1

00:00:07,290 --> 00:00:11,068

[SOUND] This lecture is

a continued discussion of

2

00:00:11,068 --> 00:00:15,610

Discriminative Classifiers for

Text Categorization.

3

00:00:15,610 --> 00:00:18,096

So, in this lecture,

we're going to introduce, yet

4

00:00:18,096 --> 00:00:22,450

another Discriminative Classifier called

the Support Vector Machine or SVM.

5

00:00:22,450 --> 00:00:25,050

Which is a very popular

classification method and

6

00:00:25,050 --> 00:00:28,790

it has been also shown to be effective for

text categorization.

7

00:00:31,350 --> 00:00:34,380

So to introduce this classifier,

8

00:00:34,380 --> 00:00:38,060

let's also think about the simple

case of two categories.

9

00:00:38,060 --> 00:00:43,300

We have two topic categories,

01 and 02 here.

10

00:00:43,300 --> 00:00:47,760

And we want to classify documents

into these two categories and

11

00:00:47,760 --> 00:00:51,820

we're going to represent again

a document by a feature factor x here.

12

00:00:53,200 --> 00:00:58,020

Now, the idea of this classifier is

to design also a linear separator

13

00:00:59,150 --> 00:01:01,360

here that you'll see and

14

00:01:01,360 --> 00:01:05,820

it's very similar to what you have

seen not just for regression, right?

15

00:01:05,820 --> 00:01:11,240

And we're going to do also say

that if the sign of this function

16

00:01:11,240 --> 00:01:16,690

value is positive then we're going to

say the objective is in category one.

17

00:01:16,690 --> 00:01:20,470

Otherwise, we're going to

say it's in category 2.

18

00:01:20,470 --> 00:01:27,700

So that makes 0 that is the decision

boundary between the few categories.

19

00:01:28,830 --> 00:01:33,990

So, in generally hiding

marginal space such as, 0.

20

00:01:33,990 --> 00:01:37,070

corresponds to a hyper plain.

21

00:01:38,210 --> 00:01:43,180

Now I've shown you a simple case of two

dimensional space it was just X1 and

22

00:01:43,180 --> 00:01:49,910

X2 and this case this corresponds

to a line that you can see here.

23

00:01:51,220 --> 00:01:55,980

So, this is a line defined by

24

00:01:55,980 --> 00:02:00,970

just three parameters here,

beta zero, beta one, and beta two.

25

00:02:02,390 --> 00:02:07,320

Now, this line is heading

in this direction so

26

00:02:07,320 --> 00:02:13,450

it shows that as we increase X1,

X2 will also increase.

27

00:02:13,450 --> 00:02:17,780

So we know that beta one and beta two have

different assigns, one is negative and

28

00:02:17,780 --> 00:02:18,920

the other is positive.

29

00:02:20,800 --> 00:02:26,790

So let's just assume that beta one is

negative and beta two Is positive.

30

00:02:28,810 --> 00:02:31,250

Now, it's interesting to examine, then,

31

00:02:31,250 --> 00:02:34,800

the data instances on

the two sides of the slide.

32

00:02:34,800 --> 00:02:39,690

So, here, the data instance are visualized

as circles for one class and

33

00:02:39,690 --> 00:02:41,800

diamonds for the other class.

34

00:02:43,140 --> 00:02:49,090

Now, one question is to take a point

like this one and to ask the question

35

00:02:49,090 --> 00:02:54,110

what's the value of this expression, or

this classifier, for this data point?

36

00:02:55,350 --> 00:02:57,000

So what do you think?

37

00:02:57,000 --> 00:03:00,650

Basically, we're going to evaluate

its value by using this function.

38

00:03:01,740 --> 00:03:06,190

And as we said, if this value's positive

we're going to say this is in category

39

00:03:06,190 --> 00:03:09,610

one, and if it's negative,

it's going to be in category two.

40

00:03:09,610 --> 00:03:15,343

Intuitively, this line separates these two

categories, so we expect the points on

41

00:03:15,343 --> 00:03:19,870

one side would be positive and the points

on the other side would be negative.

42

00:03:19,870 --> 00:03:23,200

Our question is under the assumption

that I just mentioned,

43

00:03:23,200 --> 00:03:25,320

let's examine a particular

point like this one.

44

00:03:27,590 --> 00:03:30,480

So what do you think is

the sine of this expression?

45

00:03:31,610 --> 00:03:37,830

Well, to examine the sine we can

simply look at this expression here.

46

00:03:37,830 --> 00:03:40,950

And we can compare this with let's say,

47

00:03:42,050 --> 00:03:46,950

value on the line, let's see,

compare this with this point.

48

00:03:48,440 --> 00:03:53,520

While they have identical X1, but

then one has a higher value for X2.

49

00:03:54,740 --> 00:03:59,790

Now, let's look at the sin

of the coefficient for X2.

50

00:03:59,790 --> 00:04:01,610

Well, we know this is a positive.

51

00:04:02,850 --> 00:04:06,260

So, what that means is

that the f value for

52

00:04:06,260 --> 00:04:10,400

this point should be higher

than the f value for

53

00:04:10,400 --> 00:04:14,800

this point on the line that means

this will be positive, right?

