Designing, Visualizing and Understanding Deep Neural Networks

Lecture 2: Machine Learning Background I

CS 182/282A Spring 2020
John Canny

Last Time

- Hype, Hope and Risks for Deep Networks.
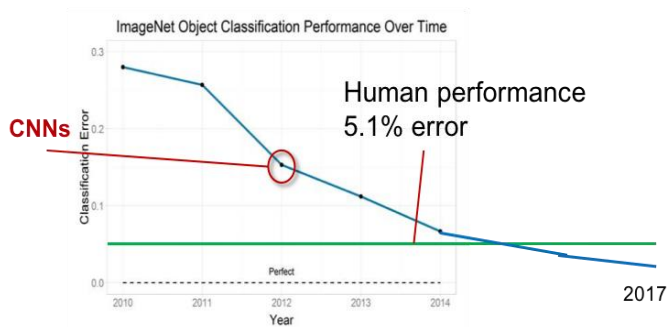
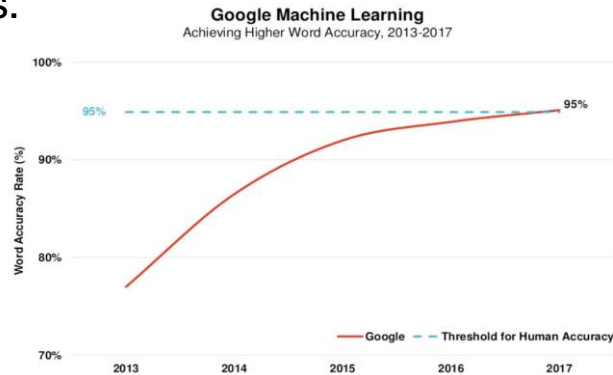- Performance of Deep networks:
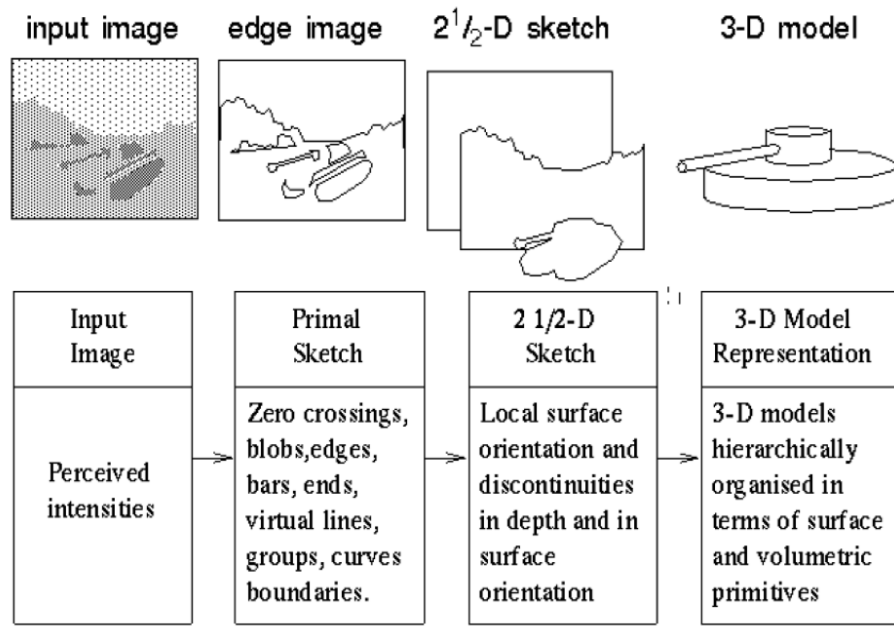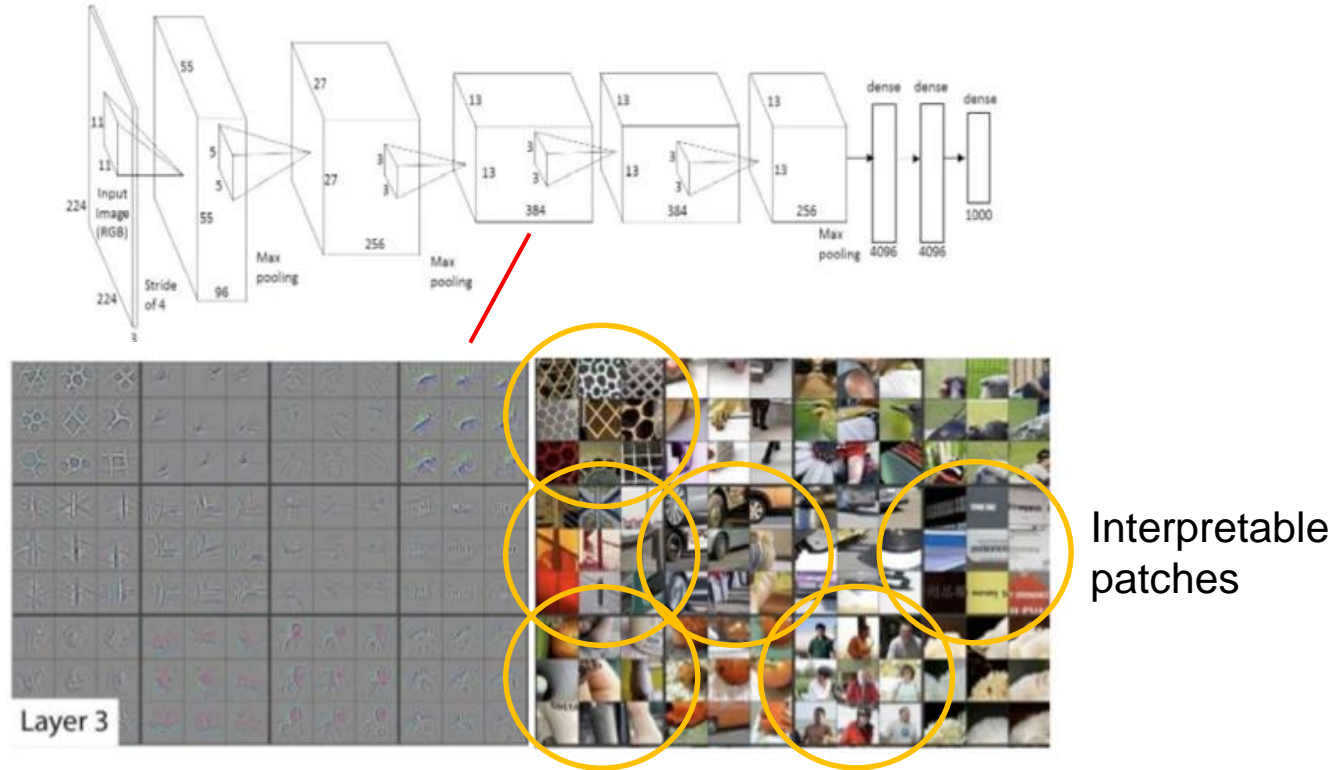

