

Designing, Visualizing and Understanding Deep Neural Networks

Lecture 2: Machine Learning Background I

CS 194/294-129 Spring 2018
John Canny

Last Time

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

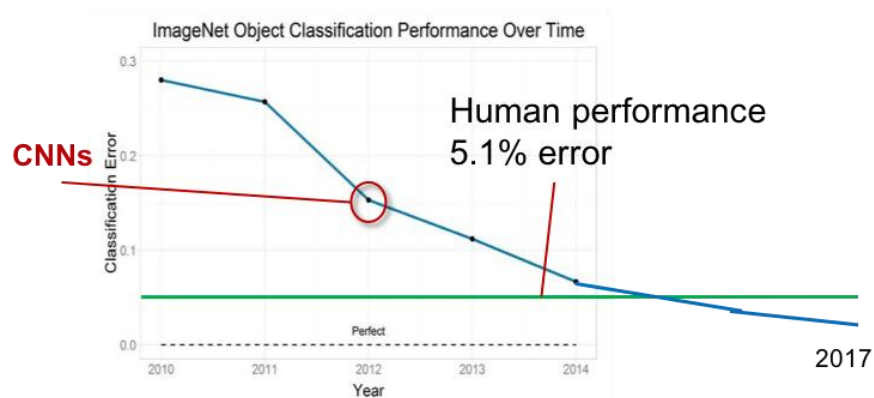
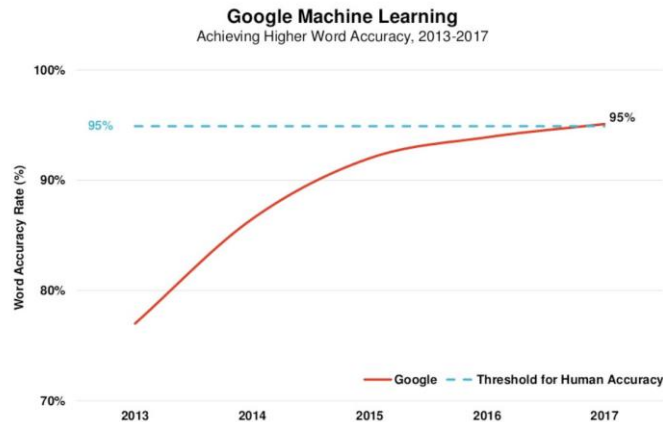


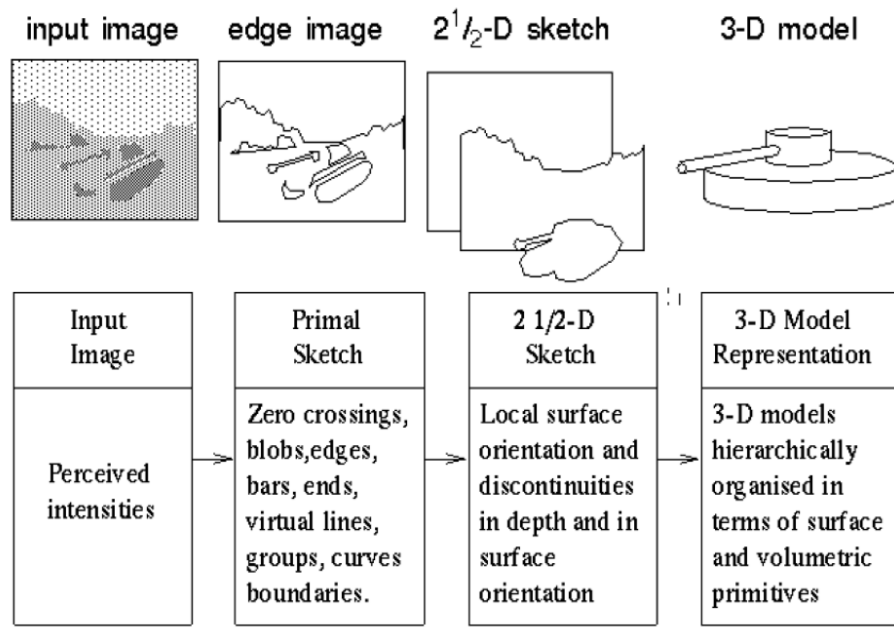
Image Classification



Speech Recognition

Last Time

Hierarchies of Representation in Human Vision:



