CSCI 699: Trustworthy ML (from an optimization lens)

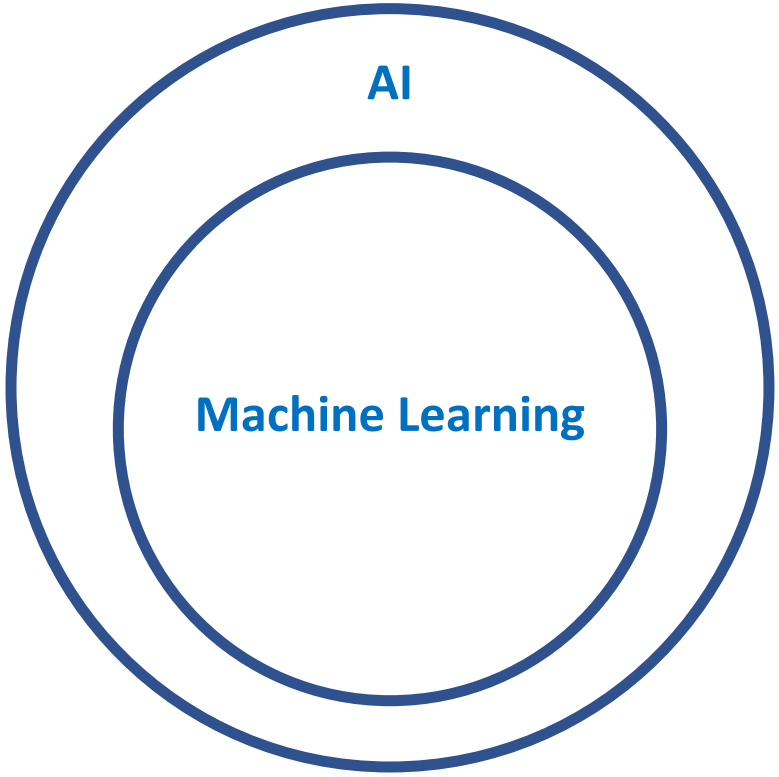
Vatsal Sharan

Fall 2025

Lecture 1, Aug 27

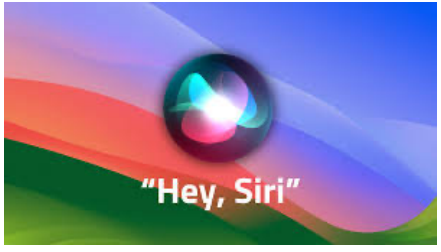
USC University of Southern California


MACHINE LEARNING
Machine Learning
LIGHTYEAR
MACHINE LEARNING EVERYWHERE
imgflip.com



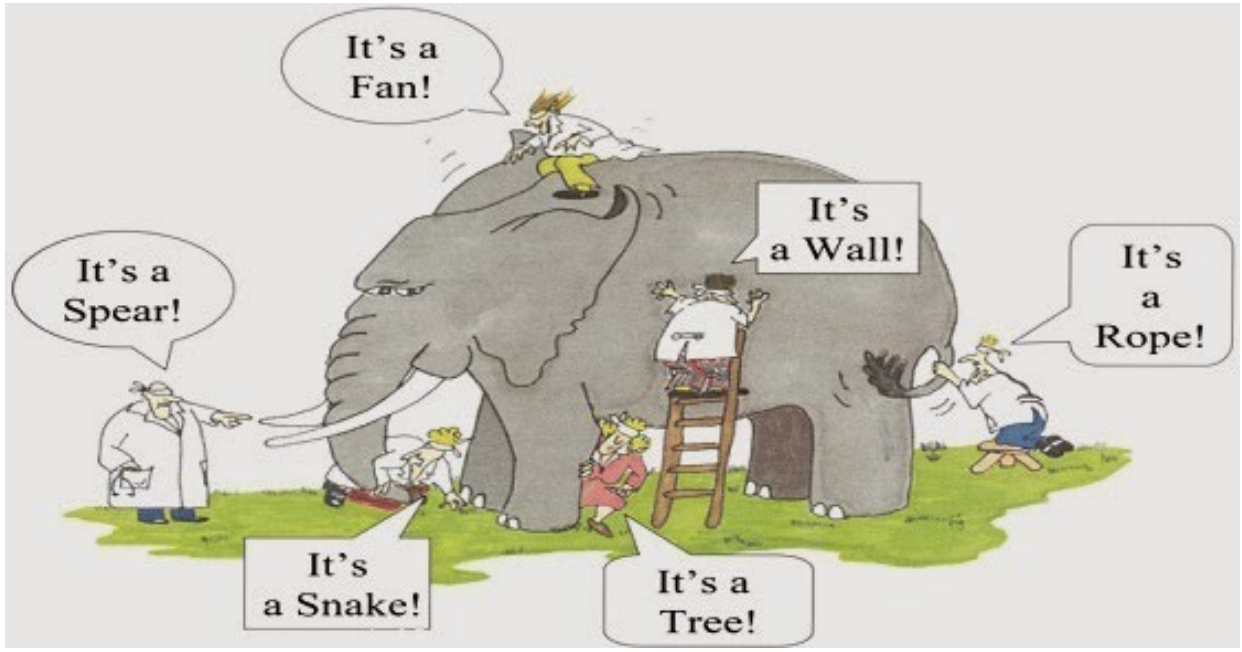
ML has been driving the recent advances in AI

ML has been having a pretty good run..



What do you think are other important advances?

However, machine learning can be *brittle*



The Blind Men and the Elephant

It was six men of Indostan
To learning much inclined,
Who went to see the Elephant
(Though all of them were blind),
That each by observation
Might satisfy his mind.

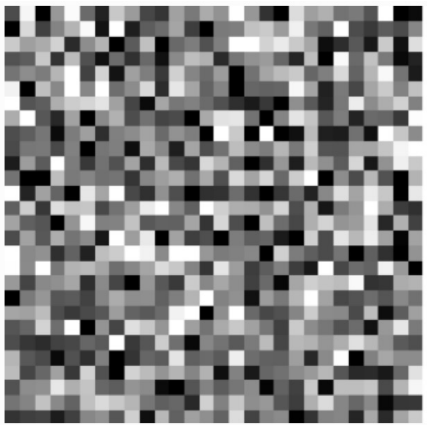
The First approached the Elephant,
And happening to fall
Against his broad and sturdy side,
At once began to bawl:
"God bless me! but the Elephant
Is very like a WALL!"

….

Models can be very sensitive to small variations in the input

Pig
(90% confidence)

+



Small amount of
adversarial noise

=

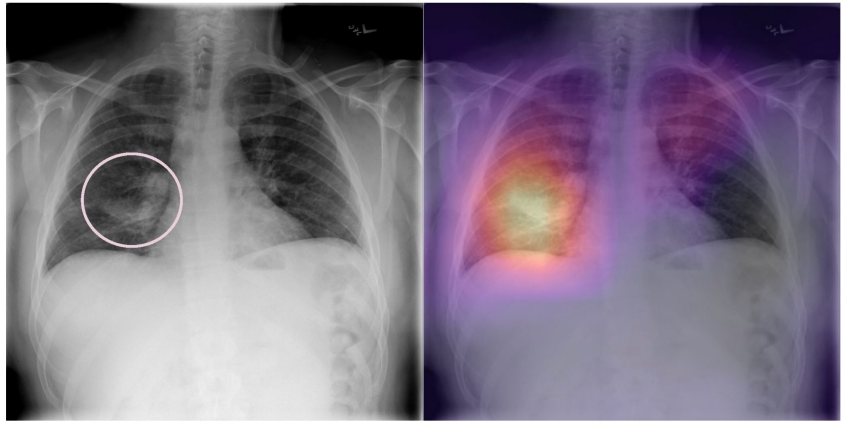
Airplane!
(99.9% confidence)

Image adapted from https://adversarial-ml-tutorial.org/

ML models can latch onto spurious features to make predictions

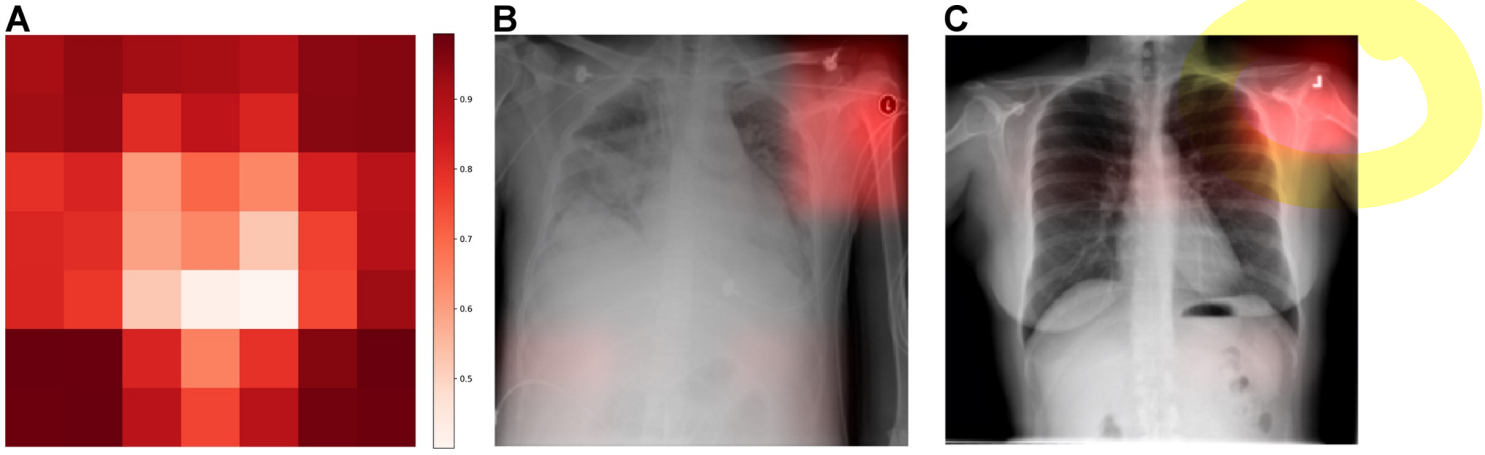
CNN models have obtained impressive results for diagnosing X-rays

E.g. *ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases*, Wang et a;. 2017



Source: *Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists,* Rajpurkar et al. 2018

But the models may not generalize as well to data from new hospitals because they can learn to pickup on spurious correlations such as the type of scanner and marks used by technicians in specific hospitals!



CNN to predict hospital system detects both general and specific image features.
(A) We obtained activation heatmaps from our trained model and averaged over a sample of images to reveal which subregions tended to contribute to a hospital system classification decision. Many different subregions strongly predicted the correct hospital system, with especially strong contributions from image corners. (B-C) On individual images, which have been normalized to highlight only the most influential regions and not all those that contributed to a positive classification, we note that the CNN has learned to detect a metal token that radiology technicians place on the patient in the corner of the image field of view at the time they capture the image. When these strong features are correlated with disease prevalence, models can leverage them to indirectly predict disease.

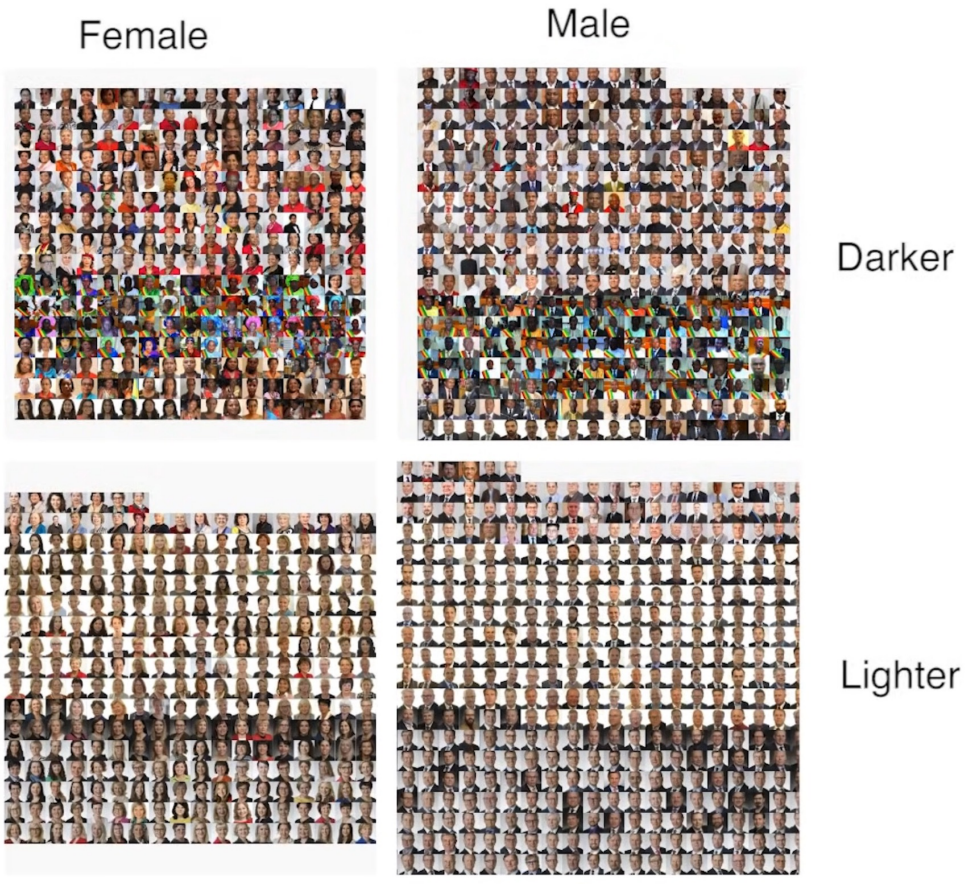
Source: *Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study,* Zech et al. 2018

Unequal accuracy: The GenderShades project

Models can do well on average but not on sub-populations

http://gendershades.org/

How well do facial recognition tools perform on various demographics?

