MachineLearning-Lecture01

Instructor (Andrew Ng): Okay. Good morning. Welcome to CS229, the machine learning class. So what I wanna do today is just spend a little time going over the logistics of the class, and then we'll start to talk a bit about machine learning.

By way of introduction, my name's Andrew Ng and I'll be instructor for this class. And so I personally work in machine learning, and I've worked on it for about 15 years now, and I actually think that machine learning is the most exciting field of all the computer sciences. So I'm actually always excited about teaching this class. Sometimes I actually think that machine learning is not only the most exciting thing in computer science, but the most exciting thing in all of human endeavor, so maybe a little bias there.

I also want to introduce the TAs, who are all graduate students doing research in or related to the machine learning and all aspects of machine learning. Paul Baumstarck works in machine learning and computer vision. Catie Chang is actually a neuroscientist who applies machine learning algorithms to try to understand the human brain. Tom Do is another PhD student, works in computational biology and in sort of the basic fundamentals of human learning. Zico Kolter is the head TA — he's head TA two years in a row now — works in machine learning and applies them to a bunch of robots. And Daniel Ramage is — I guess he's not here — Daniel applies learning algorithms to problems in natural language processing.

So you'll get to know the TAs and me much better throughout this quarter, but just from the sorts of things the TA's do, I hope you can already tell that machine learning is a highly interdisciplinary topic in which just the TAs find learning algorithms to problems in computer vision and biology and robots and language. And machine learning is one of those things that has and is having a large impact on many applications.

So just in my own daily work, I actually frequently end up talking to people like helicopter pilots to biologists to people in computer systems or databases to economists and sort of also an unending stream of people from industry coming to Stanford interested in applying machine learning methods to their own problems.

So yeah, this is fun. A couple of weeks ago, a student actually forwarded to me an article in "Computer World" about the 12 IT skills that employers can't say no to. So it's about sort of the 12 most desirable skills in all of IT and all of information technology, and topping the list was actually machine learning. So I think this is a good time to be learning this stuff and learning algorithms and having a large impact on many segments of science and industry.

I'm actually curious about something. Learning algorithms is one of the things that touches many areas of science and industries, and I'm just kind of curious. How many people here are computer science majors, are in the computer science department? Okay. About half of you. How many people are from EE? Oh, okay, maybe about a fifth. How

many biologers are there here? Wow, just a few, not many. I'm surprised. Anyone from statistics? Okay, a few. So where are the rest of you from?

Student : iCME.

Instructor (Andrew Ng) : Say again?

Student : iCME.

Instructor (Andrew Ng) : iCME. Cool.

Student : [Inaudible].

Instructor (Andrew Ng) : Civi and what else?

Student : [Inaudible]

Instructor (Andrew Ng) : Synthesis, [inaudible] systems. Yeah, cool.

Student : Chemi.

Instructor (Andrew Ng) : Chemi. Cool.

Student : [Inaudible].

Instructor (Andrew Ng) : Aero/astro. Yes, right. Yeah, okay, cool. Anyone else?

Student : [Inaudible].

Instructor (Andrew Ng) : Pardon? MSNE. All right. Cool. Yeah.

Student : [Inaudible].

Instructor (Andrew Ng) : Pardon?

Student : [Inaudible].

Instructor (Andrew Ng) : Endo —

Student : [Inaudible].

Instructor (Andrew Ng) : Oh, I see, industry. Okay. Cool. Great, great. So as you can tell from a cross-section of this class, I think we're a very diverse audience in this room, and that's one of the things that makes this class fun to teach and fun to be in, I think.

So in this class, we've tried to convey to you a broad set of principles and tools that will be useful for doing many, many things. And every time I teach this class, I can actually very confidently say that after December, no matter what you're going to do after this December when you've sort of completed this class, you'll find the things you learn in this class very useful, and these things will be useful pretty much no matter what you end up doing later in your life.

So I have more logistics to go over later, but let's say a few more words about machine learning. I feel that machine learning grew out of early work in AI, early work in artificial intelligence. And over the last — I wanna say last 15 or last 20 years or so, it's been viewed as a sort of growing new capability for computers. And in particular, it turns out that there are many programs or there are many applications that you can't program by hand.

For example, if you want to get a computer to read handwritten characters, to read sort of handwritten digits, that actually turns out to be amazingly difficult to write a piece of software to take this input, an image of something that I wrote and to figure out just what it is, to translate my cursive handwriting into — to extract the characters I wrote out in longhand. And other things: One thing that my students and I do is autonomous flight. It turns out to be extremely difficult to sit down and write a program to fly a helicopter.

But in contrast, if you want to do things like to get software to fly a helicopter or have software recognize handwritten digits, one very successful approach is to use a learning algorithm and have a computer learn by itself how to, say, recognize your handwriting. And in fact, handwritten digit recognition, this is pretty much the only approach that works well. It uses applications that are hard to program by hand.

Learning algorithms has also made I guess significant inroads in what's sometimes called database mining. So, for example, with the growth of IT and computers, increasingly many hospitals are keeping around medical records of what sort of patients, what problems they had, what their prognoses was, what the outcome was. And taking all of these medical records, which started to be digitized only about maybe 15 years, applying learning algorithms to them can turn raw medical records into what I might loosely call medical knowledge in which we start to detect trends in medical practice and even start to alter medical practice as a result of medical knowledge that's derived by applying learning algorithms to the sorts of medical records that hospitals have just been building over the last 15, 20 years in an electronic format.

