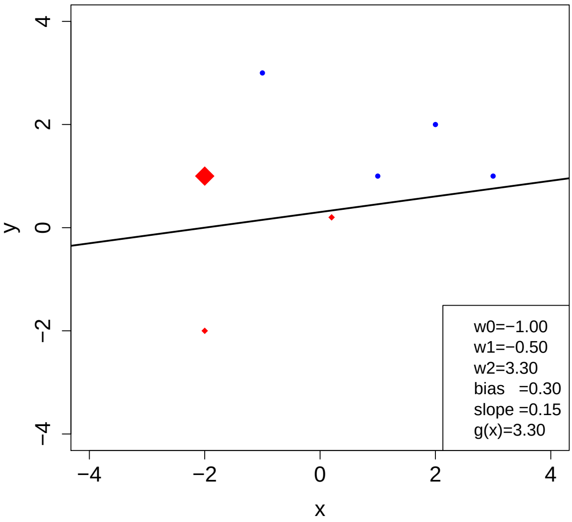

w0=0.00
w1=−0.30
w2=3.50
bias =−0.00
slope =0.09
g(x)=6.40
x
y
−4
−2
0
2
4

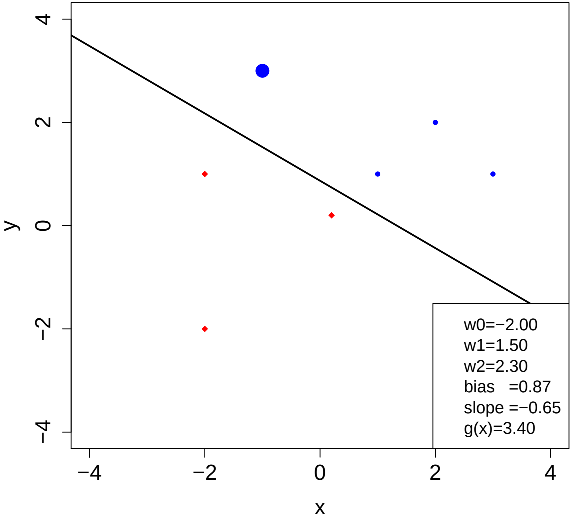

w0=0.00
w1=−0.30
w2=3.50
bias =−0.00
slope =0.09
g(x)=0.64
x
y
−4
−2
0
2
4



4
2
0
−2
−4
y
−4
−2
0
2
4
x
w0=−1.00
w1=−0.50
w2=3.30
bias =0.30
slope =0.15
g(x)=0.80


w0=−1.00
w1=−0.50
w2=3.30
bias =0.30
slope =0.15
g(x)=1.80
x
y
−4
−2
0
2
4


w0=−1.00
w1=−0.50
w2=3.30
bias =0.30
slope =0.15
g(x)=3.30
x
y
−4
−2
0
2
4


4
2
0
−2
−4
y
−4
−2
0
2
4
x
w0=−2.00
w1=1.50
w2=2.30
bias =0.87
slope =−0.65
g(x)=3.40

Outline

Probabilistic model

- Instead of thinking in terms of multidimensional space...

Probabilistic model

- Instead of thinking in terms of multidimensional space...
- Classification can be approached as a probability estimation problem

Probabilistic model

- Instead of thinking in terms of multidimensional space...
- Classification can be approached as a probability estimation problem
- We will try to find a probability distribution which

Probabilistic model

- Instead of thinking in terms of multidimensional space...
- Classification can be approached as a probability estimation problem
- We will try to find a probability distribution which
  - Describes well our training data

Probabilistic model

- Instead of thinking in terms of multidimensional space...
- Classification can be approached as a probability estimation problem
- We will try to find a probability distribution which
  - Describes well our training data
  - Allows us to make accurate predictions

Probabilistic model

- Instead of thinking in terms of multidimensional space...
- Classification can be approached as a probability estimation problem
- We will try to find a probability distribution which
  - Describes well our training data
  - Allows us to make accurate predictions
- We'll look at Naive Bayes as a simplest example of a probabilistic classifier

Representation of examples

- We are trying to classify documents. Let's represent a document as a sequence of terms (words) it contains $\mathbf{t} = (t_1 ... t_n)$

Representation of examples

- We are trying to classify documents. Let's represent a document as a sequence of terms (words) it contains $\mathbf{t} = (t_1 ... t_n)$
- For (binary) classification we want to find the most probable class:

$$\hat{y} = \operatorname*{argmax}_{y \in \{-1,+1\}} P(Y = y | \mathbf{t})$$

Representation of examples

- We are trying to classify documents. Let's represent a document as a sequence of terms (words) it contains $\mathbf{t} = (t_1 ... t_n)$
- For (binary) classification we want to find the most probable class:

$$\hat{y} = \operatorname*{argmax}_{y \in \{-1,+1\}} P(Y = y|\mathbf{t})$$

- Documents are close to unique: how do we condition on $\mathbf{t}$?