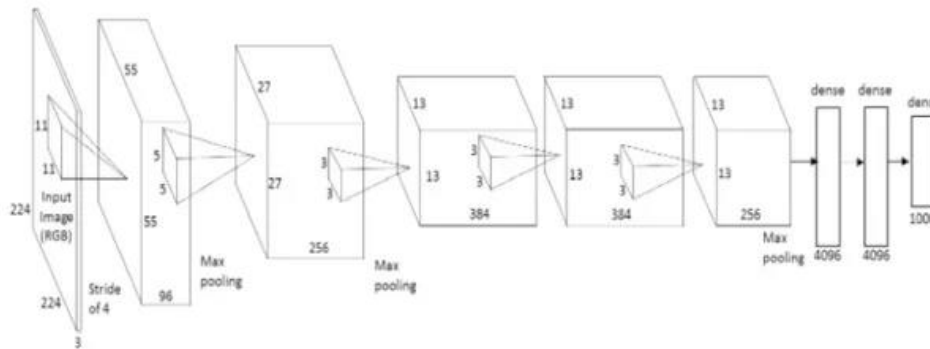
David Marr, 1979

Last Time

Learning Layered Representations: Training a deep Imagenet classifier:



1.4 million images



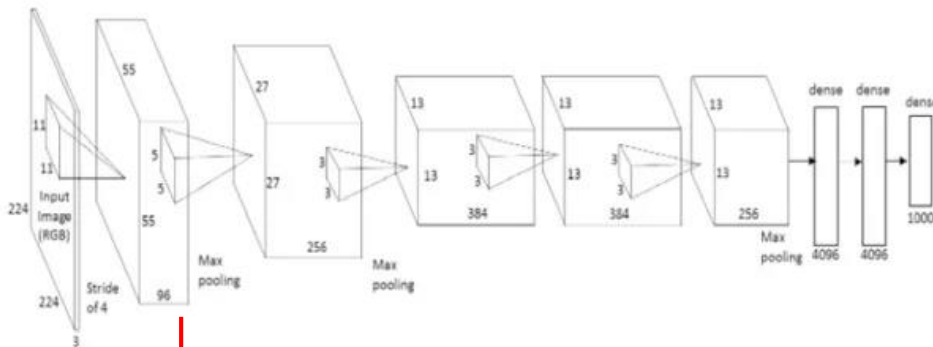
imagenet1000_clsidx_to_human.txt

```
1 {0: 'tench, Tinca tinca',
2 1: 'goldfish, Carassius auratus',
3 2: 'great white shark, white shark, man-
4 3: 'tiger shark, Galeocerdo cuvieri',
5 4: 'hammerhead, hammerhead shark',
6 5: 'electric ray, crampfish, numbfish, f
7 6: 'stingray',
8 7: 'cock',
9 8: 'hen',
10 9: 'ostrich, Struthio camelus',
11 10: 'brambling, Fringilla montifringilla',
12 11: 'goldfinch, Carduelis carduelis',
13 12: 'house finch, linnet, Carpodacus me
14 13: 'junco, snowbird',
15 14: 'indigo bunting, indigo finch, indi
16 15: 'robin, American robin, Turdus migr
17 16: 'bulbul',
18 17: 'jay',
19 18: 'magpie',
20 19: 'chickadee',
21 20: 'water ouzel, dipper',
22 21: 'kite',
23 22: 'bald eagle, American eagle, Haliae
24 23: 'vulture',
25 24: 'great grey owl, great gray owl, Str
```

1000 labels

Representation Learning

After a lot of training...