Turns out that most of you probably use learning algorithms — I don't know — I think half a dozen times a day or maybe a dozen times a day or more, and often without knowing it. So, for example, every time you send mail via the US Postal System, turns out there's an algorithm that tries to automatically read the zip code you wrote on your envelope, and that's done by a learning algorithm. So every time you send US mail, you are using a learning algorithm, perhaps without even being aware of it.

Similarly, every time you write a check, I actually don't know the number for this, but a significant fraction of checks that you write are processed by a learning algorithm that's learned to read the digits, so the dollar amount that you wrote down on your check. So every time you write a check, there's another learning algorithm that you're probably using without even being aware of it.

If you use a credit card, or I know at least one phone company was doing this, and lots of companies like eBay as well that do electronic transactions, there's a good chance that there's a learning algorithm in the background trying to figure out if, say, your credit card's been stolen or if someone's engaging in a fraudulent transaction.

If you use a website like Amazon or Netflix that will often recommend books for you to buy or movies for you to rent or whatever, these are other examples of learning algorithms that have learned what sorts of things you like to buy or what sorts of movies you like to watch and can therefore give customized recommendations to you.

Just about a week ago, I had my car serviced, and even there, my car mechanic was trying to explain to me some learning algorithm in the innards of my car that's sort of doing its best to optimize my driving performance for fuel efficiency or something.

So, see, most of us use learning algorithms half a dozen, a dozen, maybe dozens of times without even knowing it.

And of course, learning algorithms are also doing things like giving us a growing understanding of the human genome. So if someday we ever find a cure for cancer, I bet learning algorithms will have had a large role in that. That's sort of the thing that Tom works on, yes?

So in teaching this class, I sort of have three goals. One of them is just to I hope convey some of my own excitement about machine learning to you.

The second goal is by the end of this class, I hope all of you will be able to apply state-of-the-art machine learning algorithms to whatever problems you're interested in. And if you ever need to build a system for reading zip codes, you'll know how to do that by the end of this class.

And lastly, by the end of this class, I realize that only a subset of you are interested in doing research in machine learning, but by the conclusion of this class, I hope that all of you will actually be well qualified to start doing research in machine learning, okay?

So let's say a few words about logistics. The prerequisites of this class are written on one of the handouts, are as follows: In this class, I'm going to assume that all of you have sort of basic knowledge of computer science and knowledge of the basic computer skills and principles. So I assume all of you know what big?O notation, that all of you know about sort of data structures like queues, stacks, binary trees, and that all of you know enough programming skills to, like, write a simple computer program. And it turns out that most

of this class will not be very programming intensive, although we will do some programming, mostly in either MATLAB or Octave. I'll say a bit more about that later.

I also assume familiarity with basic probability and statistics. So most undergraduate statistics class, like Stat 116 taught here at Stanford, will be more than enough. I'm gonna assume all of you know what random variables are, that all of you know what expectation is, what a variance or a random variable is. And in case of some of you, it's been a while since you've seen some of this material. At some of the discussion sections, we'll actually go over some of the prerequisites, sort of as a refresher course under prerequisite class. I'll say a bit more about that later as well.

Lastly, I also assume familiarity with basic linear algebra. And again, most undergraduate linear algebra courses are more than enough. So if you've taken courses like Math 51, 103, Math 113 or CS205 at Stanford, that would be more than enough. Basically, I'm gonna assume that all of you know what matrixes and vectors are, that you know how to multiply matrices and vectors and multiply matrix and matrices, that you know what a matrix inverse is. If you know what an eigenvector of a matrix is, that'd be even better. But if you don't quite know or if you're not quite sure, that's fine, too. We'll go over it in the review sections.

So there are a couple more logistical things I should deal with in this class. One is that, as most of you know, CS229 is a televised class. And in fact, I guess many of you are probably watching this at home on TV, so I'm gonna say hi to our home viewers.

So earlier this year, I approached SCPD, which televises these classes, about trying to make a small number of Stanford classes publicly available or posting the videos on the web. And so this year, Stanford is actually starting a small pilot program in which we'll post videos of a small number of classes online, so on the Internet in a way that makes it publicly accessible to everyone. I'm very excited about that because machine learning in school, let's get the word out there.

One of the consequences of this is that — let's see — so videos or pictures of the students in this classroom will not be posted online, so your images — so don't worry about being by seeing your own face appear on YouTube one day. But the microphones may pick up your voices, so I guess the consequence of that is that because microphones may pick up your voices, no matter how irritated you are at me, don't yell out swear words in the middle of class, but because there won't be video you can safely sit there and make faces at me, and that won't show, okay?

Let's see. I also handed out this — there were two handouts I hope most of you have, course information handout. So let me just say a few words about parts of these. On the third page, there's a section that says Online Resources.

Oh, okay. Louder? Actually, could you turn up the volume? Testing. Is this better? Testing, testing. Okay, cool. Thanks.

So all right, online resources. The class has a home page, so it's in on the handouts. I won't write on the chalkboard — http://cs229.stanford.edu. And so when there are homework assignments or things like that, we usually won't sort of — in the mission of saving trees, we will usually not give out many handouts in class. So homework assignments, homework solutions will be posted online at the course home page.

As far as this class, I've also written, and I guess I've also revised every year a set of fairly detailed lecture notes that cover the technical content of this class. And so if you visit the course homepage, you'll also find the detailed lecture notes that go over in detail all the math and equations and so on that I'll be doing in class.

There's also a newsgroup, su.class.cs229, also written on the handout. This is a newsgroup that's sort of a forum for people in the class to get to know each other and have whatever discussions you want to have amongst yourselves. So the class newsgroup will not be monitored by the TAs and me. But this is a place for you to form study groups or find project partners or discuss homework problems and so on, and it's not monitored by the TAs and me. So feel free to talk trash about this class there.