Image Classification

Speech Recognition

Games: AlphaGo

Last Time: Hierarchical Representations

Hierarchies of Representation in Human Vision:

David Marr, 1979

Last Time: Representation Learning

Interpretable patches

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", 2013

4

iClickers

Please go ahead and acquire an iClicker if you are in a lecture section.

There are plenty of units at the campus bookstore.

Any iClicker (iClicker+ or iClicker2) will work.

We'll start using them in a couple of weeks.

Machine Learning Background I

Start with a space of "observations" $\mathcal{X}$ and a space of "targets" or "labels" $\mathcal{Y}$.

We are interested in how the observations determine the targets.

Data: many pairs (x_i, y_i) with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$.

Prediction: given a new observation x, predict the corresponding y.

Typically observations are "cheap" (raw data) while targets are "expensive" (may be generated by a human). Computer predictions therefore have economic value.

Prediction Problems

Observation Space $\mathcal{X}$:	Target Space $\mathcal{Y}$:
Images	Image class: "cat", "dog" etc.
Images	Caption: "kids playing soccer"
Face Images	User's identity
Natural Images	Stylized Images (e.g. cartoons)
Human Speech Waveform	Text transcript of the speech
Scene Description in English	**Sketch of the Scene**
Video from an Automobile Camera	**Steering, Accelerator and Braking**
General Video Segment	**Closed Caption Text**

Generative vs. Discriminative Models

Generative:

Discriminative:

Generative vs. Discriminative Models

Generative:

- Compute a model of the full joint data distribution $P(X, Y)$.

- Allow you to "generate" new synthetic data pairs (x_i, y_i).

Discriminative:

Generative vs. Discriminative Models

Generative:

- Compute a model of the full joint data distribution $P(X, Y)$.

- Allow you to "generate" new synthetic data pairs (x_i, y_i).

Discriminative:

- Compute only a model of target values conditioned on the data: $P(Y|X)$.

Quick Review

The conditional probability $P(Y|X)$ is (select one):

A. $P(X, Y)\, P(X)$

B. $P(X, Y)/P(X)$

C. $P(X, Y)/P(Y)$

D. $P(X, Y)\, P(Y)$

Skip this question

Quick Review

Sensei Wu! your spinjitzu looks rusty!

The conditional probability $P(Y|X)$ is:

Sorry, you're a little rusty. Do you want to (select one):

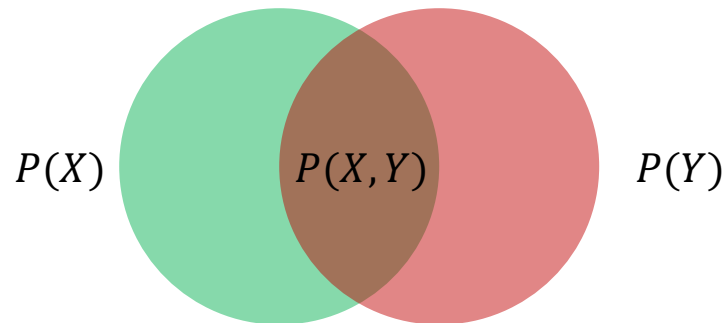Go back and try again?   Continue?

Quick Review

The conditional probability $P(Y|X)$ is:

$P(X,Y)/P(X)$

Congratulations, Sensei! The force is still with you.

e.g. for X, Y taking on Boolean values over a sample space:

$P(X)$ $P(X,Y)$ $P(Y)$

Please continue.

Generative vs. Discriminative Models

Generative:

Linear Functions plus Gaussian Noise

Naïve Bayes

Hidden Markov Models

Gaussian mixture models

Latent Dirichlet Allocation

Discriminative:

Linear Least Squares Regression

Logistic Regression

Conditional Random Fields

Support Vector Machines (SVM)

Decision trees + Random Forests

Neural Networks

Note: the correspondence is not 1-1, generative models usually include additional assumptions about the data

A Generative Model (Naïve Bayes)

Example:

- A Naïve Bayes model for images.