Representation of examples

- We are trying to classify documents. Let's represent a document as a sequence of terms (words) it contains $\mathbf{t} = (t_1 ... t_n)$
- For (binary) classification we want to find the most probable class:

$$\hat{y} = \underset{y \in \{-1,+1\}}{\operatorname{argmax}} P(Y = y | \mathbf{t})$$

- Documents are close to unique: how do we condition on $\mathbf{t}$?
- Bayes' rule and independence assumptions

Bayes rule

Bayes rule determines how joint and conditional probabilities are related.

Bayes rule

Bayes rule determines how joint and conditional probabilities are related.

$$P(Y = y|X = x) = \frac{P(X = x|Y = y)P(Y = y)}{\sum_{y'} P(X = x|Y = y')P(Y = y')}$$

Bayes rule

Bayes rule determines how joint and conditional probabilities are related.

$$P(Y = y|X = x) = \frac{P(X = x|Y = y)P(Y = y)}{\sum_{y'} P(X = x|Y = y')P(Y = y')}$$

That is:

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

Prior and likelihood

- With Bayes' rule we can invert the direction of conditioning

Prior and likelihood

- With Bayes' rule we can invert the direction of conditioning

$$\begin{aligned}\hat{y} &= \operatorname*{argmax}_y \frac{P(Y=y)P(\mathbf{t}|Y=y)}{\sum_{y'} P(Y=y')P(\mathbf{t}|Y=y')} \\ &= \operatorname*{argmax}_y \frac{P(Y=y)P(\mathbf{t}|Y=y)}{Z} \\ &= \operatorname*{argmax}_y P(Y=y)P(\mathbf{t}|Y=y)\end{aligned}$$

Prior and likelihood

- With Bayes' rule we can invert the direction of conditioning

$$
\begin{aligned}
\hat{y} &= \operatorname*{argmax}_y \frac{P(Y=y)P(\mathbf{t}|Y=y)}{\sum_{y'} P(Y=y')P(\mathbf{t}|Y=y')} \\
&= \operatorname*{argmax}_y \frac{P(Y=y)P(\mathbf{t}|Y=y)}{Z} \\
&= \operatorname*{argmax}_y P(Y=y)P(\mathbf{t}|Y=y)
\end{aligned}
$$

- Decomposed the task into estimating the prior $P(Y)$ (easy) and the likelihood $P(\mathbf{t}|Y=y)$

Conditional independence

- How to estimate $P(\mathbf{t}|Y = y)$?

Conditional independence

- How to estimate $P(\mathbf{t}|Y = y)$?
- Naively assume the occurrence of each word in the document is independent of the others, when conditioned on the class

Conditional independence

- How to estimate $P(\mathbf{t}|Y = y)$?
- Naively assume the occurrence of each word in the document is independent of the others, when conditioned on the class

$$P(\mathbf{t}|Y = y) = \prod_{i=1}^{|\mathbf{t}|} P(t_i|Y = y)$$

Naive Bayes

Putting it all together

Naive Bayes

Putting it all together

$$\hat{y} = \operatorname*{argmax}_y P(Y = y) \prod_{i=1}^{|\mathbf{t}|} P(t_i | Y = y)$$

Decision function

- For binary classification:

Decision function

- For binary classification:

$$g(\mathbf{t}) = \frac{P(Y=+1)\prod_{i=1}^{|\mathbf{t}|} P(t_i|Y=+1)}{P(Y=-1)\prod_{i=1}^{|\mathbf{t}|} P(t_i|Y=-1)}$$

$$= \frac{P(Y=+1)}{P(Y=-1)} \prod_{i=1}^{|\mathbf{t}|} \frac{P(t_i|Y=+1)}{P(t_i|Y=-1)}$$