If you want to contact the teaching staff, please use the email address written down here, cs229-qa@cs.stanford.edu. This goes to an account that's read by all the TAs and me. So rather than sending us email individually, if you send email to this account, it will actually let us get back to you maximally quickly with answers to your questions.

If you're asking questions about homework problems, please say in the subject line which assignment and which question the email refers to, since that will also help us to route your question to the appropriate TA or to me appropriately and get the response back to you quickly.

Let's see. Skipping ahead — let's see — for homework, one midterm, one open and term project. Notice on the honor code. So one thing that I think will help you to succeed and do well in this class and even help you to enjoy this class more is if you form a study group.

So start looking around where you're sitting now or at the end of class today, mingle a little bit and get to know your classmates. I strongly encourage you to form study groups and sort of have a group of people to study with and have a group of your fellow students to talk over these concepts with. You can also post on the class newsgroup if you want to use that to try to form a study group.

But some of the problems sets in this class are reasonably difficult. People that have taken the class before may tell you they were very difficult. And just I bet it would be more fun for you, and you'd probably have a better learning experience if you form a study group of people to work with. So I definitely encourage you to do that.

And just to say a word on the honor code, which is I definitely encourage you to form a study group and work together, discuss homework problems together. But if you discuss

homework problems with other students, then I'll ask you to sort of go home and write down your own solutions independently without referring to notes that were taken in any of your joint study sessions.

So in other words, when you turn in a homework problem, what you turn in should be something that was reconstructed independently by yourself and without referring to notes that you took during your study sessions with other people, okay? And obviously, showing your solutions to others or copying other solutions directly is right out.

We occasionally also reuse problem set questions from previous years so that the problems are a bit more debugged and work more smoothly. And as a result of that, I also ask you not to look at solutions from previous years, and this includes both sort of official solutions that we've given out to previous generations of this class and previous solutions that people that have taken this class in previous years may have written out by themselves, okay?

Sadly, in this class, there are usually — sadly, in previous years, there have often been a few honor code violations in this class. And last year, I think I prosecuted five honor code violations, which I think is a ridiculously large number. And so just don't work without solutions, and hopefully there'll be zero honor code violations this year. I'd love for that to happen.

The section here on the late homework policy if you ever want to hand in a homework late, I'll leave you to read that yourself.

We also have a midterm, which is scheduled for Thursday, 8th of November at 6:00 p.m., so please keep that evening free.

And let's see. And one more administrative thing I wanted to say is about the class project. So part of the goal of this class is to leave you well equipped to apply machine learning algorithms to a problem or to do research in machine learning. And so as part of this class, I'll ask you to execute a small research project sort of as a small term project.

And what most students do for this is either apply machine learning to a problem that you find interesting or investigate some aspect of machine learning. So to those of you that are either already doing research or to those of you who are in industry, you're taking this from a company, one fantastic sort of way to do a class project would be if you apply machine learning algorithms to a problem that you're interested in, to a problem that you're already working on, whether it be a science research problem or sort of a problem in industry where you're trying to get a system to work using a learning algorithm.

To those of you that are not currently doing research, one great way to do a project would be if you apply learning algorithms to just pick a problem that you care about. Pick a problem that you find interesting, and apply learning algorithms to that and play with the ideas and see what happens.

And let's see. Oh, and the goal of the project should really be for you to do a publishable piece of research in machine learning, okay?

And if you go to the course website, you'll actually find a list of the projects that students had done last year. And so I'm holding the list in my hand. You can go home later and take a look at it online.

But reading down this list, I see that last year, there were students that applied learning algorithms to control a snake robot. There was a few projects on improving learning algorithms. There's a project on flying autonomous aircraft. There was a project actually done by our TA Paul on improving computer vision algorithms using machine learning.

There are a couple of projects on Netflix rankings using learning algorithms; a few medical robots; ones on segmenting [inaudible] to segmenting pieces of the body using learning algorithms; one on musical instrument detection; another on irony sequence alignment; and a few algorithms on understanding the brain neuroscience, actually quite a few projects on neuroscience; a couple of projects on undescending fMRI data on brain scans, and so on; another project on market makings, the financial trading. There was an interesting project on trying to use learning algorithms to decide what is it that makes a person's face physically attractive. There's a learning algorithm on optical illusions, and so on.

And it goes on, so lots of fun projects. And take a look, then come up with your own ideas. But whatever you find cool and interesting, I hope you'll be able to make machine learning a project out of it. Yeah, question?

Student : Are these group projects?

Instructor (Andrew Ng): Oh, yes, thank you.

Student : So how many people can be in a group?

Instructor (Andrew Ng): Right. So projects can be done in groups of up to three people. So as part of forming study groups, later today as you get to know your classmates, I definitely also encourage you to grab two other people and form a group of up to three people for your project, okay? And just start brainstorming ideas for now amongst yourselves. You can also come and talk to me or the TAs if you want to brainstorm ideas with us.

Okay. So one more organizational question. I'm curious, how many of you know MATLAB? Wow, cool, quite a lot. Okay. So as part of the — actually how many of you know Octave or have used Octave? Oh, okay, much smaller number.

So as part of this class, especially in the homeworks, we'll ask you to implement a few programs, a few machine learning algorithms as part of the homeworks. And most of

those homeworks will be done in either MATLAB or in Octave, which is sort of — I know some people call it a free version of MATLAB, which it sort of is, sort of isn't.

So I guess for those of you that haven't seen MATLAB before, and I know most of you have, MATLAB is I guess part of the programming language that makes it very easy to write codes using matrices, to write code for numerical routines, to move data around, to plot data. And it's sort of an extremely easy to learn tool to use for implementing a lot of learning algorithms.