- Assume the image is binary (black or white pixels), the naïve Bayes model is specified by

the probabilities $P_q(X = \text{white} \mid Y = "woman")$ for each pixel q,

the probabilities $P_q(X = \text{white} \mid Y = "man")$ for each pixel q,

the probabilities $P_q(X = \text{white} \mid Y = "cat")$ for each pixel q,…

and the class probabilities $P(Y = "woman"),\ P(Y = "man"), …$

Generate an image for the class "woman": sample each pixel q **_independently_** according $P_q(X = \text{white} \mid Y = "woman")$

A Generative Model (Naïve Bayes)

- Assume the image is binary (black or white pixels), and given

the probabilities $P_q(X = \text{white} \,|\, Y = "woman")$ for each pixel q,

the probabilities $P_q(X = \text{white} \,|\, Y = "man")$ for each pixel q,

the probabilities $P_q(X = \text{white} \,|\, Y = "cat")$ for each pixel q,…

The function $P_q(X|Y)$ is called a **class-conditional density**.

Generative vs. Discriminative Tradeoffs

Example:

- A Naïve Bayes model for images.

- The independence assumption is not realistic for images.

Gives a very bad generator, and also a bad classifier.

But efficient: size of the model is proportional to
number of pixels x number of classes (so its small)

Generative vs. Discriminative Tradeoffs

Generative:

- Strong assumptions about $P(X, Y)$, especially re: independence.

- Insights into the physical process generating the data.

- Faster training.

- Better performance with sparse data.

- Biased if assumptions are violated, asymptotic accuracy may be poor.

Discriminative:

- Weak assumptions about data and dependencies.

- Little or no insight into data generation.

- May require more training data for modest accuracy.

- Accuracy continues to improve.

- Fewer forms of bias.

Generative vs. Discriminative Tradeoffs

For a quantitative analysis, see:

Andrew Ng., Michael Jordan, "On Discriminative vs. Generative classifiers, a comparison of logistic regression and naïve bayes", NIPS 2001.

Summary:

- The Naïve Bayes model has higher asymptotic error (error limit with unlimited training).
- The Naïve Bayes model approaches its asymptotic error much faster, i.e. logarithmic in the number of model parameters, while the logistic model is linear.

Generative vs. Discriminative Tradeoffs

Understanding the Data:

- **Generative models** are favored by many researchers as true "models" or "theories" of the dataset, i.e. they enlighten the physical processes that created the data.

- If a model has better accuracy than a baseline model, then it serves as evidence that the theory is right.

Fitting the Data:

- Few real-world datasets are well-explained by simple models.

- For every rule there are myriad adjustments and corrections, e.g. tax codes.

- **Discriminative models** make fewer assumptions, and can therefore typically model more complex dataset dependencies.

20

Intuition

Consider a model for the set X of 256 x 256 images.

- Let $P(X)$ be a probability density for images of "cat" or "snake".
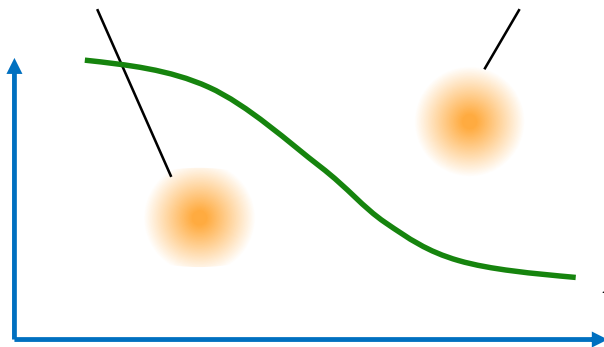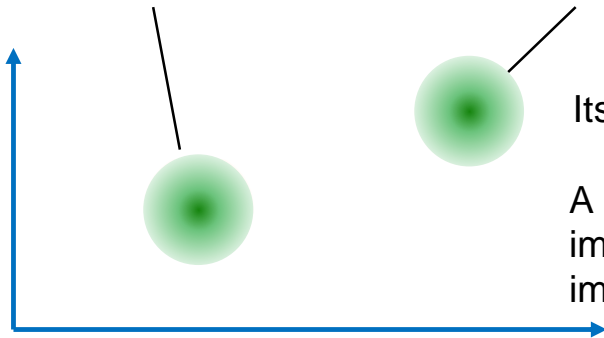

A discriminative model is easy to compute:

$P(Y = \text{"snake"}|X) = 0.5$

Feature coordinates (actually 65,536 dimensions)

Intuition

Consider a model for the set X of 256 x 256 images.

- Let $P(X)$ be a probability density for images of "cat" or "snake".

Its **much** harder to create a generative model $P(X, Y)$:

A generative model has to produce photo-realistic images of both cats and snakes, or at least do so implicitly (by assigning a density to every possible image).

22

Limitations of Discriminative Models

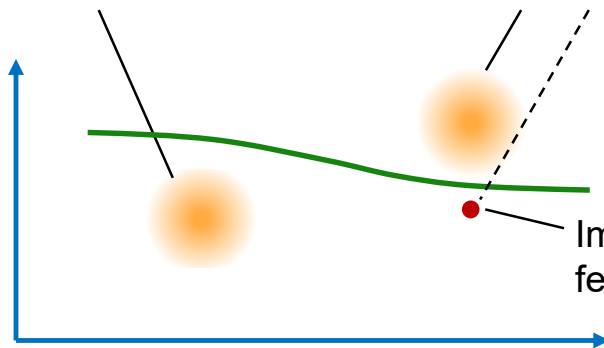- Models $P(Y|X)$ will often perform poorly off the "manifold" of natural images.