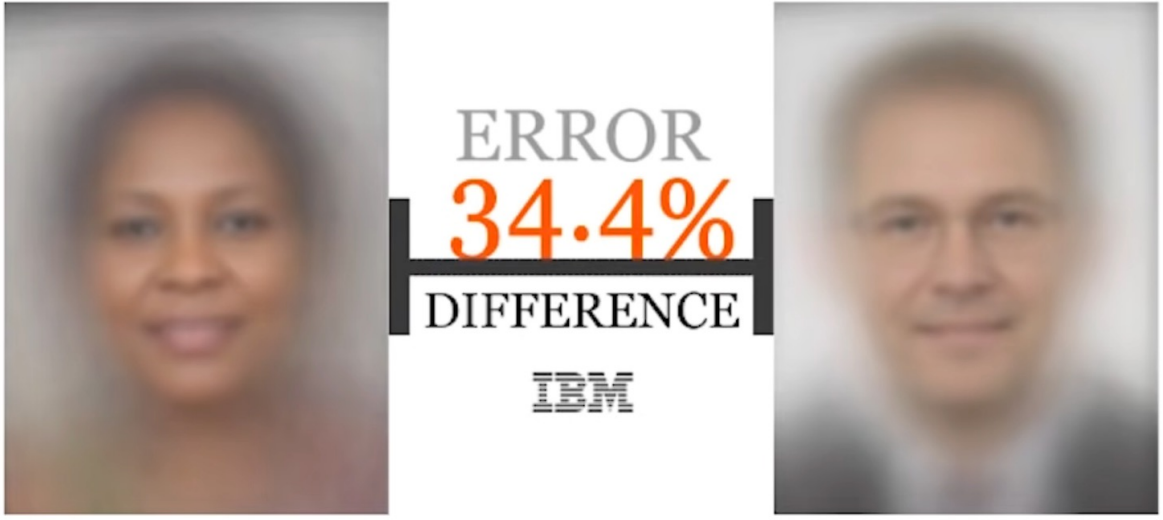


http://gendershades.org/

Ans: *Not very well*

	TYPE I	TYPE II	TYPE III	TYPE IV	TYPE V	TYPE VI
Microsoft	1.7%	1.1%	3.3%	0%	23.2%	25.0%
IBM	5.1%	7.4%	8.2%	8.3%	33.3%	**46.8%**
FACE++	11.9%	9.7%	8.2%	13.9%	32.4%	**46.5%**

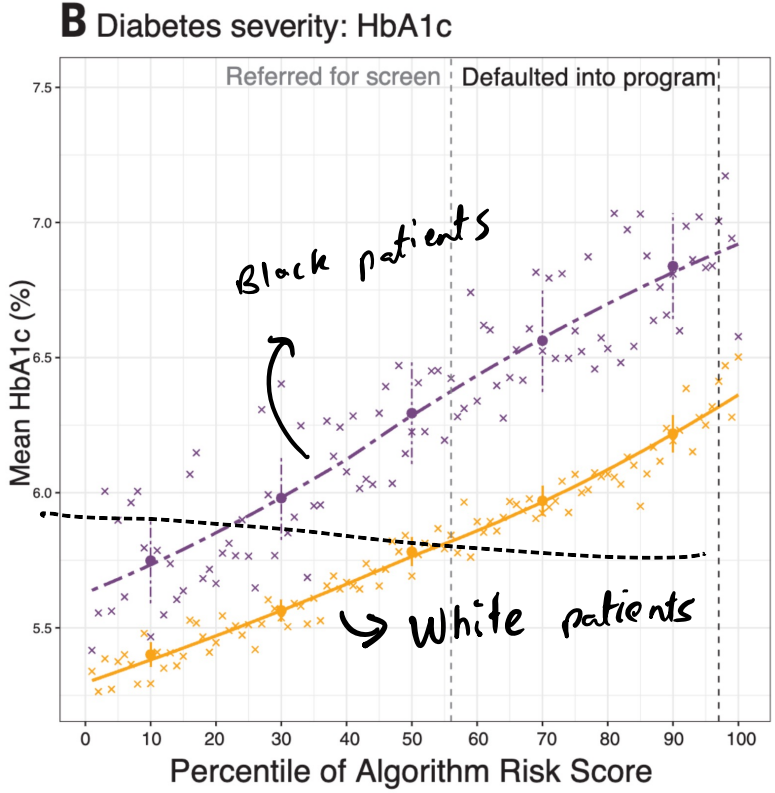
Ans: *Not very well*



Bias in predictions: Predicting disease severity

Quoting from the paper:

- Health systems rely on commercial prediction algorithms to identify and help patients with complex health needs.
- A widely used algorithm affecting millions of patients, exhibits significant racial bias: At a given risk score, Black patients are considerably sicker than White patients, as evidenced by signs of uncontrolled illnesses.
- Remedying this disparity would increase the percentage of Black patients receiving additional help from 17.7 to 46.5%.
- Bias arises because the algorithm predicts health care costs rather than illness, but unequal access to care means that we spend less money caring for Black patients than for White patients.



Dissecting racial bias in an algorithm used to manage the health of populations, Obermeyer et al., Science 2019

Privacy & Denonymization

The Netflix prize:

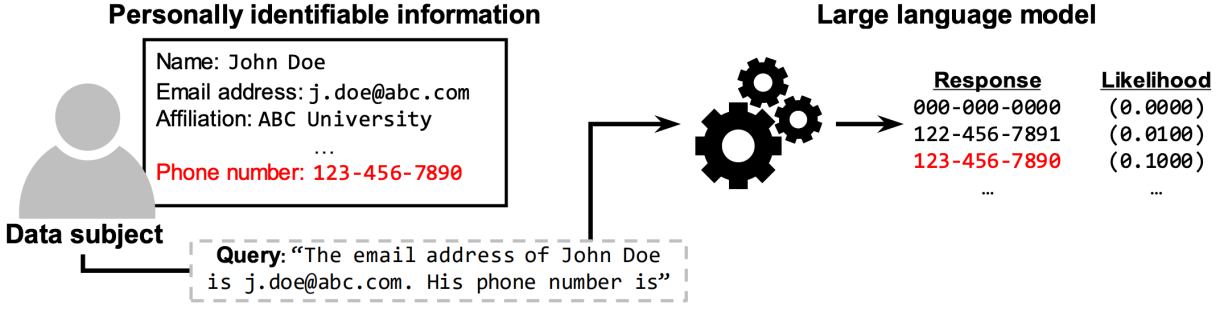
- Launched in 2006, $1M cash prize
- Dataset: 100 million movie ratings from nearly 500 thousand Netflix subscribers on a set of 17770 movies. Each data point corresponds to (anonymized user id, movie, date of rating, rating).
- Researchers were able to de-anonymize some of the subscribers by linking their rating with ratings on IMDB!
- Some Netflix subscribers had also publicly rated an overlapping set of movies on IMDB under their real identities.
- Lawsuit against Netflix, subsequent competition was cancelled.

From the book *Fairness And ML: Limitations and Opportunities*

Privacy & Denonymization

In some cases, it is possible to recover some of the original training data of the model using only API access to the model. The following (left) is an example of an image recovered by an attacker who only knows the name of the person, and the original training image (right) from [1]

Some evidence that LLMs could also leak private information:



[1] Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures, Fredrikson et al., 2015
[2] ProPILE: Probing Privacy Leakage in Large Language Models, Kim et al., 2023,

This class: Aspects of trustworthy ML

Rough plan:

Weeks 1-3: Robustness (Vatsal)
Weeks 4-6: Fairness, calibration (Vatsal)
Weeks 7-12: Privacy (Meisam)
Week 13: Project presentation (Meisam & Vatsal)
Week 14: AI Safety, Summary (Meisam & Vatsal)

Logistics

- Course website: https://vatsalsharan.github.io/fall2025/
- Communication: Slack
- Class structure (most weeks):
  - First part: Lecture
  - Second part: Student presentations (~30 min/presentation)

Overview of structure, see course website for details:

Homeworks (15%): 2 homeworks

Mini-homeworks (15%): Reading class papers before class

Class presentations (15%): Present a paper in class

Project (45%): You can choose your topic, groups of 2

Other components (10%): Scribing, class participation


Supervised Machine
Learning: Basics

Supervised ML: Predict future outcomes using past outcomes

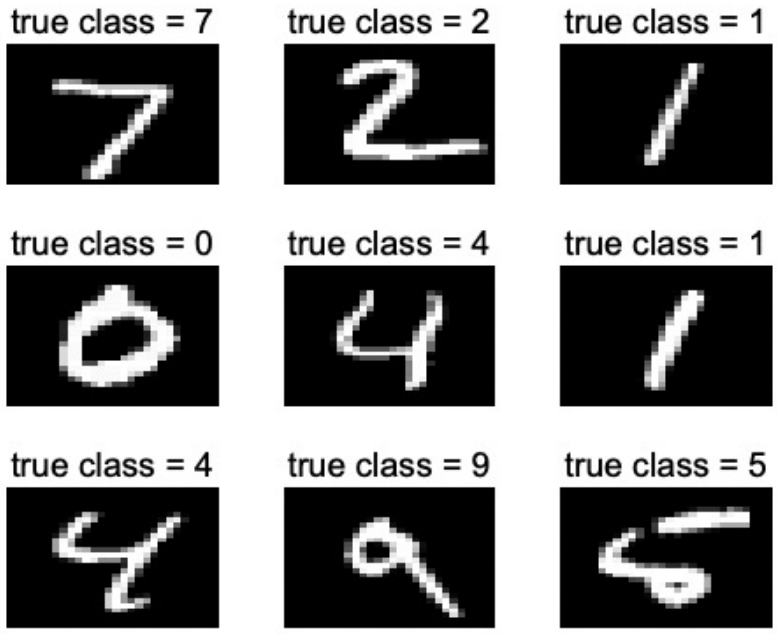
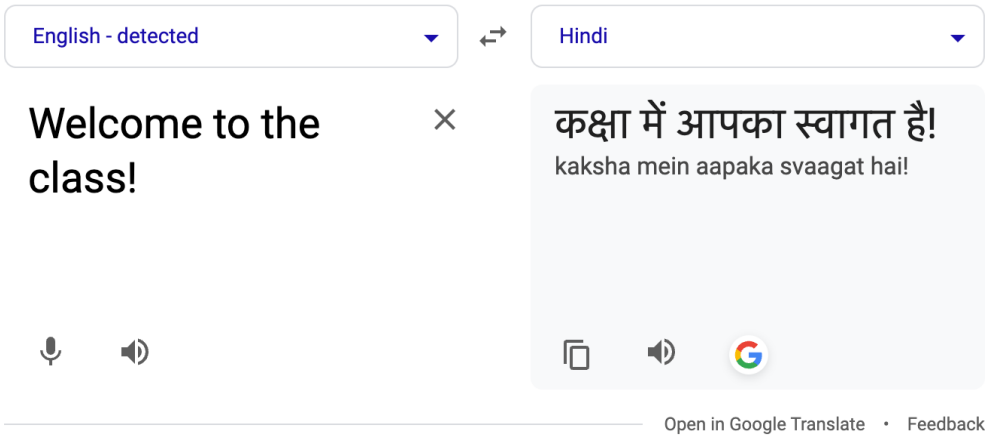
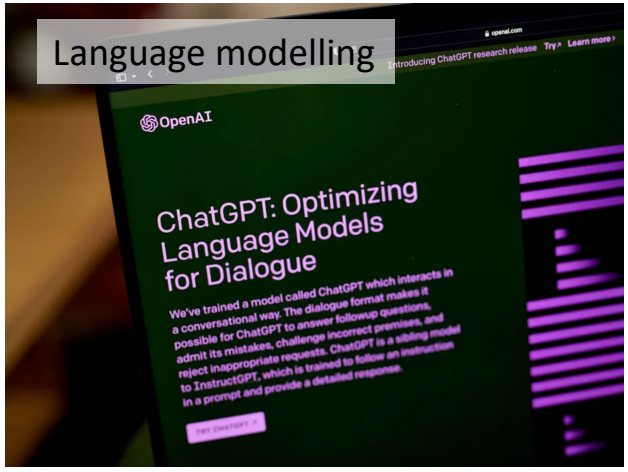