And in case some of you want to work on your own home computer or something if you don't have a MATLAB license, for the purposes of this class, there's also — [inaudible] write that down [inaudible] MATLAB — there's also a software package called Octave that you can download for free off the Internet. And it has somewhat fewer features than MATLAB, but it's free, and for the purposes of this class, it will work for just about everything.

So actually I, well, so yeah, just a side comment for those of you that haven't seen MATLAB before I guess, once a colleague of mine at a different university, not at Stanford, actually teaches another machine learning course. He's taught it for many years. So one day, he was in his office, and an old student of his from, like, ten years ago came into his office and he said, "Oh, professor, professor, thank you so much for your machine learning class. I learned so much from it. There's this stuff that I learned in your class, and I now use every day. And it's helped me make lots of money, and here's a picture of my big house."

So my friend was very excited. He said, "Wow. That's great. I'm glad to hear this machine learning stuff was actually useful. So what was it that you learned? Was it logistic regression? Was it the PCA? Was it the data networks? What was it that you learned that was so helpful?" And the student said, "Oh, it was the MATLAB."

So for those of you that don't know MATLAB yet, I hope you do learn it. It's not hard, and we'll actually have a short MATLAB tutorial in one of the discussion sections for those of you that don't know it.

Okay. The very last piece of logistical thing is the discussion sections. So discussion sections will be taught by the TAs, and attendance at discussion sections is optional, although they'll also be recorded and televised. And we'll use the discussion sections mainly for two things. For the next two or three weeks, we'll use the discussion sections to go over the prerequisites to this class or if some of you haven't seen probability or statistics for a while or maybe algebra, we'll go over those in the discussion sections as a refresher for those of you that want one.

Later in this quarter, we'll also use the discussion sections to go over extensions for the material that I'm teaching in the main lectures. So machine learning is a huge field, and there are a few extensions that we really want to teach but didn't have time in the main lectures for.

So later this quarter, we'll use the discussion sections to talk about things like convex optimization, to talk a little bit about hidden Markov models, which is a type of machine learning algorithm for modeling time series and a few other things, so extensions to the materials that I'll be covering in the main lectures. And attendance at the discussion sections is optional, okay?

So that was all I had from logistics. Before we move on to start talking a bit about machine learning, let me check what questions you have. Yeah?

Student : [Inaudible] R or something like that?

Instructor (Andrew Ng) : Oh, yeah, let's see, right. So our policy has been that you're welcome to use R, but I would strongly advise against it, mainly because in the last problem set, we actually supply some code that will run in Octave but that would be somewhat painful for you to translate into R yourself. So for your other assignments, if you wanna submit a solution in R, that's fine. But I think MATLAB is actually totally worth learning. I know R and MATLAB, and I personally end up using MATLAB quite a bit more often for various reasons. Yeah?

Student : For the [inaudible] project [inaudible]?

Instructor (Andrew Ng) : So for the term project, you're welcome to do it in smaller groups of three, or you're welcome to do it by yourself or in groups of two. Grading is the same regardless of the group size, so with a larger group, you probably — I recommend trying to form a team, but it's actually totally fine to do it in a smaller group if you want.

Student : [Inaudible] what language [inaudible]?

Instructor (Andrew Ng): So let's see. There is no C programming in this class other than any that you may choose to do yourself in your project. So all the homeworks can be done in MATLAB or Octave, and let's see. And I guess the program prerequisites is more the ability to understand big?O notation and knowledge of what a data structure, like a linked list or a queue or binary treatments, more so than your knowledge of C or Java specifically. Yeah?

Student : Looking at the end semester project, I mean, what exactly will you be testing over there? [Inaudible]?

Instructor (Andrew Ng) : Of the project?

Student : Yeah.

Instructor (Andrew Ng) : Yeah, let me answer that later. In a couple of weeks, I shall give out a handout with guidelines for the project. But for now, we should think of the goal as being to do a cool piece of machine learning work that will let you experience the

joys of machine learning firsthand and really try to think about doing a publishable piece of work.

So many students will try to build a cool machine learning application. That's probably the most common project. Some students will try to improve state-of-the-art machine learning. Some of those projects are also very successful. It's a little bit harder to do. And there's also a smaller minority of students that will sometimes try to prove — develop the theory of machine learning further or try to prove theorems about machine learning. So they're usually great projects of all of those types with applications and machine learning being the most common. Anything else? Okay, cool.

So that was it for logistics. Let's talk about learning algorithms. So can I have the laptop display, please, or the projector? Actually, could you lower the big screen? Cool. This is amazing customer service. Thank you. I see. Okay, cool. Okay. No, that's fine. I see. Okay. That's cool. Thanks. Okay.

Big screen isn't working today, but I hope you can read things on the smaller screens out there. Actually, [inaudible] I think this room just got a new projector that — someone sent you an excited email — was it just on Friday? — saying we just got a new projector and they said 4,000-to-1 something or other brightness ratio. I don't know. Someone was very excited about the new projector in this room, but I guess we'll see that in operation on Wednesday.

So start by talking about what machine learning is. What is machine learning? Actually, can you read the text out there? Raise your hand if the text on the small screens is legible. Oh, okay, cool, mostly legible. Okay. So I'll just read it out.

So what is machine learning? Way back in about 1959, Arthur Samuel defined machine learning informally as the [inaudible] that gives computers to learn — [inaudible] that gives computers the ability to learn without being explicitly programmed. So Arthur Samuel, so way back in the history of machine learning, actually did something very cool, which was he wrote a checkers program, which would play games of checkers against itself.

