F20DL and F21DL:
Part 2: Machine Learning
Lecture 6: Linear Classifiers

Katya Komendantskaya

So far

... last time we covered:

- Supervised Learning.
- This kind of learning is about finding a good hypothesis in the hypothesis space, with the purpose of making (class) predictions.
- Hypotheses were given by decision trees.

So far

... last time we covered:

- Supervised Learning.
- This kind of learning is about finding a good hypothesis in the hypothesis space, with the purpose of making (class) predictions.
- Hypotheses were given by decision trees.

Today:
We look at another kind of a hypothesis used in classification – linear function.

Linear regression

... is the problem of fitting a linear function to a set of input-output pairs given by a set of training examples, in which the input and output features are numeric.

Linear regression

... is the problem of fitting a linear function to a set of input-output pairs given by a set of training examples, in which the input and output features are numeric.

Suppose the input (non-class) features are $X_1, \ldots, X_n$. A linear function of these features is a function of the form:

$$f(X_1, \ldots, X_n) = w_0 + w_1 \times X_1 + \ldots + w_n \times X_n,$$

where $w_0, \ldots w_n$ are weights.

Linear regression

... is the problem of fitting a linear function to a set of input-output pairs given by a set of training examples, in which the input and output features are numeric.
Suppose the input (non-class) features are $X_1, \ldots, X_n$. A linear function of these features is a function of the form:

$$f(X_1, \ldots, X_n) = w_0 + w_1 \times X_1 + \ldots + w_n \times X_n,$$

where $w_0, \ldots w_n$ are weights.
Suppose E is a set of examples, where each example $e \in E$ has values $val(e, X_i)$ for feature X_i and has an observed value $val(e, Y)$. The predicted value is
$pval^{\overline{w}}(e, Y) = w_0 + w_1 \times val(e, X_1) + \ldots + w_n \times val(e, X_n)$

Linear regression

... is the problem of fitting a linear function to a set of input-output pairs given by a set of training examples, in which the input and output features are numeric.

Suppose the input (non-class) features are $X_1, \ldots, X_n$. A linear function of these features is a function of the form:

$$f(X_1, \ldots, X_n) = w_0 + w_1 \times X_1 + \ldots + w_n \times X_n,$$

where $w_0, \ldots w_n$ are weights.

Suppose E is a set of examples, where each example $e \in E$ has values $val(e, X_i)$ for feature X_i and has an observed value $val(e, Y)$. The predicted value is

$pval^{\overline{w}}(e, Y) = w_0 + w_1 \times val(e, X_1) + \ldots + w_n \times val(e, X_n)$

$$= \sum_{i=0}^{n}(w_i \times val(e, X_i)), \text{ with } val(e, X_0) = 1.$$

Examples: Customer transactions

Trans.	Music on CD?	Music on MP3?	Board Games	On-line Games	Output
T1	No	Yes	No	Yes	Buys
T2	Yes	No	No	No	Cancels
T3	Yes	No	No	Yes	Buys
T4	Yes	No	Yes	No	Cancels
T5	No	Yes	No	No	Cancels
T6	No	Yes	Yes	No	Cancels
T7	No	No	No	Yes	Buys
T8	No	Yes	Yes	Yes	Cancels
T9	Yes	Yes	No	No	Cancels
T10	Yes	Yes	No	Yes	Buys

Customer transactions, converted to numeric

Trans.	Music on CD?	Music on MP3?	Board Games	On-line Games	Output
T1	0	1	0	1	1
T2	1	0	0	0	0
T3	1	0	0	1	1
T4	1	0	1	0	0
T5	0	1	0	0	0
T6	0	1	1	0	0
T7	0	0	0	1	1
T8	0	1	1	1	0
T9	1	1	0	0	0
T10	1	1	0	1	1