54

00:04:16,190 --> 00:04:19,900

So we know in general of

all points on this side,

55

00:04:20,960 --> 00:04:25,380

the function's value will be positive and

56

00:04:25,380 --> 00:04:29,380

you can also verify all the points

on this side will be negative.

57

00:04:29,380 --> 00:04:31,750

And so this is how this kind

of linear classifier or

58

00:04:31,750 --> 00:04:35,940

linear separator can then separate

the points in the two categories.

59

00:04:37,810 --> 00:04:42,830

So, now the natural question is,

which linear separator is the best?

60

00:04:42,830 --> 00:04:47,687

Now, I've get you one line here

that can separate the two classes.

61

00:04:47,687 --> 00:04:53,190

And this line, of course, is determined

by the vector beta, the coefficients.

62

00:04:53,190 --> 00:04:55,210

Different coefficients will

give us different lines.

63

00:04:55,210 --> 00:04:58,770

So, we could imagine there are other

lines that can do the same job.

64

00:04:58,770 --> 00:05:00,630

Gamma, for example,

65

00:05:00,630 --> 00:05:04,860

could give us another line that counts

a separator to these instances.

66

00:05:06,010 --> 00:05:09,710

Of course, there are also lines that won't

separate to them and those are bad lines.

67

00:05:09,710 --> 00:05:12,310

But, the question is,

when we have multiple lines that can

68

00:05:12,310 --> 00:05:15,950

separate both clauses,

which align the best?

69

00:05:15,950 --> 00:05:21,740

In fact, you can imagine, there are many

different ways of choosing the line.

70

00:05:21,740 --> 00:05:27,310

So, the logistical regression classifier

that you have seen earlier actually uses

71

00:05:27,310 --> 00:05:33,060

some criteria to determine where this line

should be and so linear separate as well.

72

00:05:33,060 --> 00:05:36,610

And uses a conditional likelihood

on the training that it determines

73

00:05:36,610 --> 00:05:38,310

which line is the best.

74

00:05:38,310 --> 00:05:41,130

But in SVM we're going to

look at another criteria for

75

00:05:41,130 --> 00:05:43,500

determining which line is the best.

76

00:05:43,500 --> 00:05:44,230

And this time,

77

00:05:44,230 --> 00:05:48,300

the criteria is more tied to

the classification arrow as you will see.

78

00:05:49,460 --> 00:05:56,120

So, the basic idea is to choose

the separator to maximize the margin.

79

00:05:56,120 --> 00:05:57,180

So what is a margin?

80

00:05:57,180 --> 00:06:03,540

So, I choose some dotted

lines here to indicate

81

00:06:03,540 --> 00:06:09,020

the boundaries of those

data points in each class.

82

00:06:09,020 --> 00:06:13,890

And the margin is simply

the distance between the line,

83

00:06:13,890 --> 00:06:17,420

the separator, and

the closest point from each class.

84

00:06:18,490 --> 00:06:23,830

So you can see the margin of this

side is as I've shown here and

85

00:06:23,830 --> 00:06:25,810

you can also define

the margin on the other side.

86

00:06:27,020 --> 00:06:31,190

In order for

the separator to maximize the margin,

87

00:06:31,190 --> 00:06:35,700

it has to be kind of in the middle

of the two boundaries and

88

00:06:35,700 --> 00:06:40,050

you don't want this separator to

be very close to one side, and

89

00:06:40,050 --> 00:06:42,800

that in intuition makes a lot of sense.

90

00:06:44,460 --> 00:06:47,050

So this is basic idea of SVM.

91

00:06:47,050 --> 00:06:50,020

We're going to choose a linear

separator to maximize the margin.

92

00:06:52,130 --> 00:06:55,450

Now on this slide,

I've also changed the notation so

93

00:06:55,450 --> 00:06:58,460

that I'm not going to use beta

to denote the parameters.

94

00:06:58,460 --> 00:07:03,740

But instead, I'm going to use w although

w was used to denote the words before so

95

00:07:03,740 --> 00:07:05,370

don't be confused here.

96

00:07:05,370 --> 00:07:09,618

W here is actually a width,

a certain width.

97

00:07:12,734 --> 00:07:19,030

So I'm also using lowercase b to

denote the beta 0, a biased constant.

98

00:07:20,030 --> 00:07:24,100

And there are instances do

represent that as x and

99

00:07:24,100 --> 00:07:28,790

I also use the vector form

of multiplication here.

100

00:07:28,790 --> 00:07:34,110

So we see a transpose of w vector

multiply by the future vector.

101

00:07:35,290 --> 00:07:42,080

So b is a bias constant and w is a set of

weights with one way for each feature.

102

00:07:42,080 --> 00:07:45,260

We have m features and

so we have m weights and

103

00:07:45,260 --> 00:07:46,420

that will represent as a vector.

104

00:07:47,640 --> 00:07:51,260

And similarly, the data instance here,

the text object,

105

00:07:51,260 --> 00:07:55,940

is represented by also a feature

vector of the same number of elements.

106

00:07:55,940 --> 00:07:59,100

Xi is a feature value.

107

00:07:59,100 --> 00:08:04,418

For example, word count and

you can verify, when we.