Image classified as "cat" which is extremely close in feature space to (and appears to humans as) "snake"

Feature coordinates (actually 65,536 dimensions)

Prediction Functions

We can simplify the modeling task by making a strong assumption about the model $\hat{P}(X, Y)$, namely that $y = f(x)$, i.e. y **takes a single value** given x.

Examples:

- Linear regression, $y = f(x)$ is a linear function, and $y \in \mathbb{R}$ is a real value.

- Classifiers (SVM, random forest etc.), $y = f(x)$ is the predicted class of x, and $y \in \{1, \ldots, k\}$ is the class number.

Loss Functions

- There may be no "true" target value y for an observation x, i.e. there may be several different y's for the same x.

- There may also be noise or unmodeled effects in the dataset, so even if there is a single y for a given x, it may be impossible to predict it exactly.

- Instead we try to predict a value that is "close to" the observed target values. We use a loss function to measure closeness.
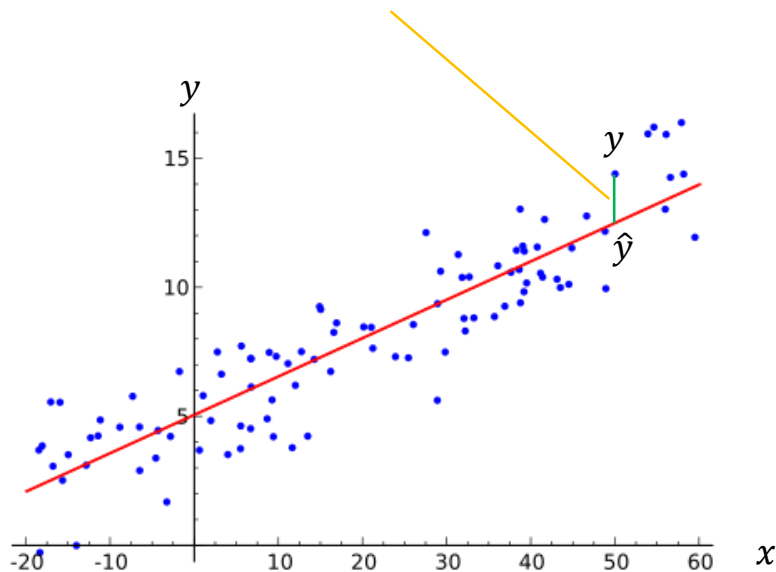
A **loss function** measures the difference between a target prediction and target data value.

e.g. squared loss $L_2(\hat{y}, y) = (\hat{y} - y)^2$ where $\hat{y} = f(x)$ is the prediction, (x, y) is the data pair.

Linear Regression

Simplest case, $\hat{y} = f(x) = ax + b$ with x a real value, and real constants a and b.
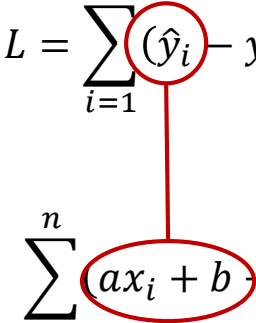
The loss is the squared loss $L_2(\hat{y}, y) = (\hat{y} - y)^2$

Data (x, y) pairs are the blue points.
The model is the red line.

Linear Regression

The total loss across all points is

$$L = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

$$= \sum_{i=1}^{n} (ax_i + b - y_i)^2$$

We want the optimum values of a and b, and since the loss is differentiable, we set

$$\frac{dL}{da} = 0 \qquad \text{and} \qquad \frac{dL}{db} = 0$$

Linear Regression

We want the loss-minimizing values of a and b, so we set

$$\frac{dL}{da} = 0 = 2a \sum_{i=1}^{n} x_i^2 + 2b \sum_{i=1}^{n} x_i - 2 \sum_{i=1}^{n} x_i y_i$$

$$\frac{dL}{db} = 0 = 2a \sum_{i=1}^{n} x_i + 2bn - 2 \sum_{i=1}^{n} y_i$$

Two linear equations in a and b, easily solved.

The model $f(x) = ax + b$ with these values is the unique minimum-loss model.

Aside: the least-squares loss is convex, making this work.

Risk Minimization

Wait a minute, what did we just do?

We found constants a and b which minimize the squared loss on some data ***we already have***.

But what we really want to do is predict the y values for points x ***we haven't seen yet***. i.e. we would like to minimize the expected loss on some new data:

$$\mathbb{E}[(\hat{y} - y)^2]$$

The expected loss is called **risk**.

What we actually minimized was an averaged loss across a finite number of data points. This averaged loss is called **empirical risk.**

Machine learning approximates risk-minimizing models with empirical-risk minimizing ones.

Risk Minimization

Generally minimizing empirical risk (loss on the data) instead of true risk works fine, but it can fail if:

- The **data sample is biased**. e.g. you cant build a (good) classifier with observations of only one class.

- There is **not enough data** to accurately estimate the parameters of the model. Depends on the complexity (number of parameters, variation in gradients, complexity of the loss function, generative vs. discriminative etc.).

Multivariate Linear Regression

This time, $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^k$, and the model is

$$y = Ax$$

for a $k \times m$ matrix A.