Decision function

- For binary classification:

$$g(\mathbf{t}) = \frac{P(Y=+1)\prod_{i=1}^{|\mathbf{t}|} P(t_i|Y=+1)}{P(Y=-1)\prod_{i=1}^{|\mathbf{t}|} P(t_i|Y=-1)}$$

$$= \frac{P(Y=+1)}{P(Y=-1)} \prod_{i=1}^{|\mathbf{t}|} \frac{P(t_i|Y=+1)}{P(t_i|Y=-1)}$$

$$\hat{y} = \begin{cases} +1 \text{ if } g(\mathbf{t}) \geq 1 \\ -1 \text{ otherwise} \end{cases}$$

Documents in vector notation

- Let's represent documents as vocabulary-size-dimensional binary vectors

Documents in vector notation

- Let's represent documents as vocabulary-size-dimensional binary vectors

	V_1	V_2	V_3	V_4	
	Obama	Ferrari	voters	movies	
$\mathbf{x} = ($	1	0	2	0	$)$

Documents in vector notation

- Let's represent documents as vocabulary-size-dimensional binary vectors

	V_1	V_2	V_3	V_4	
	Obama	Ferrari	voters	movies	
$\mathbf{x} = ($	1	0	2	0	$)$

- Dimension i indicates how many times the i^{th} vocabulary item appears in document $\mathbf{x}$

Naive Bayes in vector notation

Naive Bayes in vector notation

- Counts appear as exponents:

$$g(\mathbf{x}) = \frac{P(+1)}{P(-1)} \prod_{i=1}^{|V|} \left(\frac{P(V_i|+1)}{P(V_i|-1)} \right)^{x_i}$$

Naive Bayes in vector notation

- Counts appear as exponents:

$$g(\mathbf{x}) = \frac{P(+1)}{P(-1)} \prod_{i=1}^{|V|} \left(\frac{P(V_i|+1)}{P(V_i|-1)} \right)^{x_i}$$

- If we take the logarithm of the threshold ($\ln 1 = 0$) and of $g(x)$, we'll get the same decision function

Naive Bayes in vector notation

- Counts appear as exponents:

$$g(\mathbf{x}) = \frac{P(+1)}{P(-1)} \prod_{i=1}^{|V|} \left(\frac{P(V_i|+1)}{P(V_i|-1)} \right)^{x_i}$$

- If we take the logarithm of the threshold ($\ln 1 = 0$) and of $g(x)$, we'll get the same decision function

$$h(\mathbf{x}) = \ln \left(\frac{P(+1)}{P(-1)} \right) + \sum_{i=1}^{|V|} \ln \left(\frac{P(V_i|+1)}{P(V_i|-1)} \right) x_i$$

Linear classifier

- Remember the linear classifier?

Linear classifier

- Remember the linear classifier?

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i \, x_i$$

$$g(\mathbf{x}) = \ln\left(\frac{P(+1)}{P(-1)}\right) + \sum_{i=1}^{|V|} \ln\left(\frac{P(V_i|+1)}{P(V_i|-1)}\right) x_i$$

Linear classifier

- Remember the linear classifier?

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i \, x_i$$

$$g(\mathbf{x}) = \ln\left(\frac{P(+1)}{P(-1)}\right) + \sum_{i=1}^{|V|} \ln\left(\frac{P(V_i|+1)}{P(V_i|-1)}\right) x_i$$

- Log prior ratio corresponds to the bias term

Linear classifier

- Remember the linear classifier?

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i \quad x_i$$

$$g(\mathbf{x}) = \ln\left(\frac{P(+1)}{P(-1)}\right) + \sum_{i=1}^{|V|} \ln\left(\frac{P(V_i|+1)}{P(V_i|-1)}\right) x_i$$

- Log prior ratio corresponds to the bias term
- Log likelihood ratios correspond to feature weights

What is the difference

What is the difference

Training criterion and procedure

What is the difference

Training criterion and procedure

Perceptron

- Perceptron loss function

$$error(\mathbf{w}, D) = \sum_{(\mathbf{x},y)\in D} \begin{cases} 0 \text{ if } \operatorname{sign}(\mathbf{w}\cdot\mathbf{x}) = y \\ -yw \cdot x \text{ otherwise} \end{cases}$$

What is the difference

Training criterion and procedure

Perceptron

- Perceptron loss function

$$error(\mathbf{w}, D) = \sum_{(\mathbf{x},y)\in D} \begin{cases} 0 \text{ if } \operatorname{sign}(\mathbf{w} \cdot \mathbf{x}) = y \\ -yw \cdot x \text{ otherwise} \end{cases}$$