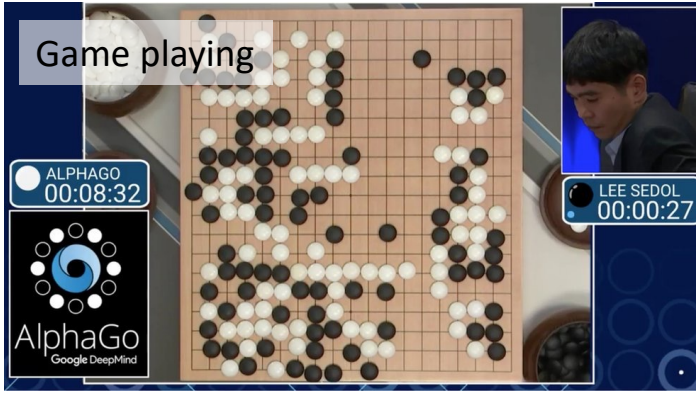
Image classification

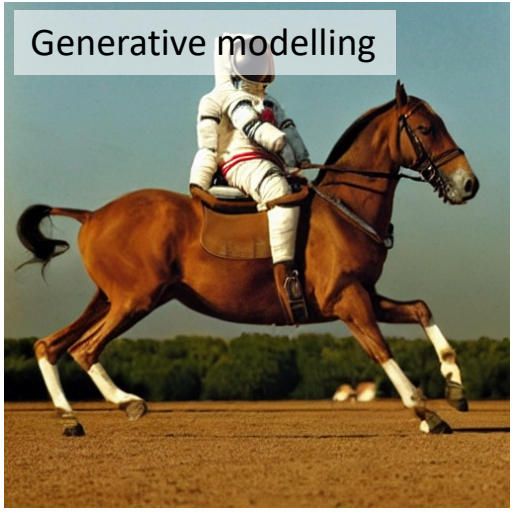


Machine translation

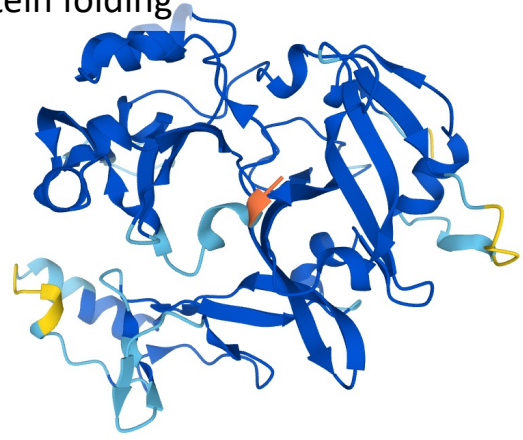
Supervised ML is at the heart of many AI advances

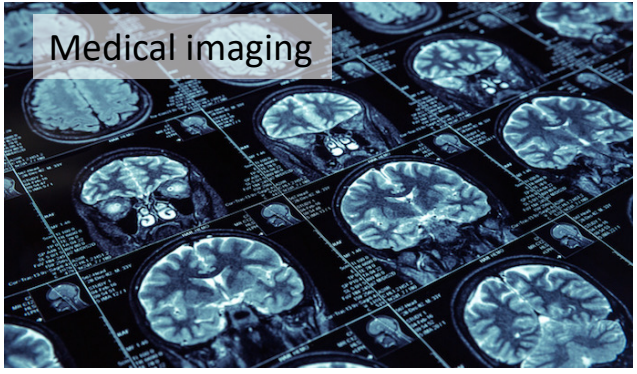






Protein folding

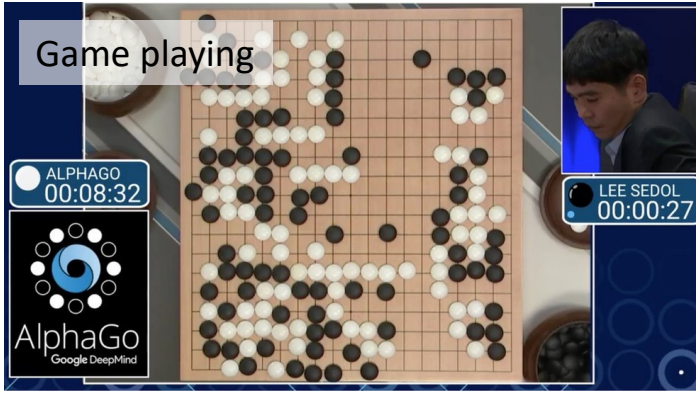


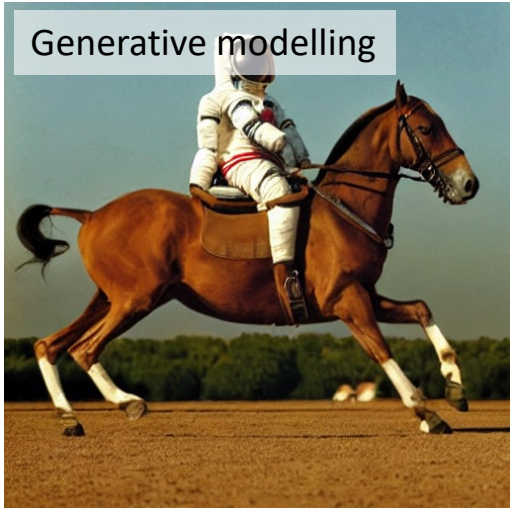


Supervised ML is at the heart of many AI advances

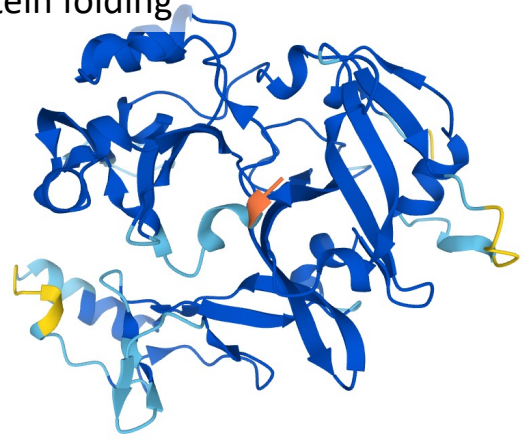
Language modelling

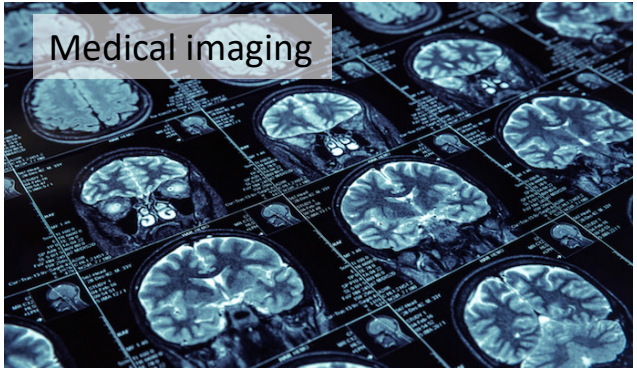
Given previous words ->
Predict next word





Protein folding





Supervised ML is at the heart of many AI advances

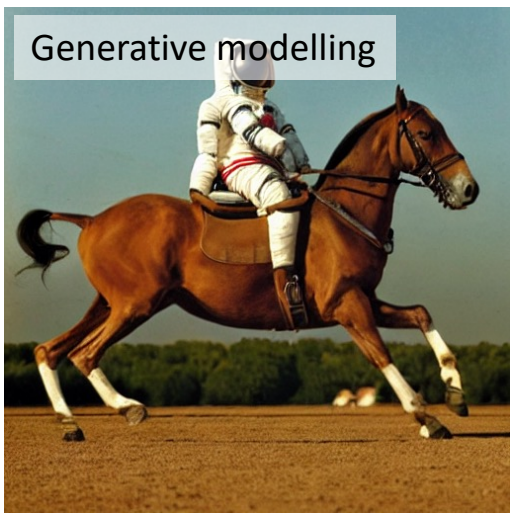
Language modelling

Given previous words ->
Predict next word

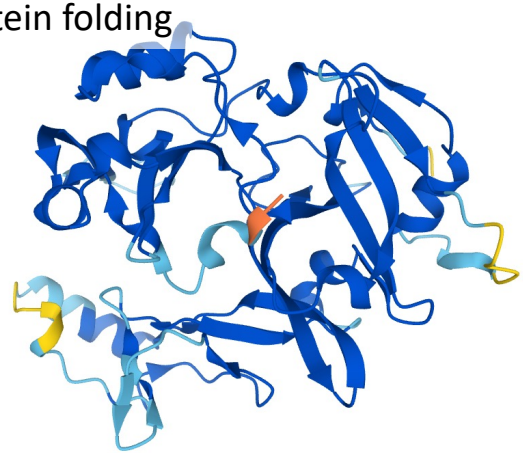
Game playing

Given current board state ->
Predict probability of winning

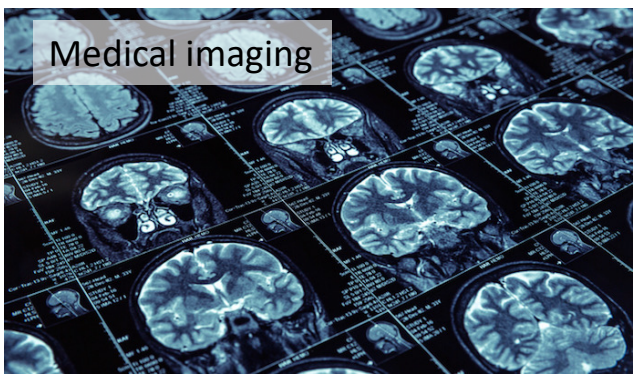
Generative modelling



Protein folding



Medical imaging



Supervised ML is at the heart of many AI advances

Language modelling

Given previous words ->
Predict next word

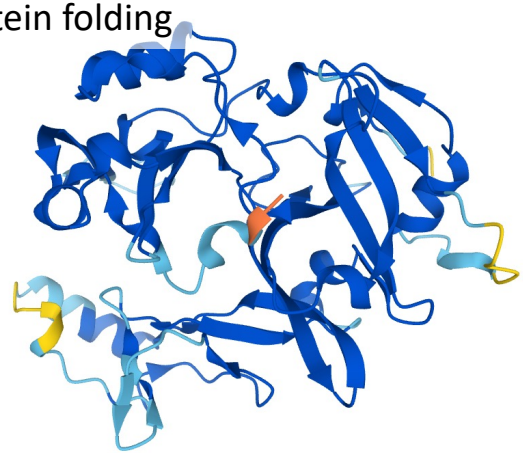
Game playing

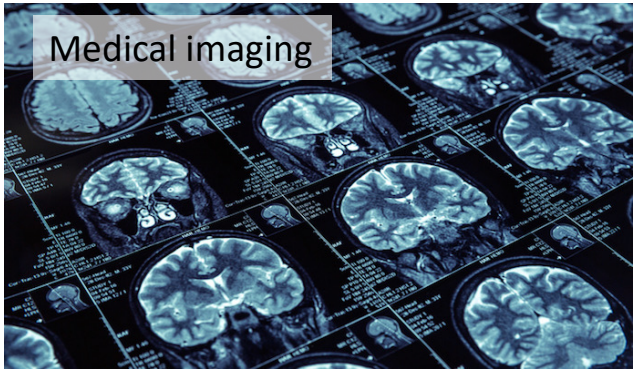
Given current board state ->
Predict probability of winning

Generative modelling

Given noisy image ->
Predict denoised image

Protein folding





Supervised ML is at the heart of many AI advances

Language modelling

Given previous words ->
Predict next word

Game playing

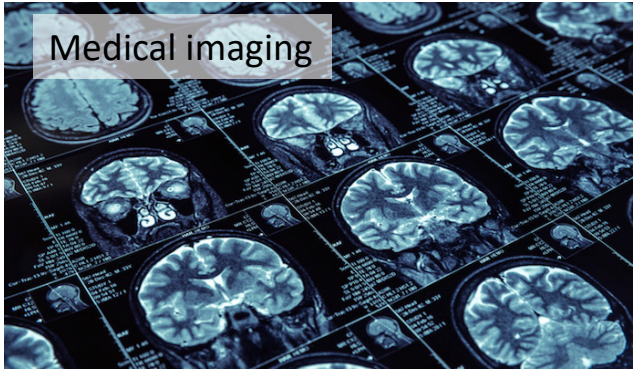
Given current board state ->
Predict probability of winning

Generative modelling

Given noisy image ->
Predict denoised image

Protein folding

Given protein chain ->
Predict 3D structure



Supervised ML is at the heart of many AI advances

Language modelling

Given previous words ->
Predict next word

Game playing

Given current board state ->
Predict probability of winning

Generative modelling

Given noisy image ->
Predict denoised image

Protein folding

Given protein chain ->
Predict 3D structure

Medical imaging

Given image ->
Predict if there is tumor etc.

General framework for supervised learning

- An input space $\mathcal{X} \subseteq \mathbb{R}^d$:
  - Representing inputs as datapoints in d dimensions.
  - E.g. for an image with 30×30 pixels, $d = 30 \times 30 = 900$.
- An output space $\mathcal{Y}$:
  - For predicting sale price of a house, $\mathcal{Y} = \mathbb{R}$ (regression)
  - For binary classification (cat vs. dog), $\mathcal{Y} = \{\pm 1\}$ (classification)
- Goal: Learn a predictor $f(x) : \mathcal{X} \to \mathcal{Y}$.