108

00:08:04,418 --> 00:08:08,960

Multiply these two vectors together,

take the dot product,

109

00:08:08,960 --> 00:08:14,335

we get the same form of the linear

separator as you have seen before.

110

00:08:14,335 --> 00:08:16,713

It's just a different way

of representing this.

111

00:08:16,713 --> 00:08:21,267

Now I use this way so that it's

more consistent with what notations

112

00:08:21,267 --> 00:08:24,750

people usually use when

they talk about SVM.

113

00:08:24,750 --> 00:08:29,470

This way you can better connect the slides

with some other readings you might do.

114

00:08:31,190 --> 00:08:39,780

Okay, so when we maximize

the margins of a separator,

115

00:08:39,780 --> 00:08:44,730

it just means the boundary of

the separator is only determined by

116

00:08:44,730 --> 00:08:49,800

a few data points, and these are the data

points that we call support vectors.

117

00:08:49,800 --> 00:08:54,600

So here illustrated are two support

vectors for one class and two for

118

00:08:54,600 --> 00:08:56,220

the other class.

119

00:08:56,220 --> 00:09:00,900

And these quotas define

the margin basically, and

120

00:09:00,900 --> 00:09:05,350

you can imagine once we know which

are supportive vectors then this

121

00:09:06,430 --> 00:09:09,750

center separator line will

be determined by them.

122

00:09:09,750 --> 00:09:16,320

So the other data points actually

don't really matter that much.

123

00:09:16,320 --> 00:09:20,420

And you can see if you change the other

data points it won't really affect

124

00:09:20,420 --> 00:09:22,905

the margin, so

the separator will stay the same.

125

00:09:22,905 --> 00:09:26,514

Mainly affected by

the the support vector machines.

126

00:09:26,514 --> 00:09:29,705

Sorry, it's mainly affected

by the support vectors and

127

00:09:29,705 --> 00:09:32,639

that's why it's called

a support vector machine.

128

00:09:32,639 --> 00:09:37,968

Okay, so now the next question is,

of course,

129

00:09:37,968 --> 00:09:42,730

how can we set it up to optimize the line?

130

00:09:42,730 --> 00:09:47,430

How can we actually find the line or

the separator?

131

00:09:47,430 --> 00:09:51,390

Now this is equivalent to

finding values for w and

132

00:09:51,390 --> 00:09:55,779

b, because they will determine

where exactly the separator is.

133

00:09:58,010 --> 00:10:04,700

So in the simplest case, the linear SVM

is just a simple optimization problem.

134

00:10:04,700 --> 00:10:10,230

So again, let's recall that our classifier

is such a linear separator, where we

135

00:10:10,230 --> 00:10:15,980

have weights for all the features, and the

main goal is remove these weights w and b.

136

00:10:15,980 --> 00:10:21,040

And the classifier will say X is in

category theta 1 if it's positive.

137

00:10:21,040 --> 00:10:23,950

Otherwise, it's going to say

it's in the other category.

138

00:10:23,950 --> 00:10:27,220

So this is our assumption, our setup.

139

00:10:27,220 --> 00:10:32,406

So in the linear SVM,

we are going to then seek these parameter

140

00:10:32,406 --> 00:10:37,510

values to optimize the margins and

then the training error.

141

00:10:38,800 --> 00:10:41,920

The training data would be basically

like in other classifiers.

142

00:10:41,920 --> 00:10:45,940

We have a set of training points

where we know the x vector, and

143

00:10:45,940 --> 00:10:50,290

then we also know the corresponding label,

y i.

144

00:10:50,290 --> 00:10:54,310

And here we define y i as two values, but

145

00:10:54,310 --> 00:10:58,358

these values are not 0, 1 as you

have seen before, but rather -1 and

146

00:10:58,358 --> 00:11:03,990

positive 1, and they're corresponding to

these two categories, as I've shown here.

147

00:11:03,990 --> 00:11:08,330

Now you might wonder why we

don't define them as 0 and

148

00:11:08,330 --> 00:11:11,770

1 instead of having -1, 1.

149

00:11:11,770 --> 00:11:15,520

And this is purely for mathematical

convenience, as you will see in a moment.

150

00:11:16,700 --> 00:11:19,450

So the goal of optimization first is

151

00:11:19,450 --> 00:11:23,700

to make sure the labeling of

training data is all correct.

152

00:11:23,700 --> 00:11:28,240

So that just means if y i,

the norm label for instance x i,

153

00:11:28,240 --> 00:11:33,610

is 1, we would like this

classified value to be large.

154

00:11:33,610 --> 00:11:36,740

And here we just choose

a threshold of 1 here.

155

00:11:36,740 --> 00:11:41,875

But if you use another threshold,

you can easily fit that constant

156

00:11:41,875 --> 00:11:47,300

into the parameter values b and

w to make the right-hand side just 1.

157

00:11:48,950 --> 00:11:54,780

Now if, on the other hand, y i is -1,

that means it's in a different class,

158

00:11:54,780 --> 00:11:58,460

then we want this classifier

to give us a very small value,

159

00:11:58,460 --> 00:12:04,860

in fact a negative value, and we want this

value to be less than or equal to -1.

160

00:12:04,860 --> 00:12:11,110

Now these are the two different instances,

different kinds of cases.