And so because a computer can play thousands of games against itself relatively quickly, Arthur Samuel had his program play thousands of games against itself, and over time it would start to learn to recognize patterns which led to wins and patterns which led to losses. So over time it learned things like that, "Gee, if I get a lot of pieces taken by the opponent, then I'm more likely to lose than win," or, "Gee, if I get my pieces into a certain position, then I'm especially likely to win rather than lose."

And so over time, Arthur Samuel had a checkers program that would actually learn to play checkers by learning what are the sort of board positions that tend to be associated with wins and what are the board positions that tend to be associated with losses. And way back around 1959, the amazing thing about this was that his program actually learned to play checkers much better than Arthur Samuel himself could.

So even today, there are some people that say, well, computers can't do anything that they're not explicitly programmed to. And Arthur Samuel's checkers program was maybe the first I think really convincing refutation of this claim. Namely, Arthur Samuel managed to write a checkers program that could play checkers much better than he personally could, and this is an instance of maybe computers learning to do things that they were not programmed explicitly to do.

Here's a more recent, a more modern, more formal definition of machine learning due to Tom Mitchell, who says that a well-posed learning problem is defined as follows: He says that a computer program is set to learn from an experience E with respect to some task T and some performance measure P if its performance on T as measured by P improves with experience E. Okay. So not only is it a definition, it even rhymes.

So, for example, in the case of checkers, the experience E that a program has would be the experience of playing lots of games of checkers against itself, say. The task T is the task of playing checkers, and the performance measure P will be something like the fraction of games it wins against a certain set of human opponents. And by this definition, we'll say that Arthur Samuel's checkers program has learned to play checkers, okay?

So as an overview of what we're going to do in this class, this class is sort of organized into four major sections. We're gonna talk about four major topics in this class, the first of which is supervised learning. So let me give you an example of that.

So suppose you collect a data set of housing prices. And one of the TAs, Dan Ramage, actually collected a data set for me last week to use in the example later. But suppose that you go to collect statistics about how much houses cost in a certain geographic area. And Dan, the TA, collected data from housing prices in Portland, Oregon. So what you can do is let's say plot the square footage of the house against the list price of the house, right, so you collect data on a bunch of houses. And let's say you get a data set like this with houses of different sizes that are listed for different amounts of money.

Now, let's say that I'm trying to sell a house in the same area as Portland, Oregon as where the data comes from. Let's say I have a house that's this size in square footage, and I want an algorithm to tell me about how much should I expect my house to sell for. So there are lots of ways to do this, and some of you may have seen elements of what I'm about to say before.

So one thing you could do is look at this data and maybe put a straight line to it. And then if this is my house, you may then look at the straight line and predict that my house is gonna go for about that much money, right? There are other decisions that we can make, which we'll talk about later, which is, well, what if I don't wanna put a straight line? Maybe I should put a quadratic function to it. Maybe that fits the data a little bit better. You notice if you do that, the price of my house goes up a bit, so that'd be nice.

And this sort of learning problem of learning to predict housing prices is an example of what's called a supervised learning problem. And the reason that it's called supervised learning is because we're providing the algorithm a data set of a bunch of square footages, a bunch of housing sizes, and as well as sort of the right answer of what the actual prices of a number of houses were, right?

So we call this supervised learning because we're supervising the algorithm or, in other words, we're giving the algorithm the, quote, right answer for a number of houses. And then we want the algorithm to learn the association between the inputs and the outputs and to sort of give us more of the right answers, okay?

It turns out this specific example that I drew here is an example of something called a regression problem. And the term regression sort of refers to the fact that the variable you're trying to predict is a continuous value and price.

There's another class of supervised learning problems which we'll talk about, which are classification problems. And so, in a classification problem, the variable you're trying to predict is discreet rather than continuous. So as one specific example — so actually a standard data set you can download online [inaudible] that lots of machine learning people have played with. Let's say you collect a data set on breast cancer tumors, and you want to learn the algorithm to predict whether or not a certain tumor is malignant. Malignant is the opposite of benign, right, so malignancy is a sort of harmful, bad tumor. So we collect some number of features, some number of properties of these tumors, and for the sake of sort of having a simple [inaudible] explanation, let's just say that we're going to look at the size of the tumor and depending on the size of the tumor, we'll try to figure out whether or not the tumor is malignant or benign.

So the tumor is either malignant or benign, and so the variable in the Y axis is either zero or 1, and so your data set may look something like that, right? And that's 1 and that's zero, okay? And so this is an example of a classification problem where the variable you're trying to predict is a discreet value. It's either zero or 1.

And in fact, more generally, there will be many learning problems where we'll have more than one input variable, more than one input feature and use more than one variable to try to predict, say, whether a tumor is malignant or benign. So, for example, continuing with this, you may instead have a data set that looks like this. I'm gonna part this data set in a slightly different way now. And I'm making this data set look much cleaner than it really is in reality for illustration, okay?

For example, maybe the crosses indicate malignant tumors and the "O"s may indicate benign tumors. And so you may have a data set comprising patients of different ages and who have different tumor sizes and where a cross indicates a malignant tumor, and an "O" indicates a benign tumor. And you may want an algorithm to learn to predict, given a new patient, whether their tumor is malignant or benign.

So, for example, what a learning algorithm may do is maybe come in and decide that a straight line like that separates the two classes of tumors really well, and so if you have a new patient who's age and tumor size fall over there, then the algorithm may predict that the tumor is benign rather than malignant, okay? So this is just another example of another supervised learning problem and another classification problem.