Examples

For our favourite data set, we want to find a function

Example

$$f(musicCD, musicMP3, gamesHard, gamesOnline) =$$

$$w_0 + w_1 \times musicCD + w_2 \times musicMP3 + w_3 \times gamesHard + w_4 \times gamesOnline$$

Examples

For our favourite data set, we want to find a function

Example

$f(musicCD, musicMP3, gamesHard, gamesOnline) =$

$w_0 + w_1 \times musicCD + w_2 \times musicMP3 + w_3 \times gamesHard + w_4 \times gamesOnline$

We need to LEARN: w_0, w_1, w_2, w_3, w_4

Examples

For our favourite data set, we want to find a function

Example

$f(musicCD, musicMP3, gamesHard, gamesOnline) =$

$w_0 + w_1 \times musicCD + w_2 \times musicMP3 + w_3 \times gamesHard + w_4 \times gamesOnline$

We need to LEARN: w_0, w_1, w_2, w_3, w_4
For it, we can predict a class for a new example e using
$pval^{\overline{w}}(e, Buys) =$
$w_0 + w_1 \times val(e, musicCD) + w_2 \times val(e, musicMP3) + w_3 \times val(e, gamesHard) + w_4 \times val(e, gamesOnline)$

Computing an error

Whatever w_0, w_1, w_2, w_3, w_4 are, we want them to minimize the error between actual and predicted classes.

Computing an error

Whatever w_0, w_1, w_2, w_3, w_4 are, we want them to minimize the error between actual and predicted classes.

- Absolute Error:

$$Error(\bar{w}) = \sum_{e \in E} val(e, Y) - pval(e, Y)$$

Computing an error

Whatever w_0, w_1, w_2, w_3, w_4 are, we want them to minimize the error between actual and predicted classes.

- Absolute Error:

$$Error(\bar{w}) = \sum_{e \in E} val(e, Y) - pval(e, Y)$$

- Sum of squared errors ("sum-of-squares"):

$$Error(\bar{w}) = \sum_{e \in E} (val(e, Y) - pval(e, Y))^2$$

Finding weights that minimize $Error(\overline{w})$

- Find the minimum analytically.
  Effective when it can be done.

Finding weights that minimize $Error(\overline{w})$

- Find the minimum analytically.
  Effective when it can be done.
- Find the minimum iteratively.
  Works for larger classes of problems.
  Gradient descent: starts with random values and changes each weight in proportion to the partial derivative of the error for the weight:

  $$w_i = w_i - \eta \frac{\partial Error_E(\overline{w})}{\partial w_i}$$

  η is the gradient descent step size, the learning rate.

Finding weights that minimize $Error(\overline{w})$

- Find the minimum analytically.
  Effective when it can be done.
- Find the minimum iteratively.
  Works for larger classes of problems.
  Gradient descent: starts with random values and changes each weight in proportion to the partial derivative of the error for the weight:

$$w_i = w_i - \eta \frac{\partial Error_E(\overline{w})}{\partial w_i}$$

  η is the gradient descent step size, the learning rate.

How to compute the derivative

$$\frac{\partial Error_E(\overline{w})}{\partial w_i}?$$

– depends on the chosen error function.

Some maths calculations

Gradient descent that minimizes the sum of squares error $Error(\bar{w}) = \sum_{e \in E}(val(e, Y) - pval(e, Y))^2$

- Partial derivative of the sum is the sum of partial derivatives – so consider each example e in turn

Some maths calculations

Gradient descent that minimizes the sum of squares error $Error(\bar{w}) = \sum_{e \in E}(val(e, Y) - pval(e, Y))^2$

- Partial derivative of the sum is the sum of partial derivatives – so consider each example e in turn
- Given example e, $Error(\bar{w}) = (val(e, Y) - pval(e, Y))^2$, so we take the derivative: $((val(e, Y) - pval(e, Y))^2)'$

Lets calculate this derivative by hand.