161

00:12:11,110 --> 00:12:13,714

How can we combine them together?

162

00:12:13,714 --> 00:12:18,622

Now this is where it's convenient

when we have chosen y i as -1 for

163

00:12:18,622 --> 00:12:20,200

the other category,

164

00:12:20,200 --> 00:12:25,830

because it turns out that we can either

combine the two into one constraint.

165

00:12:26,832 --> 00:12:32,085

y i multiplied by the classifier value

must be larger than or equal to 1.

166

00:12:33,210 --> 00:12:35,484

And obviously when y i is just 1,

167

00:12:35,484 --> 00:12:39,968

you see this is the same as

the constraint on the left-hand side.

168

00:12:39,968 --> 00:12:48,020

But when y i is -1, you also see that this

is equivalent to the other inequality.

169

00:12:48,020 --> 00:12:53,060

So this one actually captures both

constraints in a unified way,

170

00:12:53,060 --> 00:12:56,960

and that's a convenient way of

capturing these constraints.

171

00:12:56,960 --> 00:12:58,137

What's our second goal?

172

00:12:58,137 --> 00:13:00,414

Well, that's to maximize margin, so

173

00:13:00,414 --> 00:13:04,600

we want to ensure that separator

can do well on the training data.

174

00:13:04,600 --> 00:13:08,109

But then, among all the cases

where we can separate the data,

175

00:13:08,109 --> 00:13:12,172

we also would like to choose the separator

that has the largest margin.

176

00:13:12,172 --> 00:13:18,758

Now the margin can be assumed to be

related to the magnitude of the weight.

177

00:13:18,758 --> 00:13:23,777

And so

w transform multiplied by w would give

178

00:13:23,777 --> 00:13:29,893

us basically the sum of

squares of all those weights.

179

00:13:29,893 --> 00:13:35,691

So to have a small value for

this expression,

180

00:13:35,691 --> 00:13:40,430

it means all the w i's must be small.

181

00:13:42,440 --> 00:13:45,710

So we've just assumed that

we have a constraint for

182

00:13:46,930 --> 00:13:50,890

getting the data on the training

set to be classified correctly.

183

00:13:50,890 --> 00:13:57,649

Now we also have the objective that's

tied into a maximization of margin,

184

00:13:57,649 --> 00:14:03,013

and this is simply to minimize

w transpose multiplied by w,

185

00:14:03,013 --> 00:14:06,251

and we often denote this by phi of w.

186

00:14:06,251 --> 00:14:10,616

So now you can see this is

basically a optimization problem.

187

00:14:10,616 --> 00:14:15,044

We have some variables to optimize,

and these are the weights and

188

00:14:15,044 --> 00:14:17,540

b and we have some constraints.

189

00:14:17,540 --> 00:14:18,949

These are linear constraints and

190

00:14:18,949 --> 00:14:22,380

the objective function is

a quadratic function of the weights.

191

00:14:22,380 --> 00:14:25,370

So this a quadratic program

with linear constraints,

192

00:14:25,370 --> 00:14:30,050

and there are standard algorithm that

are variable for solving this problem.

193

00:14:30,050 --> 00:14:34,190

And once we solve the problem

we obtain the weights w and b.

194

00:14:34,190 --> 00:14:37,080

And then this would give us

a well-defined classifier.

195

00:14:37,080 --> 00:14:42,160

So we can then use this classifier

to classify any new text objects.

196

00:14:42,160 --> 00:14:47,190

Now the previous formulation did not

allow any error in the classification,

197

00:14:47,190 --> 00:14:50,448

but sometimes the data may not

be linear to the separator.

198

00:14:50,448 --> 00:14:54,690

That means that they may not

look as nice as you have seen on

199

00:14:54,690 --> 00:14:59,300

the previous slide where a line

can separate all of them.

200

00:14:59,300 --> 00:15:02,850

And what would happen if

we allowed some errors?

201

00:15:02,850 --> 00:15:04,980

Well, the principle can stay.

202

00:15:04,980 --> 00:15:09,305

We want to minimize the training error but

try to also maximize the margin.

203

00:15:09,305 --> 00:15:12,270

But in this case we have a soft margin,

204

00:15:12,270 --> 00:15:16,000

because the data points may

not be completely separable.

205

00:15:17,030 --> 00:15:24,650

So it turns out that we can easily

modify SVM to accommodate this.

206

00:15:24,650 --> 00:15:28,090

So what you see here is very similar

to what you have seen before,

207

00:15:28,090 --> 00:15:31,760

but we have introduced

the extra variable xi i.

208

00:15:31,760 --> 00:15:35,610

And we in fact will have one for

each data instance, and

209

00:15:35,610 --> 00:15:40,780

this is going to model the error

that we allow for each instance.

210

00:15:40,780 --> 00:15:43,245

But the optimization problem

would be very similar.

211

00:15:43,245 --> 00:15:44,783

So specifically,

212

00:15:44,783 --> 00:15:50,170

you will see we have added something

to the optimization problem.

213

00:15:50,170 --> 00:15:56,861

First we have added some

error to the constraint so

214

00:15:56,861 --> 00:16:02,119

that now we allow a Allow the classifier

215

00:16:02,119 --> 00:16:06,760

to make some mistakes here.