- *Loss function:* $\ell(f(x), y)$. Depends on the task.
  - e.g. squared loss $\ell(f(x), y) = (f(x) - y)^2$.
- *Loss minimization:* Given some loss function and labelled datapoints, a natural goal is to find some predictor $f : \mathcal{X} \to \mathcal{Y}$ such that it gets small average error on these datapoints.
  - More formally, consider a set $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ of n labelled datapoints. The *training loss or empirical risk* $\hat{R}_S(f)$ of any predictor $f : \mathcal{X} \to \mathcal{Y}$ is given by:

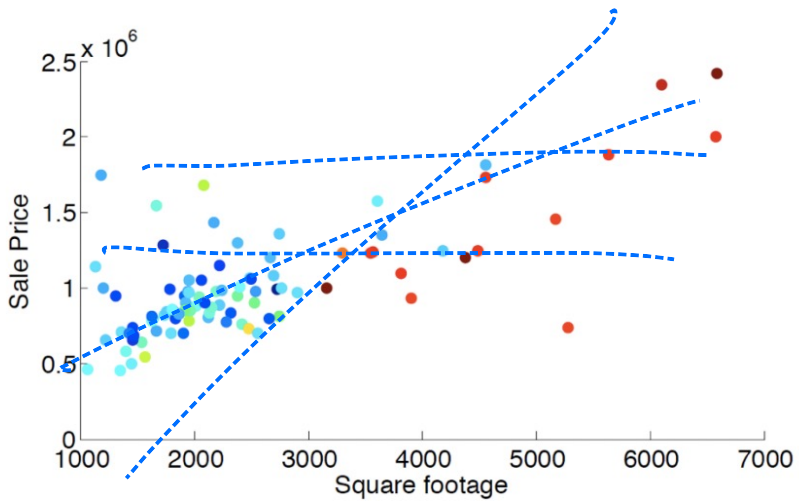
$$\hat{R}_S(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$$

    We should find some predictor f which has small empirical risk.

Function class

- *Function class:* A function class is defined as a collection of functions $f : \mathcal{X} \to \mathcal{Y}$.
  - e.g. $\mathcal{X} = \mathbb{R}, \mathcal{Y} = \mathbb{R}, \mathcal{F} = \{f : y = wx + c, w \in \mathbb{R}, c \in \mathbb{R}\}$.

- Each of these is a linear function.
- The class of all linear functions is a function class.



Empirical risk minimizer (ERM)

- *Definition:* Given a function class $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y}\}$ and set of labelled datapoints S, empirical risk minimization (ERM) corresponds to finding:

$$\min_{f \in \mathcal{F}} \hat{R}_S(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) \rightarrow \text{Optimization}$$

- In words, we want to find the predictor $f \in \mathcal{F}$ which achieves lowest loss on the training datapoints among all predictors in $\mathcal{F}$.

Generalization

- ML wouldn't be so useful if it only did well on datapoints which have been seen at training time. We want our predictor to *generalize to unseen datapoints*.
- To measure performance on new datapoints we measure *test loss*. The test loss of a predictor f is measured as the average loss on a "new" set of m points:

$$\frac{1}{m} \sum_{i=1}^{m} \ell(f(x_i), y_i)$$

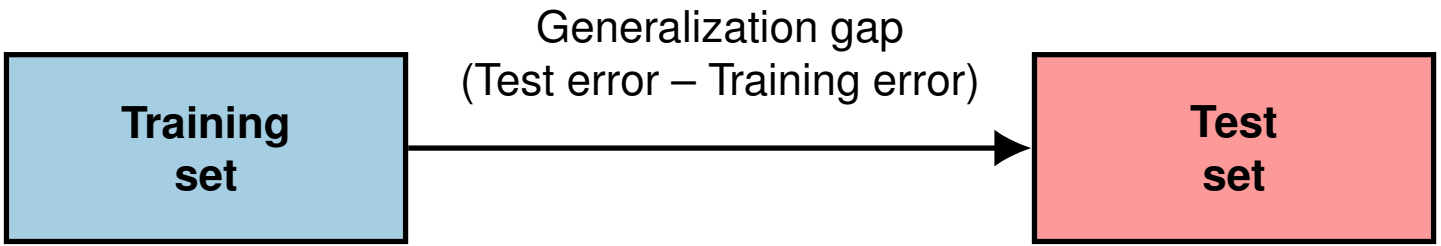
- **Training/test paradigm:** Assume training set and test set are drawn from the same distribution.

Measuring generalization: Training/Test paradigm

Data Splitting. We randomly divide data into two disjoint subsets:

- **Training set:** subset of data used to train the model.
- **Test set:** subset of data used to evaluate the model.

Generalization gap: Test error – Training error.



We usually add a third split as well.

- **Validation set:** subset of data used to measure generalization, fit hyperparameters

Generalization: More formally

Risk of a predictor. The population risk of f under distribution $\mathcal{D}$ is

$$R(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ \ell(f(x), y)\ ]$$

$$= \sum_{x',y'} \Pr_{\mathcal{D}}(x = x', y = y')\ \ell(f(x'), y').$$

How to empirically evaluate this? The average loss on a test set $S' = \{(x'_i, y'_i)\}_{i=1}^m$, where $(x_i, y_i) \sim \mathcal{D}$:

$$R(f) \approx \frac{1}{m} \sum_{i=1}^{m} \ell\big(f(x'_i), y'_i\big).$$

A tautology.

$$R(f) = \widehat{R}_S(f) \; + \; \Big(R(f) - \widehat{R}_S(f) \Big).$$

To minimize $R(f)$:

- First try to minimize the empirical risk $\widehat{R}_S(f)$.
- What remains is the term

  $$R(f) - \widehat{R}_S(f),$$

  which is the *generalization gap*.

Supervised learning in one slide

Loss function: What is the right loss function for the task?

Representation: What class of functions should we use?

Also known as the "inductive bias".

No-free lunch theorem from learning theory tells us that

no model can do well on every task

"All models are wrong, but some are useful", George Box

Supervised learning in one slide

Loss function: What is the right loss function for the task?

Representation: What class of functions should we use?

Optimization: How can we efficiently solve the empirical risk minimization problem?

Depends on all the above and also…

Supervised learning in one slide

Loss function: What is the right loss function for the task?

Representation: What class of functions should we use?

Optimization: How can we efficiently solve the empirical risk minimization problem?

Generalization: Will the predictions of our model transfer gracefully to unseen examples?

Supervised learning in one slide

Loss function: What is the right loss function for the task?

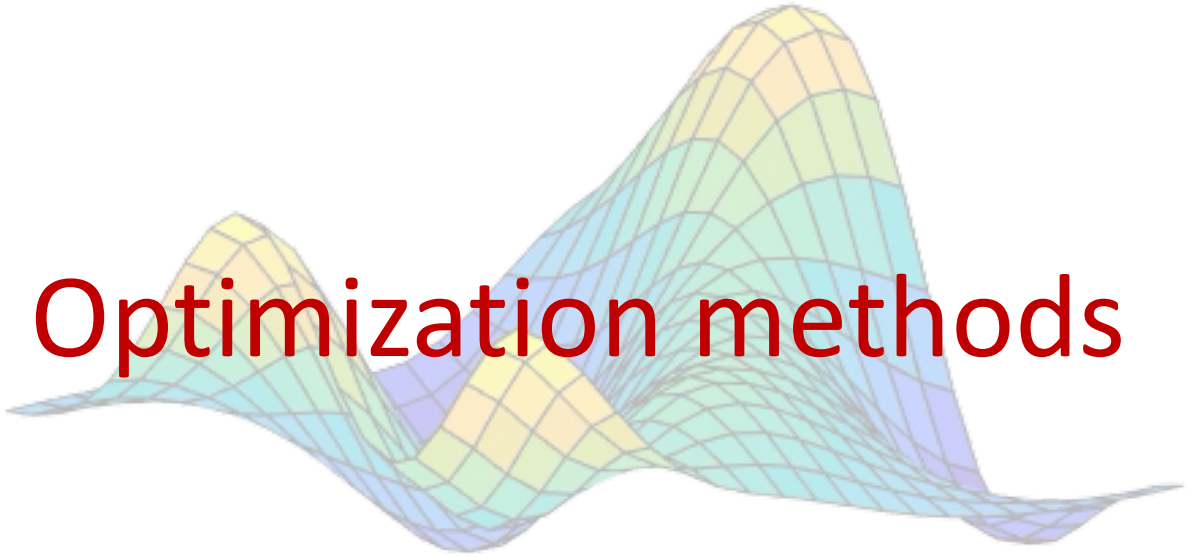
Representation: What class of functions should we use?

Optimization: How can we efficiently solve the empirical risk minimization problem?

Generalization: Will the predictions of our model transfer gracefully to unseen examples?

All related! And the fuel which powers everything is data.

Optimization methods



Problem setup

Given: a function $\mathrm{F}(\boldsymbol{w})$

Goal: minimize $\mathrm{F}(\boldsymbol{w})$ (approximately)

Two simple yet extremely popular methods

Gradient Descent (GD): simple and fundamental

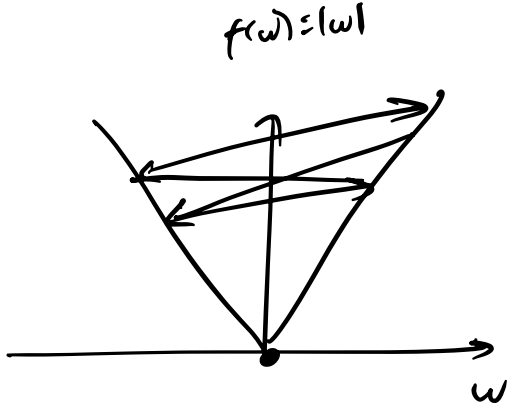
Stochastic Gradient Descent (SGD): faster, effective for large-scale problems

Gradient is the *first-order information* of a function.

Therefore, these methods are called *first-order methods*.

Gradient descent

GD: keep moving in the *negative gradient direction*



Start from some $w^{(0)}$. For $t = 0, 1, 2, \ldots$

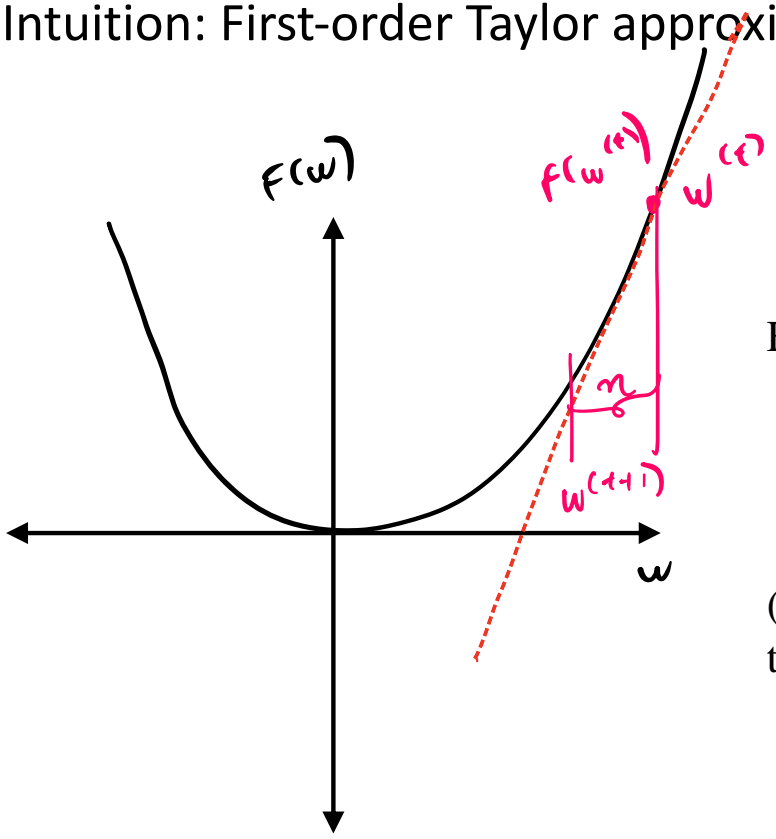
$$w^{(t+1)} \leftarrow w^{(t)} - \eta \nabla F(w^{(t)}),$$

where $\eta > 0$ is called *step size* or *learning rate*.

- in theory η should be set in terms of some parameters of f
- in practice we just try several small values
- might need to be changing over iterations (think $f(w) = |w|$)
- adaptive and automatic step size tuning is an active research area

Why GD?

Intuition: First-order Taylor approximation



$$F(\boldsymbol{w}) \approx F(\boldsymbol{w}^{(t)}) + \nabla F(\boldsymbol{w}^{(t)})^T (\boldsymbol{w} - \boldsymbol{w}^{(t)})$$

For $\boldsymbol{w} = \boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla F(\boldsymbol{w}^{(t)})$, we can write,

$$F(\boldsymbol{w}^{(t+1)}) \approx F(\boldsymbol{w}^{(t)}) - \eta \|\nabla F(\boldsymbol{w}^{(t)})\|_2^2$$

$$\implies F(\boldsymbol{w}^{(t+1)}) \lessapprox F(\boldsymbol{w}^{(t)})$$

(Note that this is only an approximation, and can be invalid if the step size is too large.)