And so it turns out that one of the issues we'll talk about later in this class is in this specific example, we're going to try to predict whether a tumor is malignant or benign based on two features or based on two inputs, namely the age of the patient and the tumor size. It turns out that when you look at a real data set, you find that learning algorithms often use other sets of features. In the breast cancer data example, you also use properties of the tumors, like clump thickness, uniformity of cell size, uniformity of cell shape, [inaudible] adhesion and so on, so various other medical properties.

And one of the most interesting things we'll talk about later this quarter is what if your data doesn't lie in a two-dimensional or three-dimensional or sort of even a finite dimensional space, but is it possible — what if your data actually lies in an infinite dimensional space? Our plots here are two-dimensional space. I can't plot you an infinite dimensional space, right? And so it turns out that one of the most successful classes of machine learning algorithms — some may call support vector machines — actually takes data and maps data to an infinite dimensional space and then does classification using not two features like I've done here, but an infinite number of features.

And that will actually be one of the most fascinating things we talk about when we go deeply into classification algorithms. And it's actually an interesting question, right, so think about how do you even represent an infinite dimensional vector in computer memory? You don't have an infinite amount of computers. How do you even represent a point that lies in an infinite dimensional space? We'll talk about that when we get to support vector machines, okay?

So let's see. So that was supervised learning. The second of the four major topics of this class will be learning theory. So I have a friend who teaches math at a different university, not at Stanford, and when you talk to him about his work and what he's really out to do, this friend of mine will — he's a math professor, right? — this friend of mine will sort of get the look of wonder in his eyes, and he'll tell you about how in his mathematical work, he feels like he's discovering truth and beauty in the universe. And he says it in sort of a really touching, sincere way, and then he has this — you can see it in his eyes — he has this deep appreciation of the truth and beauty in the universe as revealed to him by the math he does.

In this class, I'm not gonna do any truth and beauty. In this class, I'm gonna talk about learning theory to try to convey to you an understanding of how and why learning algorithms work so that we can apply these learning algorithms as effectively as possible.

So, for example, it turns out you can prove surprisingly deep theorems on when you can guarantee that a learning algorithm will work, all right? So think about a learning

algorithm for reading zip codes. When can you prove a theorem guaranteeing that a learning algorithm will be at least 99.9 percent accurate on reading zip codes? This is actually somewhat surprising. We actually prove theorems showing when you can expect that to hold.

We'll also sort of delve into learning theory to try to understand what algorithms can approximate different functions well and also try to understand things like how much training data do you need? So how many examples of houses do I need in order for your learning algorithm to recognize the pattern between the square footage of a house and its housing price? And this will help us answer questions like if you're trying to design a learning algorithm, should you be spending more time collecting more data or is it a case that you already have enough data; it would be a waste of time to try to collect more. Okay?

So I think learning algorithms are a very powerful tool that as I walk around sort of industry in Silicon Valley or as I work with various businesses in CS and outside CS, I find that there's often a huge difference between how well someone who really understands this stuff can apply a learning algorithm versus someone who sort of gets it but sort of doesn't.

The analogy I like to think of is imagine you were going to a carpentry school instead of a machine learning class, right? If you go to a carpentry school, they can give you the tools of carpentry. They'll give you a hammer, a bunch of nails, a screwdriver or whatever. But a master carpenter will be able to use those tools far better than most of us in this room. I know a carpenter can do things with a hammer and nail that I couldn't possibly. And it's actually a little bit like that in machine learning, too. One thing that's sadly not taught in many courses on machine learning is how to take the tools of machine learning and really, really apply them well.

So in the same way, so the tools of machine learning are I wanna say quite a bit more advanced than the tools of carpentry. Maybe a carpenter will disagree. But a large part of this class will be just giving you the raw tools of machine learning, just the algorithms and so on. But what I plan to do throughout this entire quarter, not just in the segment of learning theory, but actually as a theme running through everything I do this quarter, will be to try to convey to you the skills to really take the learning algorithm ideas and really to get them to work on a problem.

It's sort of hard for me to stand here and say how big a deal that is, but when I walk around companies in Silicon Valley, it's completely not uncommon for me to see someone using some machine learning algorithm and then explain to me what they've been doing for the last six months, and I go, oh, gee, it should have been obvious from the start that the last six months, you've been wasting your time, right?

And so my goal in this class, running through the entire quarter, not just on learning theory, is actually not only to give you the tools of machine learning, but to teach you how to use them well. And I've noticed this is something that really not many other

classes teach. And this is something I'm really convinced is a huge deal, and so by the end of this class, I hope all of you will be master carpenters. I hope all of you will be really good at applying these learning algorithms and getting them to work amazingly well in many problems. Okay?

Let's see. So [inaudible] the board. After learning theory, there's another class of learning algorithms that I then want to teach you about, and that's unsupervised learning. So you recall, right, a little earlier I drew an example like this, right, where you have a couple of features, a couple of input variables and sort of malignant tumors and benign tumors or whatever. And that was an example of a supervised learning problem because the data you have gives you the right answer for each of your patients. The data tells you this patient has a malignant tumor; this patient has a benign tumor. So it had the right answers, and you wanted the algorithm to just produce more of the same.

In contrast, in an unsupervised learning problem, this is the sort of data you get, okay? Where speaking loosely, you're given a data set, and I'm not gonna tell you what the right answer is on any of your data. I'm just gonna give you a data set and I'm gonna say, "Would you please find interesting structure in this data set?" So that's the unsupervised learning problem where you're sort of not given the right answer for everything.

So, for example, an algorithm may find structure in the data in the form of the data being partitioned into two clusters, or clustering is sort of one example of an unsupervised learning problem.

So I hope you can see this. It turns out that these sort of unsupervised learning algorithms are also used in many problems. This is a screen shot — this is a picture I got from Sue Emvee, who's a PhD student here, who is applying unsupervised learning algorithms to try to understand gene data, so is trying to look at genes as individuals and group them into clusters based on properties of what genes they respond to — based on properties of how the genes respond to different experiments.