216

00:16:06,760 --> 00:16:12,860

So, this Xi i is allowed an error.

217

00:16:12,860 --> 00:16:16,560

If we set Xi i to 0, then we go

back to the original constraint.

218

00:16:16,560 --> 00:16:20,260

We want every instance to

be classified accurately.

219

00:16:20,260 --> 00:16:26,420

But, if we allow this to be non-zero,

then we allow some errors here.

220

00:16:26,420 --> 00:16:30,730

In fact, if the length of the Xi i is very

large, the error can be very, very large.

221

00:16:30,730 --> 00:16:33,270

So naturally,

we don't want this to happen.

222

00:16:33,270 --> 00:16:37,570

So we want to then also

minimize this Xi i.

223

00:16:37,570 --> 00:16:41,940

So, because Xi i needs to be minimized

in order to control the error.

224

00:16:42,940 --> 00:16:46,020

And so, as a result,

in the objective function,

225

00:16:46,020 --> 00:16:50,910

we also add more to the original one,

which is only W,

226

00:16:50,910 --> 00:16:55,190

by basically ensuring that we not

only minimize the weights, but

227

00:16:55,190 --> 00:16:59,130

also minimize the errors, as you see here.

228

00:16:59,130 --> 00:17:02,705

Here we simply take a sum

over all the instances.

229

00:17:02,705 --> 00:17:07,695

Each one has a Xi i to model

the error allowed for that instance.

230

00:17:07,695 --> 00:17:10,413

And when we combine them together,

231

00:17:10,413 --> 00:17:14,680

we basically want to minimize

the errors on all of them.

232

00:17:16,350 --> 00:17:21,001

Now you see there's a parameter C here,

and that's a constant to control

233

00:17:21,001 --> 00:17:25,740

the trade-off between minimizing

the errors and maximizing the margin.

234

00:17:25,740 --> 00:17:27,888

If C is set to zero, you can see,

235

00:17:27,888 --> 00:17:33,070

we go back to the original object function

where we only maximize the margin.

236

00:17:34,340 --> 00:17:38,368

We don't really optimize

the training errors and

237

00:17:38,368 --> 00:17:43,730

then Xi i can be set to a very large value

to make the constraints easy to satisfy.

238

00:17:43,730 --> 00:17:46,512

That's not very good of course, so

239

00:17:46,512 --> 00:17:50,884

C should be set to a non-zero value,

a positive value.

240

00:17:50,884 --> 00:17:53,412

But when C is set to a very,

very large value,

241

00:17:53,412 --> 00:17:58,143

we'll see the object of the function will

be dominated mostly by the training errors

242

00:17:58,143 --> 00:18:02,420

and so the optimization of margin

will then play a secondary role.

243

00:18:02,420 --> 00:18:06,350

So if that happens, what would happen is

244

00:18:07,420 --> 00:18:11,420

then we will try to do our best to

minimize the training errors, but

245

00:18:11,420 --> 00:18:14,730

then we're not going to

take care of the margin and

246

00:18:14,730 --> 00:18:19,270

that affects the generalization factors

of the classify for future data.

247

00:18:19,270 --> 00:18:20,548

So it's also not good.

248

00:18:20,548 --> 00:18:28,175

So in particular, this parameter C

has to be actually set carefully.

249

00:18:28,175 --> 00:18:32,045

And this is just like in the case of

k-nearest neighbor where you need

250

00:18:32,045 --> 00:18:34,080

to optimize a number of neighbors.

251

00:18:34,080 --> 00:18:35,510

Here you need to optimize the C.

252

00:18:35,510 --> 00:18:40,510

And this is, in general,

also achievable by doing cross-validation.

253

00:18:40,510 --> 00:18:43,331

Basically, you look at

the empirical data and

254

00:18:43,331 --> 00:18:47,610

see what value C should be set to in

order to optimize the performance.

255

00:18:49,050 --> 00:18:50,390

Now with this modification,

256

00:18:50,390 --> 00:18:54,250

the problem is still quadratic programming

with linear constraints so the optimizing

257

00:18:54,250 --> 00:19:00,003

algorithm can be actually applied to solve

this different version of the program.

258

00:19:02,080 --> 00:19:05,780

Again, once we have obtained

the weights and the bias,

259

00:19:05,780 --> 00:19:11,360

then we can have classifier that's

ready for classifying new objects.

260

00:19:11,360 --> 00:19:13,566

So that's the basic idea of SVM.

261

00:19:16,993 --> 00:19:20,402

So to summarize the text

categorization methods,

262

00:19:20,402 --> 00:19:25,170

where we introduce the many methods,

and some are generative models.

263

00:19:25,170 --> 00:19:27,140

Some are discriminative methods.

264

00:19:27,140 --> 00:19:32,230

And these tend to perform

similarly when optimized.

265

00:19:32,230 --> 00:19:37,920

So there's still no clear winner,

although each one has its pros and cons.

266

00:19:37,920 --> 00:19:42,460

And the performance might also

vary on different data sets for

267

00:19:42,460 --> 00:19:44,320

different problems.

268

00:19:44,320 --> 00:19:50,610

And one reason is also because the feature

representation is very critical

269

00:19:52,280 --> 00:19:56,470

and these methods all require

effective feature representation.