We assume the last coordinate of each x vector is 1. Then the last column of A acts as the bias vector b we had before, simplifying the model.

The empirical risk is

$$L = \sum_{i=1}^{n} (Ax_i - y_i)^2 = \sum_{i=1}^{n} (x_i^T A^T - y_i^T)(Ax_i - y_i)$$

Multivariate Linear Regression

We want to optimize L with respect to the model matrix A, so this time we need a gradient:

$$\nabla_A L = 0$$

- we are treating the loss as a function $L(A)$ of the matrix A.
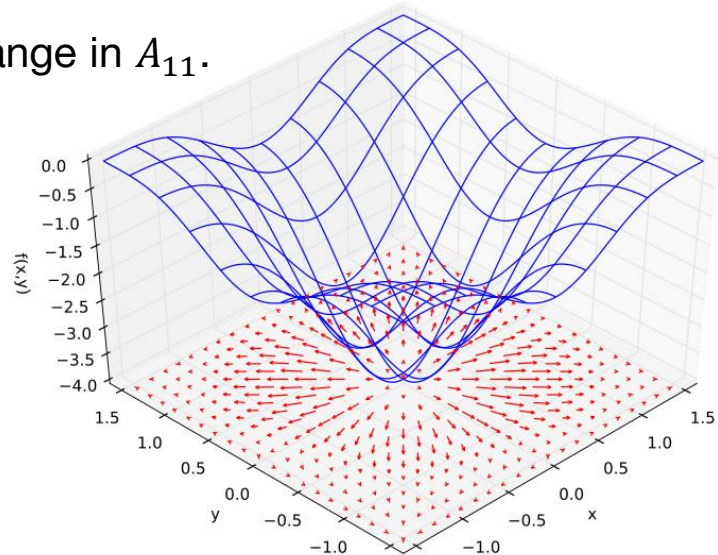
Aside: Gradients

When we write $\nabla_A L(A)$, we mean the vector of partial derivatives:

$$\nabla_A L(A) = \left[\frac{\partial L}{\partial A_{11}} , \frac{\partial L}{\partial A_{12}} , \frac{\partial L}{\partial A_{21}} , \frac{\partial L}{\partial A_{22}} \right]^T$$

Where $\frac{\partial L}{\partial A_{11}}$ measures how fast the loss changes vs. change in A_{11}.
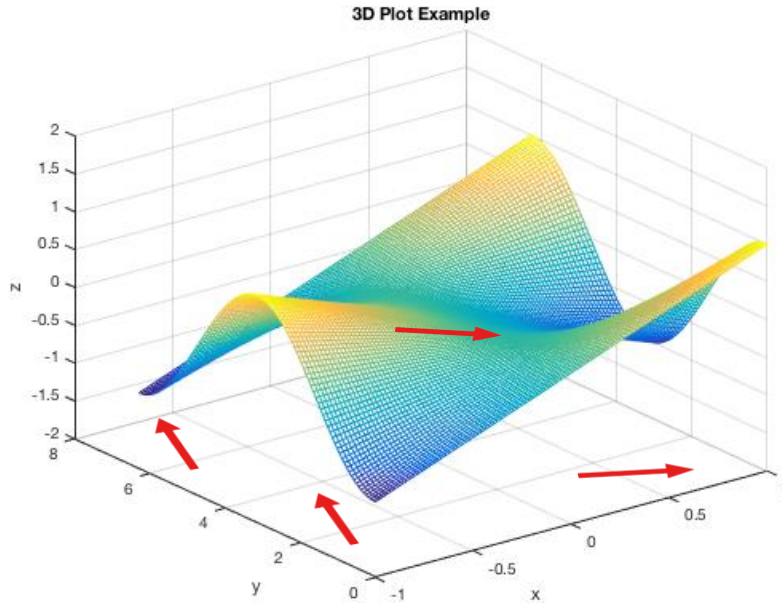
When $\nabla_A L(A) = 0$, it means all the partials are zero.
i.e. the loss is not changing in any direction.

Thus we are at a local optimum
(or at least a saddle point).

Aside: Gradients

Click on some vectors that represent the gradient of the plotted function $z = f(x, y)$:

3D Plot Example

Skip this question
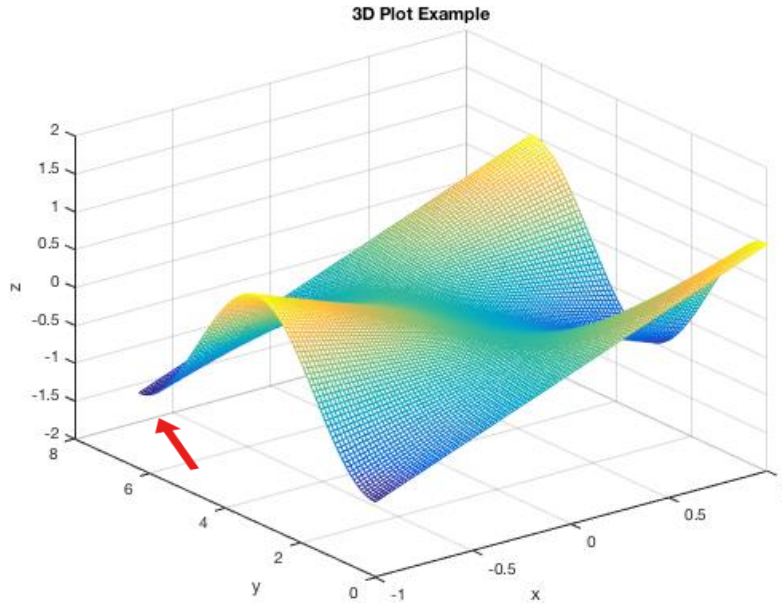
Aside: Gradients

Sorry, you missed.