Neural receptive field
gray is zero
black is negative
white is positive



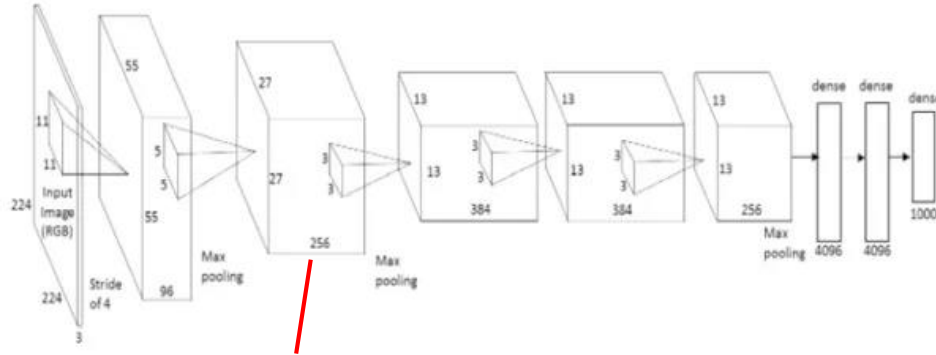
Layer 1



Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, 2013

Image patches with
strongest responses

Representation Learning



Neural receptive fields
gray is zero
black is negative
white is positive

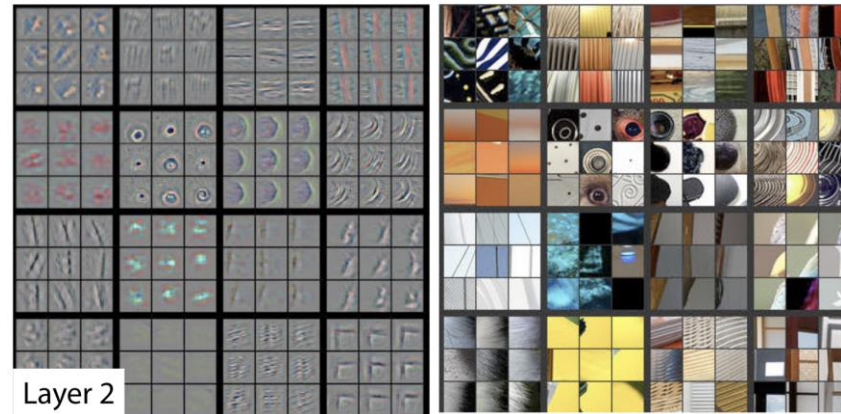
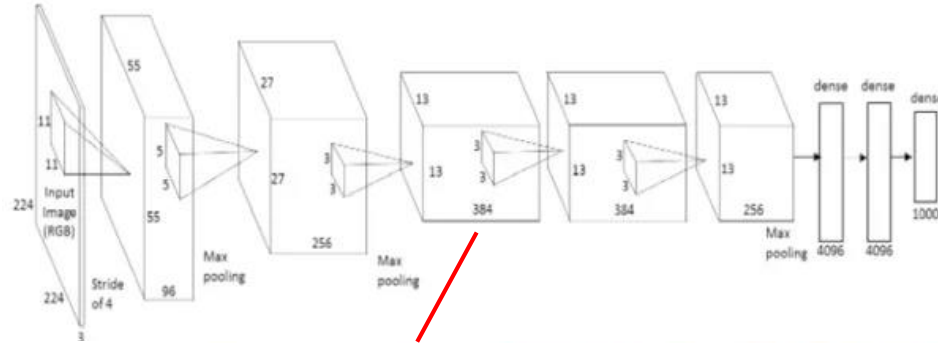