Some maths calculations

NB: Standard rules for computing derivatives

1. $(Cx)' = C$, for constant C
2. $C' = 0$, for constant C
3. $(x^2)' = 2x$
4. $(h(g(x)))' = h'(g(x)) \times g'(x)$
5. $(f \times g)' = f'g + fg'$
6. $(\alpha f + \beta g)' = \alpha f' + \beta g'$, for constants α, β

Note that we compute the partial derivative with respect to a weight w_i:
$((val(e, Y) - pval(e, Y))^2)' = \ldots$
exercise on the Board

Some maths calculations

- partial derivative of the error wrt the weight w_i:
$((val(e, Y) - pval(e, Y))^2)' =$

$$-2 \times [val(e, Y) - pval^{\overline{w}}(e, Y)] \times val(e, X_i)$$

Some maths calculations

- partial derivative of the error wrt the weight w_i:
  $((val(e, Y) - pval(e, Y))^2)' =$

$$-2 \times [val(e, Y) - pval^{\overline{w}}(e, Y)] \times val(e, X_i)$$

- For each example, let $\delta = val(e, Y) - pval^{\overline{w}}(e, Y)$

Some maths calculations

- partial derivative of the error wrt the weight w_i:
  $((val(e,Y) - pval(e,Y))^2)' =$

  $$-2 \times [val(e,Y) - pval^{\overline{w}}(e,Y)] \times val(e,X_i)$$

- For each example, let $\delta = val(e,Y) - pval^{\overline{w}}(e,Y)$
- We needed to compute an update for each weight:

  $$w_i = w_i - \eta \frac{\partial Error_E(\overline{w})}{\partial w_i}$$

- So $w_i := w_i + \eta \times \delta \times val(e,X_i)$ for a constant learning rate η (assume 2 is absorbed by η)

Gradient Descent

```
Algorithm LinearLearner(X,Y,E,η)
Inputs:
    X: set of input features, X = {X_1, ..., X_n}
    Y: target feature
    E: set of examples from which to learn
    η: - learning rate.
```

Gradient Descent

1: **Algorithm** LinearLearner(X,Y,E,η)
2: **Inputs:**
3: X: set of input features, $X = \{X_1, \ldots, X_n\}$
4: Y: target feature
5: E: set of examples from which to learn
6: η: - learning rate.
7: **Output:** parameters $w_0, \ldots, w_n$.

Gradient Descent

Algorithm LinearLearner(X,Y,E,η)
Inputs:
    X: set of input features, $X = \{X_1, \ldots, X_n\}$
    Y: target feature
    E: set of examples from which to learn
    η: - learning rate.
Output: parameters $w_0, \ldots, w_n$.
    Local $w_0, \ldots, w_n$ - real numbers
    $pval(e, Y) = w_0 + w_1 \times val(e, X_1) + \ldots + w_n \times val(e, X_n)$

Gradient Descent

Algorithm LinearLearner(X,Y,E,η)
Inputs:
    X: set of input features, $X = \{X_1, \ldots, X_n\}$
    Y: target feature
    E: set of examples from which to learn
    η: - learning rate.
Output: parameters $w_0, \ldots, w_n$.
    Local $w_0, \ldots, w_n$ - real numbers
    $pval(e, Y) = w_0 + w_1 \times val(e, X_1) + \ldots + w_n \times val(e, X_n)$
initialise $w_0, \ldots, w_n$ randomly
repeat
    for each example e in E **do**
        $\delta := val(e, Y) - pval(e, Y)$
        for each $i \in [0, n]$ **do**
            $w_i = w_i + \eta \times \delta \times val(e, X_i)$
until termination
return $w_0, \ldots, w_n$