270

00:19:56,470 --> 00:19:59,400

And to design an effective feature set,

271

00:19:59,400 --> 00:20:03,530

we need domain knowledge and humans

definitely play an important role here,

272

00:20:03,530 --> 00:20:05,608

although there are new

machine learning methods and

273

00:20:05,608 --> 00:20:10,020

algorithm representation learning

that can help with learning features.

274

00:20:12,640 --> 00:20:18,169

And another common thing

is that they might

275

00:20:18,169 --> 00:20:23,546

be performing similarly on the data set,

276

00:20:23,546 --> 00:20:28,220

but with different mistakes.

277

00:20:28,220 --> 00:20:30,913

And so,

their performance might be similar, but

278

00:20:30,913 --> 00:20:34,070

then the mistakes they

make might be different.

279

00:20:34,070 --> 00:20:37,630

So that means it's useful to

compare different methods for

280

00:20:37,630 --> 00:20:42,690

a particular problem and

then maybe combine multiple methods

281

00:20:42,690 --> 00:20:49,092

because this can improve the robustness

and they won't make the same mistakes.

282

00:20:49,092 --> 00:20:54,192

So assemble approaches that

would combine different

283

00:20:54,192 --> 00:20:59,990

methods tend to be more robust and

can be useful in practice.

284

00:20:59,990 --> 00:21:04,530

Most techniques that we introduce

use the supervised machine learning,

285

00:21:04,530 --> 00:21:06,990

which is a very general method.

286

00:21:06,990 --> 00:21:10,975

So that means that these methods can

be actually applied to any text or

287

00:21:10,975 --> 00:21:12,580

categorization problem.

288

00:21:12,580 --> 00:21:17,554

As long as we have humans to help

annotate some training data sets and

289

00:21:17,554 --> 00:21:23,493

design features, then supervising machine

learning and all these classifiers

290

00:21:23,493 --> 00:21:29,255

can be easily applied to those problems

to solve the categorization problem to

291

00:21:29,255 --> 00:21:34,431

allow us to characterize content

of text concisely with categories.

292

00:21:34,431 --> 00:21:38,716

Or to predict the sum

properties of real world

293

00:21:38,716 --> 00:21:43,250

variables that are associated

with text data.

294

00:21:43,250 --> 00:21:47,875

The computers, of course, here are trying

to optimize the combinations of

295

00:21:47,875 --> 00:21:49,908

the features provided by human.

296

00:21:49,908 --> 00:21:53,357

And as I said, there are many

different ways of combining them and

297

00:21:53,357 --> 00:21:56,130

they also optimize different object or

functions.

298

00:21:58,180 --> 00:22:02,240

But in order to achieve good performance,

they all require effective features and

299

00:22:02,240 --> 00:22:03,750

also plenty of training data.

300

00:22:04,770 --> 00:22:08,870

So as a general rule, and if you can

improve the feature representation,

301

00:22:08,870 --> 00:22:13,860

and then provide more training data,

then you can generally do better.

302

00:22:13,860 --> 00:22:18,390

Performance is often much more

affected by the effectiveness of

303

00:22:18,390 --> 00:22:23,030

features than by the choice

of specific classifiers.

304

00:22:23,030 --> 00:22:26,972

So feature design tends to be more

important than the choice of specific

305

00:22:26,972 --> 00:22:27,768

classifier.

306

00:22:30,844 --> 00:22:34,170

So, how do we design effective features?

307

00:22:34,170 --> 00:22:37,360

Well, unfortunately,

this is very application-specific.

308

00:22:37,360 --> 00:22:43,108

So there's no really much

general thing to say here.

309

00:22:43,108 --> 00:22:47,672

But we can do some analysis of

the categorization problem and

310

00:22:47,672 --> 00:22:54,400

try to understand what kind of features

might help us distinguish categories.

311

00:22:54,400 --> 00:22:59,720

And in general, we can use a lot of domain

knowledge to help us design features.

312

00:23:01,640 --> 00:23:06,180

And another way to figure out

the effective features is

313

00:23:06,180 --> 00:23:10,230

to do error analysis on

the categorization results.

314

00:23:10,230 --> 00:23:11,080

You could, for example,

315

00:23:11,080 --> 00:23:16,110

look at which category tends to be

confused with which other categories.

316

00:23:16,110 --> 00:23:20,890

And you can use a confusion matrix

to examine the errors systematically

317

00:23:20,890 --> 00:23:22,340

across categories.

318

00:23:22,340 --> 00:23:25,320

And then,

you can look into specific instances to

319

00:23:25,320 --> 00:23:29,780

see why the mistake has been made and

what features can prevent the mistake.

320

00:23:29,780 --> 00:23:35,260

And this can allow you to obtain

insights for design new features.

321

00:23:35,260 --> 00:23:37,840

So error analysis is very

important in general, and

322

00:23:37,840 --> 00:23:40,860

that's where you can get the insights

about your specific problem.

323

00:23:42,150 --> 00:23:45,220

And finally, we can leverage this

on machine learning techniques.

324

00:23:45,220 --> 00:23:48,710

So, for example, feature selection is

a technique that we haven't really talked

325

00:23:48,710 --> 00:23:50,390

about, but is very important.