Switch to Colab

optimization.ipynb ☆

File Edit View Insert Runtime Tools Help

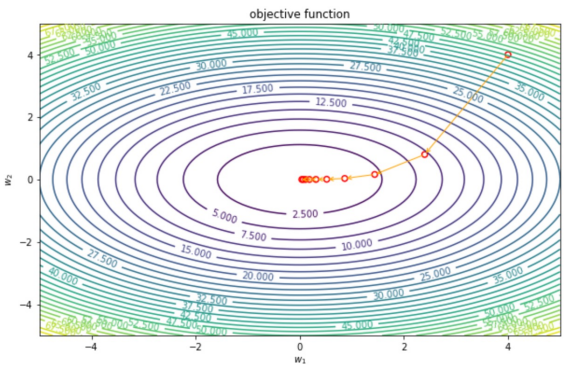
+ Code + Text

```
        this_theta[1] = last_theta[1] - eta * grad1
        theta.append(this_theta)
        J.append(cost_func(*this_theta))


    # Annotate the objective function plot with coloured points indicating the
    # parameters chosen and red arrows indicating the steps down the gradient.
    for j in range(1,N):
        ax.annotate('', xy=theta[j], xytext=theta[j-1],
                    arrowprops={'arrowstyle': '->', 'color': 'orange', 'lw': 1},
                    va='center', ha='center')
    ax.scatter(*zip(*theta), facecolors='none', edgecolors='r', lw=1.5)

    # Labels, titles and a legend.
    ax.set_xlabel(r'$w_1$')
    ax.set_ylabel(r'$w_2$')
    ax.set_title('objective function')

    plt.show()
```



Convergence guarantees for GD

Many results for GD (and many variants) on *convex objectives*.
They tell you how many iterations t (in terms of ε) are needed to achieve

$$F(\boldsymbol{w}^{(t)}) - F(\boldsymbol{w}^*) \leq \varepsilon$$

Even for *nonconvex objectives*, some guarantees exist:
e.g. how many iterations t (in terms of ε) are needed to achieve

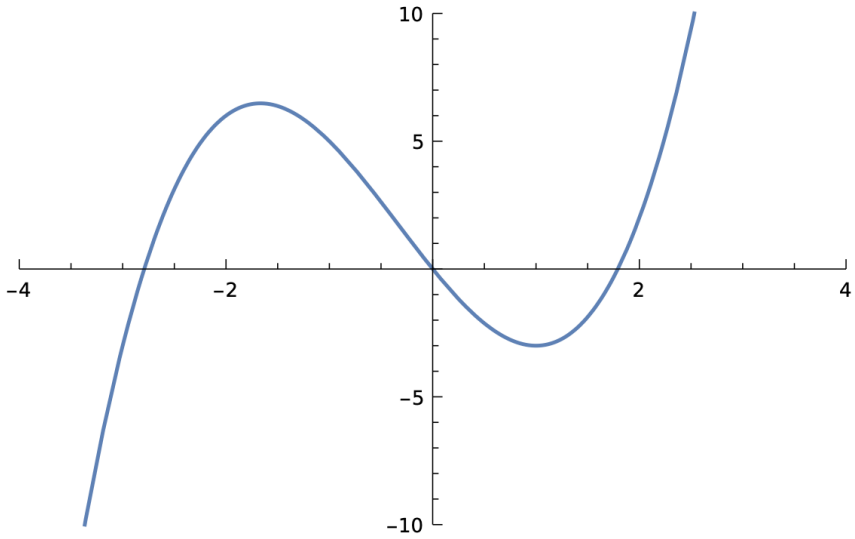
$$\left\|\nabla F(\boldsymbol{w}^{(t)})\right\| \leq \varepsilon$$

that is, how close is $\boldsymbol{w}^{(t)}$ as an approximate stationary point

for convex objectives, stationary point ⇒ global minimizer
for nonconvex objectives, what does it mean?

Stationary points: non-convex objectives

A stationary point can be a local minimizer or even a local/global maximizer (but the latter is not an issue for GD).

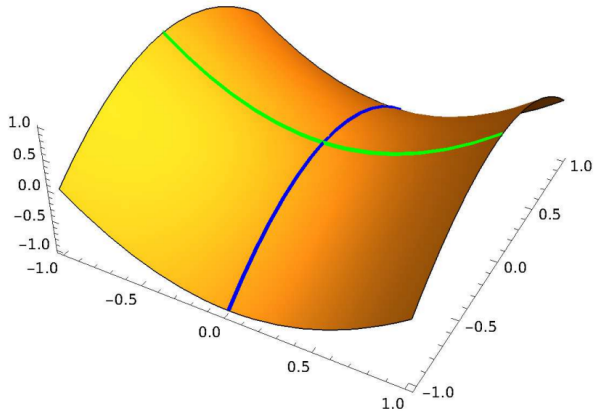


$$f(w) = w^3 + w^2 - 5w$$

Stationary points: non convex objectives

A stationary point can also be *neither a local minimizer nor a local maximizer!*

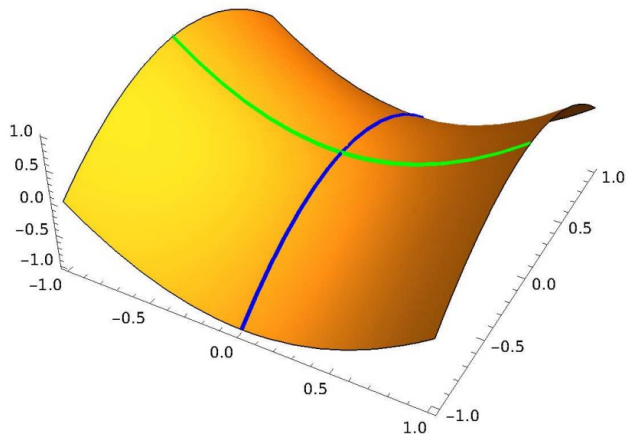
- $f(\boldsymbol{w}) = w_1^2 - w_2^2$
- $\nabla f(\boldsymbol{w}) = (2w_1, -2w_2)$
- so $\boldsymbol{w} = (0, 0)$ is stationary
- local max for blue direction $(w_1 = 0)$
- local min for green direction $(w_2 = 0)$



Switch to Colab

Stationary points: non convex objectives

This is known as a saddle point

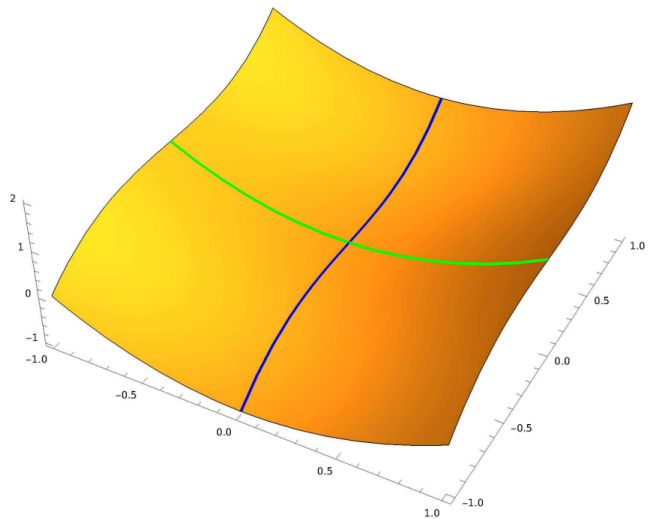


- but GD gets stuck at $(0, 0)$ only if initialized along the green direction
- so not a real issue especially *when initialized randomly*

Stationary points: non convex objectives

But not all saddle points look like a "saddle"...

- $f(\boldsymbol{w}) = w_1^2 + w_2^3$
- $\nabla f(\boldsymbol{w}) = (2w_1, 3w_2^2)$
- so $\boldsymbol{w} = (0, 0)$ is stationary
- not local min/max for blue direction $(w_1 = 0)$

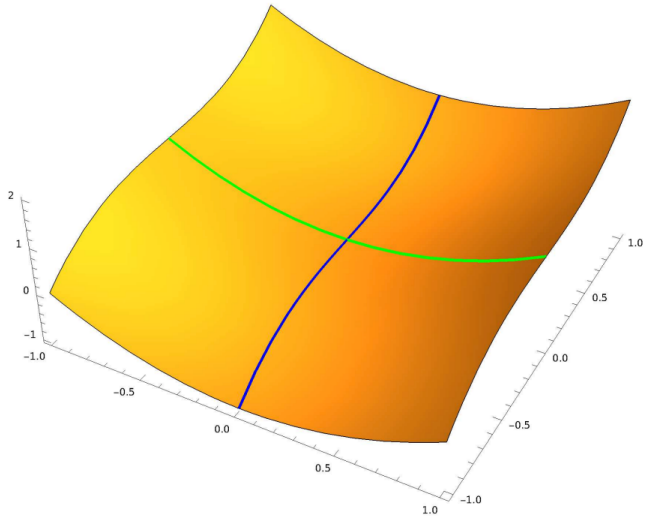


Switch to Colab

Stationary points: non convex objectives

But not all saddle points look like a "saddle"...

- $f(\boldsymbol{w}) = w_1^2 + w_2^3$
- $\nabla f(\boldsymbol{w}) = (2w_1, 3w_2^2)$
- so $\boldsymbol{w} = (0, 0)$ is stationary
- not local min/max for blue direction $(w_1 = 0)$
- GD gets stuck at $(0, 0)$ for *any initial point with $w_2 \geq 0$ and small η*



Even worse, distinguishing local min and saddle point is generally *NP-hard*.

Stochastic Gradient descent

GD: keep moving in the *negative gradient direction*

SGD: keep moving in the *noisy negative gradient direction*

$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta \tilde{\nabla} F(\boldsymbol{w}^{(t)})$$

where $\tilde{\nabla} F(\boldsymbol{w}^{(t)})$ is a random variable (called **stochastic gradient**) s.t.

$$\mathbb{E}\left[\tilde{\nabla} F(\boldsymbol{w}^{(t)})\right] = \nabla F(\boldsymbol{w}^{(t)}) \qquad \text{(unbiasedness)}$$

Stochastic Gradient descent

GD: keep moving in the *negative gradient direction*

SGD: keep moving in the *noisy negative gradient direction*

$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta \tilde{\nabla} F(\boldsymbol{w}^{(t)})$$

where $\tilde{\nabla} F(\boldsymbol{w}^{(t)})$ is a random variable (called **stochastic gradient**) s.t.

$$\mathbb{E}\left[\tilde{\nabla} F(\boldsymbol{w}^{(t)})\right] = \nabla F(\boldsymbol{w}^{(t)}) \qquad \text{(unbiasedness)}$$

- Key point: it could be much faster to obtain a stochastic gradient!
- Similar convergence guarantees, usually needs more iterations but each iteration takes less time.

Switch to Colab

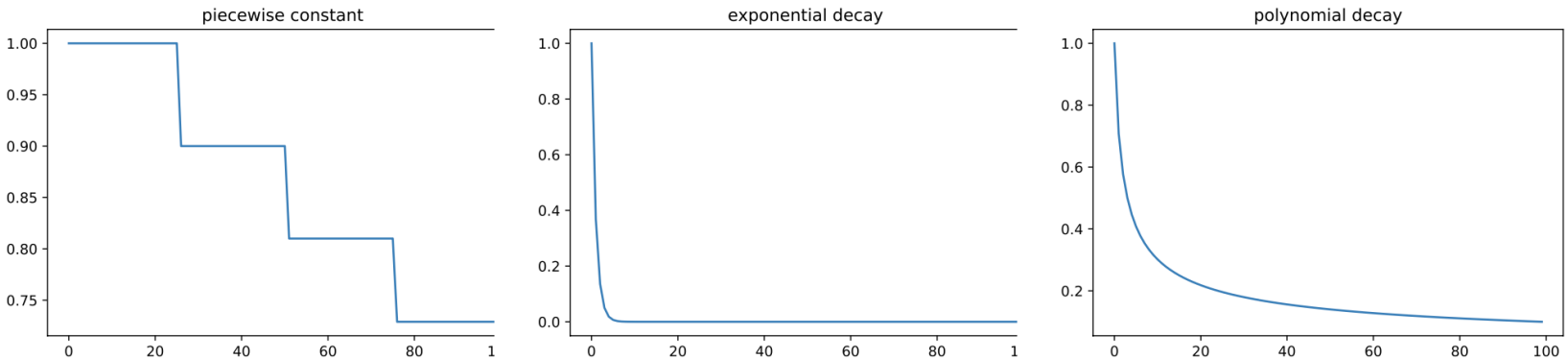
Summary: Gradient descent & Stochastic Gradient descent

- GD/SGD coverages to a stationary point. For convex objectives, this is all we need.
- For nonconvex objectives, can get stuck at local minimizers or “bad” saddle points (random initialization escapes “good” saddle points)
- Recent research shows that *many problems have no “bad” saddle points or even “bad” local minimizers*
- SGD is very popular, another very popular optimization technique is *Adam*
- Adam has two key additional ingredients: adaptive step size & momentum

Adaptive learning rate tuning

``The learning rate is perhaps the most important hyperparameter.
If you have time to tune only one hyperparameter, tune the learning rate."
-Deep learning (Book by Goodfellow, Bengio, Courville)

We often use a **learning rate schedule**.



(a) *(b)* *(c)*

Some common learning rate schedules (figure from PML)

Adaptive learning rate methods (Adagrad, RMSProp) scale the learning rate of each parameter based on some moving average of the magnitude of the gradients.

[PML] *Probabilistic Machine Learning: An Introduction* by Kevin Murphy.