Image patches with
strongest responses

Representation Learning



Neural receptive fields

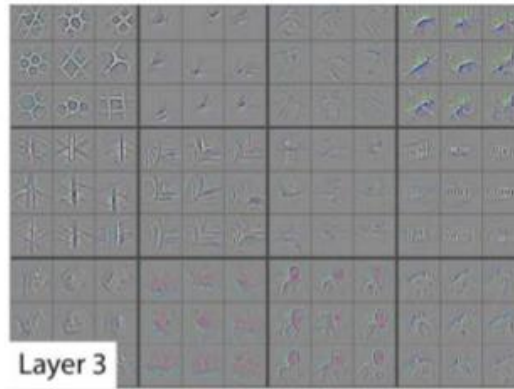


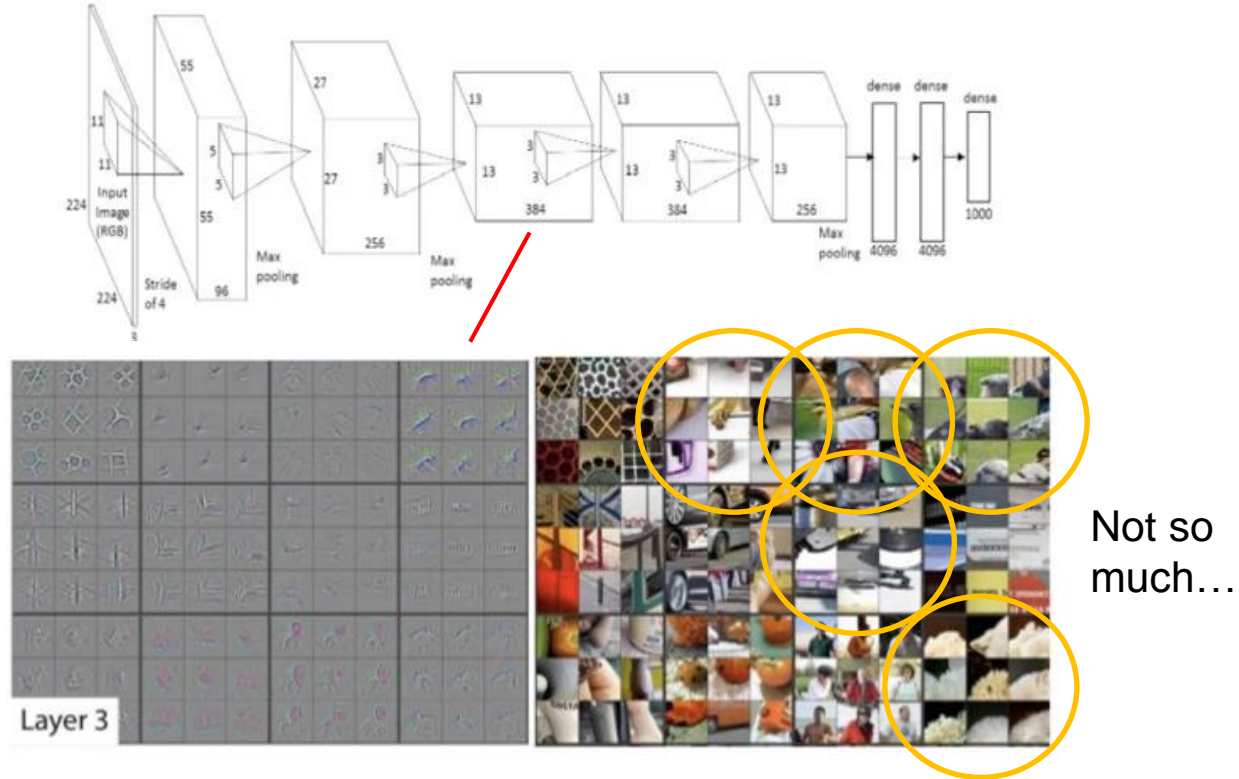
Image patches with strongest responses



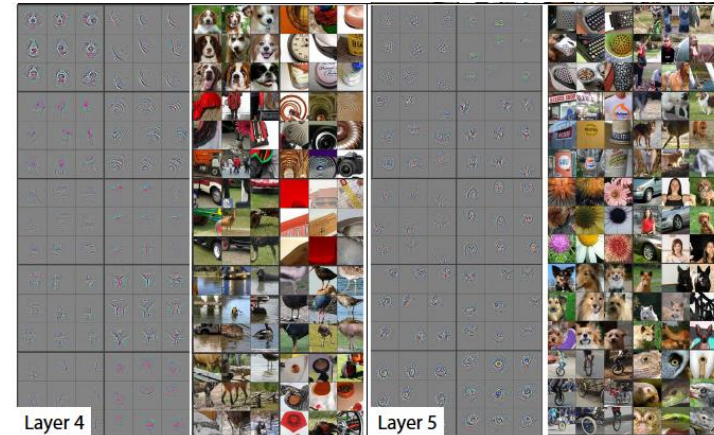
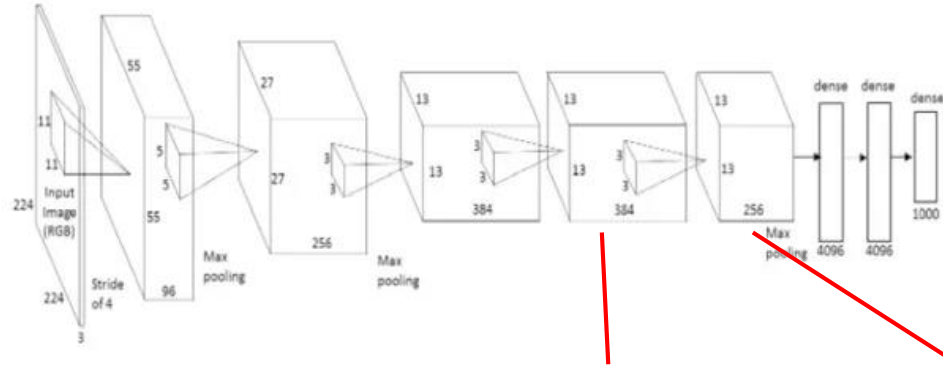
Representation Learning