Gradient Descent

Algorithm LinearLearner(X,Y,E,η)
Inputs:
X: set of input features, $X = \{X_1, \ldots, X_n\}$
Y: target feature
E: set of examples from which to learn
η: - learning rate.
Output: parameters $w_0, \ldots, w_n$.
Local $w_0, \ldots, w_n$ - real numbers
$pval(e, Y) = w_0 + w_1 \times val(e, X_1) + \ldots + w_n \times val(e, X_n)$
initialise $w_0, \ldots, w_n$ randomly
repeat
for each example e in E **do**
$\delta := val(e, Y) - pval(e, Y)$
for each $i \in [0, n]$ **do**
$w_i = w_i + \eta \times \delta \times val(e, X_i)$
until termination
return $w_0, \ldots, w_n$

Gradient Descent

Algorithm LinearLearner(X,Y,E,η)
Inputs:
    X: set of input features, $X = \{X_1, \ldots, X_n\}$
    Y: target feature
    E: set of examples from which to learn
    η: - learning rate.
Output: parameters $w_0, \ldots, w_n$.
    Local $w_0, \ldots, w_n$ - real numbers
    $pval(e, Y) = w_0 + w_1 \times val(e, X_1) + \ldots + w_n \times val(e, X_n)$
initialise $w_0, \ldots, w_n$ randomly
repeat
    for each example e in E **do**
        $\delta := val(e, Y) - pval(e, Y)$
        for each $i \in [0, n]$ **do**
            $w_i = w_i + \eta \times \delta \times val(e, X_i)$
until termination
return $w_0, \ldots, w_n$

Gradient Descent

Algorithm LinearLearner(X,Y,E,η)

Inputs:

X: set of input features, $X = \{X_1, \ldots, X_n\}$

Y: target feature

E: set of examples from which to learn

η: - learning rate.

Output: parameters $w_0, \ldots, w_n$.

Local $w_0, \ldots, w_n$ - real numbers

$pval(e, Y) = w_0 + w_1 \times val(e, X_1) + \ldots + w_n \times val(e, X_n)$

initialise $w_0, \ldots, w_n$ randomly

repeat

for each example e in E **do**

$\delta := val(e, Y) - pval(e, Y)$

for each $i \in [0, n]$ **do**

$w_i = w_i + \eta \times \delta \times val(e, X_i)$

until termination

return $w_0, \ldots, w_n$

Gradient Descent

Algorithm LinearLearner(X,Y,E,η)
Inputs:
    X: set of input features, $X = \{X_1, \ldots, X_n\}$
    Y: target feature
    E: set of examples from which to learn
    η: - learning rate.
Output: parameters $w_0, \ldots, w_n$.
    Local $w_0, \ldots, w_n$ - real numbers
    $pval(e, Y) = w_0 + w_1 \times val(e, X_1) + \ldots + w_n \times val(e, X_n)$
initialise $w_0, \ldots, w_n$ randomly
repeat
    for each example e in E **do**
        $\delta := val(e, Y) - pval(e, Y)$
        for each $i \in [0, n]$ **do**
            $w_i = w_i + \eta \times \delta \times val(e, X_i)$
until termination
return $w_0, \ldots, w_n$

Gradient Descent

Algorithm LinearLearner(X,Y,E,η)
Inputs:
  X: set of input features, $X = \{X_1, \ldots, X_n\}$
  Y: target feature
  E: set of examples from which to learn
  η: - learning rate.
Output: parameters $w_0, \ldots, w_n$.
  Local $w_0, \ldots, w_n$ - real numbers
  $pval(e, Y) = w_0 + w_1 \times val(e, X_1) + \ldots + w_n \times val(e, X_n)$
initialise $w_0, \ldots, w_n$ randomly
repeat
  for each example e in E **do**
    $\delta := val(e, Y) - pval(e, Y)$
    for each $i \in [0, n]$ **do**
      $w_i = w_i + \eta \times \delta \times val(e, X_i)$
until termination
return $w_0, \ldots, w_n$

Worked-out example

Lets start with: $pval^{\overline{w}}(e, Buys) =$
$w_0 + w_1 \times val(e, musicCD) + w_2 \times val(e, musicMP3) + w_3 \times val(e, gamesHard) + w_4 \times val(e, gamesOnline)$