Momentum

"move faster along directions that were previously good, and to slow down along directions where the gradient has suddenly changed, just like a ball rolling downhill." [PML]

Initialize $\boldsymbol{w}_0$ and (velocity) $\boldsymbol{v} = \mathbf{0}$

For $t = 1, 2, \ldots$

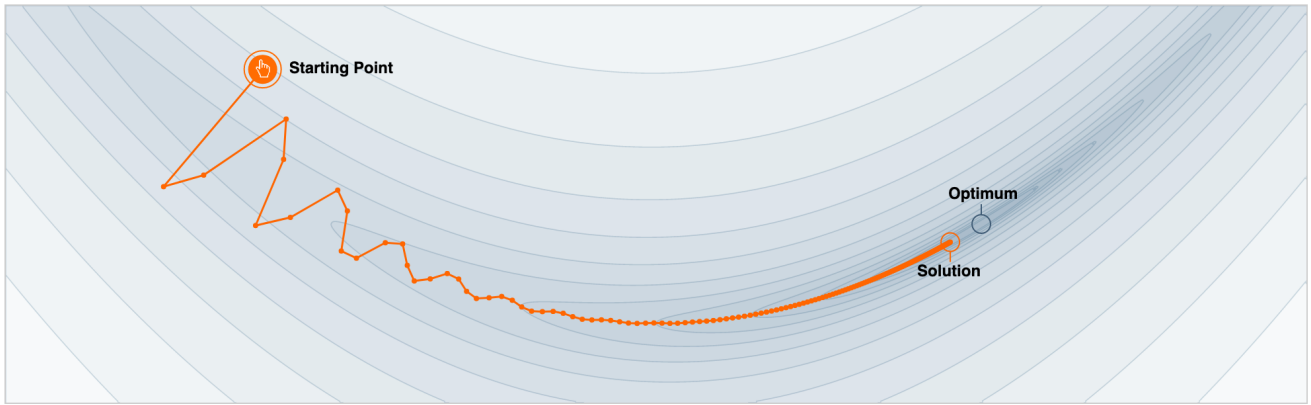
- estimate a stochastic gradient $\boldsymbol{g}_t$
- update $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} + \boldsymbol{g}_t$ for some discount factor $\alpha \in (0, 1)$
- update weight $\boldsymbol{w}_t \leftarrow \boldsymbol{w}_{t-1} - \eta \boldsymbol{v}$

Updates for first few rounds:

- $\boldsymbol{w}_1 = \boldsymbol{w}_0 - \eta \boldsymbol{g}_1$
- $\boldsymbol{w}_2 = \boldsymbol{w}_1 - \alpha\eta \boldsymbol{g}_1 - \eta \boldsymbol{g}_2$
- $\boldsymbol{w}_3 = \boldsymbol{w}_2 - \alpha^2\eta \boldsymbol{g}_1 - \alpha\eta \boldsymbol{g}_2 - \eta \boldsymbol{g}_3$
- $\ldots$

Momentum

Why Momentum Really Works



Step-size α = 0.02

0 0.003 0.006

Momentum β = 0.99

0.00 0.500 0.990

We often think of Momentum as a means of dampening oscillations and speeding up the iterations, leading to faster convergence. But it has other interesting behavior. It allows a larger range of step-sizes to be used, and creates its own oscillations. What is going on?

GABRIEL GOH
UC Davis

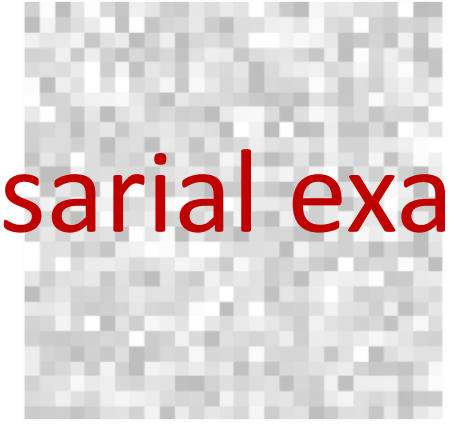
April. 4
2017

Citation:
Goh, 2017

https://distill.pub/2017/momentum/

Adversarial examples

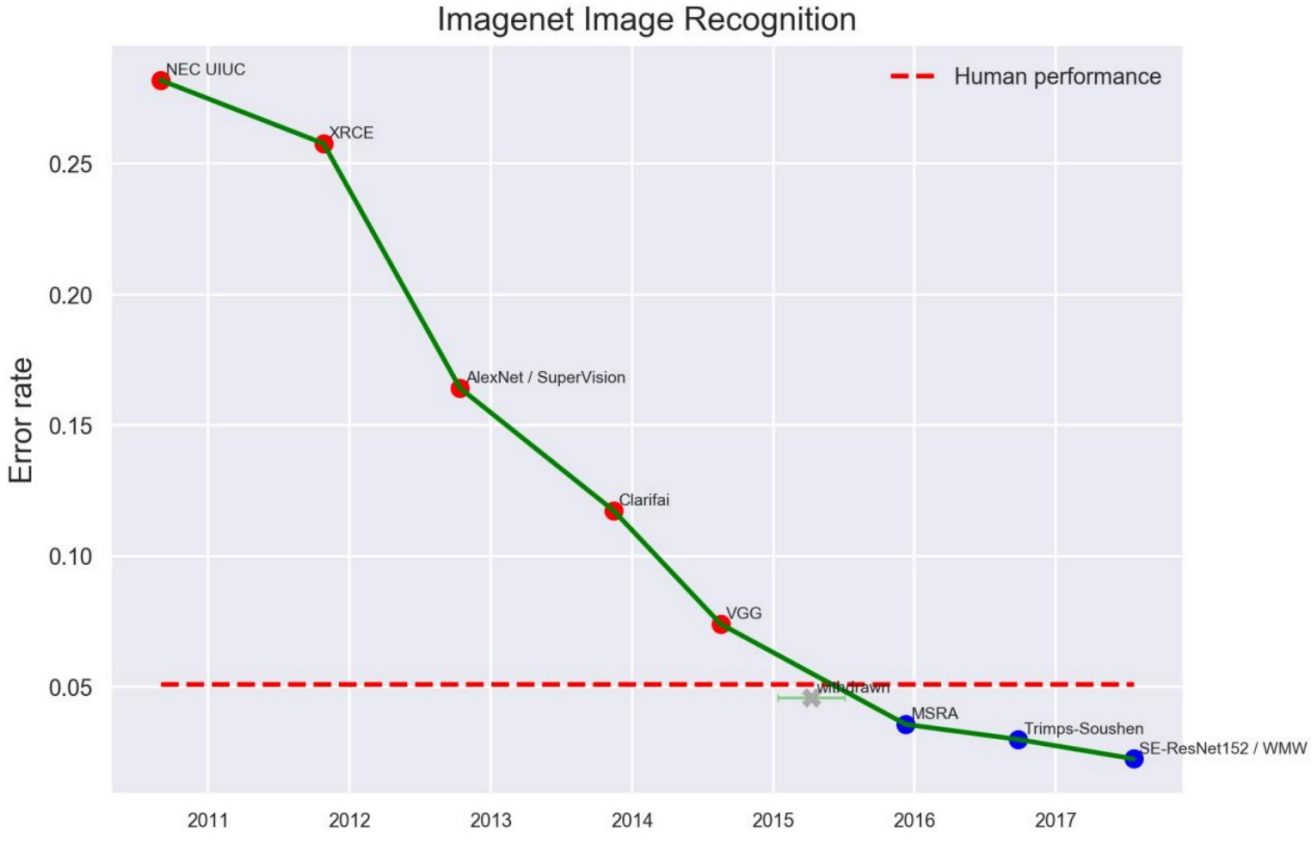
Pig
(90% confidence)



Small amount of
adversarial noise

Airplane!
(99.9% confidence)

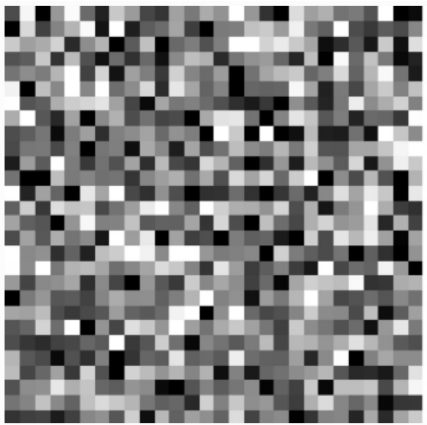
CNNs are great at image classification



However, models can also be very sensitive to small variations in the input

Pig
(90% confidence)

+



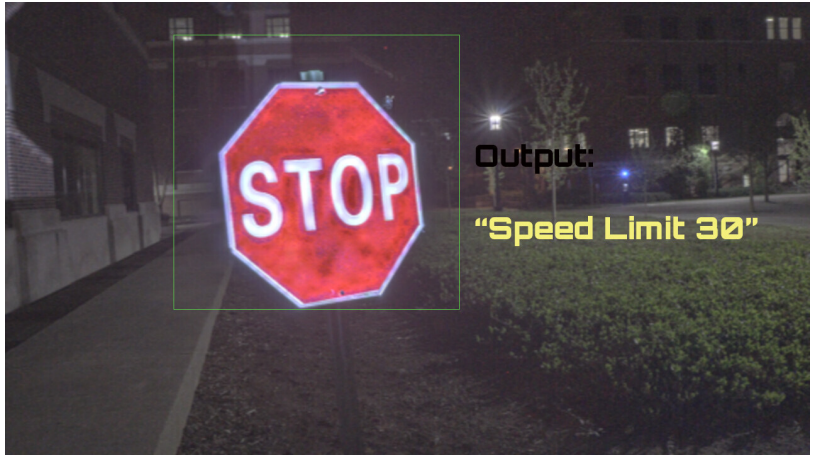
Small amount of
adversarial noise

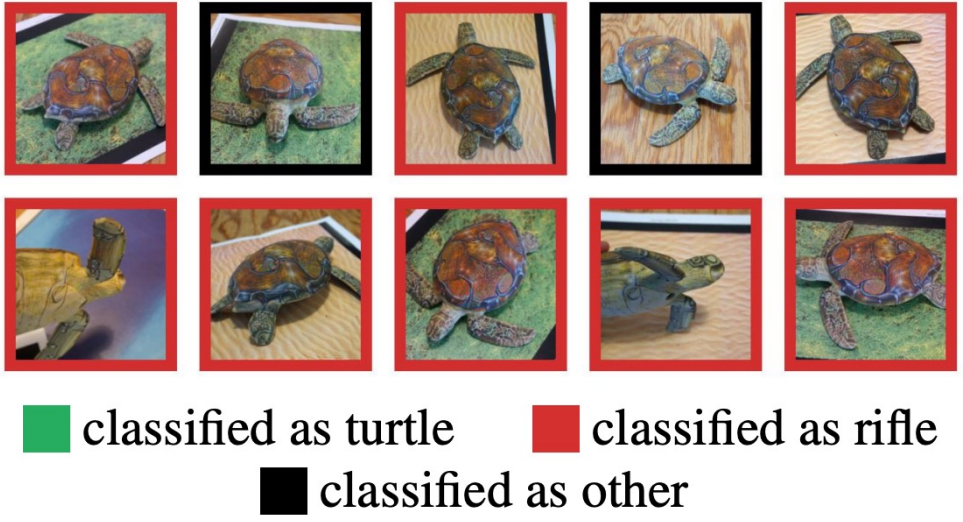
=

Airplane!
(99.9% confidence)

Image adapted from https://adversarial-ml-tutorial.org/

These are known as ***adversarial examples***





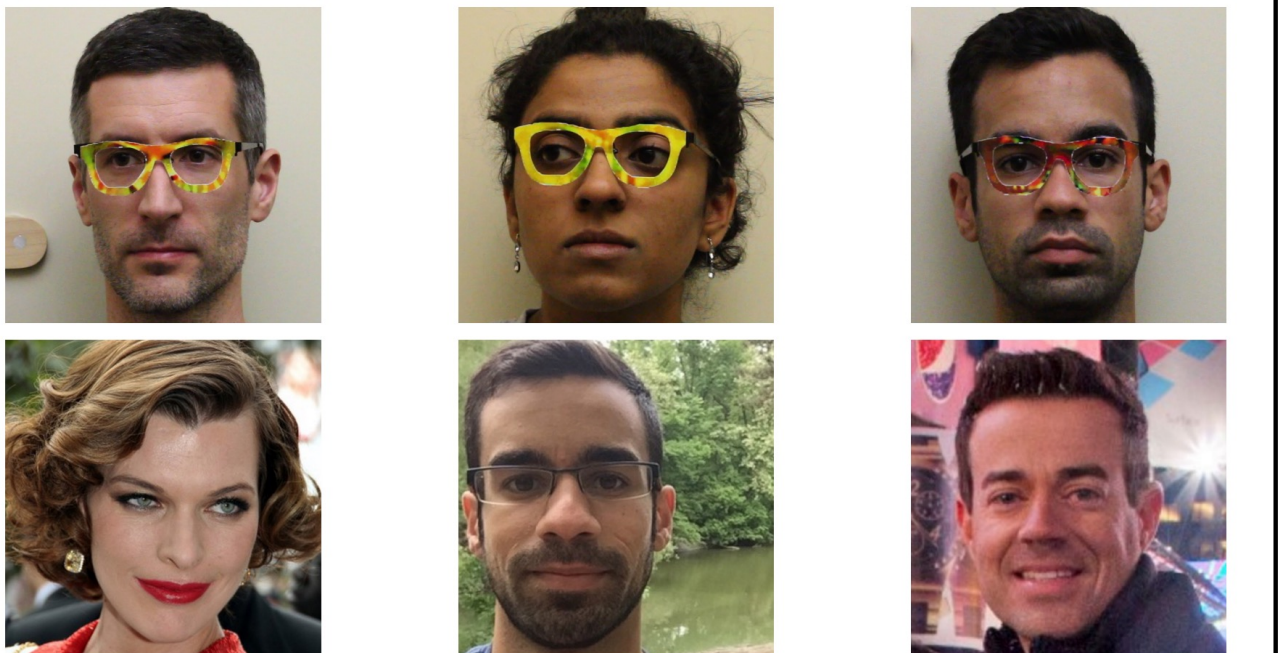
Adversarial examples have been shown to also hold for real-world tasks.