Representation Learning



Representation Learning



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

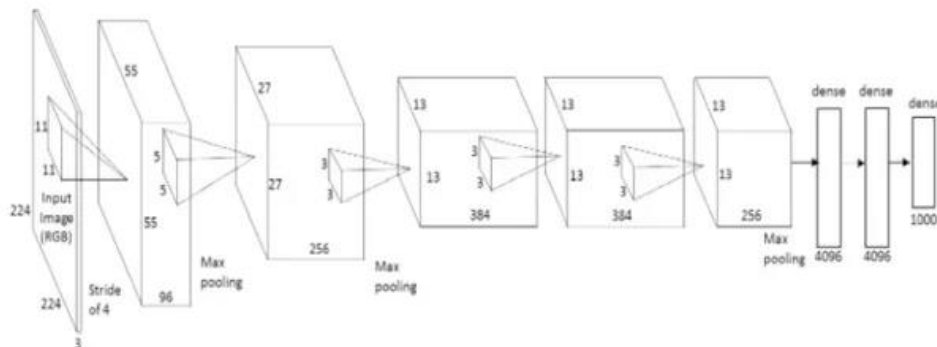
Representation Learning

These inner representations were learned **only** from input image/output labels.

Early and intermediate weights are typically task-independent.



1.4 million images



imagenet1000_clsidx_to_human.txt

```
1 {0: 'tench, Tinca tinca',
2 1: 'goldfish, Carassius auratus',
3 2: 'great white shark, white shark, man
4 3: 'tiger shark, Galeocerdo cuvieri',
5 4: 'hammerhead, hammerhead shark',
6 5: 'electric ray, crampfish, numbfish,
7 6: 'stingray',
8 7: 'cock',
9 8: 'hen',
10 9: 'ostrich, Struthio camelus',
11 10: 'brambling, Fringilla montifringilli
12 11: 'goldfinch, Carduelis carduelis',
13 12: 'house finch, linnet, Carpodacus me
14 13: 'junco, snowbird',
15 14: 'indigo bunting, indigo finch, indi
16 15: 'robin, American robin, Turdus migr
17 16: 'bulbul',
18 17: 'jay',
19 18: 'magpie',
20 19: 'chickadee',
21 20: 'water ouzel, dipper',
22 21: 'kite',
23 22: 'bald eagle, American eagle, Haliae
24 23: 'vulture',
25 24: 'great grey owl, great gray owl, Str
```

1000 labels

Consequences

You can reuse or customize models trained for different task.

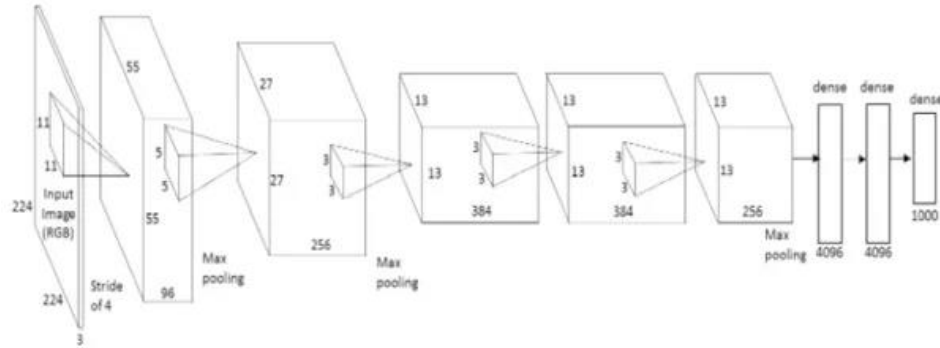
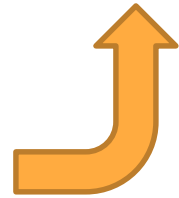


Image Classification



Reuse/Fine Tuning

You can reuse or customize models trained for different task, with much less data.