Another interesting application of [inaudible] sorts of clustering algorithms is actually image processing, this which I got from Steve Gules, who's another PhD student. It turns out what you can do is if you give this sort of data, say an image, to certain unsupervised learning algorithms, they will then learn to group pixels together and say, gee, this sort of pixel seems to belong together, and that sort of pixel seems to belong together.

And so the images you see on the bottom — I guess you can just barely see them on there — so the images you see on the bottom are groupings — are what the algorithm has done to group certain pixels together. On a small display, it might be easier to just look at the image on the right. The two images on the bottom are two sort of identical visualizations of the same grouping of the pixels into [inaudible] regions.

And so it turns out that this sort of clustering algorithm or this sort of unsupervised learning algorithm, which learns to group pixels together, it turns out to be useful for many applications in vision, in computer vision image processing.

I'll just show you one example, and this is a rather cool one that two students, Ashutosh Saxena and Min Sun here did, which is given an image like this, right? This is actually a picture taken of the Stanford campus. You can apply that sort of clustering algorithm and group the picture into regions. Let me actually blow that up so that you can see it more clearly. Okay. So in the middle, you see the lines sort of grouping the image together, grouping the image into [inaudible] regions.

And what Ashutosh and Min did was they then applied the learning algorithm to say can we take this clustering and use it to build a 3D model of the world? And so using the clustering, they then had a learning algorithm try to learn what the 3D structure of the world looks like so that they could come up with a 3D model that you can sort of fly through, okay? Although many people used to think it's not possible to take a single image and build a 3D model, but using a learning algorithm and that sort of clustering algorithm is the first step. They were able to.

I'll just show you one more example. I like this because it's a picture of Stanford with our beautiful Stanford campus. So again, taking the same sort of clustering algorithms, taking the same sort of unsupervised learning algorithm, you can group the pixels into different regions. And using that as a pre-processing step, they eventually built this sort of 3D model of Stanford campus in a single picture. You can sort of walk into the ceiling, look around the campus. Okay? This actually turned out to be a mix of supervised and unsupervised learning, but the unsupervised learning, this sort of clustering was the first step.

So it turns out these sorts of unsupervised — clustering algorithms are actually routinely used for many different problems, things like organizing computing clusters, social network analysis, market segmentation, so if you're a marketer and you want to divide your market into different segments or different groups of people to market to them separately; even for astronomical data analysis and understanding how galaxies are formed. These are just a sort of small sample of the applications of unsupervised learning algorithms and clustering algorithms that we'll talk about later in this class.

Just one particularly cool example of an unsupervised learning algorithm that I want to tell you about. And to motivate that, I'm gonna tell you about what's called the cocktail party problem, which is imagine that you're at some cocktail party and there are lots of people standing all over. And you know how it is, right, if you're at a large party, everyone's talking, it can be sometimes very hard to hear even the person in front of you. So imagine a large cocktail party with lots of people. So the problem is, is that all of these people talking, can you separate out the voice of just the person you're interested in talking to with all this loud background noise?

So I'll show you a specific example in a second, but here's a cocktail party that's I guess rather sparsely attended by just two people. But what we're gonna do is we'll put two microphones in the room, okay? And so because the microphones are just at slightly different distances to the two people, and the two people may speak in slightly different volumes, each microphone will pick up an overlapping combination of these two people's

voices, so slightly different overlapping voices. So Speaker 1's voice may be more loud on Microphone 1, and Speaker 2's voice may be louder on Microphone 2, whatever.

But the question is, given these microphone recordings, can you separate out the original speaker's voices? So I'm gonna play some audio clips that were collected by Tai Yuan Lee at UCSD. I'm gonna actually play for you the original raw microphone recordings from this cocktail party. So this is the Microphone 1:

Microphone 1:

One, two, three, four, five, six, seven, eight, nine, ten.

Microphone 2:

Uno, dos, tres, cuatro, cinco, seis, siete, ocho, nueve, diez.

Instructor (Andrew Ng) : So it's a fascinating cocktail party with people counting from one to ten. This is the second microphone:

Microphone 1:

One, two, three, four, five, six, seven, eight, nine, ten.

Microphone 2:

Uno, dos, tres, cuatro, cinco, seis, siete, ocho, nueve, diez.

Instructor (Andrew Ng) : Okay. So in supervised learning, we don't know what the right answer is, right? So what we're going to do is take exactly the two microphone recordings you just heard and give it to an unsupervised learning algorithm and tell the algorithm which of these discover structure in the data [inaudible] or what structure is there in this data? And we actually don't know what the right answer is offhand.

So give this data to an unsupervised learning algorithm, and what the algorithm does in this case, it will discover that this data can actually be explained by two independent speakers speaking at the same time, and it can further separate out the two speakers for you. So here's Output 1 of the algorithm:

Microphone 1:

One, two, three, four, five, six, seven, eight, nine, ten.

Instructor (Andrew Ng) : And there's the second algorithm:

Microphone 2:

Uno, dos, tres, cuatro, cinco, seis, siete, ocho, nueve, diez.

Instructor (Andrew Ng): And so the algorithm discovers that, gee, the structure underlying the data is really that there are two sources of sound, and here they are. I'll show you one more example. This is a, well, this is a second sort of different pair of microphone recordings:

Microphone 1:

One, two, three, four, five, six, seven, eight, nine, ten.

Microphone 2:

[Music playing.]

Instructor (Andrew Ng): So the poor guy is not at a cocktail party. He's talking to his radio. There's the second recording:

Microphone 1:

One, two, three, four, five, six, seven, eight, nine, ten.