Trans.	musicCD	musicMP3	gamesHard	gamesOnline	Buys
T1	0	1	0	1	1
T2	1	0	0	0	0
T3	1	0	0	1	1

- Take $\eta = 1$
- $\delta := val(e, Y) - pval(e, Y)$
- $w_i = w_i + \eta \times \delta \times val(e, X_i)$
- take random weights:
  $pval(e, Buys) = 1 + 2 \times val(e, musicCD) + 3 \times val(e, musicMP3) + 1 \times val(e, gamesHard) + 2 \times val(e, gamesOnline)$

Worked-out example

Trans.	musicCD	musicMP3	gamesHard	gamesOnline	Buys
T1	0	1	0	1	1

- $\delta := val(e, Y) - pval(e, Y)$
- take random weights: $pval(e, Buys) = 1 + 2 \times val(e, musicCD) + 3 \times val(e, musicMP3) + 1 \times val(e, gamesHard) + 2 \times val(e, gamesOnline)$

Example $T1$:

- $\delta = 1-$

Worked-out example

Trans.	musicCD	musicMP3	gamesHard	gamesOnline	Buys
T1	0	1	0	1	1

- $\delta := val(e, Y) - pval(e, Y)$
- take random weights:
  $pval(e, Buys) = 1 + 2 \times val(e, musicCD) + 3 \times val(e, musicMP3) + 1 \times val(e, gamesHard) + 2 \times val(e, gamesOnline)$

Example $T1$:

- $\delta = 1-\ (1 + 0 + 3 + 0 + 2) = 1 - 6 = -5$

Worked-out example

Trans.	musicCD	musicMP3	gamesHard	gamesOnline	Buys
T1	0	1	0	1	1

- Take $\eta = 1$
- $w_i = w_i + \eta \times \delta \times val(e, X_i)$
- take random weights:
  $1 + 2 \times val(e, musicCD) + 3 \times val(e, musicMP3) + 1 \times val(e, gamesHard) + 2 \times val(e, gamesOnline)$

- $\delta = -5$
- $w_0 =$

Worked-out example

Trans.	musicCD	musicMP3	gamesHard	gamesOnline	Buys
T1	0	1	0	1	1

- Take $\eta = 1$
- $w_i = w_i + \eta \times \delta \times val(e, X_i)$
- take random weights:
  $1 + 2 \times val(e, musicCD) + 3 \times val(e, musicMP3) + 1 \times val(e, gamesHard) + 2 \times val(e, gamesOnline)$

- $\delta = -5$
- $w_0 = 1 - 5 = -4;\ w_1 =$

Worked-out example

Trans.	musicCD	musicMP3	gamesHard	gamesOnline	Buys
T1	0	1	0	1	1

- Take $\eta = 1$
- $w_i = w_i + \eta \times \delta \times val(e, X_i)$
- take random weights:
  $1 + 2 \times val(e, musicCD) + 3 \times val(e, musicMP3) + 1 \times val(e, gamesHard) + 2 \times val(e, gamesOnline)$

- $\delta = -5$
- $w_0 = 1 - 5 = -4;\ w_1 = 2 + 0 = 2;\ w_2 = 3 - 5 = -2;\ w_3 =$

Worked-out example

Trans.	musicCD	musicMP3	gamesHard	gamesOnline	Buys
T1	0	1	0	1	1

- Take $\eta = 1$
- $w_i = w_i + \eta \times \delta \times val(e, X_i)$
- take random weights: $1 + 2 \times val(e, musicCD) + 3 \times val(e, musicMP3) + 1 \times val(e, gamesHard) + 2 \times val(e, gamesOnline)$

- $\delta = -5$
- $w_0 = 1 - 5 = -4$; $w_1 = 2 + 0 = 2$; $w_2 = 3 - 5 = -2$; $w_3 = 1 + 0 = 1$; $w_4 =$