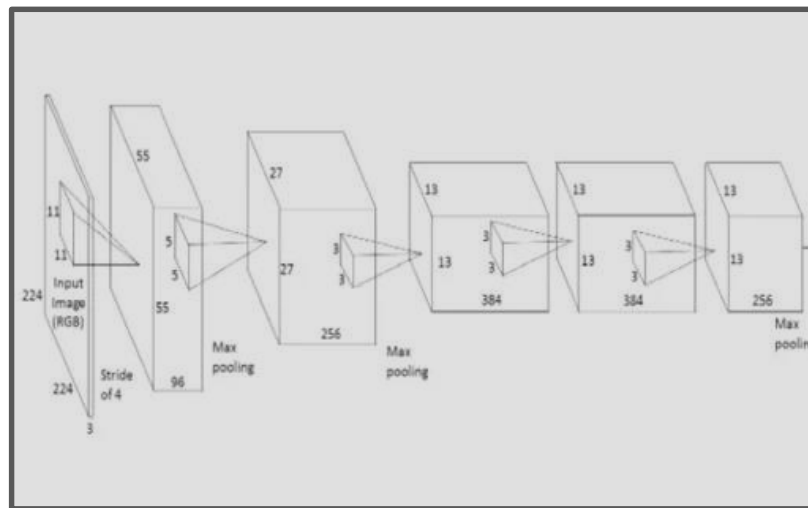
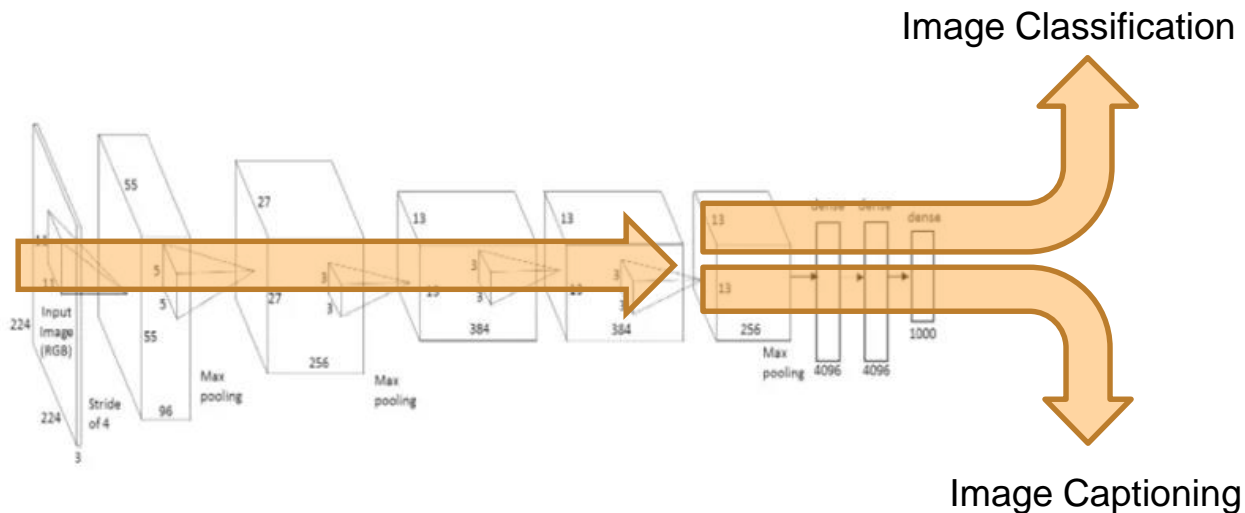


Image Captioning

Multi-task Learning

Sharing model weight across tasks often improves performance on both.



Summary of Deep Nets

- Layered architecture (the deep part) of simple units.
- Inner layer representations are learned only from end-to-end tasks.
- Depth and complexity seem to be only limited by the amount of data needed to train without overfitting. More complex models → better accuracy.
- Inner layer representations are typically task-independent → easy to re-use models in different applications that don't have large training datasets.
- Building blocks (layer types) are standard across different toolkits and across domains (image processing, speech, text, robotics)... “tricks” transfer.
- Real-world problems addressed by deep networks are usually “easy” to learn.

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” while targets are “expensive” and non-observable. Computer predictions therefore have economic value.

Prediction Problems

Observation Space \mathcal{X} :

Images

Images

Face Images

Natural Images

Signals of Human Speech

Sentence in English

Demographic info: age, income

Diet, lifestyle

Target Space \mathcal{Y} :

Image class: “cat”, “dog” etc.