- Error-driven algorithm

Naive Bayes

Naive Bayes

- Maximum Likelihood criterion

Naive Bayes

- Maximum Likelihood criterion

$$P(D|\theta) = \prod_{(\mathbf{x},y) \in D} P(Y = y|\theta)P(x|Y = y, \theta)$$

Naive Bayes

- Maximum Likelihood criterion

$$P(D|\theta) = \prod_{(\mathbf{x},y)\in D} P(Y=y|\theta)P(x|Y=y,\theta)$$

- Find parameters which maximize the log likelihood

Naive Bayes

- Maximum Likelihood criterion

$$P(D|\theta) = \prod_{(\mathbf{x},y)\in D} P(Y = y|\theta)P(x|Y = y,\theta)$$

- Find parameters which maximize the log likelihood

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \log(P(D|\theta))$$

Naive Bayes

- Maximum Likelihood criterion

$$P(D|\theta) = \prod_{(\mathbf{x},y)\in D} P(Y = y|\theta)P(x|Y = y, \theta)$$

- Find parameters which maximize the log likelihood

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \log(P(D|\theta))$$

Parameters reduce to relative counts

Naive Bayes

- Maximum Likelihood criterion

$$P(D|\theta) = \prod_{(\mathbf{x},y)\in D} P(Y = y|\theta)P(x|Y = y, \theta)$$

- Find parameters which maximize the log likelihood

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \log(P(D|\theta))$$

  Parameters reduce to relative counts
- Ad-hoc smoothing, maximum *a posteriori*) estimation , ...

Comparison

Model	Naive Bayes	Perceptron
Model power	Linear	Linear
Type	Generative	Discriminative
Distribution modeled	$P(\mathbf{x}, y)$	N/A
Independence assumptions	Strong	None

Outline

Probabilistic conditional model

Probabilistic conditional model

- Let's try to come up with a probabilistic model which has some of the advantages of perceptron

Probabilistic conditional model

- Let's try to come up with a probabilistic model which has some of the advantages of perceptron
- Model $P(y|\mathbf{x})$ directly, and not via $P(\mathbf{x}, y)$ and Bayes rule as in Naive Bayes

Probabilistic conditional model

- Let's try to come up with a probabilistic model which has some of the advantages of perceptron
- Model $P(y|\mathbf{x})$ directly, and not via $P(\mathbf{x}, y)$ and Bayes rule as in Naive Bayes
- Avoid issue of dependencies between features of $\mathbf{x}$

Probabilistic conditional model

- Let's try to come up with a probabilistic model which has some of the advantages of perceptron
- Model $P(y|\mathbf{x})$ directly, and not via $P(\mathbf{x}, y)$ and Bayes rule as in Naive Bayes
- Avoid issue of dependencies between features of $\mathbf{x}$
- We'll take linear regression as a starting point

Probabilistic conditional model

- Let's try to come up with a probabilistic model which has some of the advantages of perceptron
- Model $P(y|\mathbf{x})$ directly, and not via $P(\mathbf{x}, y)$ and Bayes rule as in Naive Bayes
- Avoid issue of dependencies between features of $\mathbf{x}$
- We'll take linear regression as a starting point
  - The goal is to adapt regression to model class-conditional probability

Multiple linear regression

- Regression: $y = w_0 + \sum_{i=1}^{d} w_i x_i$, where
  - y = outcome
  - w_0 = intercept
  - $x_1..x_d$ = features vector and $w_1..w_d$ weight vector
  - More compact:

$$g(\mathbf{x}) = \sum_{i=0}^{d} w_i x_i = \mathbf{w} \cdot \mathbf{x}$$

Logistic regression

Logistic regression

- In logistic regression we use the linear model to assign probabilities to class labels

Logistic regression

- In logistic regression we use the linear model to assign probabilities to class labels
- For binary classification, predict $p = P(Y = 1|\mathbf{x})$. But predictions of linear regression model are $\in \mathbb{R}$, whereas $p \in [0, 1]$

Logistic function

- Instead predict logit function of the probability:

Logistic function

- Instead predict logit function of the probability:

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

Logistic function

- Instead predict logit function of the probability:

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

- After solving for p, we end up passing the dot product through the inverse logit or logistic or sigmoid function