Worked-out example

Trans.	musicCD	musicMP3	gamesHard	gamesOnline	Buys
T1	0	1	0	1	1

- Take $\eta = 1$
- $w_i = w_i + \eta \times \delta \times val(e, X_i)$
- take random weights:
  $1 + 2 \times val(e, musicCD) + 3 \times val(e, musicMP3) + 1 \times val(e, gamesHard) + 2 \times val(e, gamesOnline)$

- $\delta = -5$
- $w_0 = 1 - 5 = -4$; $w_1 = 2 + 0 = 2$; $w_2 = 3 - 5 = -2$; $w_3 = 1 + 0 = 1$; $w_4 = 2 - 5 = -3$.

Worked-out example

Trans.	musicCD	musicMP3	gamesHard	gamesOnline	Buys
T1	0	1	0	1	1
T2	1	0	0	0	0

- Take $\eta = 1$
- $\delta := val(e, Y) - pval(e, Y)$
- UPDATE weights:
  $-4 + 2 \times val(e, musicCD) + -2 \times val(e, musicMP3) + 1 \times val(e, gamesHard) + -3 \times val(e, gamesOnline)$

Start all over again, for Example $T2$:

- $\delta = 0-$

Worked-out example

Trans.	musicCD	musicMP3	gamesHard	gamesOnline	Buys
T1	0	1	0	1	1
T2	1	0	0	0	0

- Take $\eta = 1$
- $\delta := val(e, Y) - pval(e, Y)$
- UPDATE weights:
  $-4 + 2 \times val(e, musicCD) + -2 \times val(e, musicMP3) + 1 \times val(e, gamesHard) + -3 \times val(e, gamesOnline)$

Start all over again, for Example $T2$:

- $\delta = 0- \ (-4 + 2) = 0 + 2 = 2$ Repeat weight update with new δ

Worked-out example

Trans.	musicCD	musicMP3	gamesHard	gamesOnline	Buys
T1	0	1	0	1	1
T2	1	0	0	0	0

- Take $\eta = 1$
- $w_i = w_i + \eta \times \delta \times val(e, X_i)$
- Current weights:
  $-4 + 2 \times val(e, musicCD) + (-2) \times val(e, musicMP3) + 1 \times val(e, gamesHard) + (-3) \times val(e, gamesOnline)$

- $\delta = 2$
- $w_0 = -4 + 2 = -2$; $w_1 = 2 + 2 = 4$; $w_2 = -2 + 0 = -2$; $w_3 = 1 + 0 = 1$; $w_4 = -3 + 0 = -3$.

Worked-out example

Trans.	musicCD	musicMP3	gamesHard	gamesOnline	Buys
T1	0	1	0	1	1
T2	1	0	0	0	0

- Take $\eta = 1$
- $w_i = w_i + \eta \times \delta \times val(e, X_i)$
- NEW weights:
  $-2 + 4 \times val(e, musicCD) + (-2) \times val(e, musicMP3) + 1 \times val(e, gamesHard) + (-3) \times val(e, gamesOnline)$

Now we got to example $T3$ and would repeat the same iteration for $T3$...

Example conclusions

- Iterate like this using all your examples
- Repeat again on all examples...
- until you meet the termination condition
- Usually given by some accuracy measure on all examples: $val - pval$ can be set arbitrary small.

Test 3. Linear Regression Component

- Take the small emotion recognition set from Lecture 1
- Convert it to numeric form: Black $\rightarrow$ 1, White $\rightarrow$ 0, Happy $\rightarrow$ 1, Sad $\rightarrow$ 0
- Execute the Linear Regression algorithm for it, taking first three examples in turn
- Random weight initialisation: $w_0 = 1$, $w_1 = 2$, $w_2 = 1$, $w_3 = -2$, $w_4 = -1$.
- Record your results, as well as intermediate values in the computation, be ready to answer questions.