Caption: “kids playing soccer”

User’s identity

Stylized Images (e.g. cartoons)

Text transcript of the speech

Translation into Spanish

Other info: education, employment

Risk of Heart Disease

Probabilistic Framing (not required)

We assume that x and y are **samples of random variables** X and Y .

These random variables have a **joint distribution** $P(X, Y)$.

For prediction, we want at least the conditional distribution $P(Y|X)$, so we can determine the distribution of targets y given an observation x .

An approximation $\hat{P}(Y|X)$ to the true distribution $P(Y|X)$ is called a **model**. The goal of machine learning is to construct models that are:

1. Computable (and perhaps simple, efficient)
2. Provide label predictions that are close to those from the true distribution $P(Y|X)$

Generative vs. Discriminative Models (not req'd)

Generative:

Discriminative:

Generative vs. Discriminative Models (not req'd)

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 (not req'd)

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: $P(Y|X)$.

Generative vs. Discriminative Models (not req'd)

Generative:

Noisy linear functions

Naïve Bayes

Hidden Markov Models

Gaussian mixture models

Latent Dirichlet Allocation

Discriminative:

Linear Least Squares

Logistic Regression

Conditional Random Fields

Support Vector Machines (SVM)

Decision trees + Random Forests

Neural Networks

Generative vs. Discriminative Tradeoffs (not req'd)

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

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

Generative vs. Discriminative Tradeoffs (not req'd)

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 (not req'd)

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 (not req'd)

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 (not req'd)

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.

Prediction Functions (required)

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 $y \in \{1, \dots, k\}$ 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.

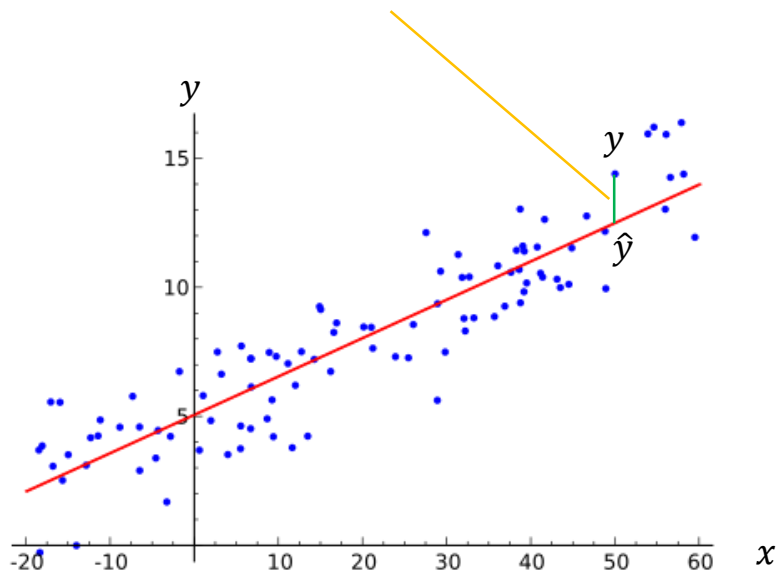
A **loss function** measures the difference between a target prediction and target data value.

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 .

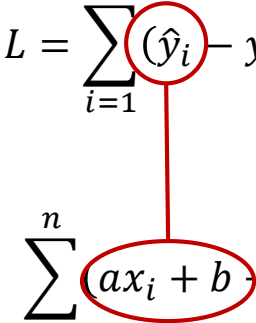
The loss is the squared loss $L_2(\hat{y}, y) = (\hat{y} - y)^2$



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

Linear Regression

The total loss across all points is

$$\begin{aligned} L &= \sum_{i=1}^n (\hat{y}_i - y_i)^2 \\ &= \sum_{i=1}^n (ax_i + b - y_i)^2 \end{aligned}$$