They are an issue because

1. Can pose potential security risks
2. Indicate that even though models are good, they don't quite work the same way as we do

More studies on adversarial examples

Dodging detection from face detection using glasses



Person in top row impersonating person in bottom row using glasses

From *Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition, Sharif et al. '16*

Examples from other modalities

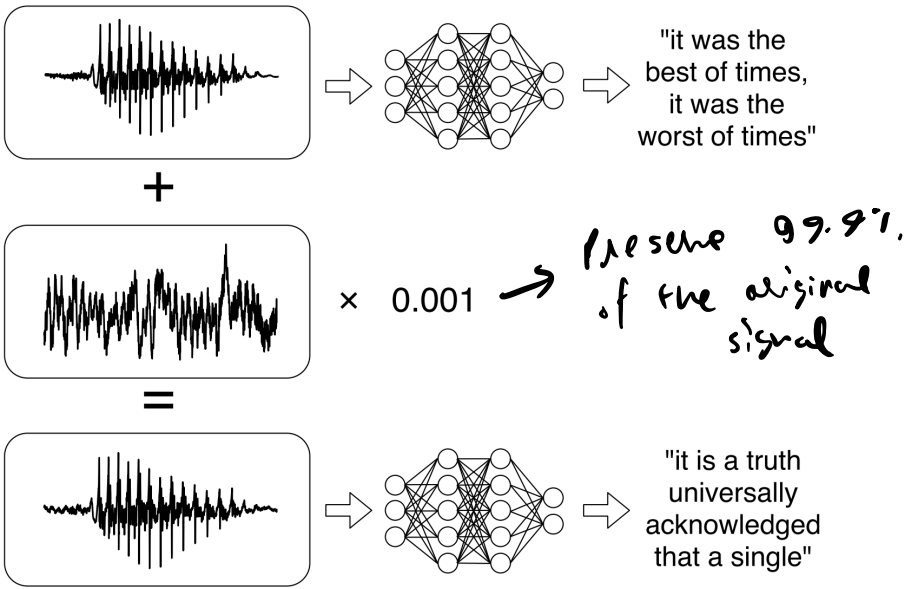


Figure 1. Illustration of our attack: given any waveform, adding a small perturbation makes the result transcribe as any desired target phrase.

From *Audio Adversarial Examples: Targeted Attacks on Speech-to-Text, Carlini & Wagner '18*

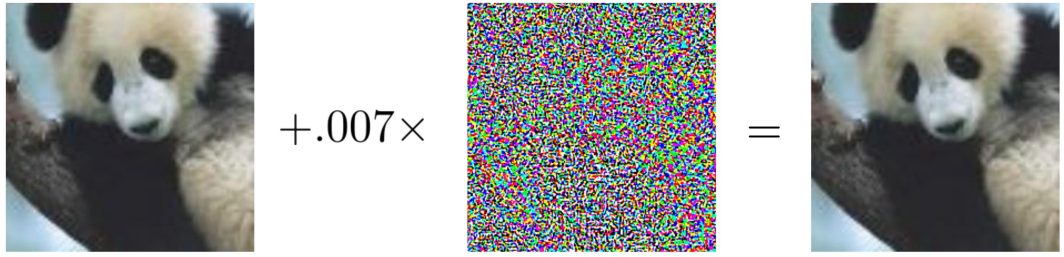
Article: Super Bowl 50
Paragraph: "*Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*"
Question: "*What is the name of the quarterback who was 38 in Super Bowl XXXIII?*"
Original Prediction: John Elway
Prediction under adversary: Jeff Dean

From *Adversarial Examples for Evaluating Reading Comprehension Systems, Jia & Liang '17*

Adversarial examples

Part 1: Basics of finding adversarial examples

Adversarial examples: Setup



Panda
58% confidence

Nematode
8% confidence

Gibbon
99% confidence

Adversary: Given an image x and classifier $f(x)$, comes up with some other image x' which is "similar" to x, such that $f(x) \neq f(x')$.

How to define similarity? One notion is small perturbations based on some norm. We typically consider the ℓ_∞ norm: $\| x - x' \|_\infty \leq \epsilon$, where ϵ is the allowed perturbation level.

This means: can perturb every pixel by a perturbation in $[-\epsilon, \epsilon]$.

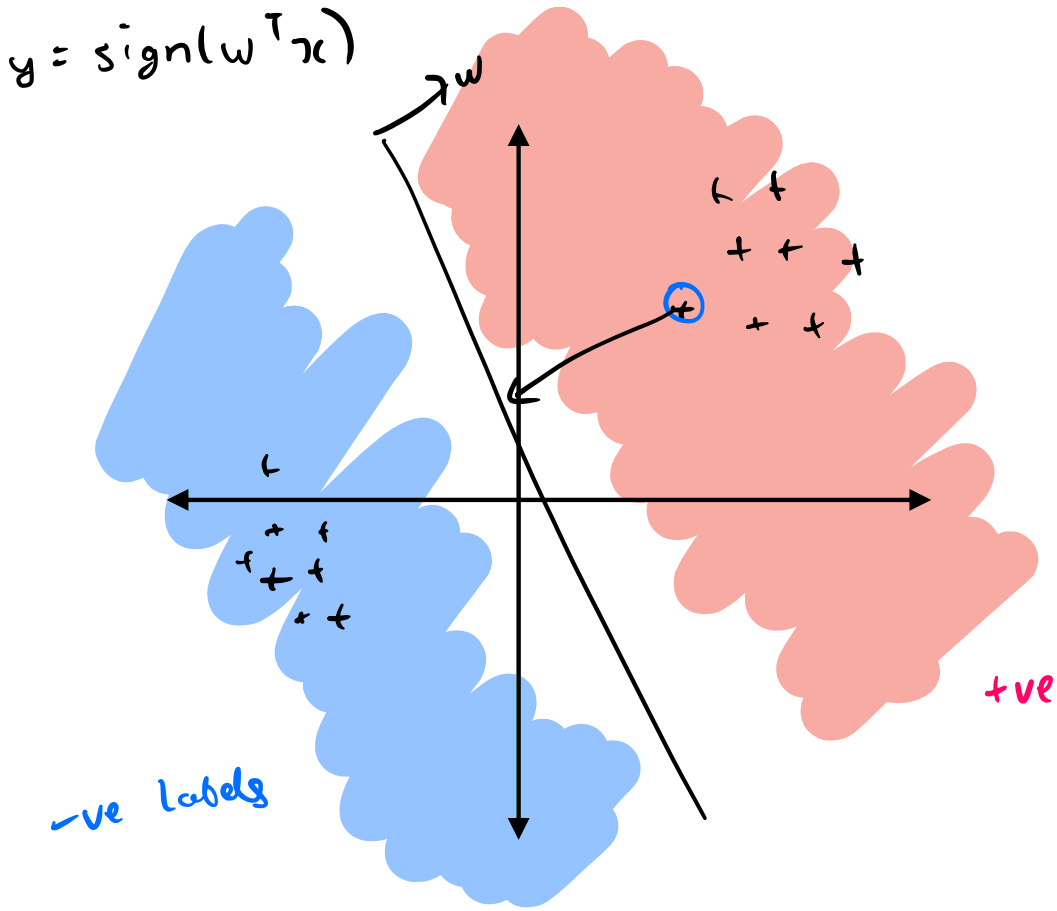
Finding Adversarial examples: Optimization problem

Adversary: Given an image x and classifier $f(x)$, comes up with some other image x' which is "similar" to x, such that $f(x) \neq f(x')$.

$$\Delta = \{\delta \in \mathbb{R}^d : \|\delta\|_\infty \leq \varepsilon\}.$$

$$\ell_{\text{adv}}(f; x, y) := \max_{\delta \in \Delta} \ell\big(f(x+\delta),\, y\big).$$

Detour: Linearity as a source of brittleness



$$x' = x - \text{sign}(w)\,\delta$$

$$w^T x' = w^T x - w^T \text{sign}(w)\cdot\delta$$

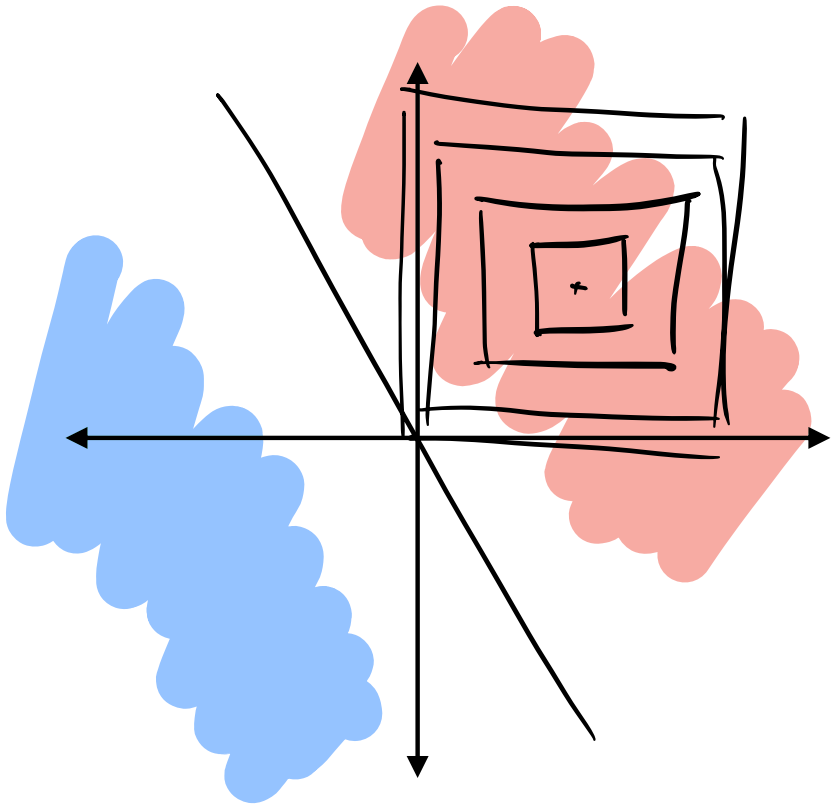
$$= w^T x - \|w\|_1\,\delta$$

$\|w\|_1$: grows with dimension

(in particular, typically grows faster than $w^T x$ with dimension)

For more details, see *Explaining and Harnessing Adversarial Examples, Goodfellow et al. '15*

Finding adversarial perturbation for linear models



Claim: $x_{adv} = x - \text{sign}(w)\cdot\delta$

changes predictions the most among

all x' s.t. $\|x - x'\|_\infty = \delta$.

For more details, see *Explaining and Harnessing Adversarial Examples, Goodfellow et al. '15*

Back to our optimization problem

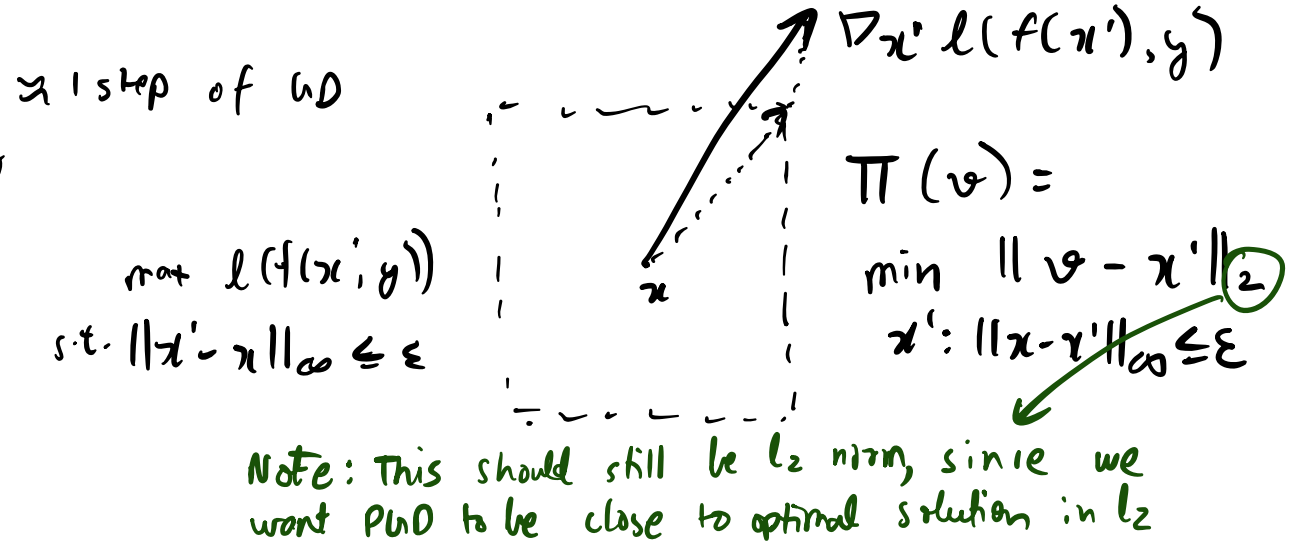
Adversary: Given an image x and classifier $f(x)$, comes up with some other image x' which is "similar" to x, such that $f(x) \neq f(x')$.