326

00:23:50,390 --> 00:23:54,830

And it has to do with trying to select the

most useful features before you actually

327

00:23:54,830 --> 00:23:56,276

train a full classifier.

328

00:23:56,276 --> 00:24:00,900

Sometimes training a classifier will also

help you identify which features have high

329

00:24:00,900 --> 00:24:01,419

values.

330

00:24:01,419 --> 00:24:04,658

There are also other ways

to ensure this sparsity.

331

00:24:04,658 --> 00:24:07,538

Of the model,

meaning to recognize the widths.

332

00:24:07,538 --> 00:24:12,870

For example, the SVM actually tries

to minimize the weights on features.

333

00:24:12,870 --> 00:24:16,630

But you can further force some features,

334

00:24:16,630 --> 00:24:19,019

force to use only a small

number of features.

335

00:24:21,080 --> 00:24:25,030

There are also techniques for

dimension reduction.

336

00:24:25,030 --> 00:24:29,450

And that's to reduce a high dimensional

feature space into a low dimensional

337

00:24:29,450 --> 00:24:33,150

space typically by clustering

of features in various ways.

338

00:24:33,150 --> 00:24:38,150

So metrics factorization

has been used to do

339

00:24:38,150 --> 00:24:42,860

such a job, and this is some of the

techniques are actually very similar to

340

00:24:42,860 --> 00:24:44,820

the talking models that we'll discuss.

341

00:24:44,820 --> 00:24:48,220

So talking morals like psa or

342

00:24:48,220 --> 00:24:52,570

lda can actually help us reduce

the dimension of features.

343

00:24:52,570 --> 00:24:56,331

Like imagine the words

our original feature.

344

00:24:56,331 --> 00:25:01,970

But the can be matched to the topic

space .Let's say we have k topics.

345

00:25:01,970 --> 00:25:04,380

So a document can now be represented

346

00:25:04,380 --> 00:25:08,750

as a vector of just k values

corresponding to the topics.

347

00:25:08,750 --> 00:25:12,380

So we can let each topic define one

dimension, so we have a k dimensional

348

00:25:12,380 --> 00:25:17,920

space instead of the original high

dimensional space corresponding to words.

349

00:25:17,920 --> 00:25:21,720

And this is often another way

to learn effective features.

350

00:25:21,720 --> 00:25:26,200

Especially, we could also use the

categories to supervise the learning of

351

00:25:26,200 --> 00:25:28,370

such low dimensional structures.

352

00:25:29,850 --> 00:25:36,070

And so, the original worth features

can be also combined with such

353

00:25:36,070 --> 00:25:40,480

amazing dimension features or

lower dimensional space features

354

00:25:40,480 --> 00:25:44,810

to provide a multi resolution

which is often very useful.

355

00:25:44,810 --> 00:25:49,940

Deep learning is a new technique that

has been developed the machine learning.

356

00:25:51,190 --> 00:25:54,890

It's particularly useful for

learning representations.

357

00:25:54,890 --> 00:25:59,840

So deep learning refers to deep neural

network, it's another kind of classifier,

358

00:25:59,840 --> 00:26:07,110

where you can have intermediate

features embedded in the models.

359

00:26:07,110 --> 00:26:11,570

That it's highly non-linear transpire, and

360

00:26:11,570 --> 00:26:17,220

some recent events that's allowed us to

train such a complex network effectively.

361

00:26:17,220 --> 00:26:23,300

And the technique has been shown to be

quite effective for speech recognition,

362

00:26:23,300 --> 00:26:27,620

computer reasoning, and

recently has been applied to text as well.

363

00:26:27,620 --> 00:26:29,530

It has shown some promise.

364

00:26:29,530 --> 00:26:33,010

And one important advantage

of this approach in

365

00:26:34,270 --> 00:26:39,010

relationship with the featured design,

is that they can

366

00:26:39,010 --> 00:26:43,920

learn intermediate replantations or

compound the features automatically.

367

00:26:43,920 --> 00:26:49,193

And this is very valuable for

learning effective replantation,

368

00:26:49,193 --> 00:26:51,660

for text recalibration.

369

00:26:51,660 --> 00:26:57,390

Although in text domain, because words are

exemplary representation of text content,

370

00:26:57,390 --> 00:27:01,620

because these are human's imaging for

communication.

371

00:27:01,620 --> 00:27:08,160

And they are generally sufficient for

For representing content for many tasks.

372

00:27:08,160 --> 00:27:11,430

If there's a need for

some new representation,

373

00:27:11,430 --> 00:27:15,250

people would have invented a new word.

374

00:27:15,250 --> 00:27:18,320

So because of this we think

of value of deep learning for

375

00:27:18,320 --> 00:27:22,610

text processing tends to be lower than for

[INAUDIBLE].

376

00:27:22,610 --> 00:27:26,490

And the speech revenue where

they are anchored corresponding

377

00:27:26,490 --> 00:27:29,920

where the design that worked as features.

378

00:27:31,160 --> 00:27:35,020

But people only still very promising for

learning effective features especially for

379

00:27:35,020 --> 00:27:35,857

complicated tasks.

380

00:27:35,857 --> 00:27:39,850

Like a analysis it has

been shown to be effective

381

00:27:41,230 --> 00:27:44,760

because it can provide that

goes beyond that of words.