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

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

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

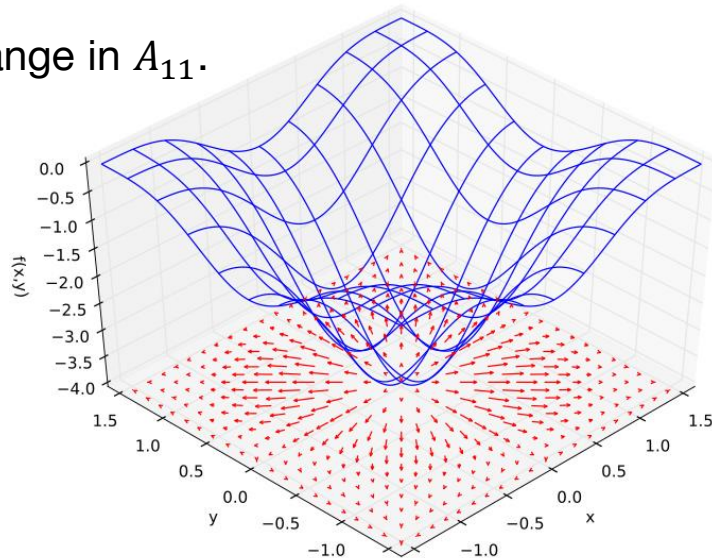
When we write $\nabla_A L(A)$, we mean the vector of partial derivatives:

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

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).



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):

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

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 $2AM_{xx} - 2M_{yx} = 0$

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** $A_0 = \mu_{yx}\mu_{xx}^{-1}$

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.



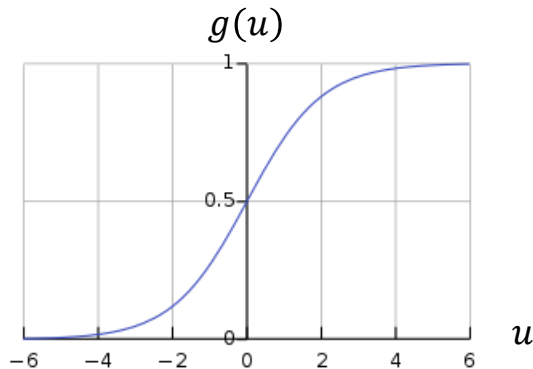
Logistic Regression

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

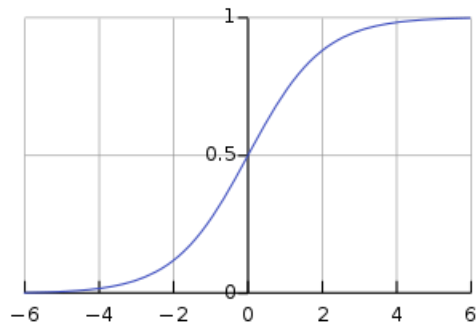
Let $w \in \mathbb{R}^m$ be a weight vector (the logistic model). Then

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

Notice that the output is:

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

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



Logistic Regression

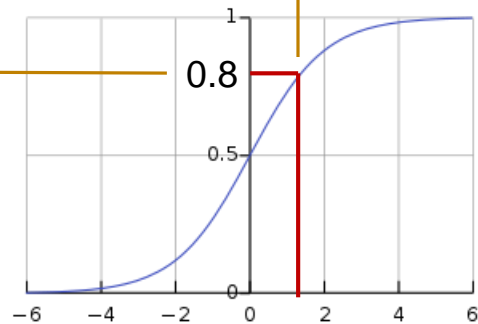
Let $w \in \mathbb{R}^m$ be a weight vector (the logistic model). Then

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

Notice that the output is:

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

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:

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

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**:

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

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 y_i (which may not be binary) with a model distribution \hat{y} .

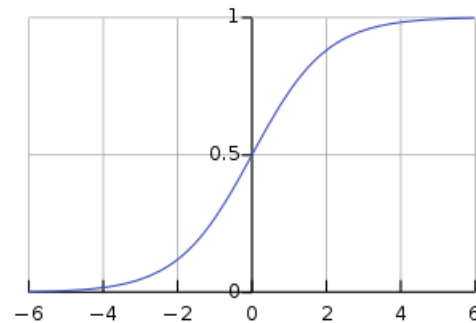
$$L = - \sum_{i=1}^n y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

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?

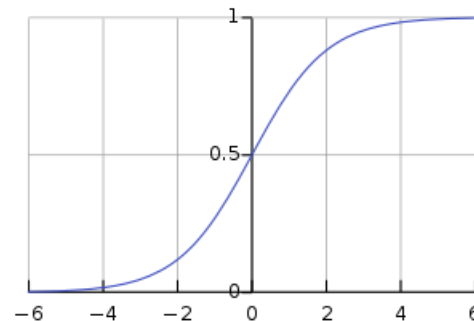


Logistic Regression

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 (not req'd)

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

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

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 (not req'd)

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

- Deep networks learn good internal representations with only end-to-end supervision.
- 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.