Logistic function

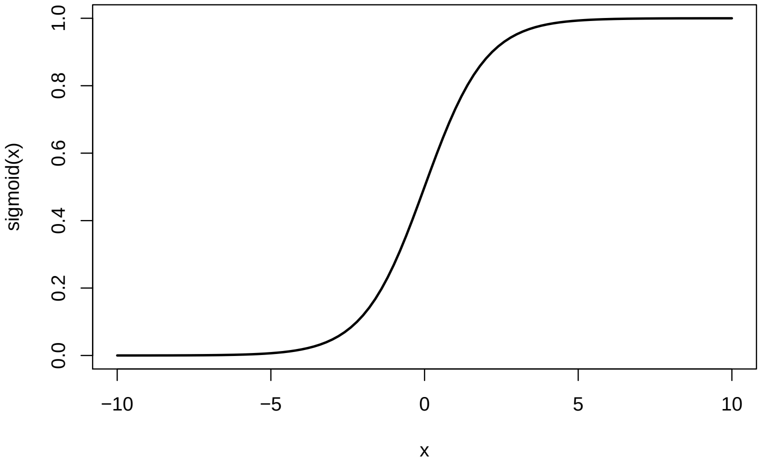
- Instead predict logit function of the probability:

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

- After solving for p, we end up passing the dot product through the inverse logit or logistic or sigmoid function

$$\begin{aligned} p &= \frac{\exp(\mathbf{w} \cdot \mathbf{x})}{1 + \exp(\mathbf{w} \cdot \mathbf{x})} \\ &= \frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x})} \end{aligned}$$

Logistic function



Logistic regression - classification

- Still a linear model

Logistic regression - classification

- Still a linear model
- Example $\mathbf{x}$ belongs to class 1 if:

$$\frac{p}{1-p} > 1$$

$$e^{\mathbf{w}\cdot\mathbf{x}} > 1$$

$$\mathbf{w} \cdot \mathbf{x} > 0$$

$$\sum_{i=0}^{d} w_i x_i > 0$$

Logistic regression - classification

- Still a linear model
- Example $\mathbf{x}$ belongs to class 1 if:

$$\frac{p}{1-p} > 1$$

$$e^{\mathbf{w}\cdot\mathbf{x}} > 1$$

$$\mathbf{w}\cdot\mathbf{x} > 0$$

$$\sum_{i=0}^{d} w_i x_i > 0$$

- Equation $\mathbf{w}\cdot\mathbf{x} = 0$ defines a hyperplane with points above belonging to class 1

Multinomial logistic regression

Logistic regression generalized to more than two classes

$$P(Y = y|\mathbf{x}, \mathbf{W}) = \frac{\exp(\mathbf{W}_{y\bullet} \cdot \mathbf{x})}{\sum_{y'} \exp(\mathbf{W}_{y'\bullet} \cdot \mathbf{x})}$$

Learning parameters

Learning parameters

- Conditional likelihood estimation: choose the weights which make the probability of the observed values y be the highest, given the observations $\mathbf{x}_i$

Learning parameters

- Conditional likelihood estimation: choose the weights which make the probability of the observed values y be the highest, given the observations $\mathbf{x}_i$
- For the training set with N examples:

Learning parameters

- Conditional likelihood estimation: choose the weights which make the probability of the observed values y be the highest, given the observations $\mathbf{x}_i$
- For the training set with N examples:

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}} \prod_{i=1}^{N} P(Y = y^{(n)} | \mathbf{x}^{(n)}, \mathbf{W})$$

$$= \underset{\mathbf{W}}{\operatorname{argmax}} \sum_{i=1}^{N} \log P(Y = y^{(n)} | \mathbf{x}^{(n)}, \mathbf{W})$$

Error function

Error function

Equivalently, we seek the value of the parameters which minimize the error function:

$$\mathrm{Err}(\mathbf{W}, D) = -\sum_{n=1}^{N} \log P(Y = y^{(n)}|\mathbf{x}^{(n)}, \mathbf{W})$$

Error function

Equivalently, we seek the value of the parameters which minimize the error function:

$$\mathrm{Err}(\mathbf{W}, D) = -\sum_{n=1}^{N} \log P(Y = y^{(n)}|\mathbf{x}^{(n)}, \mathbf{W})$$

where $D = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$

A problem in convex optimization

- L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno method)
- gradient descent
- conjugate gradient
- iterative scaling algorithms