Go back and try again?

Continue?

Aside: Gradients

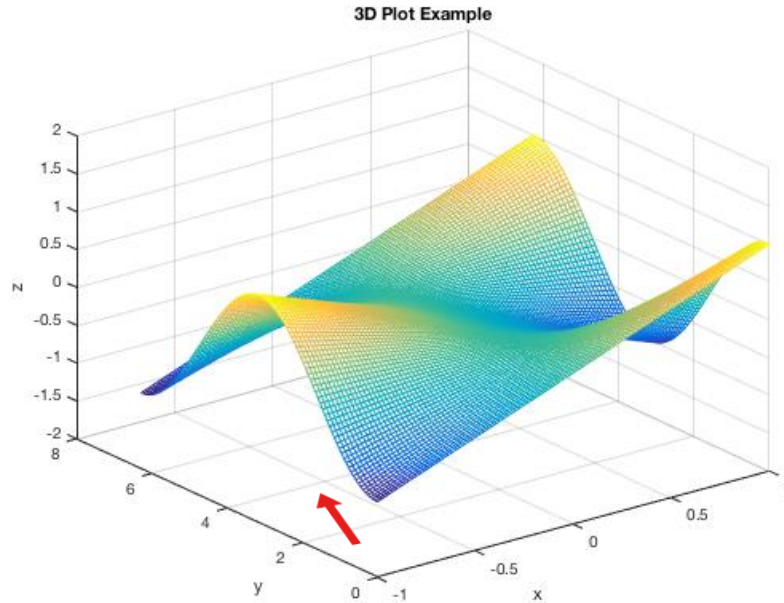Sorry, the function's gradient in y is negative here. The arrow shows a positive y gradient.

Try again?

3D Plot Example

Continue?

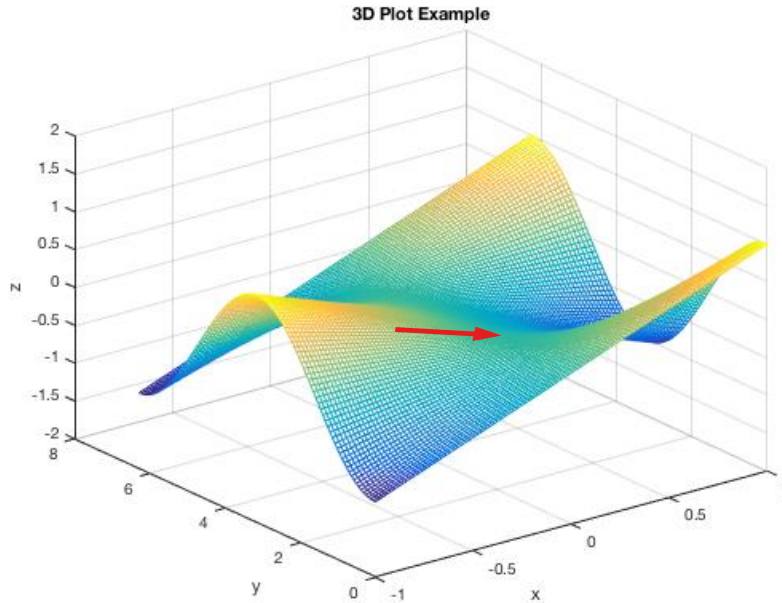# Aside: Gradients

Yes, Correct!

3D Plot Example

Please click
to Continue

Aside: Gradients

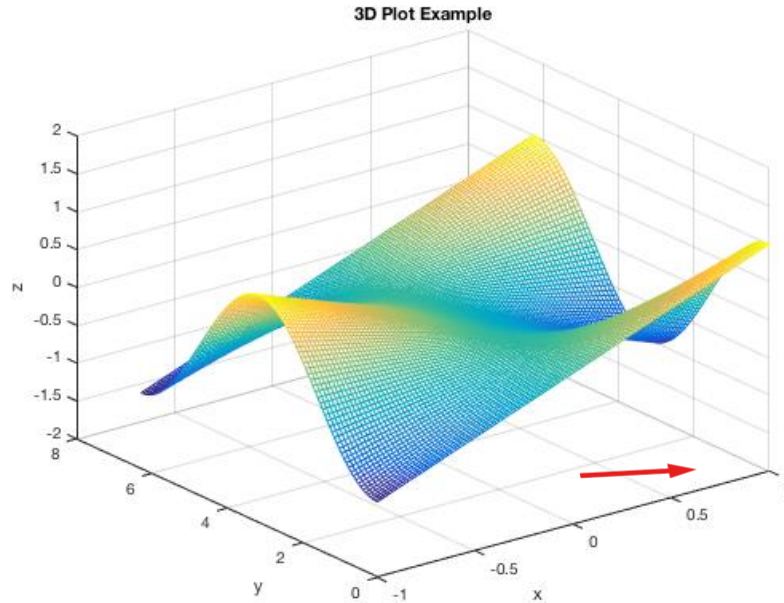The gradients of this function are two-dimensional. This arrow is meaningless.

Try again?

Continue?

3D Plot Example

Aside: Gradients

Yes, Correct!