Microphone 2:

[Music playing.]

Instructor (Andrew Ng) : Right. And we get this data. It's the same unsupervised learning algorithm. The algorithm is actually called independent component analysis, and later in this quarter, you'll see why. And then output's the following:

Microphone 1:

One, two, three, four, five, six, seven, eight, nine, ten.

Instructor (Andrew Ng): And that's the second one:

Microphone 2:

[Music playing.]

Instructor (Andrew Ng): Okay. So it turns out that beyond solving the cocktail party algorithm, this specific class of unsupervised learning algorithms are also applied to a bunch of other problems, like in text processing or understanding functional grading and machine data, like the magneto-encephalogram would be an EEG data. We'll talk about that more when we go and describe ICA or independent component analysis algorithms, which is what you just saw.

And as an aside, this algorithm I just showed you, it seems like it must be a pretty complicated algorithm, right, to take this overlapping audio streams and separate them out. It sounds like a pretty complicated thing to do. So you're gonna ask how complicated is it really to implement an algorithm like this? It turns out if you do it in MATLAB, you can do it in one line of code.

So I got this from Samuel Wyse at Toronto, U of Toronto, and the example I showed you actually used a more complicated ICA algorithm than this. But nonetheless, I guess this is why for this class I'm going to ask you to do most of your programming in MATLAB and Octave because if you try to implement the same algorithm in C or Java or something, I can tell you from personal, painful experience, you end up writing pages and pages of code rather than relatively few lines of code. I'll also mention that it did take researchers many, many years to come up with that one line of code, so this is not easy.

So that was unsupervised learning, and then the last of the four major topics I wanna tell you about is reinforcement learning. And this refers to problems where you don't do one-shot decision-making. So, for example, in the supervised learning cancer prediction problem, you have a patient come in, you predict that the cancer is malignant or benign. And then based on your prediction, maybe the patient lives or dies, and then that's it, right? So you make a decision and then there's a consequence. You either got it right or wrong. In reinforcement learning problems, you are usually asked to make a sequence of decisions over time.

So, for example, this is something that my students and I work on. If I give you the keys to an autonomous helicopter — we actually have this helicopter here at Stanford, — how do you write a program to make it fly, right? You notice that if you make a wrong decision on a helicopter, the consequence of crashing it may not happen until much later. And in fact, usually you need to make a whole sequence of bad decisions to crash a helicopter. But conversely, you also need to make a whole sequence of good decisions in order to fly a helicopter really well.

So I'm gonna show you some fun videos of learning algorithms flying helicopters. This is a video of our helicopter at Stanford flying using a controller that was learned using a reinforcement learning algorithm. So this was done on the Stanford football field, and we'll zoom out the camera in a second. You'll sort of see the trees planted in the sky. So maybe this is one of the most difficult aerobatic maneuvers flown on any helicopter under computer control. And this controller, which is very, very hard for a human to sit down and write out, was learned using one of these reinforcement learning algorithms.

Just a word about that: The basic idea behind a reinforcement learning algorithm is this idea of what's called a reward function. What we have to think about is imagine you're trying to train a dog. So every time your dog does something good, you say, "Good dog," and you reward the dog. Every time your dog does something bad, you go, "Bad dog," right? And hopefully, over time, your dog will learn to do the right things to get more of the positive rewards, to get more of the "Good dogs" and to get fewer of the "Bad dogs."

So the way we teach a helicopter to fly or any of these robots is sort of the same thing. Every time the helicopter crashes, we go, "Bad helicopter," and every time it does the right thing, we go, "Good helicopter," and over time it learns how to control itself so as to get more of these positive rewards.

So reinforcement learning is — I think of it as a way for you to specify what you want done, so you have to specify what is a "good dog" and what is a "bad dog" behavior. And then it's up to the learning algorithm to figure out how to maximize the "good dog" reward signals and minimize the "bad dog" punishments.

So it turns out reinforcement learning is applied to other problems in robotics. It's applied to things in web crawling and so on. But it's just cool to show videos, so let me just show a bunch of them. This learning algorithm was actually implemented by our head TA, Zico, of programming a four-legged dog. I guess Sam Shriver in this class also worked on the project and Peter Renfrew and Mike and a few others. But I guess this really is a good dog/bad dog since it's a robot dog.

The second video on the right, some of the students, I guess Peter, Zico, Tonca working on a robotic snake, again using learning algorithms to teach a snake robot to climb over obstacles.

Below that, this is kind of a fun example. Ashutosh Saxena and Jeff Michaels used learning algorithms to teach a car how to drive at reasonably high speeds off roads avoiding obstacles.

And on the lower right, that's a robot programmed by PhD student Eva Roshen to teach a sort of somewhat strangely configured robot how to get on top of an obstacle, how to get over an obstacle. Sorry. I know the video's kind of small. I hope you can sort of see it. Okay?

So I think all of these are robots that I think are very difficult to hand-code a controller for by learning these sorts of learning algorithms. You can in relatively short order get a robot to do often pretty amazing things.

Okay. So that was most of what I wanted to say today. Just a couple more last things, but let me just check what questions you have right now. So if there are no questions, I'll just close with two reminders, which are after class today or as you start to talk with other people in this class, I just encourage you again to start to form project partners, to try to find project partners to do your project with. And also, this is a good time to start forming study groups, so either talk to your friends or post in the newsgroup, but we just encourage you to try to start to do both of those today, okay? Form study groups, and try to find two other project partners.

So thank you. I'm looking forward to teaching this class, and I'll see you in a couple of days.

[End of Audio]

Duration: 69 minutes