Stochastic gradient descent

Stochastic gradient descent

Gradient descent

- A gradient is a slope of a function

Stochastic gradient descent

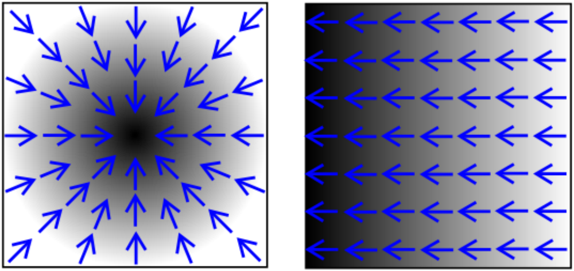
Gradient descent

- A gradient is a slope of a function
- That is, a set of partial derivatives, one for each dimension

Stochastic gradient descent

Gradient descent

- A gradient is a slope of a function
- That is, a set of partial derivatives, one for each dimension
- By following the gradient of a convex function we can descend to the bottom (minimum)



Gradient descent example

Gradient descent example

- Find $\operatorname{argmin}_\theta f(\theta)$ where $f(\theta) = \theta^2$

Gradient descent example

- Find $\operatorname{argmin}_\theta f(\theta)$ where $f(\theta) = \theta^2$
- Initial value of $\theta_1 = -1$

Gradient descent example

- Find $\operatorname{argmin}_\theta f(\theta)$ where $f(\theta) = \theta^2$
- Initial value of $\theta_1 = -1$
- Gradient function: $\nabla f(\theta) = 2\theta$

Gradient descent example

- Find $\operatorname{argmin}_\theta f(\theta)$ where $f(\theta) = \theta^2$
- Initial value of $\theta_1 = -1$
- Gradient function: $\nabla f(\theta) = 2\theta$
- Update: $\theta^{(n+1)} = \theta^{(n)} - \eta \nabla f(\theta^{(n)})$

Gradient descent example

- Find $\operatorname{argmin}_\theta f(\theta)$ where $f(\theta) = \theta^2$
- Initial value of $\theta_1 = -1$
- Gradient function: $\nabla f(\theta) = 2\theta$
- Update: $\theta^{(n+1)} = \theta^{(n)} - \eta\nabla f(\theta^{(n)})$
- The learning rate η $(= 0.2)$ controls the speed of the descent

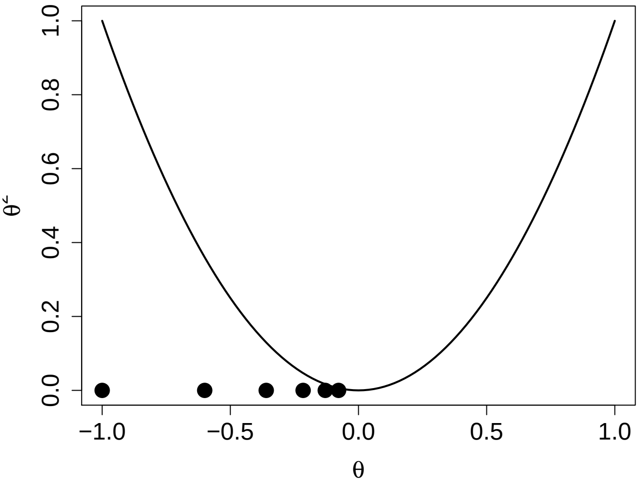
Gradient descent example

- Find $\operatorname{argmin}_\theta f(\theta)$ where $f(\theta) = \theta^2$
- Initial value of $\theta_1 = -1$
- Gradient function: $\nabla f(\theta) = 2\theta$
- Update: $\theta^{(n+1)} = \theta^{(n)} - \eta \nabla f(\theta^{(n)})$
- The learning rate η $(= 0.2)$ controls the speed of the descent
- After first iteration: $\theta^{(2)} = -1 - 0.2(-2) = -0.6$

Gradient descent example

- Find $\operatorname{argmin}_\theta f(\theta)$ where $f(\theta) = \theta^2$
- Initial value of $\theta_1 = -1$
- Gradient function: $\nabla f(\theta) = 2\theta$
- Update: $\theta^{(n+1)} = \theta^{(n)} - \eta \nabla f(\theta^{(n)})$
- The learning rate η $(= 0.2)$ controls the speed of the descent
- After first iteration: $\theta^{(2)} = -1 - 0.2(-2) = -0.6$
- After second iteration: $\theta^{(3)} = -0.6 - 0.2(-1.2) = -0.36$