3D Plot Example

Multivariate Linear Regression

We want to optimize L with respect to the model matrix A, so we use the gradient:
$$\nabla_A L = 0$$

- we are treating the loss as a function $L(A)$ of the matrix A (fixing x's and y's):

$$L(A) = \sum_{i=1}^{n} \left(x_i^T A^T - y_i^T \right) (A x_i - y_i)$$

The gradient of the loss is

$$\nabla_A L = 2A \sum_{i=1}^{n} x_i x_i^T - 2 \sum_{i=1}^{n} y_i x_i^T = 0$$

Which we can write as $2A M_{xx} - 2 M_{yx} = 0$

where $M_{xx} = \sum_{i=1}^{n} x_i x_i^T$ and $M_{yx} = \sum_{i=1}^{n} y_i x_i^T$

Multivariate Linear Regression

The solution for zero loss gradient: $\nabla_A L(A) = 2AM_{xx} - 2M_{yx} = 0$ is:

$$A = M_{yx}M_{xx}^{-1}$$

Where $M_{xx} = \sum_{i=1}^{n} x_i x_i^T$ and $M_{yx} = \sum_{i=1}^{n} y_i x_i^T$

Also works if we define $M_{xx} = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$ and $M_{yx} = \frac{1}{n}\sum_{i=1}^{n} y_i x_i^T$

These are called *sample moments*, and they converge to the true moments

$$M_{xx} \rightarrow \mu_{xx} = \mathbb{E}[xx^T]$$
$$M_{yx} \rightarrow \mu_{yx} = \mathbb{E}[yx^T]$$

So our *empirical risk-minimizing model* $A = M_{yx}M_{xx}^{-1}$ converges to the *true risk minimizing model* $A_0 = \mu_{yx}\mu_{xx}^{-1}$

41

Logistic Regression

Setup:

This time x is an ***m-dimensional binary vector***, i.e. $x \in \{0,1\}^m$, it could be the pixels in a binary image.

The target values ***y will also be binary***, $y \in \{0,1\}$, but our prediction $\hat{y} = f(x)$ ***will actually be a real value*** in the interval $[0,1]$.

We can apply a threshold $\hat{y} > y_{threshold}$ later to get a class assignment to $\{0,1\}$

The target class could be "cat". The data are pairs (x, y) of images and labels, and $y = 1$ means x is a cat image, while $y = 0$ means x is no cat.

Think of $\hat{y} = f(x)$, the output of our model, as being the probability that the image x is cat.
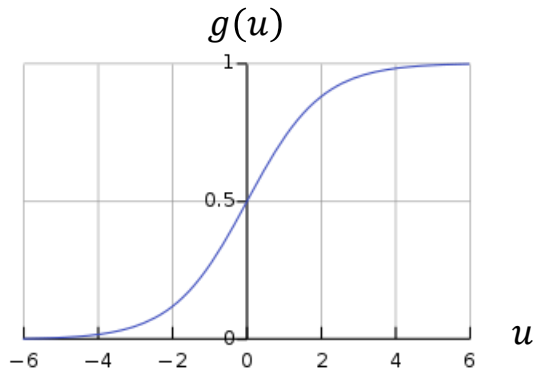
Logistic Regression

Let $w \in \mathbb{R}^m$ be a weight vector (the logistic model). Then

$$f(x) = \frac{1}{1 + \exp(-w^T x)}$$

Is the probability that the output should be "cat".

We can write this as $f(x) = g(w^T x)$ where $g(u) = 1/(1 + \exp(-u))$ is the ***logistic function:***
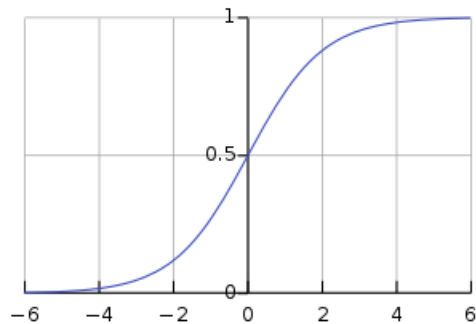
Logistic Regression

Let $w \in \mathbb{R}^m$ be a weight vector (the logistic model). Then

$$f(x) = \frac{1}{1 + \exp(-w^T x)}$$

Notice that the output is:

$$f(x) = \begin{cases} < 0.5 & \text{if} \quad w^T x < 0 \\ 0.5 & \text{if} \quad w^T x = 0 \\ > 0.5 & \text{if} \quad w^T x > 0 \end{cases}$$

So choosing a threshold of $y_{threshold} = 0.5$ gives the output of the linear classifier $w^T x > 0$.
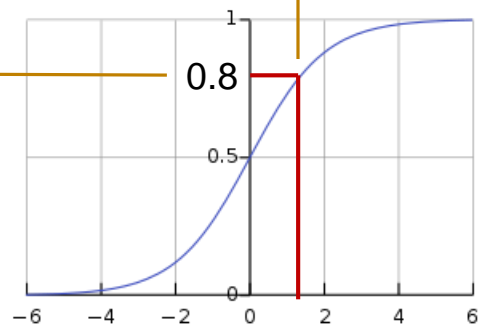

44

Logistic Regression

Let $w \in \mathbb{R}^m$ be a weight vector (the logistic model). Then

$$f(x) = \frac{1}{1 + \exp(-w^T x)}$$

Notice that the output is:

$$f(x) = \begin{cases} < 0.8 & \text{if} \quad w^T x < v \\ 0.8 & \text{if} \quad w^T x = v \\ > 0.8 & \text{if} \quad w^T x > v \end{cases}$$

So choosing a threshold of $y_{threshold} = 0.8$ gives the output of the linear classifier $w^T x > v$.