382

00:27:47,030 --> 00:27:50,240

Now regarding the training examples.

383

00:27:50,240 --> 00:27:53,940

It's generally hard to get a lot of

training examples because it involves

384

00:27:53,940 --> 00:27:54,560

human labor.

385

00:27:56,310 --> 00:27:58,570

But there are also some

ways to help with this.

386

00:27:58,570 --> 00:28:04,830

So one is to assume in some low quality

training examples can also be used.

387

00:28:04,830 --> 00:28:07,800

So, those can be called

pseudo training examples.

388

00:28:07,800 --> 00:28:13,220

For example, if you take reviews from the

internet, they might have overall ratings.

389

00:28:13,220 --> 00:28:21,250

So, to train a of categorizer,

meaning we want to positive or negative.

390

00:28:21,250 --> 00:28:24,860

And categorize these reviews

into these two categories.

391

00:28:24,860 --> 00:28:31,570

Then we could assume five star reviews

are all positive training samples.

392

00:28:31,570 --> 00:28:33,270

One star are negative.

393

00:28:33,270 --> 00:28:34,190

But of course,

394

00:28:34,190 --> 00:28:38,520

sometimes even five star reviews will also

mention negative opinions so the training

395

00:28:38,520 --> 00:28:43,180

sample is not all of that high quality,

but they can still be useful.

396

00:28:45,200 --> 00:28:47,970

Another idea is to exploit

the unlabeled data and

397

00:28:47,970 --> 00:28:50,830

there are techniques called

the semi-supervised machine

398

00:28:50,830 --> 00:28:55,685

learning techniques that can allow you to

combine labeled data with unlabeled data.

399

00:28:55,685 --> 00:29:01,070

So, in other case it's easy to see

the next model can be used For

400

00:29:01,070 --> 00:29:03,760

both text plus read and

the categorization.

401

00:29:03,760 --> 00:29:09,220

So you can imagine, if you have a lot of

unlabeled text data for categorization,

402

00:29:09,220 --> 00:29:15,620

then you can actually do clustering

on these text data, learn categories.

403

00:29:15,620 --> 00:29:18,088

And then try to somehow

align these categories.

404

00:29:18,088 --> 00:29:23,230

With the categories defined

by the training data,

405

00:29:23,230 --> 00:29:26,390

where we already know which

documents are in which category.

406

00:29:26,390 --> 00:29:31,620

So you can in fact use the Algorithm

to actually combine both.

407

00:29:31,620 --> 00:29:37,390

That would allow you essentially also

pick up useful words and label the data.

408

00:29:37,390 --> 00:29:39,320

You can think of this in another way.

409

00:29:39,320 --> 00:29:43,804

Basically, we can use let's say a to

410

00:29:43,804 --> 00:29:48,480

classify all of the unlabeled text

documents, and then we're going to

411

00:29:48,480 --> 00:29:54,040

assume the high confidence Classification

results are actually liable.

412

00:29:54,040 --> 00:29:58,600

Then you suddenly have more training

data because from the enabler that we

413

00:29:58,600 --> 00:30:03,450

now know some are labeled as category one,

some are labeled as category two.

414

00:30:03,450 --> 00:30:06,380

All though the label is not

completely reliable But

415

00:30:06,380 --> 00:30:07,830

then they can still be useful.

416

00:30:07,830 --> 00:30:14,720

So let's assume they are actually training

label examples, and then we combine them

417

00:30:14,720 --> 00:30:19,940

with true training examples through

improved categorization method.

418

00:30:19,940 --> 00:30:22,110

And so this idea is very powerful.

419

00:30:23,980 --> 00:30:28,280

When the enabled data and

the training data are very different, and

420

00:30:28,280 --> 00:30:32,410

we might need to use other advanced

machine learning techniques

421

00:30:32,410 --> 00:30:35,150

called domain adaptation or

transfer learning.

422

00:30:35,150 --> 00:30:37,580

This is when we can

423

00:30:37,580 --> 00:30:42,450

Borrow some training examples from

a related problem that may be different.

424

00:30:42,450 --> 00:30:44,470

Or, from a categorization password

425

00:30:46,780 --> 00:30:52,130

that follow very different distribution

from what we are working on.

426

00:30:52,130 --> 00:30:54,190

But basically,

when the two domains are very different,

427

00:30:54,190 --> 00:30:57,640

then we need to be careful and

not overfit the training domain.

428

00:30:57,640 --> 00:31:02,300

But yet, we can still want to use some

signals from the related training data.

429

00:31:02,300 --> 00:31:07,270

So for example,

training categorization on news might not

430

00:31:07,270 --> 00:31:12,410

give you Effective plus y for

class vine topics and tweets.

431

00:31:12,410 --> 00:31:19,490

But you can still learn something from

news to help look at writing tweets.

432

00:31:19,490 --> 00:31:25,470

So there are mission learning techniques

that can help you do that effectively.

433

00:31:25,470 --> 00:31:30,259

Here's a suggested reading where you

can find more details about some

434

00:31:30,259 --> 00:31:33,271

more of the methods is

that we have covered.

435

00:31:33,271 --> 00:31:43,271

[MUSIC]