Five iterations of gradient descent



Stochastic

- We could compute the gradient of error for the full dataset before each update

Stochastic

- We could compute the gradient of error for the full dataset before each update
- Instead

Stochastic

- We could compute the gradient of error for the full dataset before each update
- Instead
  - Compute the gradient of the error for a single example

Stochastic

- We could compute the gradient of error for the full dataset before each update
- Instead
  - Compute the gradient of the error for a single example
  - update the weight

Stochastic

- We could compute the gradient of error for the full dataset before each update
- Instead
  - Compute the gradient of the error for a single example
  - update the weight
  - Move on to the next example

Stochastic

- We could compute the gradient of error for the full dataset before each update
- Instead
  - Compute the gradient of the error for a single example
  - update the weight
  - Move on to the next example
- On average, we'll move in the right direction

Stochastic

- We could compute the gradient of error for the full dataset before each update
- Instead
  - Compute the gradient of the error for a single example
  - update the weight
  - Move on to the next example
- On average, we'll move in the right direction
- Efficient, online algorithm

Error gradient

- The gradient of the error function is the set of partial derivatives of the error function with respect to the parameters $\mathbf{W}_{yi}$

$$\nabla_{y,i}\mathrm{Err}(D,\mathbf{W}) = \frac{\partial}{\partial \mathbf{W}_{yi}} \left(-\sum_{n=1}^{N} \log P(Y = y|\mathbf{x}^{(n)}, \mathbf{W}) \right)$$
$$= -\sum_{n=1}^{N} \frac{\partial}{\partial \mathbf{W}_{yi}} \log P(Y = y|\mathbf{x}^{(n)}, \mathbf{W})$$

Update

- For the correct class ($y = y^{(n)}$)

$$\mathbf{W}_{yi}^{(n)} = \mathbf{W}_{yi}^{(n-1)} + \eta x_i^{(n)}(1 - P(Y = y|\mathbf{x}^{(n)}, \mathbf{W}))$$

  where $1 - P(Y = y|\mathbf{x}^{(n)}, \mathbf{W})$ is the residual

- For all other classes ($y \neq y^{(n)}$)

$$\mathbf{W}_{yi}^{(n)} = \mathbf{W}_{yi}^{(n-1)} - \eta x_i^{(n)} P(Y = y|\mathbf{x}^{(n)}, \mathbf{W})$$

Logistics Regression SGD vs Perceptron

$$\mathbf{w}^{(n)} = \mathbf{w}^{(n-1)} + 1\mathbf{x}^{(n)}$$

$$\mathbf{W}_{yi}^{(n)} = \mathbf{W}_{yi}^{(n-1)} + \eta x_i^{(n)} \quad (1 - P(Y = y|\mathbf{x}^{(n)}, \mathbf{W}))$$

- Very similar update!
- Perceptron is simply an instantiation of SGD for a particular error function
- The perceptron criterion: for a correctly classified example zero error; for a misclassified example $-y^{(n)}\mathbf{w} \cdot \mathbf{x}^{(n)}$

Comparison

Model	Naive Bayes	Perceptron	Log. regression
Model power	Linear	Linear	Linear
Type	Generative	Discriminative	Discriminative
Distribution	$P(\mathbf{x}, y)$	N/A	$P(y\|\mathbf{x})$
Independence	Strong	None	None

The end

Efficient averaged perceptron algorithm

$\textsc{Perceptron}(x^{1:N}, y^{1:N}, I)$:

$\mathbf{w} \leftarrow \mathbf{0}$; $\mathbf{w_a} \leftarrow \mathbf{0}$
$b \leftarrow 0$; $b_a \leftarrow 0$
$c \leftarrow 1$
for $i = 1...I$ **do**
  for $n = 1...N$ **do**
    if $y^{(n)}(\mathbf{w} \cdot \mathbf{x}^{(n)} + b) \leq 0$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + y^{(n)}\mathbf{x}^{(n)}$; $b \leftarrow b + y^{(n)}$
      $\mathbf{w_a} \leftarrow \mathbf{w_a} + cy^{(n)}\mathbf{x}^{(n)}$; $b_a \leftarrow b_a + cy^{(n)}$
    $c \leftarrow c + 1$
return $(\mathbf{w} - \mathbf{w_a}/c, b - b_a/c)$