Loss for Logistic Regression

We could use e.g. the squared loss between y and $\hat{y} = f(x)$, but there are more natural (and effective) choices.

Its based on our assumption that $\hat{y} = f(x)$ is ***the probability that x is in the target class***.

Under this assumption, we can compute the ***probability of correct classification***, and maximize that.

The probability of correct classification (for one input) is:

$$p_{correct} = \begin{cases} \hat{y} & \text{if} \quad y = 1 \quad \text{(true positive probability)} \\ 1 - \hat{y} & \text{if} \quad y = 0 \quad \text{(true negative probability)} \end{cases}$$

Which we can turn into a simple expression as:

$$p_{correct} = y\,\hat{y} + (1 - y)(1 - \hat{y})$$

Cross-Entropy Loss

We can use $-p_{correct}$ as the loss for each observation:
$$-p_{correct} = -y\,\hat{y} - (1 - y)(1 - \hat{y})$$

We could sum this over all observations and minimize it to maximize the algorithm's overall accuracy. This is sometimes done, and may give good results.

But much more commonly we use ***the negative log of the probability of a correct result:***
$$-\log p_{correct} = -y\log\hat{y} - (1 - y)\log(1 - \hat{y})$$

This is called ***Cross-Entropy Loss*** (n is the number of observations):

$$L = -\sum_{i=1}^{n} y_i \log\hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)$$

Cross entropy loss in this case is ***the negative log probability that every label is correct*** - since label errors are independent, we should multiply them to get the overall probability that everything is correct. Taking logs turns this product into a sum.

Cross-Entropy Loss

More generally ***Cross-Entropy Loss*** compares a target distribution $\boldsymbol{y_i}$ (now a vector over the possible values of Y, i is still the observation number) with a model distribution $\boldsymbol{\hat{y}_i}$.

$$L = -\sum_{i=1}^{n} \boldsymbol{y}_i^T \log \boldsymbol{\hat{y}_i}$$
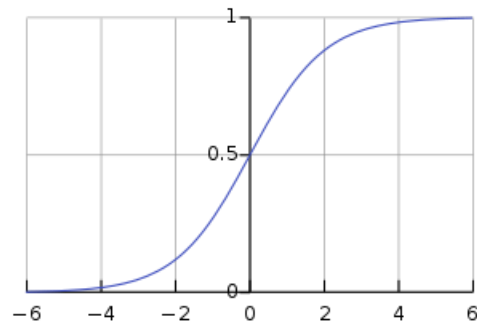
Its straightforward to show that the loss is minimized when $\boldsymbol{\hat{y}_i} = \boldsymbol{y_i}$, i.e. the predicted probability should match the observed probabilities of the labels on the data.
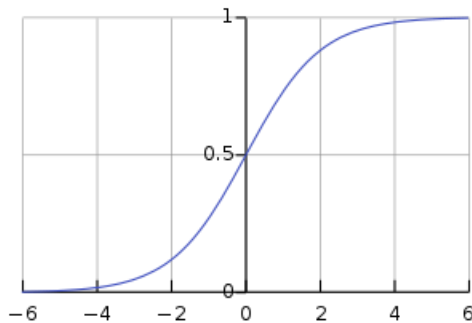
Logistic Regression

Is there something special about the logistic function

$$f(x) = \frac{1}{1 + \exp(-w^T x)}$$

or would any other "sigmoid" function work?

Logistic Regression

Is there something special about the logistic function

$$f(x) = \frac{1}{1 + \exp(-w^T x)}$$

or would any other "sigmoid" function work?

Exercise: show that a logistic regression classifier can exactly learn a naïve bayes classifier.

i.e. find weights w such that $f(x)$ is the naïve bayes probability that x is in the class.

Logistic Regression and Naïve Bayes

Exercise: show that a logistic regression classifier can exactly learn a naïve bayes classifier.

i.e. find weights w such that $f(x)$ is the naïve bayes probability that x is in the class.

Solution Sketch:

Start by adding a constant coordinate $x_0 = 1$ to x, and then set the weight w_0 to

$$\log \frac{P(C = 1)}{P(C = 0)} + \sum_{i=1}^{m} \log \frac{P(X_i = 0 | C = 1)}{P(X_i = 0 | C = 0)}$$

Then substitute in to the formula for $f(x)$ to see that

$$f(0) = \frac{P(X = 0, C = 1)}{P(X = 0, C = 1) + P(X = 0, C = 0)}$$

Which is the Bayes rule probability that the class is "1" given the data $x = 0$.

Logistic Regression and Naïve Bayes

Solution Sketch:

Finally choose the other weights w_i so that

$$f(x) = \frac{P(X = x, C = 1)}{P(X = x, C = 1) + P(X = x, C = 0)}$$

Which is the Bayesian probability $P(C = 1 | X = x)$ that the class is 1 given the data.

Summary

- Probabilistic framing of supervised learning problems.

- Generative vs. Discriminative models:

  - Deep nets are discriminative models.

  - May learn more slowly at first, but better asymptotic accuracy.

- Loss and risk: Machine learning is empirical risk minimization.

- Prediction functions $\hat{y} = f(x)$:

  - Linear regression: predict a real value.

  - Logistic regression: predict the probability of a binary target or label.