$$\Delta = \{\delta \in \mathbb{R}^d : \|\delta\|_\infty \leq \varepsilon\}$$

$$\ell_{\text{adv}}(f; x, y) := \max_{\delta \in \Delta} \ell\big(f(x+\delta), y\big)$$

Fast gradient sign method (FGSM):

$$\delta_{\text{FGSM}} = \epsilon \cdot \text{sign}(\nabla_x \ell(f(x), y)),$$
$$x^{\text{adv}} = x + \delta_{\text{FGSM}}.$$

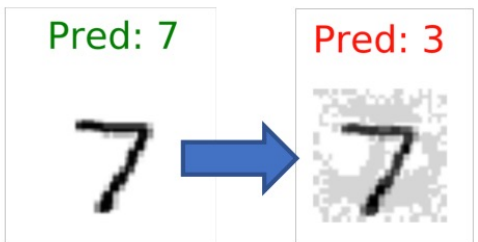


FGSM: Results

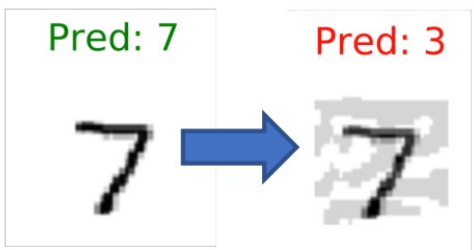
Fast gradient sign method (FGSM):

$$\delta_{\text{FGSM}} = \epsilon \cdot \text{sign}(\nabla_x \, \ell(f(x), y)) \,,$$
$$x^{\text{adv}} = x + \delta_{\text{FGSM}}.$$

MLP:



ConvNet:



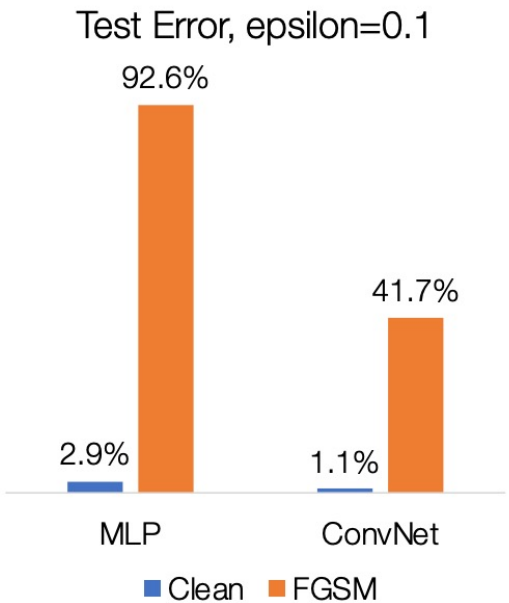


Image from https://adversarial-ml-tutorial.org/ (part 2)

Adversarial examples

Part 2: Basics of defending against adversarial examples

Adversarial robustness: The optimization problem

Adversary: Given an image x and classifier $f(x)$, comes up with some other image x' which is "similar" to x, such that $f(x) \neq f(x')$.

$$\Delta = \{\delta \in \mathbb{R}^d : \|\delta\|_\infty \leq \varepsilon\}$$

$$\ell_{\text{adv}}(f; x, y) := \max_{\delta \in \Delta} \ell\big(f(x+\delta), y\big)$$

$$\widehat{R}_S^{\text{adv}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell_{\text{adv}}(f; x_i, y_i), \quad \rightarrow \text{adversarial empirical risk}$$

Defender: Train a model such that the adversary is not effective at finding adversarial examples

$$\min_{f \in \mathcal{F}} \widehat{R}_S^{\text{adv}}(f) = \min_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \max_{\delta \in \Delta} \ell\big(f(x_i+\delta), y_i\big).$$

we can take gradients

Min-max optimization: Danskin's Theorem

Defender: Train a model such that the adversary is not effective at finding adversarial examples

$$\min_{f \in \mathcal{F}} \widehat{R}_S^{\mathrm{adv}}(f) = \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \max_{\delta \in \Delta} \ell\big(f(x_i + \delta),\, y_i\big).$$

Danskin's theorem: The gradient of the inner maximization objective with respect to f is given by the gradient at the maximizer of the inner objective.

$$\nabla_f \max_{\delta \in \Delta} \ell\big(f(x+\delta), y\big) = \nabla_f \ell\big(f(x+\delta^\star), y\big),$$

$$\text{where } \delta^\star = \arg\max_{\delta \in \Delta} \ell\big(f(x+\delta), y\big).$$

Adversarial training, inspired by Danskin's Theorem

Danskin's theorem:

$$\nabla_f \max_{\delta \in \Delta} \ell\big(f(x+\delta), y\big) = \nabla_f \, \ell\big(f(x+\delta^\star), y\big),$$

$$\text{where } \delta^\star = \arg\max_{\delta \in \Delta} \ell\big(f(x+\delta), y\big).$$

Repeat:

1. Select a minibatch B of b examples $\{(x_i, y_i)\}_{i=1}^b$ from the training set.

2. For each $(x_i, y_i) \in B$, compute adversarial perturbation using FGSM → pretending that FGSM finds the maximizer

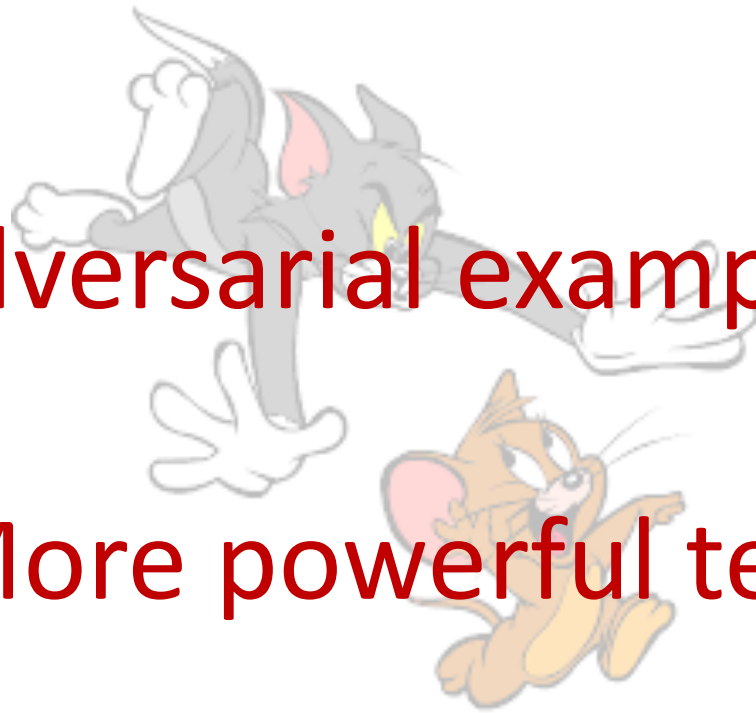
$$\delta_i^\star = \arg\max_{\delta \in \Delta} \ell\big(f(x_i+\delta), y_i\big).$$

3. Update parameters (for some learning rate α):

$$f := f - \alpha \sum_{i=1}^{b} \nabla_f \, \ell\big(f(x_i + \delta_i^\star), \, y\big).$$

	Clean Error	FGSM Error
ConvNet	1.1%	41.7%
Robust ConvNet	0.9%	2.6%

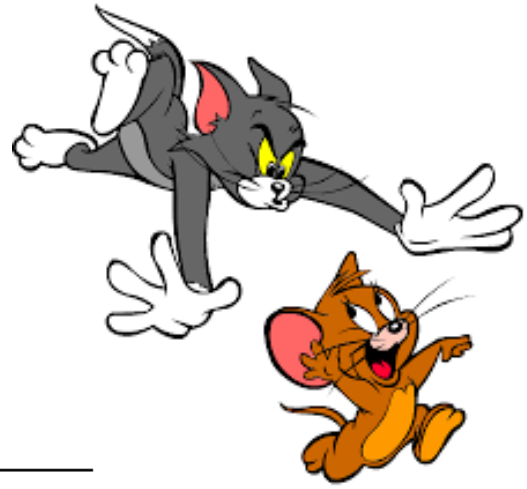
FGSM results on MNIST ($\epsilon = 0.1$)



Adversarial examples

Part 3: More powerful techniques

A cautionary tale



Adversarial examples research has had the nature of a cat-and-mouse game.

Specific defenses often work against the specific attack they were designed for, but fail more generally.

Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples

---

Anish Athalye [*1] **Nicholas Carlini** [*2] **David Wagner** [2]

This ICML 2018 paper broke 7 out of 9 defense methods appearing a few months ago at ICLR 2018!
The methods were shown to rely on “obfuscated gradients”.

Therefore, important to consider strong attack models.

Projected gradient descent (PGD) for finding adversarial examples

Feasible set: $\Delta = \{\delta \in \mathbb{R}^d : \|\delta\|_\infty \leq \epsilon\}, \quad \mathcal{S}(x) = x + \Delta.$

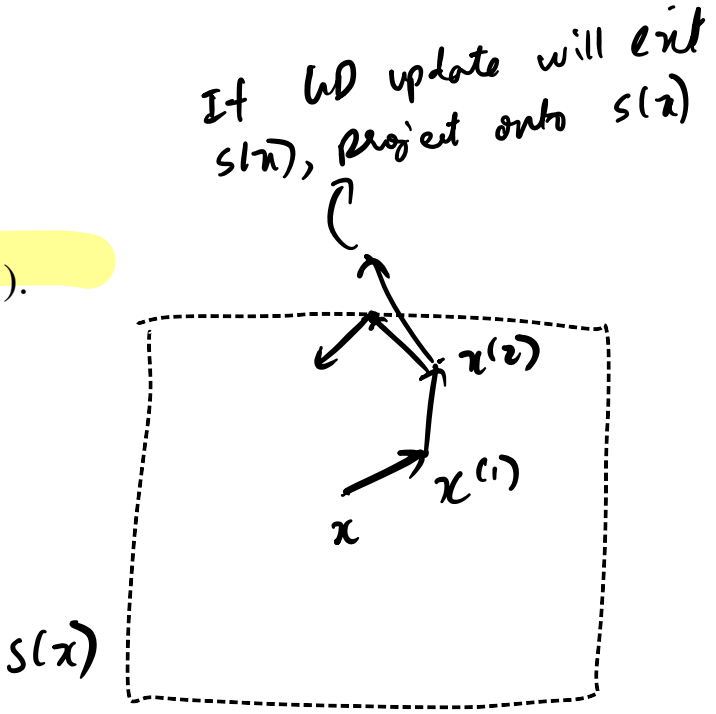
Initialize: $x^{(0)} = x$ (optionally $x^{(0)} = x + \eta,\ \eta \sim \mathrm{Unif}([-\epsilon, \epsilon]^d)$).

For $t = 0, \ldots, T-1$:

$$g^{(t)} = \nabla_x \ell\big(f(x^{(t)}), y\big),$$

$$x^{(t+1)} = \Pi_{\mathcal{S}(x)}\big(x^{(t)} + \alpha \cdot \mathrm{sign}(g^{(t)})\big).$$

Output: $x^{\mathrm{adv}} = x^{(T)}$.



From *Towards Deep Learning Models Resistant to Adversarial Attacks, Madry et al. '18*

Adversarial training using PGD

Repeat:

1. Select a minibatch B of b examples $\{(x_i, y_i)\}_{i=1}^{b}$ from the training set.
2. For each $(x_i, y_i) \in B$, compute adversarial perturbation using PGD

$$\delta_i^\star = \arg\max_{\delta \in \Delta} \ell\big(f(x_i + \delta), y_i\big).$$

→ more computationally expensive

3. Update parameters (for some learning rate α):

$$f \;:=\; f - \alpha \sum_{i=1}^{b} \nabla_f \, \ell\big(f(x_i + \delta_i^\star),\, y\big).$$

Adversarial training using PGD & FGSM, Results

Repeat:

1. Select a minibatch B of b examples $\{(x_i, y_i)\}_{i=1}^{b}$ from training set.
2. For each $(x_i, y_i) \in B$, compute adversarial perturbation

$$\delta_i^\star = \arg\max_{\delta \in \Delta} \ell\big(f(x_i + \delta), y_i\big).$$

3. Update parameters (for some learning rate α):

$$f \;:=\; f - \alpha \sum_{i=1}^{b} \nabla_f \, \ell\big(f(x_i + \delta_i^\star),\, y\big).$$

Observations

① PGD attack more effective than FGSM

② PGD gives some robustness against FGSM, but not as much as FGSM training (?)

③ FGSM better on FGSM vs Natural !

④ Natural accuracy gets worse with adversarial training

Training type →

Evaluation type ↓

	(a) Standard	(b) FGSM	(c) PGD
Natural	95.2%	90.3%	87.3%
FGSM	32.7%	95.1%	56.1%
PGD	3.5%	0.0%	45.8%

Accuracy on CIFAR10 ($\epsilon = 8$)

From *Towards Deep Learning Models Resistant to Adversarial Attacks, Madry et al. '18*